csound-ac

Generated by Doxygen 1.10.0

Chapter 1

csound-ac

Michael Gogins
https://github.com/gogins
http://michaelgogins.tumblr.com

1.1 Introduction

This repository contains:

1. CsoundAC, an algorithmic composition library, designed to be used with Csound. CsoundAC is written in C++, and has both C++ and Python interfaces. CsoundAC implements *music models*, which are kind of like scene graphs for pieces. CsoundAC has sophisticated facilities for working with tonal and non-tonal chords, progressions, and scales, and for implementing classical-style voice-leading in generated scores.
2. My computer music playpen, designed to facilitate algorithmic composition with Csound and CsoundAC by extending standard text editors. The playpen makes it possible to run various kinds of Csound pieces, and even to build C++ pieces and plugin opcodes, from the editor. For more information, see `playpen/README.md`.
3. My Visual Studio Code extension that implements the computer music playpen. Consider working in this environment. For more information, see `vscode-playpen/README.md`
4. silencio, a JavaScript library for algorithmic composition similar to CsoundAC. However, using the WebAssembly build of CsoundAC in `csound-wasm` is now recommended in place of silencio.
5. patches, a library of Csound instrument definitions, developed over many years and used in many of my pieces.

Currently, CsoundAC is supported on macOS and Linux.

Please log any bug reports or requests for enhancements at https://github.com/gogins/csound-ac/issues.

1.2 Changes

See https://github.com/gogins/csound-ac/commits/develop for the commit log.

1.3 Using

CsoundAC can be used both as a C++ library, as a Python extension module, and as a WebAssembly kodue. Python is easier to use, but C++ offers considerably more power and speed. The WebAssembly build of CsoundAC in `csound-wasm` has the same power as the C++ library and somewhat less speed.

Examples (some of which can also serve as tests) for the various aspects of csound-ac are maintained in my separate `csound-examples` repository. Some of the examples there will run in WebBrowsers using WebAssembly, and these can be viewed at `https://gogins.github.io/csound-examples`.

1.4 Installation

1. You must first install the following pre-requisites on your system:
   1.1 `Libsndfile` for reading and writing most any format of soundfile.
   1.2 `Csound` for sound synthesis.
   1.3 The `Eigen` header-file-only library for linear algebra.
   1.4 The `Boost C++ Libraries`. Only the header files are used.
   1.5 The `OpenCV` library for image processing.
   1.6 The `Python` programming language, version 3.9 or higher.
2. There are prebuilt binary releases for this package available at `https://github.com/gogins/csound-ac/releases`. These can be downloaded, unzipped to `/usr/local` (or even `~/usr/local`), and used from there. The binary files are archives, not installers. They should be installed in the same way on all platforms:
   (a) Download the archive from the releases page.
   (b) Use the 7z program to unzip the archive to the output directory, e.g. `7z x -o/usr/local csound-ac-0.5.0-Darwin.zip`. 7z will ask you what to do about any files that it might overwrite.
   (c) Run `sudo ldconfig` or take equivalent steps to ensure that the libraries can be found by the operating system. You may need to add appropriate directories to your compiler's header files path, and to the operating system `PATH` environment variable.
   (d) To uninstall, you can list the contents of the archive to a file, e.g. `unzip unzip csound-ac-0.5.0-Darwin.zip -l > listing.txt`. You can use this to identify files to remove, and you could even write a script to parse `listing.txt` and remove all files listed therein.

   Please note, on macOS the CsoundAC libraries are installed by default to `/usr/local/lib`. They can be installed elsewhere, but if so, you will probably need to set up install names and rpaths using `otool`.

1.5 Helpers

There are files and directories in the Git repository and in the packages that can be used as helpers for csound-ac. You can create symbolic links from these files to your home directory or other places.

- `build-env.sh`: Source this to set useful environment variables for the build and runtime environment on Linux. You may need to copy and modify this script.

- Creata a symbolic link from `csound-ac/playpen/playpen.py` to your home directory, to enable use of the computer music plapen.
- Copy `csound-ac/playpen/playpen.ini` to your home directory and customize it for your environment, to configure the computer music playen.
- If you use Visual Studio Code, install in it the `playpen.vsix` extension, which makes the computer music playpen part of Visual Studio Code.
- If you use the SciTE text editor, Create a symbolic link from `csound-ac/playpen/.SciTEUser.properties` to your home directory, which makes the computer music playpen part of SciTE.
- `silencio`: Create a symbolic link to this directory in every directory in which you are writing or running a piece that uses the Silencio library.
- `patches`: Include the full path of this directory in your Csound environment variable `INCDIR`.

1.6 Building On Your Local Computer

The following instructions are for macOS. Linux is similar. For more information, look at `./github/cmake.yaml`. However, on Linux it may be better to build Csound for source code.

1. Clone this Git repository.
2. Install prerequisites as follows from the repository root directory:
```
brew update
brew upgrade
brew install graphviz
brew install doxygen
brew install opencv
brew install csound
brew install bwfmetaedit
brew install sox
brew install ffmpeg
brew install lame
brew install sndfile
brew install imagemagick
git clone "https://gitlab.com/libeigen/eigen.git"
```
3. Execute `bash update-dependencies.sh`.
4. Build like this:
```
mkdir -p build-macos
cd build-macos
rm -f CMakeCache.txt
cmake -Wno-dev .. -DCMAKE_PREFIX_PATH=/usr/local:/usr
make -j6 VERBOSE=1
sudo make install
```

1.7 Release Notes

1.7.1 v7.0

- There is a new version of the Computer Music Playpen, in the form of a Visual Studio Code extension. This is now the recommnded environment for using CsoundAC.
- CsoundAC no longer implements the Soundfile class.

- CsoundAC no longer maintains the Lua binding.
- CsoundAC has internalized support for the CppSound class, because the brew package for Csound on macOS includes the the `libcsnd6` dylib without the corresponding header files. This may create a breaking change in the API for some users, but makes it possible to keep maintaining the continuous integration builds and releases of CsoundAC on GitHub.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 cmask Namespace Reference

5.2 csound Namespace Reference

C S O U N D.

Data Structures

- struct are_cl_objects
- struct are_cl_objects< Head, Tail... >
- struct AscendingDistanceComparator
- class Cell

  Score node that simplifies building up structures of motivic cells, and incrementally transforming them, as in Minimalism.
- class CellAdd

  The indicated factor is added to the indicated dimension of each note produced by the child nodes of this, beginning at the start index and proceeding up to but not including the end index, at the specified stride.
- class CellChord

  Notes produced by the child nodes of this are conformed to the chord, starting at the indicated start index, up to but not including the end index, at the indicated stride.
- class CellMultiply

  The indicated dimension of each note produced by the child nodes of this, beginning at the start index and proceeding up to but not including the end index, at the specified stride, is multiplied by the indicated factor.
- class CellRandom

  Notes produced by the child nodes of this, starting at the indicated start index, up to but not including the indicated end index, at the indicated stride, have added to them a random variable from the indicated distribution, rescaled to the indicated minimum and range.
- class CellReflect

  The indicated dimension of each note produced by the child nodes of this, beginning at the start index and proceeding up to but not including the end index, at the specified stride, is reflected (i.e.
- class CellRemove

Notes are removed from the notes produced by the child nodes of this, beginning at the indicated start index, up to but not including the end index, at the indicated stride.

- class CellRepeat

  All notes produced by child nodes are repeated for the specified number of iterations, beginning at the start index and proceeding up to but not including the end index, at the specified stride.
- class CellSelect

  The notes produced by the child nodes of this are returned as sampled from the indicated start index, up to but not including the indicated end index, at the indicated stride.
- class CellShuffle

  Notes produced by the child nodes of this, starting at the indicated start index, up to but not including the indicated end index, at the indicated stride, are randomly shuffled as to time.
- class Chord

  Chords consist of simultaneously sounding pitches.
- class ChordLindenmayer

  A Lindenmayer system consists of a turtle representing a position in musical space, that is, a note; commands for moving the turtle or writing its state into a musical score; an axiom or initial set of commands; and zero or more rules for replacing commands with arbitrary sequences of commands.
- class ChordScore

  Score equipped with chords.
- class Chunk
- class CMaskNode

  Uses the CMask library for tendency masks to generate events as a Csound score in the format determined by the CMask parameters text.
- struct compare_by_normal_form
- struct compare_by_normal_order
- struct compare_by_op
- class Composition

  Base class for user-defined musical compositions.
- class Conversions

  Conversions to and from various music and signal processing units.
- class CounterpointNode

  Uses Bill Schottstaedt's species counterpoint generator code to either (a) generate a counterpoint in species 1, 2, or 3 for a cantus firmus selected from notes generated by child nodes, or (b) attempt to correct the voice leading for species 1, 2, or 3 counterpoint in notes generated by child nodes.
- class CsoundProducer

  Optionally adds metadata, performs post-processing, translates to various soundfile formats as automatic steps in the Csound rendering of a composition to a soundfile.
- class Event
- class Exception

  Base class for C++ exceptions in the Silence system.
- class ExternalNode

  ExternalNode runs a stored script with a specified command line, and imports Csound "i" statements printed by the script to stdout as CsoundAC Event objects in a CsoundAC Score.
- class Generator

  Node that uses any callable to implement Node::generate.
- struct HarmonyEvent

  Associates a Chord with an Event representing a musical note.
- class HarmonyIFS

  HarmonyIFS is a class for doing algorithmic music composition by means of fractal interpolation functions.
- class HarmonyIFS2

HarmonyIFS is a class for doing algorithmic music composition by means of fractal interpolation functions.

- class HarmonyInterpolationPoint

  *Represents an interpolation point with scaling factors for a fractal interpolation function in the **time-harmony subspace** of the score space.*
- class HarmonyInterpolationPoint2

  *Represents an interpolation point with scaling factors for a fractal interpolation function in the **time-harmony subspace** of the score space.*
- class HarmonyPoint

  Represents a point on a time line in a score space that has a time- harmony subspace.
- class HarmonyPoint2

  Represents a point on a time line in a score space that has a time- harmony subspace.
- struct HyperplaneEquation
- class ImageToScore2

  Translates images files to scores.
- class Intercut

  The notes produced by each child node are intercut to produce the notes produced by this; e.g.
- struct is_cl_object
- struct is_cl_object< cl_object >
- class KMeansMCRM

  Uses k-means clustering to translate the accumulated samples that approximate the measure on the iterated function system implemented by the multiple copy reducing machine algorithm into a specified number of notes.
- class Koch

  All notes produced by child[N - 1] are rescaled and stacked on top of each note produced by child[N - 2], and so on.
- class Lindenmayer

  This class implements a Lindenmayer system in music space for a turtle that writes either notes into a score, or Jones-Parks grains into a memory soundfile.
- class LispGenerator

  Node that uses Lisp code to generate Events.
- class LispNode

  Base class for Nodes that can use embedded Lisp code to generate or transform Events.
- class LispTransformer

  Node that uses Lisp code to transform Events produced by child Nodes.
- class Logger
- struct MatrixCell
- class MCRM
- class MidiEvent

  This class is used to store ALL Midi messages.
- struct MidiEventComparator
- class MidiFile

  Reads and writes format 0 and format 1 standard MIDI files.
- class MidiHeader
- class MidiTrack
- class MusicModel

  A ScoreModel that uses Csound to render generated scores, via the CppSound class.
- class Node

  Base class for all music graph nodes in the Silence system.
- class PITV

  This class implements a cyclic additive group for all chords under cardinality, permutational, and range equivalence.
- class Random

A random value will be sampled from the specified distribution, translated and scaled as specified, and set in the specified row and column of the local coordinates.

- class RemoveDuplicates

  Removes all duplicate events produced by the child nodes of this.
- class Rescale

  Rescales all child events to fit a bounding hypercube in music space.
- class Scale

  Scale as a class; must be created with the name of the scale.
- struct SCOPED_DEBUGGING
- class Score

  Base class for collections of events in music space.
- class ScoreModel

  Base class for compositions that use the principle of a music graph to generate a score.
- class ScoreNode

  Node class that produces events from the contained score, which can be built up programmatically or imported from a standard MIDI file.
- class Sequence

  Node that creates a temporal sequence of child nodes.
- class Shell

  Provide a shell in which Python scripts can be loaded, saved, and executed.
- class Soundfile

  Simple, basic read/write access, in sample frames, to PCM soundfiles.
- class Stack

  The notes produced by each (not all) child node, are rescaled to all start at the same time, and last for the same duration; that of the 0th child, or a specified duration.
- class StrangeAttractor

  Generates notes by searching for a chaotic dynamical system defined by a polynomial equation or partial differential equation using Julien C.
- class System

  Abstraction layer for a minimal set of system services.
- class TempoMap
- class ThreadLock

  Encapsulates a thread monitor, such as a Windows event handle.
- struct TimeAfterComparator
- struct TimeAtComparator
- class Transformer

  Node that uses any callable to implement Node::transform.
- struct Turtle
- class Voicelead

  This class contains facilities for voiceleading, harmonic progression, and identifying chord types.
- class VoiceleadingNode

  This node class imposes a sequence of one or more "voice-leading" operations upon the pitches of notes produced by children of this node, within a segment of the notes.
- class VoiceleadingOperation

  Utility class for storing voice-leading operations within a VoiceleadNode for future application.

Typedefs

- typedef unsigned char csound_u_char
- typedef ImageToScore2 ImageToScore

  Only for backwards compatibility.
- typedef Eigen::Matrix< double, Eigen::Dynamic, Eigen::Dynamic > Matrix
- typedef void(* MessageCallbackType) (CSOUND *csound, int attribute, const char *format, va_list marker)
- typedef Node * NodePtr
- typedef void PyObject_
- typedef Eigen::Matrix< double, Eigen::Dynamic, 1 > Vector

Enumerations

- enum EQUIVALENCE_RELATIONS {
  EQUIVALENCE_RELATION_r = 0 , EQUIVALENCE_RELATION_R , EQUIVALENCE_RELATION_P ,
  EQUIVALENCE_RELATION_T ,
  EQUIVALENCE_RELATION_Tg , EQUIVALENCE_RELATION_I , EQUIVALENCE_RELATION_RP , EQUIVALENCE_RELATION_R
  ,
  EQUIVALENCE_RELATION_RPT , EQUIVALENCE_RELATION_RPTg , EQUIVALENCE_RELATION_RPI ,
  EQUIVALENCE_RELATION_RTI ,
  EQUIVALENCE_RELATION_RTgI , EQUIVALENCE_RELATION_RPTI , EQUIVALENCE_RELATION_RPTgI }

  Enums for all defined equivalence relations, used to specialize template functions.

Functions

- SILENCE_PUBLIC void add_chord (std::string, const Chord &chord)
- SILENCE_PUBLIC void add_scale (std::string, const Scale &scale)
- static void addVoice (Chord &chord)
- SILENCE_PUBLIC std::vector< Chord > allOfEquivalenceClass (int voice_count, std::string equivalence_class, double range, double g, int sector, bool printme)
- SILENCE_PUBLIC void apply (Score &score, const Chord &chord, double startTime, double endTime, bool octaveEquivalence)
- SILENCE_PUBLIC double C4 ()
- SILENCE_PUBLIC Chord chord (const Chord &scale, int scale_degree, int chord_voices, int interval=3)

  Returns the chord, in scale order, for the specified degree of the scale.
- static SILENCE_PUBLIC bool & CHORD_SPACE_DEBUGGING ()

  Returns the current state of the chord space debugging flag as a reference, which can be an lvalue or an rvalue.
- static SILENCE_PUBLIC std::string chord_space_version ()
- SILENCE_PUBLIC const Chord & chordForName (std::string name)
- SILENCE_PUBLIC std::map< std::string, Chord > & chordsForNames ()
- SILENCE_PUBLIC double closestPitch (double pitch, const Chord &chord)

  Returns the pitch in the chord that is closest to the indicated pitch.
- SILENCE_PUBLIC void conformToChord (Event &event, const Chord &chord)
- SILENCE_PUBLIC void conformToChord_equivalence (Event &event, const Chord &chord, bool octaveEquivalence)

  If the Event is a note, moves its pitch to the closest pitch of the chord.
- SILENCE_PUBLIC double conformToPitchClassSet (double pitch, const Chord &pitch_class_set)

  Conforms the pitch to the pitch-class set, but in its original register.
- std::vector< std::vector< MatrixCell > > createMatrix (const std::vector< double > &sourceMultiset_, const std::vector< double > &targetMultiset_, const std::vector< double > &sourceChord_)

- template<typename... Params>
  void defun (const std::string &name, cl_object fun(Params... params))

  Creates a DEFUN abstraction in C++.
- SILENCE_PUBLIC double distance_to_points (const Chord &chord, const std::vector< Chord > &sector_vertices)

  Returns the sum of the distances of the chord to each of the vertices of the indicated sector of a cyclical region.
- SILENCE_PUBLIC double epc (double pitch)

  Returns the equivalent of the pitch under pitch-class equivalence, i.e.
- SILENCE_PUBLIC bool eq_tolerance (double a, double b, int epsilons=20, int ulps=200)

  This is the basis of all other numeric comparisons that take floating-point limits into account.
- template<int EQUIVALENCE_RELATION>
  SILENCE_PUBLIC Chord equate (const Chord &chord)
- template<int EQUIVALENCE_RELATION>
  SILENCE_PUBLIC Chord equate (const Chord &chord, double range)
- template<int EQUIVALENCE_RELATION>
  SILENCE_PUBLIC Chord equate (const Chord &chord, double range, double g, int opt_sector)

  Template function that returns the chord sent to a fundamental domain of specialized equivalence relation, which in some cases may be defined by the indicated range, generator of transposition g, and sector of the cyclical region of OPT fundamental domains.
- template<> SILENCE_PUBLIC Chord equate< EQUIVALENCE_RELATION_I > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC Chord equate< EQUIVALENCE_RELATION_P > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC Chord equate< EQUIVALENCE_RELATION_r > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC Chord equate< EQUIVALENCE_RELATION_R > (const Chord &chord, double range_, double g, int opt_sector)
- template<> SILENCE_PUBLIC Chord equate< EQUIVALENCE_RELATION_RP > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC Chord equate< EQUIVALENCE_RELATION_RPI > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC Chord equate< EQUIVALENCE_RELATION_RPT > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC Chord equate< EQUIVALENCE_RELATION_RPTg > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC Chord equate< EQUIVALENCE_RELATION_RPTgI > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC Chord equate< EQUIVALENCE_RELATION_RPTI > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC Chord equate< EQUIVALENCE_RELATION_T > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC Chord equate< EQUIVALENCE_RELATION_Tg > (const Chord &chord, double range, double g, int opt_sector)
- static int equivalentDegree (const Scale &scale, int degree)
- SILENCE_PUBLIC double euclidean (const csound::Chord &a, const csound::Chord &b)

  Returns the Euclidean distance between the two chords.
- cl_object evaluate_form (const std::string &form)

  Evaluates a SINGLE Lisp form.
- SILENCE_PUBLIC double factorial (double n)
- void fill (std::string rootName, double rootPitch, std::string typeName, std::string typePitches, bool is_scale=false)
- template<int EQUIVALENCE_RELATION>
  SILENCE_PUBLIC std::vector< csound::Chord > fundamentalDomainByPredicate (int voiceN, double range, double g, int sector, bool printme)

Returns a set of chords in sector 0 of the cyclical region, sorted by normal order, for the indicated equivalence relation.

- template<int EQUIVALENCE_RELATION>
  SILENCE_PUBLIC std::vector< Chord > fundamentalDomainByPredicate (int voiceN, double range, double g=1., int sector=0, bool printme=false)

  Returns a set of chords in sector 0 of the cyclical region, sorted by normal order, for the indicated equivalence relation.
- template<int EQUIVALENCE_RELATION>
  SILENCE_PUBLIC std::vector< csound::Chord > fundamentalDomainByTransformation (int voiceN, double range, double g, int sector)

  Returns a set of chords in sector 0 of the cyclical region, sorted by normal order, for the indicated equivalence relation.
- template<int EQUIVALENCE_RELATION>
  SILENCE_PUBLIC std::vector< Chord > fundamentalDomainByTransformation (int voiceN, double range, double g=1., int sector=0)

  Returns a set of chords in sector 0 of the cyclical region, sorted by normal order, for the indicated equivalence relation.
- SILENCE_PUBLIC Chord gather (Score &score, double startTime, double endTime)

  Returns a chord containing all the pitches of the score beginning at or later than the start time, and up to but not including the end time.
- SILENCE_PUBLIC bool ge_tolerance (double a, double b, int epsilons=20, int ulps=200)
- bool getCorrectNegativeDurations ()
- static int getIndex (const std::string &dimension)

  Returns a zero-based numerical index for a string dimension name (for Events) or voice number (for Chords).
- static bool getIndex (int &index, const std::string &dimension)
- SILENCE_PUBLIC bool gt_tolerance (double a, double b, int epsilons=20, int ulps=200)
- SILENCE_PUBLIC HyperplaneEquation hyperplane_equation_from_random_inversion_flat (int dimensions, bool transpositional_equivalence, int opt_sector)
- SILENCE_PUBLIC HyperplaneEquation hyperplane_equation_from_singular_value_decomposition (const std↩
  ::vector< Chord > &points_, bool make_eT)
- SILENCE_PUBLIC double I (double pitch, double center=0.0)

  Returns the pitch reflected in the center, which may be any pitch.
- SILENCE_PUBLIC int indexForOctavewiseRevoicing (const Chord &chord, double range)

  Returns the index of the octavewise revoicing that this chord is, relative to its OP equivalent, within the indicated range.
- SILENCE_PUBLIC int indexForOctavewiseRevoicing (const Chord &origin, const Chord &chord, double range)

  Returns the index of the octavewise revoicing that this chord is, counting up from the origin, within the indicated range.
- void initialize_ecl (int argc, char **argv)

  This function must be called with the arc and argv from main() before any Lisp code is executed.
- void initializeNames ()
- SILENCE_PUBLIC void insert (Score &score, const Chord &chord, double time_)
- SILENCE_PUBLIC void insert (Score &score, const Chord &chord, double time_, bool voice_is_instrument)

  Inserts the notes of the chord into the score at the specified time.
- SILENCE_PUBLIC bool interpolation_point_less (const HarmonyInterpolationPoint &a, const HarmonyInterpolationPoint &b)
- SILENCE_PUBLIC bool interpolation_point_less2 (const HarmonyInterpolationPoint2 &a, const HarmonyInterpolationPoint2 &b)
- SILENCE_PUBLIC std::map< Chord, Chord > & inverse_prime_forms_for_chords ()

  Cache inverse prime forms for chords for speed.
- void inversions (const std::vector< double > &original, const std::vector< double > &iterator, size_t voice, double maximum, std::set< std::vector< double > > &chords, size_t divisionsPerOctave)
- SILENCE_PUBLIC Chord iterator (int voiceN, double first)

  Returns a chord with the specified number of voices all set to a first pitch, useful as an iterator.
- SILENCE_PUBLIC bool le_tolerance (double a, double b, int epsilons=20, int ulps=200)
- SILENCE_PUBLIC FILE *& log_file ()

- SILENCE_PUBLIC bool lt_tolerance (double a, double b, int epsilons=20, int ulps=200)
- static std::vector< double > matchContextSize (const std::vector< double > context, const std::vector< double > pcs)
- static double max (double a, double b)
- static Event mean_to_note (const std::array< double, KMeansMCRM::MEASURE_DIMENSIONS > &mean)
- MessageCallbackType & message_callback ()
- SILENCE_PUBLIC int message_level (int verbosity)
- SILENCE_PUBLIC double MIDDLE_C ()
- SILENCE_PUBLIC Chord midpoint (const Chord &a, const Chord &b)

  Returns the chord that is the midpoint between two chords, which must have the same number of voices.
- static double min (double a, double b)
- const MatrixCell & minimumCell (const MatrixCell &a, const MatrixCell &b, const MatrixCell &c)
- SILENCE_PUBLIC double modulo (double dividend, double divisor)

  Returns the remainder of the dividend divided by the divisor, according to the Euclidean definition.
- SILENCE_PUBLIC std::string nameForChord (const Chord &chord)

  Returns the first valid name for the Chord.
- SILENCE_PUBLIC std::string nameForPitchClass (double pitch)

  Returns the name of the pitch-class of the pitch.
- SILENCE_PUBLIC std::string nameForScale (const Scale &scale)

  Returns the first valid name for the Scale.
- SILENCE_PUBLIC std::vector< std::string > namesForChord (const Chord &chord)

  Returns all enharmonic names for the Chord, if any exists.
- SILENCE_PUBLIC std::multimap< Chord, std::string > & namesForChords ()
- SILENCE_PUBLIC std::vector< std::string > namesForScale (const Scale &scale)

  Returns all enharmonic names for the Scale, if any exists.
- SILENCE_PUBLIC std::multimap< Scale, std::string > & namesForScales ()
- SILENCE_PUBLIC bool next (Chord &iterator_, const Chord &minimum, double range, double g=1.)

  Increment a chord voicewise through chord space, from a low point on the unison diagonal through a high point on the unison diagonal.
- SILENCE_PUBLIC std::map< Chord, Chord > & normal_forms_for_chords ()

  Cache prime forms for chords for speed.
- Event note (const Chord &chord, int voice, double time_, double duration_=DBL_MAX, double channel_=DBL_MAX, double velocity_=DBL_MAX, double pan_=DBL_MAX)

  Creates a complete "note on" Event for the indicated voice of the chord.
- Score notes (const Chord &chord, double time_, double duration_=DBL_MAX, double channel_=DBL_MAX, double velocity_=DBL_MAX, double pan_=DBL_MAX)

  Returns an individual note for each voice of the chord.
- SILENCE_PUBLIC void numerics_information (double a, double b, int epsilons, int ulps)
- SILENCE_PUBLIC double OCTAVE ()

  The size of the octave, defined to be consistent with 12 tone equal temperament and MIDI.
- SILENCE_PUBLIC Chord octavewiseRevoicing (const Chord &chord, int revoicingNumber_, double range)

  Returns the nth octavewise revoicing of the chord that is generated by iterating revoicings within the indicated range.
- SILENCE_PUBLIC int octavewiseRevoicings (const Chord &chord, double range=OCTAVE())

  Returns the full set of octavewise revoicings of the chord within the indicated range.
- SILENCE_PUBLIC bool operator< (const Chord &a, const Chord &b)
- bool operator< (const Event &a, const Event &b)
- bool operator< (const MidiEvent &a, const MidiEvent &b)
- std::ostream & operator<< (std::ostream &stream, const VoiceleadingOperation &operation)
- SILENCE_PUBLIC bool operator<= (const Chord &a, const Chord &b)

- SILENCE_PUBLIC bool operator== (const Chord &a, const Chord &b)
- SILENCE_PUBLIC bool operator> (const Chord &a, const Chord &b)
- SILENCE_PUBLIC bool operator>= (const Chord &a, const Chord &b)
- SILENCE_PUBLIC bool parallelFifth (const Chord &a, const Chord &b)

  Returns whether the voiceleading between chords a and b contains a parallel fifth.
- static void parse_line (std::string line, Score &score)
- static bool parseIndex (int &index, const std::string &target)
- bool parseVector (std::vector< double > &elements, std::string text)
- SILENCE_PUBLIC const std::map< std::string, double > & pitchClassesForNames ()
- SILENCE_PUBLIC double pitchClassForName (std::string name)
- std::vector< std::vector< double > > pitchRotations (const std::vector< double > &chord)
- static void PostProcess (std::map< std::string, std::string > &tags, std::string filename, CsoundThreaded *csound)

  Uses ffmpeg to translate a soundfile to a normalized output file, an MP3 file, a CD audio file, a FLAC soundfile, and an MP4 video file suitable for posting to YouTube.
- template<int EQUIVALENCE_RELATION>
  SILENCE_PUBLIC bool predicate (const Chord &chord)
- template<int EQUIVALENCE_RELATION>
  SILENCE_PUBLIC bool predicate (const Chord &chord, double range)
- template<int EQUIVALENCE_RELATION>
  SILENCE_PUBLIC bool predicate (const Chord &chord, double range, double g, int opt_sector)

  Template function returning whether or not the chord is within the specialized fundamental domain, which may in some cases be defined by the indicated range, generator of transposition g, and sector of the cyclical region of OPT fundamental domains.
- template<int EQUIVALENCE_RELATION>
  SILENCE_PUBLIC bool predicate (const Chord &chord, double range, int sector)
- template<> SILENCE_PUBLIC bool predicate< EQUIVALENCE_RELATION_I > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC bool predicate< EQUIVALENCE_RELATION_P > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC bool predicate< EQUIVALENCE_RELATION_R > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC bool predicate< EQUIVALENCE_RELATION_r > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC bool predicate< EQUIVALENCE_RELATION_RP > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC bool predicate< EQUIVALENCE_RELATION_RPI > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC bool predicate< EQUIVALENCE_RELATION_RPT > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC bool predicate< EQUIVALENCE_RELATION_RPTg > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC bool predicate< EQUIVALENCE_RELATION_RPTgI > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC bool predicate< EQUIVALENCE_RELATION_RPTI > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC bool predicate< EQUIVALENCE_RELATION_T > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC bool predicate< EQUIVALENCE_RELATION_Tg > (const Chord &chord, double range, double g, int opt_sector)
- SILENCE_PUBLIC std::map< Chord, Chord > & prime_forms_for_chords ()

  Cache normal forms for chords for speed.

- SILENCE_PUBLIC const char * print_chord (const Chord &chord)

  Returns a string representation of the pitches in the chord, along with the sectors of the cyclical regions of the OPT and OPTI fundamental domains to which the chord belongs.
- static std::string print_opti_sectors (const Chord &chord)
- void SILENCE_PUBLIC printChord (std::ostream &stream, std::string label, const std::vector< double > &chord)
- void SILENCE_PUBLIC printChord (std::string label, const std::vector< double > &chord)
- static bool pythonFuncWarning (void **pythonLibrary, const char *funcName)
- static double real (const std::string &number)
- void recursiveVoicelead_ (const std::vector< double > &source, const std::vector< double > &original, const std::vector< double > &iterator, std::vector< double > &target, size_t voice, double maximum, bool avoid↩ Parallels, size_t divisionsPerOctave)
- SILENCE_PUBLIC Chord reflect_by_householder (const Chord &chord)

  Computes the Householder reflector matrix and applies it to the chord.
- SILENCE_PUBLIC Chord reflect_in_central_diagonal (const Chord &chord)
- SILENCE_PUBLIC Chord reflect_in_central_point (const Chord &chord)
- SILENCE_PUBLIC Chord reflect_in_inversion_flat (const Chord &chord, int opt_sector)
- SILENCE_PUBLIC Chord reflect_in_unison_diagonal (const Chord &chord)
- SILENCE_PUBLIC Vector reflect_vector (const Vector &point, const Vector &unit_normal_vector, double constant_term)

  Returns the point reflected in the hyperplane defined by the unit normal vector and constant term.
- SILENCE_PUBLIC Vector reflect_vectorx (const Vector &v, const Vector &u, double c)
- static void removeVoice (Chord &chord)
- double round (double x)
- SILENCE_PUBLIC Chord scale (std::string name)

  Returns the named chord as a scale, that is, starting with the chord in OP, and sorting it from the tonic pitch-class on up.
- SILENCE_PUBLIC const Scale & scaleForName (std::string name)
- SILENCE_PUBLIC std::map< std::string, Scale > & scalesForNames ()
- static SILENCE_PUBLIC bool & SCOPED_DEBUGGING_FLAG ()

  Returns the current state of the chord space scoped debugging flag as a reference, which can be an lvalue or an rvalue.
- cl_object scoreToSeq (Score &score, std::string seq_name)

  Translates a Silence Score to Common Music seq.
- void seqToScore (cl_object &seq_, Score &score)

  Translates a Common Music seq to a Silence Score.
- static SILENCE_PUBLIC bool SET_CHORD_SPACE_DEBUGGING (bool enabled)
- static SILENCE_PUBLIC bool SET_SCOPED_DEBUGGING (bool enabled)
- void setCorrectNegativeDurations (bool do_correct)
- SILENCE_PUBLIC std::vector< Event * > slice (Score &score, double startTime, double endTime)

  Returns a slice of the Score starting at the start time and extending up to but not including the end time.
- std::vector< double > sort (const std::vector< double > &chord)
- SILENCE_PUBLIC std::vector< std::string > split (std::string)
- SILENCE_PUBLIC double T (double pitch, double semitones)

  Returns the pitch transposed by semitones, which may be any scalar.
- std::string to_std_string (cl_object lisp_string)

  Translate a Lisp string to a C++ string.
- SILENCE_PUBLIC void toScore (const Chord &chord, Score &score, double time_, bool voiceIsInstrument)
- SILENCE_PUBLIC std::string toString (const Matrix &mat)
- SILENCE_PUBLIC Chord transpose_degrees (const Chord &scale, const Chord &original_chord, int transposition_degrees, int interval=3)

  Returns the chord, in scale order, transposed within the scale by the indicated number of scale degrees, which can be positive or negative.

- SILENCE_PUBLIC std::set< Chord > & unique_chords ()
- SILENCE_PUBLIC std::set< Scale > & unique_scales ()
- SILENCE_PUBLIC void *& user_data ()
- SILENCE_PUBLIC Chord voiceleading (const Chord &a, const Chord &b)

  Returns the voice-leading between chords a and b, i.e.
- SILENCE_PUBLIC Chord voiceleadingCloser (const Chord &source, const Chord &d1, const Chord &d2, bool avoidParallels=false)

  Returns which of the voiceleadings (source to d1, source to d2) is the closer (first smoother, then simpler), optionally avoiding parallel fifths.
- SILENCE_PUBLIC Chord voiceleadingClosestRange (const Chord &source, const Chord &destination, double range, bool avoidParallels)

  Returns the voicing of the destination which has the closest voice-leading from the source within the range, optionally avoiding parallel fifths.
- SILENCE_PUBLIC Chord voiceleadingSimpler (const Chord &source, const Chord &d1, const Chord &d2, bool avoidParallels=false)

  Returns which of the voiceleadings (source to d1, source to d2) is the simpler (fewest moves), optionally avoiding parallel fifths.
- SILENCE_PUBLIC Chord voiceleadingSmoother (const Chord &source, const Chord &d1, const Chord &d2, bool avoidParallels=false, double range=OCTAVE())

  Returns which of the voiceleadings (source to d1, source to d2) is the smoother (shortest moves), optionally avoiding parallel fifths.
- SILENCE_PUBLIC double voiceleadingSmoothness (const Chord &a, const Chord &b)

  Returns the smoothness of the voiceleading between chords a and b by L1 norm.

Variables

- std::map< size_t, std::map< double, double > > cForPForDivisionsPerOctave
- class SILENCE_PUBLIC Chord
- class SILENCE_PUBLIC ChordScore
- static int debug = 1
- static bool initialized__ = Conversions::initialize()
- static std::mt19937 mersenne_twister
- class SILENCE_PUBLIC MidiFile
- static const char * namesForEquivalenceRelations [ ]
- std::map< size_t, std::map< double, double > > pForCForDivisionsPerOctave
- std::map< size_t, std::map< std::vector< double >, double > > pForPrimeChordsForDivisionsPerOctave
- class SILENCE_PUBLIC PITV
- std::map< size_t, std::vector< std::vector< double > > > primeChordsForDivisionsPerOctave
- void(* Py_Finalize_)(void)=0
- void(* Py_Initialize_)(void)=0
- void(* PyErr_Print_)(void)=0
- PyObject_ *(* PyImport_ImportModule_)(char *)=0
- long(* PyLong_AsLong_)(PyObject_ *)=0
- PyObject_ *(* PyObject_CallMethod_)(PyObject_ *, char *, char *,...)=0
- PyObject_ *(* PyObject_GetAttrString_)(PyObject_ *, char *)=0
- int(* PyRun_SimpleFileEx_)(FILE *, const char *, int)=0
- int(* PyRun_SimpleString_)(const char *)=0
- void(* PySys_SetArgv_)(int, char **)=0
- class SILENCE_PUBLIC Scale
- static std::mt19937_64 twister

5.2.1 Detailed Description

C S O U N D.

L I C E N S E

This software is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This software is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this software; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

5.2.2 Typedef Documentation

5.2.2.1 csound_u_char

```
typedef unsigned char csound::csound_u_char
```

5.2.2.2 ImageToScore

```
typedef ImageToScore2 csound::ImageToScore
```

Only for backwards compatibility.

5.2.2.3 Matrix

```
typedef Eigen::Matrix<double, Eigen::Dynamic, Eigen::Dynamic> csound::Matrix
```

5.2.2.4 MessageCallbackType

```
typedef void(* csound::MessageCallbackType) (CSOUND *csound, int attribute, const char *format,
va_list marker)
```

5.2.2.5 NodePtr

```
typedef Node* csound::NodePtr
```

5.2.2.6 PyObject_

```
typedef void csound::PyObject_
```

5.2.2.7 Vector

```
typedef Eigen::Matrix<double, Eigen::Dynamic, 1> csound::Vector
```

5.2.3 Enumeration Type Documentation

5.2.3.1 EQUIVALENCE_RELATIONS

```
enum csound::EQUIVALENCE_RELATIONS
```

Enums for all defined equivalence relations, used to specialize template functions.

If relation R takes no range argument, it defaults to a range of one octave. T is transposition to layer 0, Tg is transposition to the next chord higher than layer 0 in the equal temperament generated by g.

NOTE: Not all of these are currently implemented.

Enumerator

EQUIVALENCE_RELATION_r	
EQUIVALENCE_RELATION_R	
EQUIVALENCE_RELATION_P	
EQUIVALENCE_RELATION_T	
EQUIVALENCE_RELATION_Tg	
EQUIVALENCE_RELATION_I	
EQUIVALENCE_RELATION_RP	
EQUIVALENCE_RELATION_RT	
EQUIVALENCE_RELATION_RPT	
EQUIVALENCE_RELATION_RPTg	
EQUIVALENCE_RELATION_RPI	
EQUIVALENCE_RELATION_RTI	
EQUIVALENCE_RELATION_RTgI	
EQUIVALENCE_RELATION_RPTI	
EQUIVALENCE_RELATION_RPTgI	

5.2.4 Function Documentation

5.2.4.1 add_chord()

```
SILENCE_PUBLIC void csound::add_chord (
```

```
            std::string name,
            const Chord & chord )  [inline]
```

References add_chord(), chord(), chordsForNames(), namesForChords(), and unique_chords().

Referenced by add_chord(), and fill().

5.2.4.2 add_scale()

```
SILENCE_PUBLIC void csound::add_scale (
            std::string name,
            const Scale & scale )  [inline]
```

References add_scale(), namesForScales(), scale(), scalesForNames(), and unique_scales().

Referenced by add_scale(), fill(), csound::Scale::Scale(), and csound::Scale::Scale().

5.2.4.3 addVoice()

```
static void csound::addVoice (
            Chord & chord )  [static]
```

References chord(), csound::Chord::eOP(), csound::Chord::getPitch(), csound::Chord::resize(), csound::Chord::setPitch(), and csound::Chord::voices().

Referenced by csound::ChordLindenmayer::chordOperation(), and csound::ChordLindenmayer::modalityOperation().

5.2.4.4 allOfEquivalenceClass()

```
SILENCE_PUBLIC std::vector< Chord > csound::allOfEquivalenceClass (
            int voice_count,
            std::string equivalence_class,
            double range,
            double g,
            int sector,
            bool printme )
```

Referenced by main().

5.2.4.5 apply()

```
SILENCE_PUBLIC void csound::apply (
            Score & score,
            const Chord & chord,
            double startTime,
            double endTime,
            bool octaveEquivalence )
```

References apply(), chord(), conformToChord_equivalence(), and slice().

Referenced by apply().

5.2.4.6 C4()

```
SILENCE_PUBLIC double csound::C4 ( )  [inline]
```

References C4(), and MIDDLE_C().

Referenced by C4().

5.2.4.7 chord()

```
SILENCE_PUBLIC Chord csound::chord (
            const Chord & scale,
            int scale_degree,
            int chord_voices,
            int interval = 3 )  [inline]
```

Returns the chord, in scale order, for the specified degree of the scale.

The chord can be composed of seconds, thirds, or larger intervals, and can have two or more voices. The scale can have any number of pitch-classes
and any interval content; it simply has to consists of pitch-classes sorted from the tonic pitch-class on up.

PLEASE NOTE: Scale degree is 1-based. A "third" is denoted "3" but is two scale degrees, and so on.

References chord(), csound::Chord::getPitch(), OCTAVE(), csound::Chord::resize(), scale(), csound::Chord::setPitch(), and csound::Chord::voices().

Referenced by csound::Turtle::__str__(), add_chord(), csound::HarmonyIFS::add_interpolation_point_as_chord(), csound::HarmonyIFS2::add_interpolation_point_as_chord(), addVoice(), apply(), chord(), csound::Scale::chord(), chordForName(), closestPitch(), conformToChord(), conformToChord_equivalence(), csound::Scale::degree(), distance_to_points(), csound::Chord::epcs(), csound::Chord::eppcs(), equate(), equate(), equate< EQUIVALENCE_RELATION_I >(), equate< EQUIVALENCE_RELATION_P >(), equate< EQUIVALENCE_RELATION_r >(), equate< EQUIVALENCE_RELATION_R >(), equate< EQUIVALENCE_RELATION_RP >(), equate< EQUIVALENCE_RELATION_RPI >(), equate< EQUIVALENCE_RELATION_RP equate< EQUIVALENCE_RELATION_RPTg >(), equate< EQUIVALENCE_RELATION_RPTgI >(), equate< EQUIVALENCE_RELATIO equate< EQUIVALENCE_RELATION_T >(), equate< EQUIVALENCE_RELATION_Tg >(), fill(), csound::PITV::fromChord(), gather(), csound::HarmonyEvent::get_chord(), csound::Score::getPitches(), csound::Score::getPT(), csound::Score::getPTV(), hyperplane_equation_from_random_inversion_flat(), indexForOctavewiseRevoicing(), indexForOctavewiseRevoicing(), csound::Turtle::initialize(), csound::Voicelead::initializePrimeChordsForDivisionsPerOctave(), insert(), insert(), csound::ChordScore::insertChord(), csound::Voicelead::inversions(), csound::Voicelead::invert(), csound::HarmonyIFS2::iterate(), csound::Chord::K(), csound::Chord::K_range(), csound::PITV::list(), csound::Scale::modulations(), csound::Scale::modulations_for_scale_ csound::Scale::modulations_for_voices(), csound::Chord::move(), nameForChord(), namesForChord(), csound::Voicelead::normalChord() note(), notes(), octavewiseRevoicing(), octavewiseRevoicings(), csound::Turtle::operator<(), csound::Turtle::operator=(), csound::Voicelead::orderedPcs(), csound::Voicelead::pcs(), csound::Voicelead::pitchClassSetToM(), pitchRotations(), predicate(), predicate(), predicate(), predicate< EQUIVALENCE_RELATION_I >(), predicate< EQUIVALENCE_RELATION_P >(), predicate< EQUIVALENCE_RELATION_R >(), predicate< EQUIVALENCE_RELATION_r >(), predicate< EQUIVALENCE_RELATION_ predicate< EQUIVALENCE_RELATION_RPI >(), predicate< EQUIVALENCE_RELATION_RPT >(), predicate< EQUIVALENCE_RELA predicate< EQUIVALENCE_RELATION_RPTgI >(), predicate< EQUIVALENCE_RELATION_RPTI >(), predicate< EQUIVALENCE_RE predicate< EQUIVALENCE_RELATION_Tg >(), csound::PITV::preinitialize(), csound::Voicelead::primeChord(), print_chord(), print_opti_sectors(), printChord(), printChord(), reflect_by_householder(), reflect_in_central_diagonal(), reflect_in_central_point(), reflect_in_inversion_flat(), reflect_in_unison_diagonal(), csound::Scale::relative_tonicizations_for_scale_types() removeVoice(), csound::Voicelead::rotate(), csound::Voicelead::rotations(), csound::HarmonyEvent::set_chord(), csound::ChordScore::setDuration(), csound::ChordScore::setScale(), sort(), csound::Voicelead::sortByAscendingDistance(), csound::Scale::tonicizations(), csound::Voicelead::toOrigin(), toScore(), csound::Voicelead::transpose(), csound::Scale::transpose_degree transpose_degrees(), csound::Voicelead::uniquePcs(), csound::Chord::v(), csound::Chord::voicings(), and csound::Voicelead::wrap().

5.2.4.8 CHORD_SPACE_DEBUGGING()

```
static SILENCE_PUBLIC bool & csound::CHORD_SPACE_DEBUGGING ( )  [static]
```

Returns the current state of the chord space debugging flag as a reference, which can be an lvalue or an rvalue.

References CHORD_SPACE_DEBUGGING().

Referenced by CHORD_SPACE_DEBUGGING(), fundamentalDomainByPredicate(), fundamentalDomainByTransformation(), csound::SCOPED_DEBUGGING::SCOPED_DEBUGGING(), SET_CHORD_SPACE_DEBUGGING(), and csound::SCOPED_DEBUGGIN

5.2.4.9 chord_space_version()

```
static SILENCE_PUBLIC std::string csound::chord_space_version ( )  [static]
```

Referenced by main().

5.2.4.10 chordForName()

```
SILENCE_PUBLIC const Chord & csound::chordForName (
            std::string name )  [inline]
```

References chord(), chordForName(), chordsForNames(), initializeNames(), and csound::Chord::resize().

Referenced by chordForName(), csound::Turtle::initialize(), main(), main(), scale(), test_nrL(), test_nrP(), test_nrR(), and test_pitv().

5.2.4.11 chordsForNames()

```
SILENCE_PUBLIC std::map< std::string, Chord > & csound::chordsForNames ( )  [inline]
```

References chordsForNames().

Referenced by add_chord(), chordForName(), and chordsForNames().

5.2.4.12 closestPitch()

```
SILENCE_PUBLIC double csound::closestPitch (
            double pitch,
            const Chord & chord )  [inline]
```

Returns the pitch in the chord that is closest to the indicated pitch.

References chord(), closestPitch(), csound::Chord::getPitch(), and csound::Chord::voices().

Referenced by closestPitch(), conformToChord_equivalence(), and conformToPitchClassSet().

5.2.4.13 conformToChord()

```
SILENCE_PUBLIC void csound::conformToChord (
            Event & event,
            const Chord & chord )
```

References chord(), conformToChord(), and conformToChord_equivalence().

Referenced by conformToChord(), csound::CellChord::transform(), and csound::HarmonyIFS::translate_score_attractor_to_score().

5.2.4.14 conformToChord_equivalence()

```
SILENCE_PUBLIC void csound::conformToChord_equivalence (
            Event & event,
            const Chord & chord,
            bool octaveEquivalence )
```

If the Event is a note, moves its pitch to the closest pitch of the chord.

If octaveEquivalence is true (the default), the pitch-class of the note is moved to the closest pitch-class of the chord, i.e. keeping the note more or less in its original register; otherwise, the pitch of the note is moved to the closest absolute pitch of the chord.

References chord(), closestPitch(), conformToChord_equivalence(), conformToPitchClassSet(), csound::Chord::epcs(), and csound::Event::isNoteOn().

Referenced by apply(), conformToChord(), conformToChord_equivalence(), and csound::ChordScore::conformToChords().

5.2.4.15 conformToPitchClassSet()

```
SILENCE_PUBLIC double csound::conformToPitchClassSet (
            double pitch,
            const Chord & pitch_class_set )  [inline]
```

Conforms the pitch to the pitch-class set, but in its original register.

References closestPitch(), conformToPitchClassSet(), epc(), and OCTAVE().

Referenced by conformToChord_equivalence(), conformToPitchClassSet(), and main().

5.2.4.16 createMatrix()

```
std::vector< std::vector< MatrixCell > > csound::createMatrix (
            const std::vector< double > & sourceMultiset_,
            const std::vector< double > & targetMultiset_,
            const std::vector< double > & sourceChord_ )
```

References fundamentalDomainByPredicate(), minimumCell(), csound::Voicelead::smoothness(), and csound::Voicelead::voiceleading().

Referenced by csound::Voicelead::nonBijectiveVoicelead().

5.2.4.17 defun()

```
template<typename...  Params>
void csound::defun (
             const std::string & name,
             cl_object  funParams...  params )
```

Creates a DEFUN abstraction in C++.

References defun().

Referenced by defun().

5.2.4.18 distance_to_points()

```
SILENCE_PUBLIC double csound::distance_to_points (
             const Chord & chord,
             const std::vector< Chord > & points )  [inline]
```

Returns the sum of the distances of the chord to each of the vertices of the indicated sector of a cyclical region.

Returns the sum of the distances of the chord to each of one or more chords.

References chord(), distance_to_points(), and euclidean().

Referenced by distance_to_points(), and csound::Chord::opti_domain_sectors().

5.2.4.19 epc()

```
SILENCE_PUBLIC double csound::epc (
             double pitch )  [inline]
```

Returns the equivalent of the pitch under pitch-class equivalence, i.e.

the pitch is in the interval [0, OCTAVE). Implemented using the Euclidean definition.

References epc(), modulo(), and OCTAVE().

Referenced by conformToPitchClassSet(), epc(), csound::Chord::epcs(), csound::Chord::eppcs(), csound::Chord::isepcs(), csound::Chord::K(), main(), and nameForPitchClass().

5.2.4.20 eq_tolerance()

```
SILENCE_PUBLIC bool csound::eq_tolerance (
            double a,
            double b,
            int epsilons = 20,
            int ulps = 200 )  [inline]
```

This is the basis of all other numeric comparisons that take floating-point limits into account.

It is a "close enough" comparison. If a or b equals 0, the indicated number of machine epsilons is used as the tolerance; if neither a nor b equals 0, the indicated number of units in the last place (ULPs) is used as the tolerance. These tolerances should be set to appropriate values based on the use case.

References CHORD_SPACE_DEBUG, and eq_tolerance().

Referenced by csound::Chord::contains(), csound::Chord::count(), eq_tolerance(), equals(), ge_tolerance(), gt_tolerance(), csound::Chord::isepcs(), le_tolerance(), lt_tolerance(), nameForPitchClass(), numerics_information(), operator==(), predicate< EQUIVALENCE_RELATION_T >(), predicate< EQUIVALENCE_RELATION_Tg >(), scale(), test_eq_tolerance(), and csound::Intercut::traverse().

5.2.4.21 equate() [1/3]

```
template<int EQUIVALENCE_RELATION>
SILENCE_PUBLIC Chord csound::equate (
            const Chord & chord )  [inline]
```

References chord(), equate(), and OCTAVE().

5.2.4.22 equate() [2/3]

```
template<int EQUIVALENCE_RELATION>
SILENCE_PUBLIC Chord csound::equate (
            const Chord & chord,
            double range )  [inline]
```

References chord(), and equate().

5.2.4.23 equate() [3/3]

```
template<int EQUIVALENCE_RELATION>
SILENCE_PUBLIC Chord csound::equate (
            const Chord & chord,
            double range,
            double g,
            int opt_sector )
```

Template function that returns the chord sent to a fundamental domain of specialized equivalence relation, which in some cases may be defined by the indicated range, generator of transposition g, and sector of the cyclical region of OPT fundamental domains.

References equate(), OCTAVE(), octavewiseRevoicing(), octavewiseRevoicings(), parallelFifth(), and pitchClassForName().

Referenced by equate(), equate(), and equate().

5.2.4.24 equate< EQUIVALENCE_RELATION_I >()

```
template<>
SILENCE_PUBLIC Chord csound::equate< EQUIVALENCE_RELATION_I > (
            const Chord & chord,
            double range,
            double g,
            int opt_sector )  [inline]
```

References chord(), and reflect_in_inversion_flat().

Referenced by csound::Chord::eI().

5.2.4.25 equate< EQUIVALENCE_RELATION_P >()

```
template<>
SILENCE_PUBLIC Chord csound::equate< EQUIVALENCE_RELATION_P > (
            const Chord & chord,
            double range,
            double g,
            int opt_sector )  [inline]
```

References chord(), csound::Chord::getPitch(), gt_tolerance(), and csound::Chord::voices().

Referenced by csound::Chord::eP().

5.2.4.26 equate< EQUIVALENCE_RELATION_r >()

```
template<>
SILENCE_PUBLIC Chord csound::equate< EQUIVALENCE_RELATION_r > (
            const Chord & chord,
            double range,
            double g,
            int opt_sector )  [inline]
```

References chord(), csound::Chord::getPitch(), modulo(), csound::Chord::setPitch(), and csound::Chord::voices().

5.2.4.27 equate< EQUIVALENCE_RELATION_R >()

```
template<>
SILENCE_PUBLIC Chord csound::equate< EQUIVALENCE_RELATION_R > (
            const Chord & chord,
            double range_,
            double g,
            int opt_sector )  [inline]
```

References chord(), CHORD_SPACE_DEBUG, csound::Chord::iseR(), csound::Chord::layer(), le_tolerance(), csound::Chord::max(), csound::Chord::setPitch(), and csound::Chord::toString().

Referenced by csound::Chord::eR().

5.2.4.28 equate< EQUIVALENCE_RELATION_RP >()

```
template<>
SILENCE_PUBLIC Chord csound::equate< EQUIVALENCE_RELATION_RP > (
            const Chord & chord,
            double range,
            double g,
            int opt_sector )  [inline]
```

References chord().

Referenced by csound::Chord::eRP(), indexForOctavewiseRevoicing(), and octavewiseRevoicing().

5.2.4.29 equate< EQUIVALENCE_RELATION_RPI >()

```
template<>
SILENCE_PUBLIC Chord csound::equate< EQUIVALENCE_RELATION_RPI > (
            const Chord & chord,
            double range,
            double g,
            int opt_sector )  [inline]
```

References chord(), csound::Chord::eI(), and csound::Chord::eRP().

Referenced by csound::Chord::eRPI().

5.2.4.30 equate< EQUIVALENCE_RELATION_RPT >()

```
template<>
SILENCE_PUBLIC Chord csound::equate< EQUIVALENCE_RELATION_RPT > (
            const Chord & chord,
            double range,
            double g,
            int opt_sector )  [inline]
```

CHORD_SPACE_DEBUGGING() = true; std::raise(SIGINT);

References chord(), csound::Chord::eRPTs(), and print_chord().

Referenced by csound::Chord::eRPT().

5.2.4.31 equate< EQUIVALENCE_RELATION_RPTg >()

```
template<>
SILENCE_PUBLIC Chord csound::equate< EQUIVALENCE_RELATION_RPTg > (
            const Chord & chord,
            double range,
            double g,
            int opt_sector )  [inline]
```

CHORD_SPACE_DEBUGGING() = true; std::raise(SIGINT);

References chord(), csound::Chord::eRPTTs(), and print_chord().

Referenced by csound::Chord::eRPTT().

5.2.4.32 equate< EQUIVALENCE_RELATION_RPTgI >()

```
template<>
SILENCE_PUBLIC Chord csound::equate< EQUIVALENCE_RELATION_RPTgI > (
            const Chord & chord,
            double range,
            double g,
            int opt_sector )  [inline]
```

References chord().

Referenced by csound::Chord::eRPTTI().

5.2.4.33 equate< EQUIVALENCE_RELATION_RPTI >()

```
template<>
SILENCE_PUBLIC Chord csound::equate< EQUIVALENCE_RELATION_RPTI > (
            const Chord & chord,
            double range,
            double g,
            int opt_sector )  [inline]
```

References chord().

Referenced by csound::Chord::eRPTI().

5.2.4.34 equate< EQUIVALENCE_RELATION_T >()

```
template<>
SILENCE_PUBLIC Chord csound::equate< EQUIVALENCE_RELATION_T > (
            const Chord & chord,
            double range,
            double g,
            int opt_sector )  [inline]
```

References chord(), csound::Chord::layer(), csound::Chord::T(), and csound::Chord::voices().

Referenced by csound::Chord::eT().

5.2.4.35 equate< EQUIVALENCE_RELATION_Tg >()

```
template<>
SILENCE_PUBLIC Chord csound::equate< EQUIVALENCE_RELATION_Tg > (
            const Chord & chord,
            double range,
            double g,
            int opt_sector )  [inline]
```

References csound::Chord::ceiling(), chord(), csound::Chord::eT(), lt_tolerance(), and csound::Chord::T().

Referenced by csound::Chord::eTT().

5.2.4.36 equivalentDegree()

```
static int csound::equivalentDegree (
            const Scale & scale,
            int degree )  [static]
```

References scale(), and csound::Chord::voices().

Referenced by csound::ChordLindenmayer::scaleDegreeOperation(), and csound::ChordLindenmayer::scaleOperation().

5.2.4.37 euclidean()

```
SILENCE_PUBLIC double csound::euclidean (
            const csound::Chord & a,
            const csound::Chord & b )  [inline]
```

Returns the Euclidean distance between the two chords.

References euclidean(), csound::Chord::getPitch(), and csound::Chord::voices().

Referenced by distance_to_points(), csound::Chord::distanceToOrigin(), csound::Chord::distanceToUnisonDiagonal(), and euclidean().

5.2.4.38 evaluate_form()

```
cl_object csound::evaluate_form (
            const std::string & form )
```

Evaluates a *SINGLE* Lisp form.

Please note, in Embeddable Common Lisp, `(require :xxx)` and some other forms work only if they are at the top level. That may necessitate repeated calls to this function from the embedding system.

References fundamentalDomainByPredicate().

Referenced by csound::LispGenerator::generate(), csound::LispNode::getNumberFromForm(), csound::LispNode::getStringFromForm(), main(), scoreToSeq(), and csound::LispTransformer::transform().

5.2.4.39 factorial()

```
SILENCE_PUBLIC double csound::factorial (
            double n )  [inline]
```

References factorial().

Referenced by factorial().

5.2.4.40 fill()

```
void csound::fill (
            std::string rootName,
            double rootPitch,
            std::string typeName,
            std::string typePitches,
            bool is_scale = false )  [inline]
```

chordsForNames()[chordName] = eOP_; namesForChords()[eOP_] = chordName;

scalesForNames()[chordName] = scale; namesForScales()[scale] = chordName;

chordsForNames()[chordName] = eOP_; namesForChords()[eOP_] = chordName;

scalesForNames()[chordName] = scale; namesForScales()[scale] = chordName;

References add_chord(), add_scale(), chord(), CHORD_SPACE_DEBUG, csound::Chord::eOP(), fill(), pitchClassForName(), csound::Chord::resize(), scale(), csound::Chord::setPitch(), split(), csound::Chord::T(), and csound::Chord::toString().

Referenced by fill(), initializeNames(), and main().

5.2.4.41 fundamentalDomainByPredicate() [1/2]

```
template<int EQUIVALENCE_RELATION>
SILENCE_PUBLIC std::vector< csound::Chord > csound::fundamentalDomainByPredicate (
            int voiceN,
            double range,
            double g = 1.,
            int sector = 0,
            bool printme = false )  [inline]
```

Returns a set of chords in sector 0 of the cyclical region, sorted by normal order, for the indicated equivalence relation.

If there are duplicate chords for the same equivalence, only the one closest to the origin is returned. SCOPED_DEBUGGING debugging;

References CHORD_SPACE_DEBUG, CHORD_SPACE_DEBUGGING(), fundamentalDomainByPredicate(), iterator(), namesForEquivalenceRelations, next(), print_chord(), and csound::Chord::toString().

Referenced by csound::Score::add(), csound::Node::addChild(), csound::Voicelead::addOctave(), csound::Lindenmayer::addRule(), csound::Conversions::amplitudeToDecibels(), csound::Conversions::amplitudeToGain(), csound::Conversions::amplitudeToMidi(), csound::Score::append(), csound::Score::append(), csound::Score::append_event(), csound::Score::append_note(), csound::Score::appendToCsoundScoreHeader(), csound::LispNode::appendTopLevelForm(), csound::VoiceleadingNode::apply(), csound::Voicelead::areParallel(), csound::MusicModel::arrange(), csound::Score::arrange(), csound::MusicModel::arrange(), csound::Score::arrange(), csound::MusicModel::arrange(), csound::Score::arrange(), csound::MusicModel::arrange(), csound::MusicModel::arrange(), csound::MusicModel::arrange(), csound::Score::arrange_all(), csound::AscendingDistanceComparator::a csound::Soundfile::blank(), csound::StrangeAttractor::calculateFractalDimension(), csound::StrangeAttractor::calculateLyupanovExponen csound::Voicelead::chordToPTV(), csound::VoiceleadingNode::chordVoiceleading(), csound::Chunk::Chunk(), csound::Node::clear(), csound::Soundfile::close(), csound::System::closeLibrary(), csound::Voicelead::closer(), csound::Voicelead::closest(), csound::Voicelead::closestPitch(), csound::StrangeAttractor::codeRandomize(), csound::MidiFile::computeTimes(), csound::ImageToScore2::condense(), csound::Event::conformToPitchClassSet(), csound::Voicelead::conformToPitchClassSet(),

csound::ImageToScore2::contrast(), csound::Event::correct_negative_duration(), csound::Event::correct_negative_durations(), csound::Soundfile::cosineGrain(), csound::Soundfile::create(), csound::MusicModel::createCsoundScore(), createMatrix(), csound::Event::createNoteOffEvent(), csound::Lindenmayer::createRotation(), csound::Node::createTransform(), csound::MusicModel::csoundArgv(), csound::Voicelead::cToM(), csound::Voicelead::cToP(), csound::System::debug(), csound::System::debug(), csound::Conversions::decibelsToAmplitude(), csound::KMeansMCRM::deterministic_algorithm(), csound::ImageToScore2::dilate(), csound::Conversions::doubleToString(), csound::Event::dump(), csound::MidiFile::dump(), csound::Score::dump(), csound::Conversions::dupstr(), csound::ImageToScore2::erode(), csound::Soundfile::error(), csound::System::error(), csound::System::error(), csound::Voicelead::euclideanDistance(), evaluate_form(), csound::Event::Event(), csound::Event::Event(), csound::Conversions::findClosestPitchClass(), csound::Score::findScale(), fundamentalDomainByPredicate(), csound::Conversions::gainToAmplitude(), csound::Conversions::gainToDb(), csound::ImageToScore2::gaussianBlur(), csound::MusicModel::generate(), csound::ExternalNode::generate(), csound::ScoreNode::generate(), csound::Random::generate(), csound::LispGenerator::generate(), csound::Composition::generateAllNames(), csound::Composition::generateFilename(), csound::Shell::generateFilename(), csound::CMaskNode::generateLocally(), csound::ExternalNode::generateLocally(), csound::ImageToScore2::generateLocally(), csound::Lindenmayer::generateLocally(), csound::MCRM::generateLocally(), csound::KMeansMCRM::generateLocally(), csound::Score::getBlueScore(), csound::Event::getChannel(), csound::MusicModel::getCsour
csound::Score::getCsoundScore(), csound::TempoMap::getCurrentSecondsPerTick(), csound::StrangeAttractor::getDimensionAndOrder(
csound::Score::getDuration(), csound::Score::getDurationFromZero(), csound::CsoundProducer::GetFilenameBase(), csound::CsoundProducer::GetGitCommitHash(), csound::Event::getKey_tempered(), csound::Event::getKeyNumber(), csound::Conversions::getMaximumAmplitude(), csound::Conversions::getMaximumDynamicRange(), csound::CsoundProducer::GetMeta
csound::MidiEvent::getMetaSize(), csound::Shell::getMidiFilename(), csound::Event::getMidiStatus(), csound::StrangeAttractor::getNorma
csound::StrangeAttractor::getNormalizedX(), csound::StrangeAttractor::getNormalizedY(), csound::StrangeAttractor::getNormalizedZ(), csound::LispNode::getNumberFromForm(), csound::Composition::getOutputDirectory(), csound::Shell::getOutputSoundfileName(), csound::Score::getPitches(), csound::Event::getProperties(), csound::Score::getPT(), csound::Score::getPTV(), csound::Random::getRandomCoordinates(), csound::Lindenmayer::getReplacement(), csound::Rescale::getRescale(), csound::Score::getScale(), csound::Event::getStatusNumber(), csound::LispNode::getStringFromForm(), csound::System::getSymbol(), csound::MusicModel::getThis(), csound::ScoreModel::getThis(), csound::Event::getVelocityNumber(), csound::Score::getVoicing(), csound::CsoundProducer::GitCommit(), csound::Conversions::hzToMidi(), csound::Conversions::hzToOctave(), csound::Conversions::hzToSamplingIncrement(), csound::Voicelead::I(), csound::Voicelead::I_vector(), csound::Voicelead::Iform(), csound::Score::indexAfterTime(), csound::Score::indexAtTime(), csound::Score::indexToTime(), csound::System::inform(), csound::System::inform(), csound::Conversions::initialize(), csound::Event::initialize(), initialize_ecl(), csound::Voicelead::initializePrimeCh
csound::Lindenmayer::interpret(), csound::Conversions::intToString(), csound::Voicelead::inversions(), inversions(), csound::Voicelead::invert(), csound::Event::isMatchingEvent(), csound::Event::isMatchingNoteOff(), csound::MCRM::iterate(), csound::KMeansMCRM::iterate(), csound::CsoundProducer::Join(), csound::Soundfile::jonesParksGrain(), csound::Voicelead::K(), csound::Conversions::leftPan(), csound::Score::load(), csound::MidiFile::load(), csound::Score::load(), csound::Shell::loadAppend(), log_file(), main(), csound::Shell::main(), csound::Composition::makeTimestamp(), csound::Chunk::markChunkEnd(), csound::Chunk::markChunkSize(), csound::Chunk::markChunkStart(), matchContextSize(), csound::MidiEvent::matchesNoteOffEvent(), mean_to_note(), csound::KMeansMCRM::means_to_notes(), csound::System::message(), csound::System::message(), csound::System::message(), csound::System::message(), csound::System::message(), csound::System::message(), message_callback(), message_level(), csound::Conversions::midiToAmplitude(), csound::Conversions::midiToDecibels(), csound::Conversions::midiToGain(), csound::Conversions::midiToHz(), csound::Conversions::midiToOctave(), csound::Conversions::midiToPitchClass(), csound::Conversions::midiToPitchClassSet(), csound::Conversions::midiToRoundedOctave(), csound::Conversions::midiToSamplingIncrement(), csound::Soundfile::mixFrames(), csound::Voicelead::mToC(), csound::Voicelead::mToPitchClassSet(), csound::Voicelead::nameToC(), csound::Conversions::nameToPitche
csound::Voicelead::nonBijectiveVoicelead(), csound::Voicelead::normalChord(), csound::Composition::normalizeOutputSoundfile(), csound::Conversions::octaveToHz(), csound::Conversions::octaveToMidi(), csound::Conversions::octaveToSamplingIncrement(), csound::Shell::open(), csound::Soundfile::open(), csound::System::openLibrary(), csound::TimeAtComparator::operator()(), csound::TimeAfterComparator::operator()(), csound::AscendingDistanceComparator::operator()(), operator<<(), csound::MidiTrack::operator=(), csound::Voicelead::orderedPcs(), csound::Voicelead::pAndTtoPitchClassSet(), parse_line(), csound::Voicelead::pc(), csound::Voicelead::pcs(), csound::MusicModel::perform(), csound::Composition::performAll(), csound::CsoundProducer::PerformAndPostProcess(), csound::CsoundProducer::PerformAndPostProcessRoutine(), csound::Composition::performMaster(), csound::Conversions::phaseToTableLengths(), csound::Voicelead::pitchClassSetToM(), csound::Conversions::pitchClassSetToMidi(), csound::Voicelead::pitchClassSetToPandT(), csound::Conversions::pitchClassToMidi(), pitchRotations(), csound::ImageToScore2::pixel_to_event(), PostProcess(), printChord(), printChord(), csound::Score::process(), csound::MusicModel::processArgs(), csound::Composition::processArgv(), csound::ImageToScore2::processImage(),

csound::Voicelead::pToC(), csound::Voicelead::pToPrimeChord(), csound::Voicelead::ptvToChord(), pythonFuncWarning(), csound::Voicelead::Q(), csound::KMeansMCRM::random_algorithm(), csound::Chunk::read(), csound::MidiHeader::read(), csound::MidiFile::read(), csound::MidiEvent::readByte(), csound::Soundfile::readFrame(), csound::Soundfile::readFrames(), csound::MidiEvent::readIn(), csound::MidiTrack::readIn(), csound::MidiFile::readInt(), csound::MidiFile::readShort(), csound::MidiFile::readVariableLength(), csound::Voicelead::recursiveVoicelead(), recursiveVoicelead_(), csound::Score::remove(), csound::Composition::render(), csound::MusicModel::render(), csound::StrangeAttractor::render(), csound::Composition::renderAll(), csound::Score::rescale(), csound::Score::rescale(), csound::Score::rescale_event(), csound::MCRM::resize(), csound::Lindenmayer::rewrite(), csound::Conversions::rightPan(), csound::Voicelead::rotate(), csound::Voicelead::rotations(), csound::Shell::runScript(), csound::Score::save(), csound::MidiFile::save(), csound::Score::save(), csound::Shell::save(), scoreToSeq(), csound::Soundfile::seek(), csound::Soundfile::seekSeconds(), seqToScore(), csound::Event::set(), csound::Event::setAmplitude(), csound::StrangeAttractor::setAttractorType(), csound::Event::setChannel(), csound::Soundfile::setChannel csound::ExternalNode::setCommand(), setCorrectNegativeDurations(), csound::Score::setCsoundScoreHeader(), csound::MusicModel::setCsoundScoreHeader(), csound::CsoundProducer::SetDoGitCommit(), csound::Composition::setDuration(), csound::Score::setDuration(), csound::Score::setDurationFromZero(), csound::Soundfile::setFormat(), csound::Soundfile::setFramesPerS csound::Event::setHeight(), csound::ImageToScore2::setImageFilename(), csound::StrangeAttractor::setIteration(), csound::StrangeAttractor::setIterationCount(), csound::Score::setK(), csound::Score::setKL(), csound::Score::setKV(), csound::System::setLogfile(), csound::ImageToScore2::setMaximumVoiceCount(), csound::System::setMessageCallback(), csound::System::setMessageLevel(), csound::CsoundProducer::SetMetadata(), csound::Event::setMidi(), csound::Event::setOffTime(), csound::CsoundProducer::SetOutput(), csound::Composition::setOutputDirectory(), csound::CMaskNode::setParametersText(), csound::Score::setPitchClassSet(), csound::Event::setPitches(), csound::Score::setPitches(), csound::Score::setPT(), csound::Score::setPTV(), csound::Score::setQ(), csound::Score::setQL(), csound::Score::setQV(), csound::Rescale::setRescale(), csound::Score::setScale(), csound::Composition::setScore(), csound::StrangeAttractor::setScoreType(), csound::ExternalNode::setScript(csound::Composition::setTieOverlappingNotes(), csound::System::setUserdata(), csound::Score::setVoicing(), csound::StrangeAttractor::setW(), csound::MCRM::setWeight(), csound::StrangeAttractor::setX(), csound::StrangeAttractor::setY(), csound::StrangeAttractor::setZ(), csound::ImageToScore2::sharpen(), csound::StrangeAttractor::shuffleRandomNumbers(), csound::Voicelead::simpler(), csound::Voicelead::smoothness(), csound::Score::sort(), sort(), csound::Voicelead::sortByAscendingDistanc csound::StrangeAttractor::specialFunctions(), csound::CsoundProducer::Start(), csound::CsoundProducer::startTiming(), csound::System::startTiming(), csound::ThreadLock::startWait(), csound::CsoundProducer::stopTiming(), csound::System::stopTiming(), csound::Conversions::stringToBool(), csound::Conversions::stringToDouble(), csound::Conversions::stringToInt(), csound::Conversions::stringToVector(), csound::Conversions::swapInt(), csound::Conversions::swapShort(), csound::Voicelead::T(), csound::Voicelead::T_vector(), csound::Composition::tagFile(), csound::Score::temper(), csound::Voicelead::Tform(), csound::ImageToScore2::threshhold(), csound::Score::tieOverlappingNotes(), to_std_string(), csound::Event::toBlueIStatement(), csound::Event::toCsoundIStatement(), csound::Event::toCsoundIStatementHeld(), csound::Event::toCsoundIStatementRelease(), csound::MidiFile::toInt(), csound::Score::toJson(), csound::Voicelead::toOrigin(), csound::MidiFile::toShort(), csound::Score::toString(), csound::Event::toString(), csound::MidiEvent::toString(), csound::Score::transform(), csound::CounterpointNode::transform(), csound::RemoveDuplicates::transform(), csound::Random::transform(), csound::Rescale::transform(), csound::VoiceleadingNode::transfo csound::LispTransformer::transform(), csound::CMaskNode::translate_to_silence(), csound::Composition::translateMaster(), csound::Composition::translateToCdAudio(), csound::Composition::translateToMp3(), csound::Composition::translateToMp4(), csound::Composition::translateToNotation(), csound::Voicelead::transpose(), csound::Node::traverse(), csound::Sequence::traverse(), csound::Voicelead::uniquePcs(), csound::Lindenmayer::updateActual(), csound::Soundfile::updateHeader(), user_data(), csound::Voicelead::voicelead(), csound::Score::voicelead(), csound::Score::voicelead(), csound::Score::voicelead_pitches(), csound::Score::voicelead_segments(), csound::Voicelead::voiceleading(), csound::Voicelead::voicings(), csound::System::warn(), csound::System::warn(), csound::Voicelead::wrap(), csound::Composition::write(), csound::Logger::write(), csound::Chunk::write(), csound::MidiHeader::write(), csound::MidiFile::write(), csound::ImageToScore2::write_processed_file(), csound::Soundfile::writeFrame(), csound::Soundfile::writeFrames(), csound::MidiFile::writeInt(), csound::MidiEvent::writeOut(), csound::MidiTrack::writeOut(), csound::MidiFile::writeShort(), and csound::MidiFile::writeVariableLength().

5.2.4.42 fundamentalDomainByPredicate() [2/2]

```
template<int EQUIVALENCE_RELATION>
SILENCE_PUBLIC std::vector< Chord > csound::fundamentalDomainByPredicate (
            int voiceN,
```

```
double range,
double g = 1.,
int sector = 0,
bool printme = false )  [inline]
```

Returns a set of chords in sector 0 of the cyclical region, sorted by normal order, for the indicated equivalence relation.

If there are duplicate chords for the same equivalence, only the one closest to the origin is returned. SCOPED_DEBUGGING debugging;

References CHORD_SPACE_DEBUG, CHORD_SPACE_DEBUGGING(), fundamentalDomainByPredicate(), iterator(), namesForEquivalenceRelations, next(), print_chord(), and csound::Chord::toString().

Referenced by csound::Score::add(), csound::Node::addChild(), csound::Voicelead::addOctave(), csound::Lindenmayer::addRule(), csound::Conversions::amplitudeToDecibels(), csound::Conversions::amplitudeToGain(), csound::Conversions::amplitudeToMidi(), csound::Score::append(), csound::Score::append(), csound::Score::append_event(), csound::Score::append_note(), csound::Score::appendToCsoundScoreHeader(), csound::LispNode::appendTopLevelForm(), csound::VoiceleadingNode::apply(), csound::Voicelead::areParallel(), csound::MusicModel::arrange(), csound::Score::arrange(), csound::MusicModel::arrange(), csound::Score::arrange(), csound::MusicModel::arrange(), csound::Score::arrange(), csound::MusicModel::arrange(), csound::MusicModel::arrange(), csound::MusicModel::arrange(), csound::Score::arrange_all(), csound::AscendingDistanceComparator::a csound::Soundfile::blank(), csound::StrangeAttractor::calculateFractalDimension(), csound::StrangeAttractor::calculateLyupanovExponen csound::Voicelead::chordToPTV(), csound::VoiceleadingNode::chordVoiceleading(), csound::Chunk::Chunk(), csound::Node::clear(), csound::Soundfile::close(), csound::System::closeLibrary(), csound::Voicelead::closer(), csound::Voicelead::closest(), csound::Voicelead::closestPitch(), csound::StrangeAttractor::codeRandomize(), csound::MidiFile::computeTimes(), csound::ImageToScore2::condense(), csound::Event::conformToPitchClassSet(), csound::Voicelead::conformToPitchClassSet(), csound::ImageToScore2::contrast(), csound::Event::correct_negative_duration(), csound::Event::correct_negative_durations(), csound::Soundfile::cosineGrain(), csound::Soundfile::create(), csound::MusicModel::createCsoundScore(), createMatrix(), csound::Event::createNoteOffEvent(), csound::Lindenmayer::createRotation(), csound::Node::createTransform(), csound::MusicModel::csoundArgv(), csound::Voicelead::cToM(), csound::Voicelead::cToP(), csound::System::debug(), csound::System::debug(), csound::Conversions::decibelsToAmplitude(), csound::KMeansMCRM::deterministic_algorithm(), csound::ImageToScore2::dilate(), csound::Conversions::doubleToString(), csound::Event::dump(), csound::MidiFile::dump(), csound::Score::dump(), csound::Conversions::dupstr(), csound::ImageToScore2::erode(), csound::Soundfile::error(), csound::System::error(), csound::System::error(), csound::Voicelead::euclideanDistance(), evaluate_form(), csound::Event::Event(), csound::Event::Event(), csound::Conversions::findClosestPitchClass(), csound::Score::findScale(), fundamentalDomainByPredicate(), csound::Conversions::gainToAmplitude(), csound::Conversions::gainToDb(), csound::ImageToScore2::gaussianBlur(), csound::MusicModel::generate(), csound::ExternalNode::generate(), csound::ScoreNode::generate(), csound::Random::generate(), csound::LispGenerator::generate(), csound::Composition::generateAllNames(), csound::Composition::generateFilename(), csound::Shell::generateFilename(), csound::CMaskNode::generateLocally(), csound::ExternalNode::generateLocally(), csound::ImageToScore2::generateLocally(), csound::Lindenmayer::generateLocally(), csound::MCRM::generateLocally(), csound::KMeansMCRM::generateLocally(), csound::Score::getBlueScore(), csound::Event::getChannel(), csound::MusicModel::getCsour csound::Score::getCsoundScore(), csound::TempoMap::getCurrentSecondsPerTick(), csound::StrangeAttractor::getDimensionAndOrder(csound::Score::getDuration(), csound::Score::getDurationFromZero(), csound::CsoundProducer::GetFilenameBase(), csound::CsoundProducer::GetGitCommitHash(), csound::Event::getKey_tempered(), csound::Event::getKeyNumber(), csound::Conversions::getMaximumAmplitude(), csound::Conversions::getMaximumDynamicRange(), csound::CsoundProducer::GetMeta csound::MidiEvent::getMetaSize(), csound::Shell::getMidiFilename(), csound::Event::getMidiStatus(), csound::StrangeAttractor::getNorma csound::StrangeAttractor::getNormalizedX(), csound::StrangeAttractor::getNormalizedY(), csound::StrangeAttractor::getNormalizedZ(), csound::LispNode::getNumberFromForm(), csound::Composition::getOutputDirectory(), csound::Shell::getOutputSoundfileName(), csound::Score::getPitches(), csound::Event::getProperties(), csound::Score::getPT(), csound::Score::getPTV(), csound::Random::getRandomCoordinates(), csound::Lindenmayer::getReplacement(), csound::Rescale::getRescale(), csound::Score::getScale(), csound::Event::getStatusNumber(), csound::LispNode::getStringFromForm(), csound::System::getSymbol(), csound::MusicModel::getThis(), csound::ScoreModel::getThis(), csound::Event::getVelocityNumber(), csound::Score::getVoicing(), csound::CsoundProducer::GitCommit(), csound::Conversions::hzToMidi(), csound::Conversions::hzToOctave(), csound::Conversions::hzToSamplingIncrement(), csound::Voicelead::I(), csound::Voicelead::I_vector(), csound::Voicelead::Iform(), csound::Score::indexAfterTime(), csound::Score::indexAtTime(), csound::Score::indexToTime(), csound::System::inform(),

csound::System::inform(), csound::Conversions::initialize(), csound::Event::initialize(), initialize_ecl(), csound::Voicelead::initializePrimeCh
csound::Lindenmayer::interpret(), csound::Conversions::intToString(), csound::Voicelead::inversions(), inversions(),
csound::Voicelead::invert(), csound::Event::isMatchingEvent(), csound::Event::isMatchingNoteOff(), csound::MCRM::iterate(),
csound::KMeansMCRM::iterate(), csound::CsoundProducer::Join(), csound::Soundfile::jonesParksGrain(), csound::Voicelead::K(),
csound::Conversions::leftPan(), csound::Score::load(), csound::MidiFile::load(), csound::Score::load(), csound::Shell::loadAppend(),
log_file(), main(), csound::Shell::main(), csound::Composition::makeTimestamp(), csound::Chunk::markChunkEnd(),
csound::Chunk::markChunkSize(), csound::Chunk::markChunkStart(), matchContextSize(), csound::MidiEvent::matchesNoteOffEvent(),
mean_to_note(), csound::KMeansMCRM::means_to_notes(), csound::System::message(), csound::System::message(),
csound::System::message(), csound::System::message(), csound::System::message(), csound::System::message(),
csound::System::message(), message_callback(), message_level(), csound::Conversions::midiToAmplitude(),
csound::Conversions::midiToDecibels(), csound::Conversions::midiToGain(), csound::Conversions::midiToHz(),
csound::Conversions::midiToOctave(), csound::Conversions::midiToPitchClass(), csound::Conversions::midiToPitchClassSet(),
csound::Conversions::midiToRoundedOctave(), csound::Conversions::midiToSamplingIncrement(), csound::Soundfile::mixFrames(),
csound::Voicelead::mToC(), csound::Voicelead::mToPitchClassSet(), csound::Voicelead::nameToC(), csound::Conversions::nameToPitche
csound::Voicelead::nonBijectiveVoicelead(), csound::Voicelead::normalChord(), csound::Composition::normalizeOutputSoundfile(),
csound::Conversions::octaveToHz(), csound::Conversions::octaveToMidi(), csound::Conversions::octaveToSamplingIncrement(),
csound::Shell::open(), csound::Soundfile::open(), csound::System::openLibrary(), csound::TimeAtComparator::operator()(),
csound::TimeAfterComparator::operator()(), csound::AscendingDistanceComparator::operator()(), operator<<(),
csound::MidiTrack::operator=(), csound::Voicelead::orderedPcs(), csound::Voicelead::pAndTtoPitchClassSet(),
parse_line(), csound::Voicelead::pc(), csound::Voicelead::pcs(), csound::MusicModel::perform(), csound::Composition::performAll(),
csound::CsoundProducer::PerformAndPostProcess(), csound::CsoundProducer::PerformAndPostProcessRoutine(),
csound::Composition::performMaster(), csound::Conversions::phaseToTableLengths(), csound::Voicelead::pitchClassSetToM(),
csound::Conversions::pitchClassSetToMidi(), csound::Voicelead::pitchClassSetToPandT(), csound::Conversions::pitchClassToMidi(),
pitchRotations(), csound::ImageToScore2::pixel_to_event(), PostProcess(), printChord(), printChord(), csound::Score::process(),
csound::MusicModel::processArgs(), csound::Composition::processArgv(), csound::ImageToScore2::processImage(),
csound::Voicelead::pToC(), csound::Voicelead::pToPrimeChord(), csound::Voicelead::ptvToChord(), pythonFuncWarning(),
csound::Voicelead::Q(), csound::KMeansMCRM::random_algorithm(), csound::Chunk::read(), csound::MidiHeader::read(),
csound::MidiFile::read(), csound::MidiEvent::readByte(), csound::Soundfile::readFrame(), csound::Soundfile::readFrames(),
csound::MidiEvent::readIn(), csound::MidiTrack::readIn(), csound::MidiFile::readInt(), csound::MidiFile::readShort(),
csound::MidiFile::readVariableLength(), csound::Voicelead::recursiveVoicelead(), recursiveVoicelead_(), csound::Score::remove(),
csound::Composition::render(), csound::MusicModel::render(), csound::StrangeAttractor::render(), csound::Composition::renderAll(),
csound::Score::rescale(), csound::Score::rescale(), csound::Score::rescale_event(), csound::MCRM::resize(),
csound::Lindenmayer::rewrite(), csound::Conversions::rightPan(), csound::Voicelead::rotate(), csound::Voicelead::rotations(),
csound::Shell::runScript(), csound::Score::save(), csound::MidiFile::save(), csound::Score::save(), csound::Shell::save(),
scoreToSeq(), csound::Soundfile::seek(), csound::Soundfile::seekSeconds(), seqToScore(), csound::Event::set(),
csound::Event::setAmplitude(), csound::StrangeAttractor::setAttractorType(), csound::Event::setChannel(), csound::Soundfile::setChannel
csound::ExternalNode::setCommand(), setCorrectNegativeDurations(), csound::Score::setCsoundScoreHeader(),
csound::MusicModel::setCsoundScoreHeader(), csound::CsoundProducer::SetDoGitCommit(), csound::Composition::setDuration(),
csound::Score::setDuration(), csound::Score::setDurationFromZero(), csound::Soundfile::setFormat(), csound::Soundfile::setFramesPerS
csound::Event::setHeight(), csound::ImageToScore2::setImageFilename(), csound::StrangeAttractor::setIteration(),
csound::StrangeAttractor::setIterationCount(), csound::Score::setK(), csound::Score::setKL(), csound::Score::setKV(),
csound::System::setLogfile(), csound::ImageToScore2::setMaximumVoiceCount(), csound::System::setMessageCallback(),
csound::System::setMessageLevel(), csound::CsoundProducer::SetMetadata(), csound::Event::setMidi(), csound::Event::setOffTime(),
csound::CsoundProducer::SetOutput(), csound::Composition::setOutputDirectory(), csound::CMaskNode::setParametersText(),
csound::Score::setPitchClassSet(), csound::Event::setPitches(), csound::Score::setPitches(), csound::Score::setPT(),
csound::Score::setPTV(), csound::Score::setQ(), csound::Score::setQL(), csound::Score::setQV(), csound::Rescale::setRescale(),
csound::Score::setScale(), csound::Composition::setScore(), csound::StrangeAttractor::setScoreType(), csound::ExternalNode::setScript(
csound::Composition::setTieOverlappingNotes(), csound::System::setUserdata(), csound::Score::setVoicing(),
csound::StrangeAttractor::setW(), csound::MCRM::setWeight(), csound::StrangeAttractor::setX(), csound::StrangeAttractor::setY(),
csound::StrangeAttractor::setZ(), csound::ImageToScore2::sharpen(), csound::StrangeAttractor::shuffleRandomNumbers(),
csound::Voicelead::simpler(), csound::Voicelead::smoothness(), csound::Score::sort(), sort(), csound::Voicelead::sortByAscendingDistanc
csound::StrangeAttractor::specialFunctions(), csound::CsoundProducer::Start(), csound::CsoundProducer::startTiming(),
csound::System::startTiming(), csound::ThreadLock::startWait(), csound::CsoundProducer::stopTiming(), csound::System::stopTiming(),
csound::Conversions::stringToBool(), csound::Conversions::stringToDouble(), csound::Conversions::stringToInt(),

csound::Conversions::stringToVector(), csound::Conversions::swapInt(), csound::Conversions::swapShort(), csound::Voicelead::T(), csound::Voicelead::T_vector(), csound::Composition::tagFile(), csound::Score::temper(), csound::Voicelead::Tform(), csound::ImageToScore2::threshhold(), csound::Score::tieOverlappingNotes(), to_std_string(), csound::Event::toBlueIStatement(), csound::Event::toCsoundIStatement(), csound::Event::toCsoundIStatementHeld(), csound::Event::toCsoundIStatementRelease(), csound::MidiFile::toInt(), csound::Score::toJson(), csound::Voicelead::toOrigin(), csound::MidiFile::toShort(), csound::Score::toString(), csound::Event::toString(), csound::MidiEvent::toString(), csound::Score::transform(), csound::CounterpointNode::transform(), csound::RemoveDuplicates::transform(), csound::Random::transform(), csound::Rescale::transform(), csound::VoiceleadingNode::transfo csound::LispTransformer::transform(), csound::CMaskNode::translate_to_silence(), csound::Composition::translateMaster(), csound::Composition::translateToCdAudio(), csound::Composition::translateToMp3(), csound::Composition::translateToMp4(), csound::Composition::translateToNotation(), csound::Voicelead::transpose(), csound::Node::traverse(), csound::Sequence::traverse(), csound::Voicelead::uniquePcs(), csound::Lindenmayer::updateActual(), csound::Soundfile::updateHeader(), user_data(), csound::Voicelead::voicelead(), csound::Score::voicelead(), csound::Score::voicelead(), csound::Score::voicelead_pitches(), csound::Score::voicelead_segments(), csound::Voicelead::voiceleading(), csound::Voicelead::voicings(), csound::System::warn(), csound::System::warn(), csound::Voicelead::wrap(), csound::Composition::write(), csound::Logger::write(), csound::Chunk::write(), csound::MidiHeader::write(), csound::MidiFile::write(), csound::ImageToScore2::write_processed_file(), csound::Soundfile::writeFrame(), csound::Soundfile::writeFrames(), csound::MidiFile::writeInt(), csound::MidiEvent::writeOut(), csound::MidiTrack::writeOut(), csound::MidiFile::writeShort(), and csound::MidiFile::writeVariableLength().

5.2.4.43 fundamentalDomainByTransformation() [1/2]

```
template<int EQUIVALENCE_RELATION>
SILENCE_PUBLIC std::vector< csound::Chord > csound::fundamentalDomainByTransformation (
            int voiceN,
            double range,
            double g = 1.,
            int sector = 0 )  [inline]
```

Returns a set of chords in sector 0 of the cyclical region, sorted by normal order, for the indicated equivalence relation.

All duplicate chords for the same equivalence are returned, ordered by distance from the origin.

References CHORD_SPACE_DEBUG, CHORD_SPACE_DEBUGGING(), fundamentalDomainByTransformation(), iterator(), namesForEquivalenceRelations, next(), and csound::Chord::toString().

Referenced by fundamentalDomainByTransformation().

5.2.4.44 fundamentalDomainByTransformation() [2/2]

```
template<int EQUIVALENCE_RELATION>
SILENCE_PUBLIC std::vector< Chord > csound::fundamentalDomainByTransformation (
            int voiceN,
            double range,
            double g = 1.,
            int sector = 0 )  [inline]
```

Returns a set of chords in sector 0 of the cyclical region, sorted by normal order, for the indicated equivalence relation.

All duplicate chords for the same equivalence are returned, ordered by distance from the origin.

References CHORD_SPACE_DEBUG, CHORD_SPACE_DEBUGGING(), fundamentalDomainByTransformation(), iterator(), namesForEquivalenceRelations, next(), and csound::Chord::toString().

Referenced by fundamentalDomainByTransformation().

5.2.4.45 gather()

```
SILENCE_PUBLIC Chord csound::gather (
            Score & score,
            double startTime,
            double endTime )
```

Returns a chord containing all the pitches of the score beginning at or later than the start time, and up to but not including the end time.

References chord(), gather(), csound::Chord::resize(), csound::Chord::setPitch(), and slice().

Referenced by gather().

5.2.4.46 ge_tolerance()

```
SILENCE_PUBLIC bool csound::ge_tolerance (
            double a,
            double b,
            int epsilons = 20,
            int ulps = 200 )  [inline]
```

References eq_tolerance(), and ge_tolerance().

Referenced by ge_tolerance(), csound::HarmonyIFS::initialize_hutchinson_operator(), csound::HarmonyIFS2::initialize_hutchinson_opera test_eq_tolerance(), and csound::Scale::transpose().

5.2.4.47 getCorrectNegativeDurations()

```
bool SILENCE_PUBLIC csound::getCorrectNegativeDurations ( )
```

References csound::Event::correct_negative_durations().

5.2.4.48 getIndex() [1/2]

```
static int csound::getIndex (
            const std::string & dimension )  [static]
```

Returns a zero-based numerical index for a string dimension name (for Events) or voice number (for Chords).

References csound::Event::DEPTH, csound::Event::DURATION, csound::Event::HEIGHT, csound::Event::INSTRUMENT, csound::Event::KEY, csound::Event::PAN, csound::Event::PHASE, csound::Event::PITCHES, csound::Event::STATUS, csound::Event::TIME, and csound::Event::VELOCITY.

Referenced by getIndex(), csound::ChordLindenmayer::noteOrientationOperation(), and parseIndex().

5.2.4.49 getIndex() [2/2]

```
static bool csound::getIndex (
            int & index,
            const std::string & dimension )  [static]
```

References getIndex().

5.2.4.50 gt_tolerance()

```
SILENCE_PUBLIC bool csound::gt_tolerance (
            double a,
            double b,
            int epsilons = 20,
            int ulps = 200 )  [inline]
```

References eq_tolerance(), and gt_tolerance().

Referenced by equate< EQUIVALENCE_RELATION_P >(), gt_tolerance(), csound::Chord::is_minor(), csound::Chord::max(), csound::Chord::maximumInterval(), modulo(), operator<(), operator>(), and test_eq_tolerance().

5.2.4.51 hyperplane_equation_from_random_inversion_flat()

```
SILENCE_PUBLIC HyperplaneEquation csound::hyperplane_equation_from_random_inversion_flat (
            int dimensions,
            bool transpositional_equivalence,
            int opt_sector )  [inline]
```

References csound::Chord::center(), chord(), CHORD_SPACE_DEBUG, csound::HyperplaneEquation::constant_term, csound::Chord::eP(), csound::Chord::eT(), csound::Chord::getPitch(), hyperplane_equation_from_random_inversion_flat(), hyperplane_equation_from_singular_value_decomposition(), mersenne_twister, csound::Chord::setPitch(), and csound::HyperplaneEquation::unit_normal_vector.

Referenced by hyperplane_equation_from_random_inversion_flat().

5.2.4.52 hyperplane_equation_from_singular_value_decomposition()

```
SILENCE_PUBLIC HyperplaneEquation csound::hyperplane_equation_from_singular_value_decomposition (
            const std::vector< Chord > & points_,
            bool make_eT )  [inline]
```

References csound::HyperplaneEquation::constant_term, hyperplane_equation_from_singular_value_decomposition(), and csound::HyperplaneEquation::unit_normal_vector.

Referenced by hyperplane_equation_from_random_inversion_flat(), and hyperplane_equation_from_singular_value_decomposition().

5.2.4.53 I()

```
SILENCE_PUBLIC double csound::I (
            double pitch,
            double center = 0.0 )  [inline]
```

Returns the pitch reflected in the center, which may be any pitch.

NOTE: Does NOT return an equivalent under any requivalence relation.

References I().

Referenced by csound::HarmonyIFS::add_interpolation_point(), csound::HarmonyIFS2::add_interpolation_point(), csound::PITV::fromChord(), csound::HarmonyInterpolationPoint::HarmonyInterpolationPoint(), csound::HarmonyInterpolationPoint2::Harm csound::Chord::I(), I(), csound::Chord::inverse_prime_form(), csound::HarmonyIFS2::iterate(), csound::HarmonyIFS::point_to_note(), csound::Chord::prime_form(), csound::PITV::toChord(), csound::HarmonyPoint::toString(), csound::HarmonyInterpolationPoint::toString(), csound::HarmonyPoint2::toString(), and csound::HarmonyInterpolationPoint2::toString().

5.2.4.54 indexForOctavewiseRevoicing() [1/2]

```
SILENCE_PUBLIC int csound::indexForOctavewiseRevoicing (
            const Chord & chord,
            double range )  [inline]
```

Returns the index of the octavewise revoicing that this chord is, relative to its OP equivalent, within the indicated range.

Returns -1 if there is no such chord within the range.

References chord(), equate< EQUIVALENCE_RELATION_RP >(), indexForOctavewiseRevoicing(), and OCTAVE().

5.2.4.55 indexForOctavewiseRevoicing() [2/2]

```
SILENCE_PUBLIC int csound::indexForOctavewiseRevoicing (
            const Chord & origin,
            const Chord & chord,
            double range )  [inline]
```

Returns the index of the octavewise revoicing that this chord is, counting up from the origin, within the indicated range.

Returns -1 if there is no such chord within the range.

References chord(), CHORD_SPACE_DEBUG, indexForOctavewiseRevoicing(), next(), OCTAVE(), octavewiseRevoicings(), and csound::Chord::toString().

Referenced by csound::PITV::fromChord(), indexForOctavewiseRevoicing(), and indexForOctavewiseRevoicing().

5.2.4.56 initialize_ecl()

```
void csound::initialize_ecl (
            int argc,
            char ** argv )
```

This function must be called with the arc and argv from main() before any Lisp code is executed.

References fundamentalDomainByPredicate(), and csound::System::inform().

Referenced by main().

5.2.4.57 initializeNames()

```
void csound::initializeNames ( )  [inline]
```

References CHORD_SPACE_DEBUG, fill(), initializeNames(), and pitchClassesForNames().

Referenced by chordForName(), initializeNames(), namesForChord(), namesForScale(), and scaleForName().

5.2.4.58 insert() [1/2]

```
SILENCE_PUBLIC void csound::insert (
            Score & score,
            const Chord & chord,
            double time_ )
```

References chord(), insert(), and toScore().

Referenced by insert(), and insert().

5.2.4.59 insert() [2/2]

```
SILENCE_PUBLIC void csound::insert (
            Score & score,
            const Chord & chord,
            double time_,
            bool voice_is_instrument )
```

Inserts the notes of the chord into the score at the specified time.

References chord(), insert(), and toScore().

5.2.4.60 interpolation_point_less()

```
SILENCE_PUBLIC bool csound::interpolation_point_less (
            const HarmonyInterpolationPoint & a,
            const HarmonyInterpolationPoint & b )  [inline]
```

References interpolation_point_less(), and csound::HarmonyInterpolationPoint::t.

Referenced by csound::HarmonyIFS::initialize_hutchinson_operator(), and interpolation_point_less().

5.2.4.61 interpolation_point_less2()

```
SILENCE_PUBLIC bool csound::interpolation_point_less2 (
            const HarmonyInterpolationPoint2 & a,
            const HarmonyInterpolationPoint2 & b )  [inline]
```

References interpolation_point_less2(), and csound::HarmonyInterpolationPoint2::t.

Referenced by csound::HarmonyIFS2::initialize_hutchinson_operator(), and interpolation_point_less2().

5.2.4.62 inverse_prime_forms_for_chords()

```
SILENCE_PUBLIC std::map< Chord, Chord > & csound::inverse_prime_forms_for_chords ( )  [inline]
```

Cache inverse prime forms for chords for speed.

References inverse_prime_forms_for_chords().

Referenced by csound::Chord::inverse_prime_form(), and inverse_prime_forms_for_chords().

5.2.4.63 inversions()

```
void csound::inversions (
            const std::vector< double > & original,
            const std::vector< double > & iterator,
            size_t voice,
            double maximum,
            std::set< std::vector< double > > & chords,
            size_t divisionsPerOctave )
```

References fundamentalDomainByPredicate(), inversions(), iterator(), and sort().

Referenced by inversions().

5.2.4.64 iterator()

```
SILENCE_PUBLIC Chord csound::iterator (
            int voiceN,
            double first )  [inline]
```

Returns a chord with the specified number of voices all set to a first pitch, useful as an iterator.

References iterator(), csound::Chord::resize(), and csound::Chord::setPitch().

Referenced by csound::Voicelead::chordToPTV(), fundamentalDomainByPredicate(), fundamentalDomainByTransformation(), csound::Conversions::initialize(), csound::PITV::initialize(), inversions(), iterator(), csound::Score::load(), predicate(), csound::Voicelead::ptvToChord(), csound::Voicelead::recursiveVoicelead(), recursiveVoicelead_(), csound::Composition::translateToNotat and csound::Voicelead::voicings().

5.2.4.65 le_tolerance()

```
SILENCE_PUBLIC bool csound::le_tolerance (
            double a,
            double b,
            int epsilons = 20,
            int ulps = 200 )  [inline]
```

References eq_tolerance(), and le_tolerance().

Referenced by equate< EQUIVALENCE_RELATION_R >(), csound::Chord::is_compact(), le_tolerance(), modulo(), next(), predicate(), predicate< EQUIVALENCE_RELATION_P >(), predicate< EQUIVALENCE_RELATION_R >(), and predicate< EQUIVALENCE_RELATION_r >().

5.2.4.66 log_file()

```
SILENCE_PUBLIC FILE *& csound::log_file ( )
```

References fundamentalDomainByPredicate().

Referenced by csound::System::getLogfile(), csound::System::message(), and csound::System::setLogfile().

5.2.4.67 lt_tolerance()

```
SILENCE_PUBLIC bool csound::lt_tolerance (
            double a,
            double b,
            int epsilons = 20,
            int ulps = 200 )  [inline]
```

References eq_tolerance(), and lt_tolerance().

Referenced by equate< EQUIVALENCE_RELATION_Tg >(), csound::Chord::is_minor(), lt_tolerance(), csound::Chord::min(), csound::Chord::minimumInterval(), csound::Chord::normal_order(), operator<(), operator>(), csound::Chord::opti_domain_sectors(), predicate(), predicate< EQUIVALENCE_RELATION_r >(), predicate< EQUIVALENCE_RELATION_Tg >(), and csound::Scale::transpose().

5.2.4.68 matchContextSize()

```
static std::vector< double > csound::matchContextSize (
            const std::vector< double > context,
            const std::vector< double > pcs )  [static]
```

References fundamentalDomainByPredicate().

Referenced by csound::Score::setQ(), csound::Score::setQL(), and csound::Score::setQV().

5.2.4.69 max()

```
static double csound::max (
            double a,
            double b )  [static]
```

Referenced by csound::Score::getScale().

5.2.4.70 mean_to_note()

```
static Event csound::mean_to_note (
            const std::array< double, KMeansMCRM::MEASURE_DIMENSIONS > & mean )  [static]
```

References fundamentalDomainByPredicate(), and csound::Event::setTime().

Referenced by csound::KMeansMCRM::means_to_notes().

5.2.4.71 message_callback()

```
MessageCallbackType & csound::message_callback ( )
```

References fundamentalDomainByPredicate().

Referenced by csound::System::getMessageCallback(), csound::System::message(), and csound::System::setMessageCallback().

5.2.4.72 message_level()

```
SILENCE_PUBLIC int csound::message_level (
            int verbosity )
```

References fundamentalDomainByPredicate().

Referenced by csound::System::debug(), csound::System::debug(), csound::System::error(), csound::System::error(), csound::System::getMessageLevel(), csound::System::inform(), csound::System::inform(), csound::System::message(), csound::System::message(), csound::System::message(), csound::System::message(), csound::System::setMessageLevel(), csound::System::warn(), and csound::System::warn().

5.2.4.73 MIDDLE_C()

```
SILENCE_PUBLIC double csound::MIDDLE_C ( )  [inline]
```

References MIDDLE_C().

Referenced by C4(), MIDDLE_C(), and predicate().

5.2.4.74 midpoint()

```
SILENCE_PUBLIC Chord csound::midpoint (
            const Chord & a,
            const Chord & b )  [inline]
```

Returns the chord that is the midpoint between two chords, which must have the same number of voices.

CHORD_SPACE_DEBUG("a: %s b: %s mid: %s\n", a.toString().c_str(), b.toString().c_str(), midpoint_.to↩ String().c_str());

CHORD_SPACE_DEBUG("a: %s b: %s mid: %s\n", a.toString().c_str(), b.toString().c_str(), midpoint_.to↩ String().c_str());

References csound::Chord::getPitch(), midpoint(), csound::Chord::setPitch(), and csound::Chord::voices().

Referenced by csound::Chord::initialize_sectors(), midpoint(), and predicate().

5.2.4.75 min()

```
static double csound::min (
            double a,
            double b )  [static]
```

Referenced by csound::Score::getScale().

5.2.4.76 minimumCell()

```
const MatrixCell & csound::minimumCell (
            const MatrixCell & a,
            const MatrixCell & b,
            const MatrixCell & c )
```

References csound::MatrixCell::d.

Referenced by createMatrix().

5.2.4.77 modulo()

```
SILENCE_PUBLIC double csound::modulo (
            double dividend,
            double divisor )  [inline]
```

Returns the remainder of the dividend divided by the divisor, according to the Euclidean definition.

References gt_tolerance(), le_tolerance(), and modulo().

Referenced by epc(), equate< EQUIVALENCE_RELATION_r >(), main(), modulo(), and predicate().

5.2.4.78 nameForChord()

```
SILENCE_PUBLIC std::string csound::nameForChord (
            const Chord & chord )  [inline]
```

Returns the first valid name for the Chord.

If none exists, an empty result is returned.

References chord(), nameForChord(), and namesForChord().

Referenced by csound::Chord::name(), nameForChord(), and predicate().

5.2.4.79 nameForPitchClass()

```
SILENCE_PUBLIC std::string csound::nameForPitchClass (
            double pitch )  [inline]
```

Returns the name of the pitch-class of the pitch.

The first of enharmonic names is always used, sorry. If there is no matching name, an empty string is returned.

References epc(), eq_tolerance(), nameForPitchClass(), and pitchClassesForNames().

Referenced by csound::Scale::name(), nameForPitchClass(), predicate(), and csound::Scale::transpose().

5.2.4.80 nameForScale()

```
SILENCE_PUBLIC std::string csound::nameForScale (
            const Scale & scale )  [inline]
```

Returns the first valid name for the Scale.

References nameForScale(), namesForScale(), and scale().

Referenced by nameForScale(), and predicate().

5.2.4.81 namesForChord()

```
SILENCE_PUBLIC std::vector< std::string > csound::namesForChord (
            const Chord & chord )  [inline]
```

Returns all enharmonic names for the Chord, if any exists.

If none exists, an empty result is returned.

References chord(), initializeNames(), namesForChord(), and namesForChords().

Referenced by nameForChord(), and namesForChord().

5.2.4.82 namesForChords()

```
SILENCE_PUBLIC std::multimap< Chord, std::string > & csound::namesForChords ( )  [inline]
```

References namesForChords().

Referenced by add_chord(), namesForChord(), namesForChords(), and predicate().

5.2.4.83 namesForScale()

```
SILENCE_PUBLIC std::vector< std::string > csound::namesForScale (
            const Scale & scale )  [inline]
```

Returns all enharmonic names for the Scale, if any exists.

If none exists, an empty result is returned.

References initializeNames(), namesForScale(), namesForScales(), and scale().

Referenced by nameForScale(), and namesForScale().

5.2.4.84 namesForScales()

```
SILENCE_PUBLIC std::multimap< Scale, std::string > & csound::namesForScales ( )  [inline]
```

References namesForScales().

Referenced by add_scale(), namesForScale(), namesForScales(), and predicate().

5.2.4.85 next()

```
SILENCE_PUBLIC bool csound::next (
            Chord & iterator_,
            const Chord & minimum,
            double range,
            double g = 1.  )  [inline]
```

Increment a chord voicewise through chord space, from a low point on the unison diagonal through a high point on the unison diagonal.

g is the generator of transposition. Before iterating the iterator must be set to the low point of iteration.

References csound::Chord::getPitch(), le_tolerance(), csound::Chord::min(), next(), csound::Chord::setPitch(), and csound::Chord::voices().

Referenced by fundamentalDomainByPredicate(), fundamentalDomainByTransformation(), indexForOctavewiseRevoicing(), csound::PITV::initialize(), next(), octavewiseRevoicing(), octavewiseRevoicings(), predicate(), and voiceleadingClosestRange().

5.2.4.86 normal_forms_for_chords()

```
SILENCE_PUBLIC std::map< Chord, Chord > & csound::normal_forms_for_chords ( )  [inline]
```

Cache prime forms for chords for speed.

References normal_forms_for_chords().

Referenced by csound::Chord::normal_form(), and normal_forms_for_chords().

5.2.4.87 note()

```
SILENCE_PUBLIC Event csound::note (
            const Chord & chord,
            int voice,
            double time_,
            double duration_ = DBL_MAX,
            double channel_ = DBL_MAX,
            double velocity_ = DBL_MAX,
            double pan_ = DBL_MAX )
```

Creates a complete "note on" Event for the indicated voice of the chord.

If the optional duration, channel, velocity, and pan parameters are not passed, then the Chord's own values for these are used.

References chord(), csound::Chord::getDuration(), csound::Chord::getInstrument(), csound::Chord::getLoudness(), csound::Chord::getPan(), csound::Chord::getPitch(), note(), csound::Event::setDuration(), csound::Event::setInstrument(), csound::Event::setKey(), csound::Event::setPan(), csound::Event::setTime(), and csound::Event::setVelocity().

Referenced by csound::Turtle::__str__(), csound::HarmonyEvent::get_note(), csound::Turtle::initialize(), note(), notes(), csound::Turtle::operator<(), csound::Turtle::operator=(), csound::Score::save(), csound::HarmonyEvent::set_note(), and csound::CounterpointNode::transform().

5.2.4.88 notes()

```
SILENCE_PUBLIC Score csound::notes (
            const Chord & chord,
            double time_,
            double duration_ = DBL_MAX,
            double channel_ = DBL_MAX,
            double velocity_ = DBL_MAX,
            double pan_ = DBL_MAX )
```

Returns an individual note for each voice of the chord.

If the optional duration, channel, velocity, and pan parameters are not passed, then the Chord's own values for these are used.

References csound::Score::append(), chord(), note(), notes(), and csound::Chord::voices().

Referenced by notes().

5.2.4.89 numerics_information()

```
SILENCE_PUBLIC void csound::numerics_information (
            double a,
            double b,
            int epsilons,
            int ulps )
```

References CHORD_SPACE_DEBUG, eq_tolerance(), and numerics_information().

Referenced by numerics_information(), and test_eq_tolerance().

5.2.4.90 OCTAVE()

```
SILENCE_PUBLIC double csound::OCTAVE ( )  [inline]
```

The size of the octave, defined to be consistent with 12 tone equal temperament and MIDI.

References OCTAVE().

Referenced by csound::Chord::center(), chord(), csound::Chord::clamp(), conformToPitchClassSet(), csound::Chord::eI(), csound::Chord::eO(), csound::Chord::eOP(), csound::Chord::eOPI(), csound::Chord::eOPT(), csound::Chord::eOPTI(), csound::Chord::eOPTT(), csound::Chord::eOPTTI(), csound::Chord::eP(), epc(), equate(), equate(), csound::Chord::eT(), csound::Chord::eTT(), csound::Chord::Iform(), indexForOctavewiseRevoicing(), indexForOctavewiseRevoicing(), csound::Chord::iseI_chord(), csound::Chord::iseO(), csound::Chord::iseOP(), csound::Chord::iseOPI(), csound::Chord::iseOPT(), csound::Chord::iseOPTI(), csound::Chord::iseOPTT(), csound::Chord::iseOPTTI(), csound::Chord::iseP(), csound::Chord::iseT(), csound::Chord::iseTT(), main(), csound::Chord::normal_order(), OCTAVE(), octavewiseRevoicing(), octavewiseRevoicings(), predicate(), csound::PITV::preinitialize(), csound::Chord::Tform(), csound::Scale::transpose(), csound::Chord::v(), and voiceleadingClosestRange().

5.2.4.91 octavewiseRevoicing()

```
SILENCE_PUBLIC Chord csound::octavewiseRevoicing (
            const Chord & chord,
            int revoicingNumber_,
            double range )  [inline]
```

Returns the nth octavewise revoicing of the chord that is generated by iterating revoicings within the indicated range.

References chord(), CHORD_SPACE_DEBUG, equate< EQUIVALENCE_RELATION_RP >(), next(), OCTAVE(), octavewiseRevoicing(), octavewiseRevoicings(), and csound::Chord::toString().

Referenced by csound::ChordLindenmayer::chordOperation(), equate(), csound::PITV::fromChord(), main(), octavewiseRevoicing(), and csound::PITV::toChord().

5.2.4.92 octavewiseRevoicings()

```
SILENCE_PUBLIC int csound::octavewiseRevoicings (
            const Chord & chord,
            double range = OCTAVE() )  [inline]
```

Returns the full set of octavewise revoicings of the chord within the indicated range.

References chord(), CHORD_SPACE_DEBUG, csound::Chord::eOP(), next(), OCTAVE(), octavewiseRevoicings(), and csound::Chord::toString().

Referenced by equate(), indexForOctavewiseRevoicing(), octavewiseRevoicing(), octavewiseRevoicings(), and csound::PITV::preinitialize().

5.2.4.93 operator<() [1/3]

```
SILENCE_PUBLIC bool csound::operator< (
            const Chord & a,
            const Chord & b )  [inline]
```

References csound::Chord::getPitch(), gt_tolerance(), lt_tolerance(), and csound::Chord::voices().

5.2.4.94 operator<() [2/3]

```
bool SILENCE_PUBLIC csound::operator< (
            const Event & a,
            const Event & b )
```

References csound::Event::ELEMENT_COUNT, and csound::Event::SORT_ORDER.

5.2.4.95 operator<() [3/3]

```
bool csound::operator< (
            const MidiEvent & a,
            const MidiEvent & b )  [inline]
```

5.2.4.96 operator<<()

```
std::ostream SILENCE_PUBLIC & csound::operator<< (
            std::ostream & stream,
            const VoiceleadingOperation & operation )
```

References fundamentalDomainByPredicate().

5.2.4.97 operator<=()

```
SILENCE_PUBLIC bool csound::operator<= (
            const Chord & a,
            const Chord & b )  [inline]
```

5.2.4.98 operator==()

```
SILENCE_PUBLIC bool csound::operator== (
            const Chord & a,
            const Chord & b )  [inline]
```

References eq_tolerance(), csound::Chord::getPitch(), and csound::Chord::voices().

5.2.4.99 operator>()

```
SILENCE_PUBLIC bool csound::operator> (
            const Chord & a,
            const Chord & b )  [inline]
```

References csound::Chord::getPitch(), gt_tolerance(), lt_tolerance(), and csound::Chord::voices().

5.2.4.100 operator>=()

```
SILENCE_PUBLIC bool csound::operator>= (
            const Chord & a,
            const Chord & b )  [inline]
```

5.2.4.101 parallelFifth()

```
SILENCE_PUBLIC bool csound::parallelFifth (
            const Chord & a,
            const Chord & b )  [inline]
```

Returns whether the voiceleading between chords a and b contains a parallel fifth.

References csound::Chord::count(), parallelFifth(), and voiceleading().

Referenced by equate(), parallelFifth(), voiceleadingCloser(), voiceleadingSimpler(), and voiceleadingSmoother().

5.2.4.102 parse_line()

```
static void csound::parse_line (
            std::string line,
            Score & score )  [static]
```

Csound dimensions are assumed to be the same as for Event::toCsoundIStatement.

References fundamentalDomainByPredicate(), and csound::Event::setStatus().

Referenced by csound::ExternalNode::generateLocally().

5.2.4.103 parseIndex()

```
static bool csound::parseIndex (
            int & index,
            const std::string & target )  [static]
```

References csound::System::debug(), and getIndex().

Referenced by csound::ChordLindenmayer::arithmetic(), and csound::ChordLindenmayer::arithmetic().

5.2.4.104 parseVector()

```
bool csound::parseVector (
            std::vector< double > & elements,
            std::string text )
```

References real().

Referenced by csound::ChordLindenmayer::arithmetic(), csound::ChordLindenmayer::arithmetic(), and csound::ChordLindenmayer::scale

5.2.4.105 pitchClassesForNames()

```
SILENCE_PUBLIC const std::map< std::string, double > & csound::pitchClassesForNames ( )  [inline]
```

References pitchClassesForNames().

Referenced by initializeNames(), nameForPitchClass(), pitchClassesForNames(), and pitchClassForName().

5.2.4.106 pitchClassForName()

```
SILENCE_PUBLIC double csound::pitchClassForName (
            std::string name )  [inline]
```

References pitchClassesForNames(), and pitchClassForName().

Referenced by equate(), fill(), pitchClassForName(), and scale().

5.2.4.107 pitchRotations()

```
std::vector< std::vector< double > > csound::pitchRotations (
            const std::vector< double > & chord )
```

References chord(), fundamentalDomainByPredicate(), and csound::Voicelead::rotate().

Referenced by csound::Voicelead::recursiveVoicelead().

5.2.4.108 PostProcess()

```
static void csound::PostProcess (
            std::map< std::string, std::string > & tags,
            std::string filename,
            CsoundThreaded * csound )  [static]
```

Uses ffmpeg to translate a soundfile to a normalized output file, an MP3 file, a CD audio file, a FLAC soundfile, and an MP4 video file suitable for posting to YouTube.

All files are tagged with metadata. This function is called automatically by PerformAndPostProcess.

References fundamentalDomainByPredicate().

Referenced by csound::CsoundProducer::PerformAndPostProcessRoutine().

5.2.4.109 predicate() [1/4]

```
template<int EQUIVALENCE_RELATION>
SILENCE_PUBLIC bool csound::predicate (
            const Chord & chord )  [inline]
```

References chord(), OCTAVE(), and predicate().

5.2.4.110 predicate() [2/4]

```
template<int EQUIVALENCE_RELATION>
SILENCE_PUBLIC bool csound::predicate (
            const Chord & chord,
            double range )  [inline]
```

References chord(), and predicate().

5.2.4.111 predicate() [3/4]

```
template<int EQUIVALENCE_RELATION>
SILENCE_PUBLIC bool csound::predicate (
            const Chord & chord,
            double range,
            double g,
            int opt_sector )
```

Template function returning whether or not the chord is within the specialized fundamental domain, which may in some cases be defined by the indicated range, generator of transposition g, and sector of the cyclical region of OPT fundamental domains.

References iterator(), le_tolerance(), lt_tolerance(), MIDDLE_C(), midpoint(), modulo(), nameForChord(), nameForPitchClass(), nameForScale(), namesForChords(), namesForScales(), next(), and predicate().

Referenced by predicate(), predicate(), predicate(), and predicate().

5.2.4.112 predicate() [4/4]

```
template<int EQUIVALENCE_RELATION>
SILENCE_PUBLIC bool csound::predicate (
            const Chord & chord,
            double range,
            int sector )  [inline]
```

References chord(), and predicate().

5.2.4.113 predicate< EQUIVALENCE_RELATION_I >()

```
template<>
SILENCE_PUBLIC bool csound::predicate< EQUIVALENCE_RELATION_I > (
            const Chord & chord,
            double range,
            double g,
            int opt_sector )  [inline]
```

References chord(), CHORD_SPACE_DEBUG, csound::Chord::is_opti_sector(), csound::Chord::self_inverse(), and csound::Chord::toString().

5.2.4.114 predicate< EQUIVALENCE_RELATION_P >()

```
template<>
SILENCE_PUBLIC bool csound::predicate< EQUIVALENCE_RELATION_P > (
            const Chord & chord,
            double range,
            double g,
            int opt_sector )  [inline]
```

References chord(), csound::Chord::getPitch(), le_tolerance(), and csound::Chord::voices().

5.2.4.115 predicate< EQUIVALENCE_RELATION_R >()

```
template<>
SILENCE_PUBLIC bool csound::predicate< EQUIVALENCE_RELATION_R > (
            const Chord & chord,
            double range,
            double g,
            int opt_sector )  [inline]
```

References chord(), csound::Chord::layer(), le_tolerance(), csound::Chord::max(), and csound::Chord::min().

5.2.4.116 predicate< EQUIVALENCE_RELATION_r >()

```
template<>
SILENCE_PUBLIC bool csound::predicate< EQUIVALENCE_RELATION_r > (
            const Chord & chord,
            double range,
            double g,
            int opt_sector )  [inline]
```

References chord(), csound::Chord::getPitch(), le_tolerance(), lt_tolerance(), and csound::Chord::voices().

5.2.4.117 predicate< EQUIVALENCE_RELATION_RP >()

```
template<>
SILENCE_PUBLIC bool csound::predicate< EQUIVALENCE_RELATION_RP > (
            const Chord & chord,
            double range,
            double g,
            int opt_sector )  [inline]
```

References chord().

5.2.4.118 predicate< EQUIVALENCE_RELATION_RPI >()

```
template<>
SILENCE_PUBLIC bool csound::predicate< EQUIVALENCE_RELATION_RPI > (
            const Chord & chord,
            double range,
            double g,
            int opt_sector )  [inline]
```

References chord().

5.2.4.119 predicate< EQUIVALENCE_RELATION_RPT >()

```
template<>
SILENCE_PUBLIC bool csound::predicate< EQUIVALENCE_RELATION_RPT > (
            const Chord & chord,
            double range,
            double g,
            int opt_sector )  [inline]
```

References chord(), and csound::Chord::is_opt_sector().

5.2.4.120 predicate< EQUIVALENCE_RELATION_RPTg >()

```
template<>
SILENCE_PUBLIC bool csound::predicate< EQUIVALENCE_RELATION_RPTg > (
            const Chord & chord,
            double range,
            double g,
            int opt_sector )  [inline]
```

References chord(), and csound::Chord::is_opt_sector().

5.2.4.121 predicate< EQUIVALENCE_RELATION_RPTgI >()

```
template<>
SILENCE_PUBLIC bool csound::predicate< EQUIVALENCE_RELATION_RPTgI > (
            const Chord & chord,
            double range,
            double g,
            int opt_sector )  [inline]
```

References chord(), and csound::Chord::is_opt_sector().

5.2.4.122 predicate< EQUIVALENCE_RELATION_RPTI >()

```
template<>
SILENCE_PUBLIC bool csound::predicate< EQUIVALENCE_RELATION_RPTI > (
            const Chord & chord,
            double range,
            double g,
            int opt_sector )  [inline]
```

References chord(), and csound::Chord::is_opt_sector().

5.2.4.123 predicate< EQUIVALENCE_RELATION_T >()

```
template<>
SILENCE_PUBLIC bool csound::predicate< EQUIVALENCE_RELATION_T > (
            const Chord & chord,
            double range,
            double g,
            int opt_sector )  [inline]
```

References chord(), CHORD_SPACE_DEBUG, eq_tolerance(), csound::Chord::layer(), and csound::Chord::toString().

5.2.4.124 predicate< EQUIVALENCE_RELATION_Tg >()

```
template<>
SILENCE_PUBLIC bool csound::predicate< EQUIVALENCE_RELATION_Tg > (
            const Chord & chord,
            double range,
            double g,
            int opt_sector )  [inline]
```

References csound::Chord::ceiling(), chord(), eq_tolerance(), csound::Chord::eT(), csound::Chord::layer(), lt_tolerance(), and csound::Chord::T().

5.2.4.125 prime_forms_for_chords()

```
SILENCE_PUBLIC std::map< Chord, Chord > & csound::prime_forms_for_chords ( )  [inline]
```

Cache normal forms for chords for speed.

References prime_forms_for_chords().

Referenced by csound::Chord::prime_form(), and prime_forms_for_chords().

5.2.4.126 print_chord()

```
const char * csound::print_chord (
            const Chord & chord )  [inline]
```

Returns a string representation of the pitches in the chord, along with the sectors of the cyclical regions of the OPT and OPTI fundamental domains to which the chord belongs.

References chord(), csound::Chord::opti_domain_sectors(), print_chord(), and csound::Chord::toString().

Referenced by csound::Chord::ceiling(), equate< EQUIVALENCE_RELATION_RPT >(), equate< EQUIVALENCE_RELATION_RPTg >(csound::PITV::fromChord(), fundamentalDomainByPredicate(), csound::Chord::information_sector(), csound::PITV::initialize(), csound::PITV::list(), print_chord(), and csound::PITV::toChord().

5.2.4.127 print_opti_sectors()

```
static std::string csound::print_opti_sectors (
            const Chord & chord )  [static]
```

References chord(), csound::Chord::opti_domain_sectors(), and print_opti_sectors().

Referenced by csound::Chord::information_sector(), and print_opti_sectors().

5.2.4.128 printChord() [1/2]

```
void SILENCE_PUBLIC csound::printChord (
            std::ostream & stream,
            std::string label,
            const std::vector< double > & chord )  [extern]
```

References chord(), fundamentalDomainByPredicate(), csound::System::getMessageLevel(), and csound::System::INFORMATION_LEVE

Referenced by csound::Turtle::__str__(), csound::VoiceleadingNode::apply(), csound::Score::getPitches(), csound::Score::getVoicing(), printChord(), csound::Score::setK(), csound::Score::setPT(), csound::Score::setPTV(), csound::Score::setQ(), csound::Score::setQL(), csound::Score::setQV(), csound::Score::voicelead(), and csound::Score::voicelead().

5.2.4.129 printChord() [2/2]

```
void SILENCE_PUBLIC csound::printChord (
            std::string label,
            const std::vector< double > & chord )
```

References chord(), fundamentalDomainByPredicate(), csound::System::getMessageLevel(), csound::System::inform(), csound::System::INFORMATION_LEVEL, and printChord().

5.2.4.130 pythonFuncWarning()

```
static bool csound::pythonFuncWarning (
            void ** pythonLibrary,
            const char * funcName )  [static]
```

References fundamentalDomainByPredicate(), and csound::System::warn().

Referenced by csound::Shell::open().

5.2.4.131 real()

```
static double csound::real (
            const std::string & number )  [static]
```

Referenced by csound::ChordLindenmayer::arithmetic(), csound::ChordLindenmayer::arithmetic(), csound::ChordLindenmayer::chordOpe
csound::ChordLindenmayer::modalityOperation(), csound::ChordLindenmayer::noteOperation(), csound::ChordLindenmayer::noteOrienta
parseVector(), csound::ChordLindenmayer::scaleDegreeOperation(), csound::ChordLindenmayer::scaleOperation(),
and csound::ChordLindenmayer::scoreOperation().

5.2.4.132 recursiveVoicelead_()

```
void csound::recursiveVoicelead_ (
            const std::vector< double > & source,
            const std::vector< double > & original,
            const std::vector< double > & iterator,
            std::vector< double > & target,
            size_t voice,
            double maximum,
            bool avoidParallels,
            size_t divisionsPerOctave )
```

References csound::Voicelead::closer(), fundamentalDomainByPredicate(), iterator(), and recursiveVoicelead_().

Referenced by csound::Voicelead::recursiveVoicelead(), and recursiveVoicelead_().

5.2.4.133 reflect_by_householder()

```
SILENCE_PUBLIC Chord csound::reflect_by_householder (
            const Chord & chord )  [inline]
```

Computes the Householder reflector matrix and applies it to the chord.

The transformation is: H(p) = p - 2 ∗ u ∗ (u^T ∗ p). The corresponding matrix is: I - 2 ∗ u ∗ u^T.

References csound::Chord::center(), chord(), CHORD_SPACE_DEBUG, csound::Chord::eT(), csound::Chord::hyperplane_equation(), csound::Chord::opt_domain_sectors(), reflect_by_householder(), csound::Chord::setPitch(), csound::Chord::toString(), toString(), and csound::Chord::voices().

Referenced by main(), and reflect_by_householder().

5.2.4.134 reflect_in_central_diagonal()

```
SILENCE_PUBLIC Chord csound::reflect_in_central_diagonal (
            const Chord & chord )  [inline]
```

References csound::Chord::center(), chord(), csound::Chord::getPitch(), csound::Chord::layer(), reflect_in_central_diagonal(), csound::Chord::T(), and csound::Chord::voices().

Referenced by reflect_in_central_diagonal().

5.2.4.135 reflect_in_central_point()

```
SILENCE_PUBLIC Chord csound::reflect_in_central_point (
            const Chord & chord )  [inline]
```

References csound::Chord::center(), chord(), csound::Chord::getPitch(), reflect_in_central_point(), and csound::Chord::voices().

Referenced by reflect_in_central_point().

5.2.4.136 reflect_in_inversion_flat()

```
SILENCE_PUBLIC Chord csound::reflect_in_inversion_flat (
            const Chord & chord,
            int opt_sector )  [inline]
```

References chord(), csound::HyperplaneEquation::constant_term, csound::Chord::hyperplane_equation(), reflect_in_inversion_flat(), reflect_vector(), csound::Chord::setPitch(), csound::HyperplaneEquation::unit_normal_vector, and csound::Chord::voices().

Referenced by equate< EQUIVALENCE_RELATION_I >(), csound::Chord::information_sector(), csound::Chord::reflect(), reflect_in_inversion_flat(), and csound::Chord::self_inverse().

5.2.4.137 reflect_in_unison_diagonal()

```
SILENCE_PUBLIC Chord csound::reflect_in_unison_diagonal (
            const Chord & chord )  [inline]
```

References chord(), csound::Chord::getPitch(), csound::Chord::layer(), csound::Chord::origin(), reflect_in_unison_diagonal(), csound::Chord::T(), and csound::Chord::voices().

Referenced by reflect_in_unison_diagonal().

5.2.4.138 reflect_vector()

```
SILENCE_PUBLIC Vector csound::reflect_vector (
            const Vector & point,
            const Vector & unit_normal_vector,
            double constant_term )  [inline]
```

Returns the point reflected in the hyperplane defined by the unit normal vector and constant term.

SCOPED_DEBUGGING debugging;

SCOPED_DEBUGGING debugging;

References CHORD_SPACE_DEBUG, reflect_vector(), and toString().

Referenced by reflect_in_inversion_flat(), and reflect_vector().

5.2.4.139 reflect_vectorx()

```
SILENCE_PUBLIC Vector csound::reflect_vectorx (
            const Vector & v,
            const Vector & u,
            double c )  [inline]
```

References reflect_vectorx().

Referenced by reflect_vectorx().

5.2.4.140 removeVoice()

```
static void csound::removeVoice (
            Chord & chord )  [static]
```

References chord(), csound::Chord::eOP(), csound::Chord::resize(), and csound::Chord::voices().

Referenced by csound::ChordLindenmayer::chordOperation(), and csound::ChordLindenmayer::modalityOperation().

5.2.4.141 round()

```
double csound::round (
            double x )
```

Referenced by csound::Voicelead::conformToPitchClassSet(), csound::Voicelead::mToPitchClassSet(), and csound::Voicelead::pToPrimeC

5.2.4.142 scale()

```
SILENCE_PUBLIC Chord csound::scale (
            std::string name )  [inline]
```

Returns the named chord as a scale, that is, starting with the chord in OP, and sorting it from the tonic pitch-class on up.

This enables transformations in tonal harmony such as transposing by scale degree. If no Chord exists for the name, an empty Chord is returned.

References CHORD_SPACE_DEBUG, chordForName(), eq_tolerance(), csound::Chord::getPitch(), pitchClassForName(), scale(), split(), csound::Chord::toString(), and csound::Chord::v().

Referenced by csound::Turtle::__str__(), add_scale(), chord(), equivalentDegree(), fill(), csound::Turtle::initialize(), csound::Scale::modulations_for_scale_types(), nameForScale(), namesForScale(), csound::Turtle::operator<(), csound::Turtle::operator=(), csound::Score::rescale(), csound::Scale::Scale(), scale(), scaleForName(), csound::ChordScore::setScale(), csound::Score::setScale(), csound::ChordLindenmayer::setTurtleScale(), and transpose_degrees().

5.2.4.143 scaleForName()

```
SILENCE_PUBLIC const Scale & csound::scaleForName (
            std::string name )  [inline]
```

References initializeNames(), csound::Chord::resize(), scale(), scaleForName(), and scalesForNames().

Referenced by csound::Turtle::initialize(), main(), and scaleForName().

5.2.4.144 scalesForNames()

```
SILENCE_PUBLIC std::map< std::string, Scale > & csound::scalesForNames ( )  [inline]
```

References scalesForNames().

Referenced by add_scale(), scaleForName(), and scalesForNames().

5.2.4.145 SCOPED_DEBUGGING_FLAG()

```
static SILENCE_PUBLIC bool & csound::SCOPED_DEBUGGING_FLAG ( )  [static]
```

Returns the current state of the chord space *scoped* debugging flag as a reference, which can be an lvalue or an rvalue.

References SCOPED_DEBUGGING_FLAG().

Referenced by csound::SCOPED_DEBUGGING::SCOPED_DEBUGGING(), SCOPED_DEBUGGING_FLAG(), SET_SCOPED_DEBUGGING(), and csound::SCOPED_DEBUGGING::~SCOPED_DEBUGGING().

5.2.4.146 scoreToSeq()

```
cl_object csound::scoreToSeq (
            Score & score,
            std::string seq_name )
```

Translates a Silence Score to Common Music seq.

All Silence note on Events in the Score to Common Music MIDI events in the seq. MIDI channel 0 is Csound insno 1.

References evaluate_form(), fundamentalDomainByPredicate(), and csound::Score::getDuration().

Referenced by csound::LispTransformer::transform().

5.2.4.147 seqToScore()

```
void csound::seqToScore (
            cl_object & seq_,
            Score & score )
```

Translates a Common Music seq to a Silence Score.

All MIDI events in the seq are translated to Silence note on Events in the Score.

References csound::Score::append(), and fundamentalDomainByPredicate().

Referenced by csound::LispGenerator::generate(), and csound::LispTransformer::transform().

5.2.4.148 SET_CHORD_SPACE_DEBUGGING()

```
static SILENCE_PUBLIC bool csound::SET_CHORD_SPACE_DEBUGGING (
            bool enabled )  [static]
```

References CHORD_SPACE_DEBUGGING(), and SET_CHORD_SPACE_DEBUGGING().

Referenced by SET_CHORD_SPACE_DEBUGGING().

5.2.4.149 SET_SCOPED_DEBUGGING()

```
static SILENCE_PUBLIC bool csound::SET_SCOPED_DEBUGGING (
            bool enabled )  [static]
```

References SCOPED_DEBUGGING_FLAG(), and SET_SCOPED_DEBUGGING().

Referenced by SET_SCOPED_DEBUGGING().

5.2.4.150 setCorrectNegativeDurations()

```
void SILENCE_PUBLIC csound::setCorrectNegativeDurations (
            bool do_correct )
```

References csound::Event::correct_negative_durations(), and fundamentalDomainByPredicate().

5.2.4.151 slice()

```
SILENCE_PUBLIC std::vector< Event * > csound::slice (
            Score & score,
            double startTime,
            double endTime )
```

Returns a slice of the Score starting at the start time and extending up to but not including the end time.

The slice contains pointers to the Events in the Score.

References slice().

Referenced by apply(), gather(), and slice().

5.2.4.152 sort()

```
std::vector< double > csound::sort (
            const std::vector< double > & chord )
```

References chord(), and fundamentalDomainByPredicate().

Referenced by csound::Voicelead::chordToPTV(), csound::ChordScore::conformToChords(), csound::ChordScore::getDuration(), csound::ChordScore::getScale(), csound::Voicelead::I_vector(), inversions(), csound::Voicelead::ptvToChord(), csound::ChordScore::setDuration(), csound::ChordScore::setScale(), csound::Voicelead::T_vector(), csound::Voicelead::uniquePcs(), and csound::Voicelead::voicings().

5.2.4.153 split()

```
SILENCE_PUBLIC std::vector< std::string > csound::split (
            std::string string_ )  [inline]
```

References split().

Referenced by fill(), scale(), and split().

5.2.4.154 T()

```
SILENCE_PUBLIC double csound::T (
            double pitch,
            double semitones )  [inline]
```

Returns the pitch transposed by semitones, which may be any scalar.

NOTE: Does NOT return an equivalent under any requivalence relation.

References T().

Referenced by csound::HarmonyIFS::add_interpolation_point(), csound::HarmonyIFS2::add_interpolation_point(), csound::Chord::et(), csound::PITV::fromChord(), csound::HarmonyInterpolationPoint::HarmonyInterpolationPoint(), csound::HarmonyInterpolationPoint2::HarmonyInterpolationPoint2(), csound::HarmonyIFS::iterate(), csound::HarmonyIFS2::iterate(), csound::Chord::move(), csound::Chord::nrD(), csound::HarmonyIFS::point_to_note(), csound::Chord::Q(), csound::Score::setPT(), csound::Score::setPTV(), csound::Chord::T(), T(), csound::PITV::toChord(), csound::HarmonyPoint::toString(), csound::HarmonyInterpolationPoint::toString(), csound::HarmonyPoint2::toString(), csound::HarmonyInterpolationPoint2::toString(), and csound::Scale::transpose().

5.2.4.155 to_std_string()

```
std::string csound::to_std_string (
            cl_object lisp_object )
```

Translate a Lisp string to a C++ string.

References fundamentalDomainByPredicate().

Referenced by csound::LispNode::getStringFromForm().

5.2.4.156 toScore()

```
SILENCE_PUBLIC void csound::toScore (
            const Chord & chord,
            Score & score,
            double time_,
            bool voiceIsInstrument )
```

References csound::Score::append(), chord(), csound::Chord::getDuration(), csound::Chord::getInstrument(), csound::Chord::getLoudness(), csound::Chord::getPan(), csound::Chord::getPitch(), toScore(), and csound::Chord::voices().

Referenced by insert(), insert(), csound::HarmonyIFS2::iterate(), and toScore().

5.2.4.157 toString()

```
SILENCE_PUBLIC std::string csound::toString (
            const Matrix & mat )  [inline]
```

References toString().

Referenced by csound::HarmonyIFS::generate_score_attractor(), csound::HarmonyIFS2::generate_score_attractor(), csound::Chord::information_sector(), csound::Chord::initialize_sectors(), csound::HarmonyIFS::iterate(), csound::Chord::opti_domain_sec reflect_by_householder(), reflect_vector(), csound::Chord::test(), csound::PITV::toChord(), and toString().

5.2.4.158 transpose_degrees()

```
SILENCE_PUBLIC Chord csound::transpose_degrees (
            const Chord & scale,
            const Chord & original_chord,
            int transposition_degrees,
            int interval = 3 )  [inline]
```

Returns the chord, in scale order, transposed within the scale by the indicated number of scale degrees, which can be positive or negative.

The original chord may be in any order or voicing. By default, chords are generated by thirds, but they can be at any interval in scale degrees. If the original chord does not belong to the scale, an empty Chord is returned.

References chord(), CHORD_SPACE_DEBUG, csound::Chord::eOP(), csound::Chord::information(), csound::Chord::resize(), scale(), csound::Chord::toString(), transpose_degrees(), and csound::Chord::voices().

Referenced by csound::Scale::transpose_degrees(), and transpose_degrees().

5.2.4.159 unique_chords()

```
SILENCE_PUBLIC std::set< Chord > & csound::unique_chords ( )  [inline]
```

References unique_chords().

Referenced by add_chord(), and unique_chords().

5.2.4.160 unique_scales()

```
SILENCE_PUBLIC std::set< Scale > & csound::unique_scales ( )  [inline]
```

References unique_scales().

Referenced by add_scale(), csound::Scale::modulations_for_scale_types(), and unique_scales().

5.2.4.161 user_data()

```
SILENCE_PUBLIC void *& csound::user_data ( )
```

References fundamentalDomainByPredicate().

Referenced by csound::System::debug(), csound::System::error(), csound::System::getUserdata(), csound::System::inform(), csound::System::message(), csound::System::message(), csound::System::setUserdata(), and csound::System::warn().

5.2.4.162 voiceleading()

```
SILENCE_PUBLIC Chord csound::voiceleading (
            const Chord & a,
            const Chord & b )  [inline]
```

Returns the voice-leading between chords a and b, i.e.

what you have to add to a to get b, as a chord of directed intervals.

References csound::Chord::getPitch(), csound::Chord::setPitch(), voiceleading(), and csound::Chord::voices().

Referenced by parallelFifth(), csound::Chord::T_voiceleading(), voiceleading(), and voiceleadingSimpler().

5.2.4.163 voiceleadingCloser()

```
SILENCE_PUBLIC Chord csound::voiceleadingCloser (
            const Chord & source,
            const Chord & d1,
            const Chord & d2,
            bool avoidParallels = false )  [inline]
```

Returns which of the voiceleadings (source to d1, source to d2) is the closer (first smoother, then simpler), optionally avoiding parallel fifths.

References parallelFifth(), voiceleadingCloser(), voiceleadingSimpler(), and voiceleadingSmoothness().

Referenced by voiceleadingCloser(), and voiceleadingClosestRange().

5.2.4.164 voiceleadingClosestRange()

```
SILENCE_PUBLIC Chord csound::voiceleadingClosestRange (
            const Chord & source,
            const Chord & destination,
            double range,
            bool avoidParallels )  [inline]
```

Returns the voicing of the destination which has the closest voice-leading from the source within the range, optionally avoiding parallel fifths.

References csound::Chord::eOP(), csound::Chord::getPitch(), next(), OCTAVE(), csound::Chord::setPitch(), voiceleadingCloser(), voiceleadingClosestRange(), and csound::Chord::voices().

Referenced by voiceleadingClosestRange().

5.2.4.165 voiceleadingSimpler()

```
SILENCE_PUBLIC Chord csound::voiceleadingSimpler (
            const Chord & source,
            const Chord & d1,
            const Chord & d2,
            bool avoidParallels = false )  [inline]
```

Returns which of the voiceleadings (source to d1, source to d2) is the simpler (fewest moves), optionally avoiding parallel fifths.

References csound::Chord::count(), parallelFifth(), voiceleading(), and voiceleadingSimpler().

Referenced by voiceleadingCloser(), and voiceleadingSimpler().

5.2.4.166 voiceleadingSmoother()

```
SILENCE_PUBLIC Chord csound::voiceleadingSmoother (
            const Chord & source,
            const Chord & d1,
            const Chord & d2,
            bool avoidParallels = false,
            double range = OCTAVE() )  [inline]
```

Returns which of the voiceleadings (source to d1, source to d2) is the smoother (shortest moves), optionally avoiding parallel fifths.

References parallelFifth(), voiceleadingSmoother(), and voiceleadingSmoothness().

Referenced by voiceleadingSmoother().

5.2.4.167 voiceleadingSmoothness()

```
SILENCE_PUBLIC double csound::voiceleadingSmoothness (
            const Chord & a,
            const Chord & b )  [inline]
```

Returns the smoothness of the voiceleading between chords a and b by L1 norm.

References csound::Chord::getPitch(), voiceleadingSmoothness(), and csound::Chord::voices().

Referenced by voiceleadingCloser(), voiceleadingSmoother(), and voiceleadingSmoothness().

5.2.5 Variable Documentation

5.2.5.1 cForPForDivisionsPerOctave

```
std::map<size_t, std::map<double, double> > csound::cForPForDivisionsPerOctave
```

Referenced by csound::Voicelead::initializePrimeChordsForDivisionsPerOctave(), and csound::Voicelead::pToC().

5.2.5.2 Chord

```
class SILENCE_PUBLIC csound::Chord
```

Referenced by main().

5.2.5.3 ChordScore

```
class SILENCE_PUBLIC csound::ChordScore
```

5.2.5.4 debug

```
int csound::debug = 1  [static]
```

Referenced by csound::Voicelead::addOctave(), csound::Voicelead::areParallel(), csound::Voicelead::pcs(), csound::Voicelead::recursiveV
csound::Voicelead::rotations(), and csound::Voicelead::voicelead().

5.2.5.5 initialized__

```
bool csound::initialized__ = Conversions::initialize()  [static]
```

5.2.5.6 mersenne_twister

```
std::mt19937 csound::mersenne_twister  [static]
```

Referenced by hyperplane_equation_from_random_inversion_flat().

5.2.5.7 MidiFile

```
class SILENCE_PUBLIC csound::MidiFile
```

5.2.5.8 namesForEquivalenceRelations

```
const char* csound::namesForEquivalenceRelations[]  [static]
```

Initial value:

```
= {
    "r",
    "R",
    "P",
    "T",
    "Tg",
    "I",
    "RP",
    "RT",




    "RPT",
    "RPTg",
    "RPI",
    "RTI",
    "RTgI",
    "RPTI",
    "RPTgI",
}
```

Referenced by fundamentalDomainByPredicate(), and fundamentalDomainByTransformation().

5.2.5.9 pForCForDivisionsPerOctave

```
std::map<size_t, std::map<double, double> > csound::pForCForDivisionsPerOctave
```

Referenced by csound::Voicelead::initializePrimeChordsForDivisionsPerOctave().

5.2.5.10 pForPrimeChordsForDivisionsPerOctave

```
std::map<size_t, std::map< std::vector<double>, double> > csound::pForPrimeChordsForDivisions↩
PerOctave
```

Referenced by csound::Voicelead::cToP(), and csound::Voicelead::initializePrimeChordsForDivisionsPerOctave().

5.2.5.11 PITV

```
class SILENCE_PUBLIC csound::PITV
```

5.2.5.12 primeChordsForDivisionsPerOctave

```
std::map<size_t, std::vector< std::vector<double> > > csound::primeChordsForDivisionsPerOctave
```

Referenced by csound::Voicelead::initializePrimeChordsForDivisionsPerOctave(), csound::Voicelead::pToC(), and csound::Voicelead::pToPrimeChord().

5.2.5.13 Py_Finalize_

```
void(* csound::Py_Finalize_)(void) (
            void  )
```

Referenced by csound::Shell::open().

5.2.5.14 Py_Initialize_

```
void(* csound::Py_Initialize_)(void) (
            void  )
```

Referenced by csound::Shell::open().

5.2.5.15 PyErr_Print_

```
void(* csound::PyErr_Print_)(void) (
            void  )
```

Referenced by csound::Shell::open(), and csound::Shell::runScript().

5.2.5.16 PyImport_ImportModule_

```
PyObject_ *(* csound::PyImport_ImportModule_)(char *) (
            char *  )
```

Referenced by csound::Shell::open().

5.2.5.17 PyLong_AsLong_

```
long(* csound::PyLong_AsLong_)(PyObject_ *) (
            PyObject_ *  )
```

Referenced by csound::Shell::open().

5.2.5.18 PyObject_CallMethod_

```
PyObject_ *(* csound::PyObject_CallMethod_)(PyObject_ *, char *, char *,...)  (
            PyObject_ * ,
            char * ,
            char * ,
             ...  )
```

Referenced by csound::Shell::open().

5.2.5.19 PyObject_GetAttrString_

```
PyObject_ *(* csound::PyObject_GetAttrString_)(PyObject_ *, char *) (
            PyObject_ * ,
            char *  )
```

Referenced by csound::Shell::open().

5.2.5.20 PyRun_SimpleFileEx_

```
int(* csound::PyRun_SimpleFileEx_)(FILE *, const char *, int) (
            FILE * ,
            const char * ,
            int  )
```

Referenced by csound::Shell::open().

5.2.5.21 PyRun_SimpleString_

```
int(* csound::PyRun_SimpleString_)(const char *) (
            const char *  )
```

Referenced by csound::Shell::main(), csound::Shell::open(), and csound::Shell::runScript().

5.2.5.22 PySys_SetArgv_

```
void(* csound::PySys_SetArgv_)(int, char **) (
            int ,
            char **  )
```

Referenced by csound::Shell::main(), and csound::Shell::open().

5.2.5.23 Scale

```
class SILENCE_PUBLIC csound::Scale
```

Referenced by csound::ChordLindenmayer::scaleOperation().

5.2.5.24 twister

```
std::mt19937_64 csound::twister  [static]
```

Referenced by csound::ChordLindenmayer::arithmetic(), and csound::ChordLindenmayer::scoreOperation().

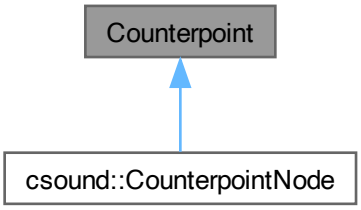
Chapter 6

Data Structure Documentation

6.1 Counterpoint Class Reference

`#include <Counterpoint.hpp>`

Inheritance diagram for Counterpoint:



Public Types

- enum { MostNotes_ = 128 , MostVoices_ = 12 }
- enum {
  Unison = 0 , MinorSecond = 1 , MajorSecond = 2 , MinorThird = 3 ,
  MajorThird = 4 , Fourth = 5 , Tritone = 6 , Fifth = 7 ,
  MinorSixth = 8 , MajorSixth = 9 , MinorSeventh = 10 , MajorSeventh = 11 ,
  Octave = 12 }
- enum {
  Aeolian = 1 , Dorian = 2 , Phrygian = 3 , Lydian = 4 ,
  Mixolydian = 5 , Ionian = 6 , Locrian = 7 }
- enum { DirectMotion = 1 , ContraryMotion = 2 , ObliqueMotion = 3 , NoMotion = 4 }

- enum {
  WholeNote = 8 , HalfNote = 4 , DottedHalfNote = 6 , QuarterNote = 2 ,
  DottedQuarterNote = 3 , EighthNote = 1 }
- enum {
  One = 0 , Two = 2 , Three = 3 , Four = 4 ,
  Five = 5 , Six = 6 , Eight = 8 }
- enum { infinity = 1000000 , Bad = 100 , RealBad = 200 }
- enum { INTERVALS_WITH_BASS_SIZE = 8 }
- enum { NumFields = 16 , Field = (MostVoices_+1) , EndF = (Field∗NumFields) }

Public Member Functions

- int ABS (int i)
- void AddInterval (int n)
- int ADissonance (int Interval, int Cn, int Cp, int v, int Species)
- int AnOctave (int Interval)
- void AnySpecies (int OurMode, int ∗StartPitches, int CurV, int CantusFirmusLength, int Species)
- void ARRBLT (int ∗dest, int ∗source, int num)
- int ASeventh (int Interval)
- int ASkip (int Interval)
- int AStep (int Interval)
- int ATenth (int Interval)
- int AThird (int Interval)
- int BadMelody (int Intv)
- int Bass (int Cn, int v)
- int Beat8 (int n)
- void BestFitFirst (int CurTime, int CurrentPenalty, int NumParts, int Species, int BrLim)
- int Cantus (int n, int v)
- int Check (int Cn, int Cp, int v, int NumParts, int Species, int CurLim)
- void CleanRhy ()
- virtual void clear ()
- int ConsecutiveSkipsInSameDirection (int Pitch1, int Pitch2, int Pitch3)
- Counterpoint ()
- void counterpoint (int OurMode, int ∗StartPitches, int CurV, int CantusFirmusLength, int Species, int ∗cantus)
- int CurRhy (int n)
- int DirectMotionToPerfectConsonance (int Pitch1, int Pitch2, int Pitch3, int Pitch4)
- int Doubled (int Pitch, int Cn, int v)
- int DownBeat (int n, int v)
- int ExtremeRange (int Pitch)
- void fillCantus (int c0, int c1, int c2, int c3, int c4, int c5, int c6, int c7, int c8, int c9, int c10, int c11, int c12, int c13, int c14)
- void FillRhyPat ()
- int FirstNote (int n, int v)
- int GoodRhy ()
- virtual void initialize (int mostnotes, int mostvoices)
- int InMode (int Pitch, int Mode)
- int LastNote (int n, int v)
- int Look (int CurPen, int CurVoice, int NumParts, int Species, int Lim, int ∗Pens, int ∗Is, int ∗CurNotes)
- int MAX (int a, int b)
- void message (const char ∗format, va_list valist)

- void message (const char *format,...)
- int MIN (int a, int b)
- int MotionType (int Pitch1, int Pitch2, int Pitch3, int Pitch4)
- int NextToLastNote (int n, int v)
- int Other (int Cn, int v, int v1)
- int OtherVoiceCheck (int Cn, int Cp, int v, int NumParts, int Species, int CurLim)
- int OutOfRange (int Pitch)
- int PitchRepeats (int Cn, int Cp, int v)
- float RANDOM (float amp)
- int SaveIndx (int indx, int *Sp)
- void SaveResults (int CurrentPenalty, int Penalty, int v1, int Species)
- void SetUs (int n, int p, int v)
- int Size (int MelInt)
- int SpecialSpeciesCheck (int Cn, int Cp, int v, int Other0, int Other1, int Other2, int NumParts, int Species, int MelInt, int Interval, int ActInt, int LastIntClass, int Pitch, int LastMelInt, int CurLim)
- void toCsoundScore (std::string filename, double secondsPerPulse)
- int TooMuchOfInterval (int Cn, int Cp, int v)
- int TotalRange (int Cn, int Cp, int v)
- int UpBeat (int n, int v)
- int Us (int n, int v)
- void UsedRhy (int n)
- int VIndex (int Time, int VNum)
- void winners (int v1, int *data, int *best, int *best1, int *best2, int *durs)
- virtual ~Counterpoint ()

Data Fields

- int AllDone
- int AllVoicesSkipPenalty
- int AscendingSixthPenalty
- int AugmentedIntervalPenalty
- int BadCadencePenalty
- int BadMelodyPenalty
- int BasePitch
- Eigen::MatrixXi BestFit
- Eigen::MatrixXi BestFit1
- Eigen::MatrixXi BestFit2
- int BestFitPenalty
- int Branches
- int CompoundPenalty
- int CrossAboveCantusPenalty
- int CrossBelowBassPenalty
- Eigen::MatrixXi Ctrpt
- int DirectMotionPenalty
- int DirectPerfectOnDownbeatPenalty
- int DirectToFifthPenalty
- int DirectToOctavePenalty
- int DirectToTritonePenalty
- int DissonanceNotFillingThirdPenalty
- int DissonancePenalty

- int DoubledFifthPenalty
- int DoubledLeadingTonePenalty
- int DoubledSixthPenalty
- int DownBeatUnisonPenalty
- Eigen::MatrixXi Dur
- int EighthJumpPenalty
- int EndOnPerfectPenalty
- int ExtremeRangePenalty
- int FifthFollowedBySameDirectionPenalty
- int FifthPrecededBySameDirectionPenalty
- int Fits [3]
- int FourRepeatedNotesPenalty
- int HalfUntiedPenalty
- int HighestSemitone
- int InnerVoicesInDirectToPerfectPenalty
- int InnerVoicesInDirectToTritonePenalty
- int IntervalsWithBass [INTERVALS_WITH_BASS_SIZE]
- int LeapAtCadencePenalty
- int LesserLigaturePenalty
- int LowerNeighborPenalty
- int LowestSemitone
- int LydianCadentialTritonePenalty
- int MaxPenalty
- int MelodicBoredomPenalty
- int MelodicTritonePenalty
- int Mode
- int MostNotes
- int MostVoices
- int NoLeadingTonePenalty
- int NoMotionAgainstOctavePenalty
- int NotaCambiataPenalty
- int NotaLigaturePenalty
- int NotBestCadencePenalty
- int NotContraryToOthersPenalty
- int NoTimeForaLigaturePenalty
- int NotTriadPenalty
- int OctaveLeapPenalty
- Eigen::MatrixXi Onset
- int OutOfModePenalty
- int OutOfRangePenalty
- int OverOctavePenalty
- int OverTwelfthPenalty
- int ParallelFifthPenalty
- int ParallelUnisonPenalty
- float PenaltyRatio
- int PerfectConsonancePenalty
- long randx
- int RepeatedPitchPenalty
- int RepetitionOnUpbeatPenalty
- Eigen::VectorXi RhyNotes
- Eigen::MatrixXi RhyPat

- int SixFiveChordPenalty
- int SixthFollowedBySameDirectionPenalty
- int SixthLeapPenalty
- int SixthPrecededBySameDirectionPenalty
- int SkipFollowedBySameDirectionPenalty
- int SkipFromUnisonPenalty
- int SkipPrecededBySameDirectionPenalty
- int SkipTo8vePenalty
- int SkipToDownBeatPenalty
- int TenthToOctavePenalty
- int ThirdDoubledPenalty
- int ThreeRepeatedNotesPenalty
- int ThreeSkipsPenalty
- Eigen::VectorXi TotalNotes
- int TotalTime
- int TripledBassPenalty
- int TwoRepeatedNotesPenalty
- int TwoSkipsNotInTriadPenalty
- int TwoSkipsPenalty
- std::normal_distribution uniform_real_generator
- int UnisonDownbeatPenalty
- int UnisonOnBeat4Penalty
- int UnisonPenalty
- int UnisonUpbeatPenalty
- int UnpreparedSixFivePenalty
- int UnresolvedLeadingTonePenalty
- int UnresolvedLigaturePenalty
- int UnresolvedSixFivePenalty
- int UpperNeighborPenalty
- int UpperVoicesTooFarApartPenalty
- Eigen::VectorXi vbs
- int VerticalTritonePenalty

Static Public Attributes

- static int _Aeolian [12] = {1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0}
- static int _Dorian [12] = {1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0}
- static int _Ionian [12] = {1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1}
- static int _Locrian [12] = {1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0}
- static int _Lydian [12] = {1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1}
- static int _Mixolydian [12] = {1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0}
- static int _Phrygian [12] = {1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0}
- static int BadMelodyInterval [13] = {0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0}
- static int Dissonance [13] = {0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0}
- static int ImperfectConsonance [13] = {0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0}
- static int Indx [17] = {0, 1, -1, 2, -2, 3, -3, 0, 4, -4, 5, 7, -5, 8, 12, -7,-12}
- static std::mt19937 mersenneTwister
- static int PerfectConsonance [13] = {1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1}

6.1.1 Member Enumeration Documentation

6.1.1.1 anonymous enum

```
anonymous enum
```

Enumerator

Most↩ Notes_	
Most↩ Voices_	

6.1.1.2 anonymous enum

```
anonymous enum
```

Enumerator

Unison	
MinorSecond	
MajorSecond	
MinorThird	
MajorThird	
Fourth	
Tritone	
Fifth	
MinorSixth	
MajorSixth	
MinorSeventh	
MajorSeventh	
Octave	

6.1.1.3 anonymous enum

```
anonymous enum
```

Enumerator

Aeolian	
Dorian	
Phrygian	
Lydian	
Mixolydian	
Ionian	
Locrian	

6.1.1.4 anonymous enum

```
anonymous enum
```

Enumerator

DirectMotion	
ContraryMotion	
ObliqueMotion	
NoMotion	

6.1.1.5 anonymous enum

```
anonymous enum
```

Enumerator

WholeNote	
HalfNote	
DottedHalfNote	
QuarterNote	
DottedQuarterNote	
EighthNote	

6.1.1.6 anonymous enum

```
anonymous enum
```

Enumerator

One	
Two	
Three	
Four	
Five	
Six	
Eight	

6.1.1.7 anonymous enum

```
anonymous enum
```

Enumerator

infinity	
Bad	
RealBad	

6.1.1.8 anonymous enum

```
anonymous enum
```

Enumerator

INTERVALS_WITH_BASS_SIZE	

6.1.1.9 anonymous enum

```
anonymous enum
```

Enumerator

NumFields	
Field	
EndF	

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Counterpoint()

```
Counterpoint::Counterpoint ( )
```

References AllVoicesSkipPenalty, AscendingSixthPenalty, AugmentedIntervalPenalty, Bad, BadCadencePenalty, BadMelodyPenalty, CompoundPenalty, CrossAboveCantusPenalty, CrossBelowBassPenalty, DirectMotionPenalty, DirectPerfectOnDownbeatPenalty, DirectToFifthPenalty, DirectToOctavePenalty, DirectToTritonePenalty, DissonanceNotFillingThirdPenalty, DissonancePenalty, DoubledFifthPenalty, DoubledLeadingTonePenalty, DoubledSixthPenalty, DownBeatUnisonPenalty, EighthJumpPenalty, EndOnPerfectPenalty, ExtremeRangePenalty, FifthFollowedBySameDirectionPenalty, FifthPrecededBySameDirectio FourRepeatedNotesPenalty, HalfUntiedPenalty, infinity, initialize(), InnerVoicesInDirectToPerfectPenalty, InnerVoicesInDirectToTritonePen LeapAtCadencePenalty, LesserLigaturePenalty, LowerNeighborPenalty, LydianCadentialTritonePenalty, MelodicBoredomPenalty, MelodicTritonePenalty, MostNotes_, MostVoices_, NoLeadingTonePenalty, NoMotionAgainstOctavePenalty, NotaCambiataPenalty, NotaLigaturePenalty, NotBestCadencePenalty, NotContraryToOthersPenalty, NoTimeForaLigaturePenalty, NotTriadPenalty, OctaveLeapPenalty, OutOfModePenalty, OutOfRangePenalty, OverOctavePenalty, OverTwelfthPenalty, ParallelFifthPenalty, ParallelUnisonPenalty, PerfectConsonancePenalty, RealBad, RepeatedPitchPenalty, RepetitionOnUpbeatPenalty, SixFiveChordPenalty, SixthFollowedBySameDirectionPenalty, SixthLeapPenalty, SixthPrecededBySameDirectionPenalty, SkipFollowedBySameDirectionPenalty, SkipFromUnisonPenalty, SkipPrecededBySameDirectionPenalty, SkipTo8vePenalty,

SkipToDownBeatPenalty, TenthToOctavePenalty, ThirdDoubledPenalty, ThreeRepeatedNotesPenalty, ThreeSkipsPenalty, TripledBassPenalty, TwoRepeatedNotesPenalty, TwoSkipsNotInTriadPenalty, TwoSkipsPenalty, UnisonDownbeatPenalty, UnisonOnBeat4Penalty, UnisonPenalty, UnisonUpbeatPenalty, UnpreparedSixFivePenalty, UnresolvedLeadingTonePenalty, UnresolvedLigaturePenalty, UnresolvedSixFivePenalty, UpperNeighborPenalty, UpperVoicesTooFarApartPenalty, and VerticalTritonePenalty.

6.1.2.2 ~Counterpoint()

```
Counterpoint::~Counterpoint ( )  [virtual]
```

6.1.3 Member Function Documentation

6.1.3.1 ABS()

```
int Counterpoint::ABS (
            int i )
```

Referenced by AnOctave(), ASkip(), AStep(), ATenth(), BadMelody(), Check(), DirectMotionToPerfectConsonance(), OtherVoiceCheck(), SaveResults(), Size(), and SpecialSpeciesCheck().

6.1.3.2 AddInterval()

```
void Counterpoint::AddInterval (
            int n )
```

References IntervalsWithBass.

Referenced by OtherVoiceCheck().

6.1.3.3 ADissonance()

```
int Counterpoint::ADissonance (
            int Interval,
            int Cn,
            int Cp,
            int v,
            int Species )
```

References AStep(), Beat8(), Dissonance, DownBeat(), Dur, FirstNote(), LastNote(), Onset, UpBeat(), Us(), and WholeNote.

Referenced by Check().

6.1.3.4 AnOctave()

```
int Counterpoint::AnOctave (
            int Interval )
```

References ABS(), and Unison.

Referenced by Check().

6.1.3.5 AnySpecies()

```
void Counterpoint::AnySpecies (
            int OurMode,
            int * StartPitches,
            int CurV,
            int CantusFirmusLength,
            int Species )
```

References AllDone, BasePitch, BestFit, BestFitFirst(), BestFitPenalty, Branches, CleanRhy(), Ctrpt, Dur, GoodRhy(), HalfNote, infinity, MaxPenalty, Mode, MostNotes, MostVoices, Onset, PenaltyRatio, QuarterNote, RealBad, RhyNotes, RhyPat, TotalNotes, TotalTime, UsedRhy(), and WholeNote.

Referenced by counterpoint(), and main().

6.1.3.6 ARRBLT()

```
void Counterpoint::ARRBLT (
            int * dest,
            int * source,
            int num )
```

Referenced by SaveIndx().

6.1.3.7 ASeventh()

```
int Counterpoint::ASeventh (
            int Interval )
```

References MajorSeventh, and MinorSeventh.

Referenced by SpecialSpeciesCheck().

6.1.3.8 ASkip()

```
int Counterpoint::ASkip (
            int Interval )
```

References ABS(), and MajorSecond.

Referenced by Check(), ConsecutiveSkipsInSameDirection(), OtherVoiceCheck(), SaveResults(), and SpecialSpeciesCheck().

6.1.3.9 AStep()

```
int Counterpoint::AStep (
            int Interval )
```

References ABS(), MajorSecond, and MinorSecond.

Referenced by ADissonance(), Check(), and SpecialSpeciesCheck().

6.1.3.10 ATenth()

```
int Counterpoint::ATenth (
            int Interval )
```

References ABS(), and AThird().

Referenced by Check().

6.1.3.11 AThird()

```
int Counterpoint::AThird (
            int Interval )
```

References MajorThird, and MinorThird.

Referenced by ATenth(), and SpecialSpeciesCheck().

6.1.3.12 BadMelody()

```
int Counterpoint::BadMelody (
            int Intv )
```

References ABS(), BadMelodyInterval, MinorSixth, and Octave.

Referenced by Check().

6.1.3.13 Bass()

```
int Counterpoint::Bass (
            int Cn,
            int v )
```

References Cantus(), MIN(), and Other().

Referenced by Check(), OtherVoiceCheck(), and SpecialSpeciesCheck().

6.1.3.14 Beat8()

```
int Counterpoint::Beat8 (
            int n )
```

Referenced by ADissonance(), DownBeat(), and SpecialSpeciesCheck().

6.1.3.15 BestFitFirst()

```
void Counterpoint::BestFitFirst (
            int CurTime,
            int CurrentPenalty,
            int NumParts,
            int Species,
            int BrLim )
```

References AllDone, BestFitFirst(), BestFitPenalty, Branches, EndF, Field, Indx, infinity, Look(), MaxPenalty, MIN(), MostVoices, NumFields, Onset, PenaltyRatio, SaveResults(), SetUs(), TotalTime, Us(), and VIndex().

Referenced by AnySpecies(), and BestFitFirst().

6.1.3.16 Cantus()

```
int Counterpoint::Cantus (
            int n,
            int v )
```

References Ctrpt, and Onset.

Referenced by Bass(), and Check().

6.1.3.17 Check()

```
int Counterpoint::Check (
            int Cn,
            int Cp,
            int v,
            int NumParts,
            int Species,
            int CurLim )
```

References ABS(), ADissonance(), Aeolian, AnOctave(), ASkip(), AStep(), ATenth(), BadCadencePenalty, BadMelody(), BadMelodyPenalty, BasePitch, Bass(), Cantus(), CompoundPenalty, ConsecutiveSkipsInSameDirection(), CrossAboveCantusPenalty, DirectMotion, DirectMotionPenalty, DirectMotionToPerfectConsonance(), DirectPerfectOnDownbeatPenalty, DirectToFifthPenalty, DirectToOctavePenalty, Dissonance, DissonanceNotFillingThirdPenalty, DissonancePenalty, Doubled(), DoubledLeadingTonePenalty, DownBeat(), EndOnPerfectPenalty, ExtremeRange(), ExtremeRangePenalty, Fifth, FifthFollowedBySameDirectionPenalty, FifthPrecededBySameDirectionPenalty, FirstNote(), FourRepeatedNotesPenalty, Fourth, InMode(), LastNote(), LeapAtCadencePenalty, LowerNeighborPenalty, Lydian, LydianCadentialTritonePenalty, MajorSixth, MajorThird, MAX(), MelodicBoredomPenalty, MelodicTritonePenalty, MinorSecond, MinorSixth, Mode, MotionType(), NextToLastNote(),

NoLeadingTonePenalty, NoMotionAgainstOctavePenalty, Octave, OctaveLeapPenalty, OtherVoiceCheck(), OutOfModePenalty, OutOfRange(), OutOfRangePenalty, OverOctavePenalty, OverTwelfthPenalty, ParallelFifthPenalty, ParallelUnisonPenalty, PerfectConsonance, PerfectConsonancePenalty, Phrygian, PitchRepeats(), RepetitionOnUpbeatPenalty, SixthFollowedBySameDirectionP SixthLeapPenalty, SixthPrecededBySameDirectionPenalty, SkipFollowedBySameDirectionPenalty, SkipFromUnisonPenalty, SkipPrecededBySameDirectionPenalty, SkipTo8vePenalty, SpecialSpeciesCheck(), TenthToOctavePenalty, ThreeRepeatedNotesPenalty, ThreeSkipsPenalty, TooMuchOfInterval(), TotalNotes, TotalRange(), Tritone, TwoRepeatedNotesPenalty, TwoSkipsNotInTriadPenalty, TwoSkipsPenalty, Unison, UnisonDownbeatPenalty, UnisonPenalty, UnresolvedLeadingTonePenalty, UpBeat(), UpperNeighborPenalty, Us(), and VerticalTritonePenalty.

Referenced by Look().

6.1.3.18 CleanRhy()

```
void Counterpoint::CleanRhy ( )
```

References RhyPat.

Referenced by AnySpecies().

6.1.3.19 clear()

```
void Counterpoint::clear ( )  [virtual]
```

References BestFit, BestFit1, BestFit2, Ctrpt, Dur, MostVoices, Onset, RhyNotes, RhyPat, TotalNotes, and vbs.

Referenced by csound::CounterpointNode::transform().

6.1.3.20 ConsecutiveSkipsInSameDirection()

```
int Counterpoint::ConsecutiveSkipsInSameDirection (
            int Pitch1,
            int Pitch2,
            int Pitch3 )
```

References ASkip().

Referenced by Check().

6.1.3.21 counterpoint()

```
void Counterpoint::counterpoint (
            int OurMode,
            int * StartPitches,
            int CurV,
            int CantusFirmusLength,
            int Species,
            int * cantus )
```

References AnySpecies(), Ctrpt, Fits, initialize(), and vbs.

Referenced by main(), and csound::CounterpointNode::transform().

6.1.3.22 CurRhy()

```
int Counterpoint::CurRhy (
            int n )
```

References RhyPat.

Referenced by GoodRhy().

6.1.3.23 DirectMotionToPerfectConsonance()

```
int Counterpoint::DirectMotionToPerfectConsonance (
            int Pitch1,
            int Pitch2,
            int Pitch3,
            int Pitch4 )
```

References ABS(), DirectMotion, MotionType(), and PerfectConsonance.

Referenced by Check(), and OtherVoiceCheck().

6.1.3.24 Doubled()

```
int Counterpoint::Doubled (
            int Pitch,
            int Cn,
            int v )
```

References Other().

Referenced by Check().

6.1.3.25 DownBeat()

```
int Counterpoint::DownBeat (
            int n,
            int v )
```

References Beat8(), and Onset.

Referenced by ADissonance(), Check(), SpecialSpeciesCheck(), and UpBeat().

6.1.3.26 ExtremeRange()

```
int Counterpoint::ExtremeRange (
            int Pitch )
```

References HighestSemitone, and LowestSemitone.

Referenced by Check().

6.1.3.27 fillCantus()

```
void Counterpoint::fillCantus (
            int c0,
            int c1,
            int c2,
            int c3,
            int c4,
            int c5,
            int c6,
            int c7,
            int c8,
            int c9,
            int c10,
            int c11,
            int c12,
            int c13,
            int c14 )
```

References Ctrpt.

Referenced by main().

6.1.3.28 FillRhyPat()

```
void Counterpoint::FillRhyPat ( )
```

References EighthNote, HalfNote, QuarterNote, RhyNotes, RhyPat, and WholeNote.

Referenced by csound::CounterpointNode::CounterpointNode(), and main().

6.1.3.29 FirstNote()

```
int Counterpoint::FirstNote (
            int n,
            int v )
```

Referenced by ADissonance(), and Check().

6.1.3.30 GoodRhy()

```
int Counterpoint::GoodRhy ( )
```

References CurRhy(), MAX(), MIN(), and RANDOM().

Referenced by AnySpecies().

6.1.3.31 initialize()

```
void Counterpoint::initialize (
            int mostnotes,
            int mostvoices )  [virtual]
```

References BestFit, BestFit1, BestFit2, Ctrpt, Dur, MostNotes, MostVoices, Onset, randx, RhyNotes, RhyPat, TotalNotes, and vbs.

Referenced by Counterpoint(), and counterpoint().

6.1.3.32 InMode()

```
int Counterpoint::InMode (
            int Pitch,
            int Mode )
```

References _Aeolian, _Dorian, _Ionian, _Locrian, _Lydian, _Mixolydian, _Phrygian, Aeolian, Dorian, Ionian, Locrian, Lydian, Mixolydian, Mode, and Phrygian.

Referenced by Check(), OtherVoiceCheck(), and SaveResults().

6.1.3.33 LastNote()

```
int Counterpoint::LastNote (
            int n,
            int v )
```

References TotalNotes.

Referenced by ADissonance(), Check(), and OtherVoiceCheck().

6.1.3.34 Look()

```
int Counterpoint::Look (
            int CurPen,
            int CurVoice,
            int NumParts,
            int Species,
            int Lim,
            int * Pens,
            int * Is,
            int * CurNotes )
```

References Check(), Ctrpt, Indx, Look(), MIN(), SaveIndx(), and SetUs().

Referenced by BestFitFirst(), and Look().

6.1.3.35 MAX()

```
int Counterpoint::MAX (
            int a,
            int b )
```

Referenced by Check(), GoodRhy(), and TotalRange().

6.1.3.36 message() [1/2]

```
void Counterpoint::message (
            const char * format,
            va_list valist )
```

References csound::System::message().

6.1.3.37 message() [2/2]

```
void Counterpoint::message (
            const char * format,
             ...  )
```

References message().

Referenced by message(), and SaveResults().

6.1.3.38 MIN()

```
int Counterpoint::MIN (
            int a,
            int b )
```

Referenced by Bass(), BestFitFirst(), GoodRhy(), Look(), SaveResults(), and TotalRange().

6.1.3.39 MotionType()

```
int Counterpoint::MotionType (
            int Pitch1,
            int Pitch2,
            int Pitch3,
            int Pitch4 )
```

References ContraryMotion, DirectMotion, NoMotion, and ObliqueMotion.

Referenced by Check(), DirectMotionToPerfectConsonance(), and OtherVoiceCheck().

6.1.3.40 NextToLastNote()

```
int Counterpoint::NextToLastNote (
            int n,
            int v )
```

References TotalNotes.

Referenced by Check(), and SpecialSpeciesCheck().

6.1.3.41 Other()

```
int Counterpoint::Other (
            int Cn,
            int v,
            int v1 )
```

References Ctrpt, Onset, and VIndex().

Referenced by Bass(), Doubled(), OtherVoiceCheck(), and SaveResults().

6.1.3.42 OtherVoiceCheck()

```
int Counterpoint::OtherVoiceCheck (
            int Cn,
            int Cp,
            int v,
            int NumParts,
            int Species,
            int CurLim )
```

References ABS(), AddInterval(), AllVoicesSkipPenalty, ASkip(), AugmentedIntervalPenalty, Bass(), ContraryMotion, CrossBelowBassPenalty, DirectMotion, DirectMotionToPerfectConsonance(), Dissonance, DoubledFifthPenalty, DoubledLeadingTonePenalty, DoubledSixthPenalty, Fifth, Fourth, InMode(), InnerVoicesInDirectToPerfectPenalty, InnerVoicesInDirectToTritonePenalty, INTERVALS_WITH_BASS_SIZE, IntervalsWithBass, LastNote(), MajorThird, Mode, MotionType(), NotContraryToOthersPenalty, NotTriadPenalty, Octave, Other(), ParallelFifthPenalty, ParallelUnisonPenalty, SixFiveChordPenalty, ThirdDoubledPenalty, TripledBassPenalty, Tritone, Unison, UnisonPenalty, UnpreparedSixFivePenalty, UnresolvedSixFivePenalty, UpperVoicesTooFarApartPenalty, Us(), and VerticalTritonePenalty.

Referenced by Check().

6.1.3.43 OutOfRange()

```
int Counterpoint::OutOfRange (
            int Pitch )
```

References HighestSemitone, and LowestSemitone.

Referenced by Check().

6.1.3.44 PitchRepeats()

```
int Counterpoint::PitchRepeats (
            int Cn,
            int Cp,
            int v )
```

References Us().

Referenced by Check().

6.1.3.45 RANDOM()

```
float Counterpoint::RANDOM (
            float amp )
```

References mersenneTwister, and uniform_real_generator.

Referenced by GoodRhy().

6.1.3.46 SaveIndx()

```
int Counterpoint::SaveIndx (
            int indx,
            int * Sp )
```

References ARRBLT(), EndF, and Field.

Referenced by Look().

6.1.3.47 SaveResults()

```
void Counterpoint::SaveResults (
            int CurrentPenalty,
            int Penalty,
            int v1,
            int Species )
```

References ABS(), ASkip(), BasePitch, BestFit, BestFit1, BestFit2, BestFitPenalty, Ctrpt, Fifth, Fits, Fourth, InMode(), MaxPenalty, message(), MIN(), MinorSecond, MinorThird, Mode, Octave, Other(), PenaltyRatio, SetUs(), TotalNotes, Unison, and Us().

Referenced by BestFitFirst().

6.1.3.48 SetUs()

```
void Counterpoint::SetUs (
            int n,
            int p,
            int v )
```

References Ctrpt.

Referenced by BestFitFirst(), Look(), and SaveResults().

6.1.3.49 Size()

```
int Counterpoint::Size (
            int MelInt )
```

References ABS(), Eight, Fifth, Five, Four, Fourth, MajorSecond, MajorThird, MinorSecond, MinorSixth, MinorThird, Octave, One, Six, Three, Two, and Unison.

Referenced by TooMuchOfInterval().

6.1.3.50 SpecialSpeciesCheck()

```
int Counterpoint::SpecialSpeciesCheck (
            int Cn,
            int Cp,
            int v,
            int Other0,
            int Other1,
            int Other2,
            int NumParts,
            int Species,
            int MelInt,
            int Interval,
            int ActInt,
            int LastIntClass,
            int Pitch,
            int LastMelInt,
            int CurLim )
```

References ABS(), ASeventh(), ASkip(), AStep(), AThird(), BadCadencePenalty, Bass(), Beat8(), Dissonance, DissonancePenalty, DownBeat(), DownBeatUnisonPenalty, Dur, EighthJumpPenalty, EighthNote, Fifth, Fourth, HalfNote, HalfUntiedPenalty, LesserLigaturePenalty, MajorSecond, MajorThird, MinorSecond, MinorSixth, Mode, NextToLastNote(), NotaCambiataPenalty, NotaLigaturePenalty, NoTimeForaLigaturePenalty, Onset, Phrygian, QuarterNote, SkipToDownBeatPenalty, Tritone, Unison, UnisonOnBeat4Penalty, UnisonUpbeatPenalty, UnresolvedLigaturePenalty, UpBeat(), and Us().

Referenced by Check().

6.1.3.51 toCsoundScore()

```
void Counterpoint::toCsoundScore (
            std::string filename,
            double secondsPerPulse )
```

References Ctrpt, Dur, csound::System::inform(), Onset, and TotalNotes.

Referenced by main().

6.1.3.52 TooMuchOfInterval()

```
int Counterpoint::TooMuchOfInterval (
            int Cn,
            int Cp,
            int v )
```

References Ctrpt, and Size().

Referenced by Check().

6.1.3.53 TotalRange()

```
int Counterpoint::TotalRange (
            int Cn,
            int Cp,
            int v )
```

References MAX(), MIN(), and Us().

Referenced by Check().

6.1.3.54 UpBeat()

```
int Counterpoint::UpBeat (
            int n,
            int v )
```

References DownBeat().

Referenced by ADissonance(), Check(), and SpecialSpeciesCheck().

6.1.3.55 Us()

```
int Counterpoint::Us (
            int n,
            int v )
```

References Ctrpt.

Referenced by ADissonance(), BestFitFirst(), Check(), OtherVoiceCheck(), PitchRepeats(), SaveResults(), SpecialSpeciesCheck(), and TotalRange().

6.1.3.56 UsedRhy()

```
void Counterpoint::UsedRhy (
            int n )
```

References RhyPat.

Referenced by AnySpecies().

6.1.3.57 VIndex()

```
int Counterpoint::VIndex (
            int Time,
            int VNum )
```

References Dur, Onset, and TotalNotes.

Referenced by BestFitFirst(), and Other().

6.1.3.58 winners()

```
void Counterpoint::winners (
            int v1,
            int * data,
            int * best,
            int * best1,
            int * best2,
            int * durs )
```

References BestFit, BestFit1, BestFit2, Dur, Fits, MostNotes, and TotalNotes.

6.1.4 Field Documentation

6.1.4.1 _Aeolian

```
int Counterpoint::_Aeolian = {1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0}  [static]
```

Referenced by InMode().

6.1.4.2 _Dorian

```
int Counterpoint::_Dorian = {1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0}  [static]
```

Referenced by InMode().

6.1.4.3 _Ionian

```
int Counterpoint::_Ionian = {1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1}  [static]
```

Referenced by InMode().

6.1.4.4 _Locrian

```
int Counterpoint::_Locrian = {1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0}  [static]
```

Referenced by InMode().

6.1.4.5 _Lydian

```
int Counterpoint::_Lydian = {1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1}  [static]
```

Referenced by InMode().

6.1.4.6 _Mixolydian

```
int Counterpoint::_Mixolydian = {1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0}  [static]
```

Referenced by InMode().

6.1.4.7 _Phrygian

```
int Counterpoint::_Phrygian = {1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0}  [static]
```

Referenced by InMode().

6.1.4.8 AllDone

```
int Counterpoint::AllDone
```

Referenced by AnySpecies(), and BestFitFirst().

6.1.4.9 AllVoicesSkipPenalty

```
int Counterpoint::AllVoicesSkipPenalty
```

Referenced by Counterpoint(), and OtherVoiceCheck().

6.1.4.10 AscendingSixthPenalty

```
int Counterpoint::AscendingSixthPenalty
```

Referenced by Counterpoint().

6.1.4.11 AugmentedIntervalPenalty

```
int Counterpoint::AugmentedIntervalPenalty
```

Referenced by Counterpoint(), and OtherVoiceCheck().

6.1.4.12 BadCadencePenalty

```
int Counterpoint::BadCadencePenalty
```

Referenced by Check(), Counterpoint(), and SpecialSpeciesCheck().

6.1.4.13 BadMelodyInterval

```
int Counterpoint::BadMelodyInterval = {0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0}  [static]
```

Referenced by BadMelody().

6.1.4.14 BadMelodyPenalty

```
int Counterpoint::BadMelodyPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.15 BasePitch

```
int Counterpoint::BasePitch
```

Referenced by AnySpecies(), Check(), and SaveResults().

6.1.4.16 BestFit

```
Eigen::MatrixXi Counterpoint::BestFit
```

Referenced by AnySpecies(), clear(), initialize(), SaveResults(), and winners().

6.1.4.17 BestFit1

```
Eigen::MatrixXi Counterpoint::BestFit1
```

Referenced by clear(), initialize(), SaveResults(), and winners().

6.1.4.18 BestFit2

```
Eigen::MatrixXi Counterpoint::BestFit2
```

Referenced by clear(), initialize(), SaveResults(), and winners().

6.1.4.19 BestFitPenalty

```
int Counterpoint::BestFitPenalty
```

Referenced by AnySpecies(), BestFitFirst(), and SaveResults().

6.1.4.20 Branches

```
int Counterpoint::Branches
```

Referenced by AnySpecies(), and BestFitFirst().

6.1.4.21 CompoundPenalty

```
int Counterpoint::CompoundPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.22 CrossAboveCantusPenalty

```
int Counterpoint::CrossAboveCantusPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.23 CrossBelowBassPenalty

```
int Counterpoint::CrossBelowBassPenalty
```

Referenced by Counterpoint(), and OtherVoiceCheck().

6.1.4.24 Ctrpt

```
Eigen::MatrixXi Counterpoint::Ctrpt
```

Referenced by AnySpecies(), Cantus(), clear(), counterpoint(), fillCantus(), initialize(), Look(), Other(), SaveResults(), SetUs(), toCsoundScore(), TooMuchOfInterval(), csound::CounterpointNode::transform(), and Us().

6.1.4.25 DirectMotionPenalty

```
int Counterpoint::DirectMotionPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.26 DirectPerfectOnDownbeatPenalty

```
int Counterpoint::DirectPerfectOnDownbeatPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.27 DirectToFifthPenalty

```
int Counterpoint::DirectToFifthPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.28 DirectToOctavePenalty

```
int Counterpoint::DirectToOctavePenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.29 DirectToTritonePenalty

```
int Counterpoint::DirectToTritonePenalty
```

Referenced by Counterpoint().

6.1.4.30 Dissonance

```
int Counterpoint::Dissonance = {0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0}  [static]
```

Referenced by ADissonance(), Check(), OtherVoiceCheck(), and SpecialSpeciesCheck().

6.1.4.31 DissonanceNotFillingThirdPenalty

```
int Counterpoint::DissonanceNotFillingThirdPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.32 DissonancePenalty

```
int Counterpoint::DissonancePenalty
```

Referenced by Check(), Counterpoint(), and SpecialSpeciesCheck().

6.1.4.33 DoubledFifthPenalty

```
int Counterpoint::DoubledFifthPenalty
```

Referenced by Counterpoint(), and OtherVoiceCheck().

6.1.4.34 DoubledLeadingTonePenalty

```
int Counterpoint::DoubledLeadingTonePenalty
```

Referenced by Check(), Counterpoint(), and OtherVoiceCheck().

6.1.4.35 DoubledSixthPenalty

```
int Counterpoint::DoubledSixthPenalty
```

Referenced by Counterpoint(), and OtherVoiceCheck().

6.1.4.36 DownBeatUnisonPenalty

```
int Counterpoint::DownBeatUnisonPenalty
```

Referenced by Counterpoint(), and SpecialSpeciesCheck().

6.1.4.37 Dur

```
Eigen::MatrixXi Counterpoint::Dur
```

Referenced by ADissonance(), AnySpecies(), clear(), initialize(), SpecialSpeciesCheck(), toCsoundScore(), csound::CounterpointNode::transform(), VIndex(), and winners().

6.1.4.38 EighthJumpPenalty

```
int Counterpoint::EighthJumpPenalty
```

Referenced by Counterpoint(), and SpecialSpeciesCheck().

6.1.4.39 EndOnPerfectPenalty

```
int Counterpoint::EndOnPerfectPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.40 ExtremeRangePenalty

```
int Counterpoint::ExtremeRangePenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.41 FifthFollowedBySameDirectionPenalty

```
int Counterpoint::FifthFollowedBySameDirectionPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.42 FifthPrecededBySameDirectionPenalty

```
int Counterpoint::FifthPrecededBySameDirectionPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.43 Fits

```
int Counterpoint::Fits[3]
```

Referenced by counterpoint(), SaveResults(), and winners().

6.1.4.44 FourRepeatedNotesPenalty

```
int Counterpoint::FourRepeatedNotesPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.45 HalfUntiedPenalty

```
int Counterpoint::HalfUntiedPenalty
```

Referenced by Counterpoint(), and SpecialSpeciesCheck().

6.1.4.46 HighestSemitone

```
int Counterpoint::HighestSemitone
```

Referenced by ExtremeRange(), OutOfRange(), and csound::CounterpointNode::transform().

6.1.4.47 ImperfectConsonance

```
int Counterpoint::ImperfectConsonance = {0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0}  [static]
```

6.1.4.48 Indx

```
int Counterpoint::Indx = {0, 1, -1, 2, -2, 3, -3, 0, 4, -4, 5, 7, -5, 8, 12, -7,-12}  [static]
```

Referenced by BestFitFirst(), and Look().

6.1.4.49 InnerVoicesInDirectToPerfectPenalty

```
int Counterpoint::InnerVoicesInDirectToPerfectPenalty
```

Referenced by Counterpoint(), and OtherVoiceCheck().

6.1.4.50 InnerVoicesInDirectToTritonePenalty

```
int Counterpoint::InnerVoicesInDirectToTritonePenalty
```

Referenced by Counterpoint(), and OtherVoiceCheck().

6.1.4.51 IntervalsWithBass

```
int Counterpoint::IntervalsWithBass[INTERVALS_WITH_BASS_SIZE]
```

Referenced by AddInterval(), and OtherVoiceCheck().

6.1.4.52 LeapAtCadencePenalty

```
int Counterpoint::LeapAtCadencePenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.53 LesserLigaturePenalty

```
int Counterpoint::LesserLigaturePenalty
```

Referenced by Counterpoint(), and SpecialSpeciesCheck().

6.1.4.54 LowerNeighborPenalty

```
int Counterpoint::LowerNeighborPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.55 LowestSemitone

```
int Counterpoint::LowestSemitone
```

Referenced by ExtremeRange(), OutOfRange(), and csound::CounterpointNode::transform().

6.1.4.56 LydianCadentialTritonePenalty

```
int Counterpoint::LydianCadentialTritonePenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.57 MaxPenalty

```
int Counterpoint::MaxPenalty
```

Referenced by AnySpecies(), BestFitFirst(), and SaveResults().

6.1.4.58 MelodicBoredomPenalty

```
int Counterpoint::MelodicBoredomPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.59 MelodicTritonePenalty

```
int Counterpoint::MelodicTritonePenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.60 mersenneTwister

```
std::mt19937 Counterpoint::mersenneTwister  [static]
```

Referenced by RANDOM().

6.1.4.61 Mode

```
int Counterpoint::Mode
```

Referenced by AnySpecies(), Check(), InMode(), OtherVoiceCheck(), SaveResults(), and SpecialSpeciesCheck().

6.1.4.62 MostNotes

```
int Counterpoint::MostNotes
```

Referenced by AnySpecies(), initialize(), and winners().

6.1.4.63 MostVoices

```
int Counterpoint::MostVoices
```

Referenced by AnySpecies(), BestFitFirst(), clear(), and initialize().

6.1.4.64 NoLeadingTonePenalty

```
int Counterpoint::NoLeadingTonePenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.65 NoMotionAgainstOctavePenalty

```
int Counterpoint::NoMotionAgainstOctavePenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.66 NotaCambiataPenalty

```
int Counterpoint::NotaCambiataPenalty
```

Referenced by Counterpoint(), and SpecialSpeciesCheck().

6.1.4.67 NotaLigaturePenalty

```
int Counterpoint::NotaLigaturePenalty
```

Referenced by Counterpoint(), and SpecialSpeciesCheck().

6.1.4.68 NotBestCadencePenalty

```
int Counterpoint::NotBestCadencePenalty
```

Referenced by Counterpoint().

6.1.4.69 NotContraryToOthersPenalty

```
int Counterpoint::NotContraryToOthersPenalty
```

Referenced by Counterpoint(), and OtherVoiceCheck().

6.1.4.70 NoTimeForaLigaturePenalty

```
int Counterpoint::NoTimeForaLigaturePenalty
```

Referenced by Counterpoint(), and SpecialSpeciesCheck().

6.1.4.71 NotTriadPenalty

```
int Counterpoint::NotTriadPenalty
```

Referenced by Counterpoint(), and OtherVoiceCheck().

6.1.4.72 OctaveLeapPenalty

```
int Counterpoint::OctaveLeapPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.73 Onset

```
Eigen::MatrixXi Counterpoint::Onset
```

Referenced by ADissonance(), AnySpecies(), BestFitFirst(), Cantus(), clear(), DownBeat(), initialize(), Other(), SpecialSpeciesCheck(), toCsoundScore(), csound::CounterpointNode::transform(), and VIndex().

6.1.4.74 OutOfModePenalty

```
int Counterpoint::OutOfModePenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.75 OutOfRangePenalty

```
int Counterpoint::OutOfRangePenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.76 OverOctavePenalty

```
int Counterpoint::OverOctavePenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.77 OverTwelfthPenalty

```
int Counterpoint::OverTwelfthPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.78 ParallelFifthPenalty

```
int Counterpoint::ParallelFifthPenalty
```

Referenced by Check(), Counterpoint(), and OtherVoiceCheck().

6.1.4.79 ParallelUnisonPenalty

```
int Counterpoint::ParallelUnisonPenalty
```

Referenced by Check(), Counterpoint(), and OtherVoiceCheck().

6.1.4.80 PenaltyRatio

```
float Counterpoint::PenaltyRatio
```

Referenced by AnySpecies(), BestFitFirst(), and SaveResults().

6.1.4.81 PerfectConsonance

```
int Counterpoint::PerfectConsonance = {1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1}  [static]
```

Referenced by Check(), and DirectMotionToPerfectConsonance().

6.1.4.82 PerfectConsonancePenalty

```
int Counterpoint::PerfectConsonancePenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.83 randx

```
long Counterpoint::randx
```

Referenced by initialize().

6.1.4.84 RepeatedPitchPenalty

```
int Counterpoint::RepeatedPitchPenalty
```

Referenced by Counterpoint().

6.1.4.85 RepetitionOnUpbeatPenalty

```
int Counterpoint::RepetitionOnUpbeatPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.86 RhyNotes

```
Eigen::VectorXi Counterpoint::RhyNotes
```

Referenced by AnySpecies(), clear(), FillRhyPat(), and initialize().

6.1.4.87 RhyPat

```
Eigen::MatrixXi Counterpoint::RhyPat
```

Referenced by AnySpecies(), CleanRhy(), clear(), CurRhy(), FillRhyPat(), initialize(), and UsedRhy().

6.1.4.88 SixFiveChordPenalty

```
int Counterpoint::SixFiveChordPenalty
```

Referenced by Counterpoint(), and OtherVoiceCheck().

6.1.4.89 SixthFollowedBySameDirectionPenalty

```
int Counterpoint::SixthFollowedBySameDirectionPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.90 SixthLeapPenalty

```
int Counterpoint::SixthLeapPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.91 SixthPrecededBySameDirectionPenalty

```
int Counterpoint::SixthPrecededBySameDirectionPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.92 SkipFollowedBySameDirectionPenalty

```
int Counterpoint::SkipFollowedBySameDirectionPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.93 SkipFromUnisonPenalty

```
int Counterpoint::SkipFromUnisonPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.94 SkipPrecededBySameDirectionPenalty

```
int Counterpoint::SkipPrecededBySameDirectionPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.95 SkipTo8vePenalty

```
int Counterpoint::SkipTo8vePenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.96 SkipToDownBeatPenalty

```
int Counterpoint::SkipToDownBeatPenalty
```

Referenced by Counterpoint(), and SpecialSpeciesCheck().

6.1.4.97 TenthToOctavePenalty

```
int Counterpoint::TenthToOctavePenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.98 ThirdDoubledPenalty

```
int Counterpoint::ThirdDoubledPenalty
```

Referenced by Counterpoint(), and OtherVoiceCheck().

6.1.4.99 ThreeRepeatedNotesPenalty

```
int Counterpoint::ThreeRepeatedNotesPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.100 ThreeSkipsPenalty

```
int Counterpoint::ThreeSkipsPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.101 TotalNotes

```
Eigen::VectorXi Counterpoint::TotalNotes
```

Referenced by AnySpecies(), Check(), clear(), initialize(), LastNote(), NextToLastNote(), SaveResults(), toCsoundScore(), csound::CounterpointNode::transform(), VIndex(), and winners().

6.1.4.102 TotalTime

```
int Counterpoint::TotalTime
```

Referenced by AnySpecies(), and BestFitFirst().

6.1.4.103 TripledBassPenalty

```
int Counterpoint::TripledBassPenalty
```

Referenced by Counterpoint(), and OtherVoiceCheck().

6.1.4.104 TwoRepeatedNotesPenalty

```
int Counterpoint::TwoRepeatedNotesPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.105 TwoSkipsNotInTriadPenalty

```
int Counterpoint::TwoSkipsNotInTriadPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.106 TwoSkipsPenalty

```
int Counterpoint::TwoSkipsPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.107 uniform_real_generator

```
std::normal_distribution Counterpoint::uniform_real_generator
```

Referenced by RANDOM().

6.1.4.108 UnisonDownbeatPenalty

```
int Counterpoint::UnisonDownbeatPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.109 UnisonOnBeat4Penalty

```
int Counterpoint::UnisonOnBeat4Penalty
```

Referenced by Counterpoint(), and SpecialSpeciesCheck().

6.1.4.110 UnisonPenalty

```
int Counterpoint::UnisonPenalty
```

Referenced by Check(), Counterpoint(), and OtherVoiceCheck().

6.1.4.111 UnisonUpbeatPenalty

```
int Counterpoint::UnisonUpbeatPenalty
```

Referenced by Counterpoint(), and SpecialSpeciesCheck().

6.1.4.112 UnpreparedSixFivePenalty

```
int Counterpoint::UnpreparedSixFivePenalty
```

Referenced by Counterpoint(), and OtherVoiceCheck().

6.1.4.113 UnresolvedLeadingTonePenalty

```
int Counterpoint::UnresolvedLeadingTonePenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.114 UnresolvedLigaturePenalty

```
int Counterpoint::UnresolvedLigaturePenalty
```

Referenced by Counterpoint(), and SpecialSpeciesCheck().

6.1.4.115 UnresolvedSixFivePenalty

```
int Counterpoint::UnresolvedSixFivePenalty
```

Referenced by Counterpoint(), and OtherVoiceCheck().

6.1.4.116 UpperNeighborPenalty

```
int Counterpoint::UpperNeighborPenalty
```

Referenced by Check(), and Counterpoint().

6.1.4.117 UpperVoicesTooFarApartPenalty

```
int Counterpoint::UpperVoicesTooFarApartPenalty
```

Referenced by Counterpoint(), and OtherVoiceCheck().

6.1.4.118 vbs

```
Eigen::VectorXi Counterpoint::vbs
```

Referenced by clear(), counterpoint(), initialize(), and main().

6.1.4.119 VerticalTritonePenalty

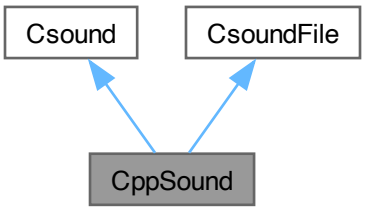
```
int Counterpoint::VerticalTritonePenalty
```

Referenced by Check(), Counterpoint(), and OtherVoiceCheck().

6.2 CppSound Class Reference

```
#include <CppSound.hpp>
```

Inheritance diagram for CppSound:



Public Member Functions

- virtual void addArrangement (std::string instrument)
- virtual void addNote (double p1, double p2, double p3)
- virtual void addNote (double p1, double p2, double p3, double p4)
- virtual void addNote (double p1, double p2, double p3, double p4, double p5)
- virtual void addNote (double p1, double p2, double p3, double p4, double p5, double p6)
- virtual void addNote (double p1, double p2, double p3, double p4, double p5, double p6, double p7)
- virtual void addNote (double p1, double p2, double p3, double p4, double p5, double p6, double p7, double p8)
- virtual void addNote (double p1, double p2, double p3, double p4, double p5, double p6, double p7, double p8, double p9)
- virtual void addNote (double p1, double p2, double p3, double p4, double p5, double p6, double p7, double p8, double p9, double p10)
- virtual void addNote (double p1, double p2, double p3, double p4, double p5, double p6, double p7, double p8, double p9, double p10, double p11)
- virtual void addScoreLine (const std::string line)
- virtual void cleanup ()
- virtual int compile ()
- virtual int compile (int argc, const char **argv)
- CppSound ()
- virtual int exportArrangement (std::ostream &stream) const
- virtual int exportArrangementForPerformance (std::ostream &stream) const
- virtual int exportArrangementForPerformance (std::string filename) const
- virtual int exportCommand (std::ostream &stream) const
- virtual bool exportForPerformance () const
- virtual int exportMidifile (std::ostream &stream) const
- virtual int exportOrchestra (std::ostream &stream) const
- virtual int exportScore (std::ostream &stream) const
- virtual std::string generateFilename ()
- virtual std::string getArrangement (int index) const
- virtual int getArrangementCount () const
- virtual std::string getCommand () const
- virtual std::string getCSD () const
- virtual CSOUND * getCsound ()
- virtual CsoundFile * getCsoundFile ()
- virtual std::string getFilename () const
- virtual std::string getInstrument (int number) const
- virtual bool getInstrument (int number, std::string &definition) const
- virtual std::string getInstrument (std::string name) const
- virtual bool getInstrument (std::string name, std::string &definition) const
- virtual std::string getInstrumentBody (int number) const
- virtual std::string getInstrumentBody (std::string name) const
- virtual int getInstrumentCount () const
- virtual std::map< int, std::string > getInstrumentNames () const
- virtual double getInstrumentNumber (std::string name) const
- virtual bool getIsCompiled () const
- virtual bool getIsGo ()
- virtual bool getIsPerforming () const
- virtual std::string getMidiFilename () const
- virtual std::string getOrcFilename () const
- virtual std::string getOrchestra () const

- virtual std::string getOrchestraHeader () const
- virtual std::string getOutputSoundfileName () const
- virtual std::string getScoFilename () const
- virtual std::string getScore () const
- virtual size_t getSpoutSize () const
- virtual intptr_t getThis ()
- virtual int importArrangement (std::istream &stream)
- virtual int importCommand (std::istream &stream)
- virtual int importFile (std::istream &stream)
- virtual int importFile (std::string filename)

  Imports the indicated file, which can be a Csound unified file (.csd), Csound orchestra (.orc), Csound score (.sco), standard MIDI file (.mid), or MusicXML v2 (.xml) file.
- virtual int importMidifile (std::istream &stream)
- virtual int importOrchestra (std::istream &stream)
- virtual int importScore (std::istream &stream)
- virtual void inputMessage (const char *istatement)
- virtual void insertArrangement (int index, std::string instrument)
- virtual int load (std::istream &stream)
- virtual int load (std::string filename)

  Clears all contents of this, then imports the indicated file, which can be a Csound unified file (.csd), Csound orchestra (.orc), Csound score (.sco), standard MIDI file (.mid), or MusicXML v2 (.xml) file.
- virtual bool loadOrcLibrary (const char *filename=0)
- virtual int perform ()
- virtual int perform (int argc, const char **argv)
- virtual int performKsmps ()
- virtual void removeAll ()
- virtual void removeArrangement ()
- virtual void removeArrangement (int index)
- virtual void removeCommand ()
- virtual void removeMidifile ()
- virtual void removeOrchestra ()
- virtual void removeScore ()
- virtual int save (std::ostream &stream) const
- virtual int save (std::string filename) const
- virtual void setArrangement (int index, std::string instrument)
- virtual void setCommand (std::string commandLine)
- virtual void setCSD (std::string xml)
- virtual void setFilename (std::string name)
- virtual void setIsPerforming (bool isPerforming)
- virtual void setOrchestra (std::string orchestra)
- virtual void setScore (std::string score)
- virtual void stop ()
- virtual void write (const char *text)
- virtual ~CppSound ()

Data Fields

- std::vector< std::string > arrangement
- std::string libraryFilename

  Patch library and arrangement.

Protected Attributes

- std::vector< std::string > args
- std::vector< char * > argv
- std::string command

  CsOptions.
- std::string filename

  What are we storing, anyway?
- std::vector< unsigned char > midifile

  CsMidi.
- std::string orchestra

  CsInstruments.
- std::string score

  CsScore.

6.2.1 Constructor & Destructor Documentation

6.2.1.1 CppSound()

```
CppSound::CppSound ( )  [inline]
```

6.2.1.2 ~CppSound()

```
CppSound::~CppSound ( )  [virtual]
```

6.2.2 Member Function Documentation

6.2.2.1 addArrangement()

```
void CsoundFile::addArrangement (
            std::string instrument )  [virtual], [inherited]
```

References CsoundFile::arrangement.

6.2.2.2 addNote() [1/9]

```
void CsoundFile::addNote (
            double p1,
            double p2,
            double p3 )  [virtual], [inherited]
```

References CsoundFile::addScoreLine().

6.2.2.3 addNote() [2/9]

```
void CsoundFile::addNote (
            double p1,
            double p2,
            double p3,
            double p4 )  [virtual], [inherited]
```

References CsoundFile::addScoreLine().

6.2.2.4 addNote() [3/9]

```
void CsoundFile::addNote (
            double p1,
            double p2,
            double p3,
            double p4,
            double p5 )  [virtual], [inherited]
```

References CsoundFile::addScoreLine().

6.2.2.5 addNote() [4/9]

```
void CsoundFile::addNote (
            double p1,
            double p2,
            double p3,
            double p4,
            double p5,
            double p6 )  [virtual], [inherited]
```

References CsoundFile::addScoreLine().

6.2.2.6 addNote() [5/9]

```
void CsoundFile::addNote (
            double p1,
            double p2,
            double p3,
            double p4,
            double p5,
            double p6,
            double p7 )  [virtual], [inherited]
```

References CsoundFile::addScoreLine().

6.2.2.7 addNote() [6/9]

```
void CsoundFile::addNote (
            double p1,
            double p2,
            double p3,
            double p4,
            double p5,
            double p6,
            double p7,
            double p8 )  [virtual], [inherited]
```

References CsoundFile::addScoreLine().

6.2.2.8 addNote() [7/9]

```
void CsoundFile::addNote (
            double p1,
            double p2,
            double p3,
            double p4,
            double p5,
            double p6,
            double p7,
            double p8,
            double p9 )  [virtual], [inherited]
```

References CsoundFile::addScoreLine().

6.2.2.9 addNote() [8/9]

```
void CsoundFile::addNote (
            double p1,
            double p2,
            double p3,
            double p4,
            double p5,
            double p6,
            double p7,
            double p8,
            double p9,
            double p10 )  [virtual], [inherited]
```

References CsoundFile::addScoreLine().

6.2.2.10 addNote() [9/9]

```
void CsoundFile::addNote (
            double p1,
            double p2,
            double p3,
            double p4,
            double p5,
            double p6,
            double p7,
            double p8,
            double p9,
            double p10,
            double p11 )  [virtual], [inherited]
```

References CsoundFile::addScoreLine().

6.2.2.11 addScoreLine()

```
void CsoundFile::addScoreLine (
            const std::string line )  [virtual], [inherited]
```

References CsoundFile::score.

Referenced by CsoundFile::addNote(), CsoundFile::addNote(), CsoundFile::addNote(), CsoundFile::addNote(), CsoundFile::addNote(), CsoundFile::addNote(), CsoundFile::addNote(), CsoundFile::addNote(), and csound::MusicModel::createCsoundScore().

6.2.2.12 cleanup()

```
void CppSound::cleanup ( )  [virtual]
```

Referenced by perform().

6.2.2.13 compile() [1/2]

```
int CppSound::compile ( )  [virtual]
```

References CsoundFile::args, CsoundFile::argv, compile(), CsoundFile::getCommand(), and scatterArgs().

Referenced by compile(), and perform().

6.2.2.14 compile() [2/2]

```
int CppSound::compile (
            int argc,
            const char ** argv )  [virtual]
```

References CsoundFile::argv, CsoundFile::getOrchestra(), CsoundFile::getScore(), and MYFLT.

6.2.2.15 exportArrangement()

```
int CsoundFile::exportArrangement (
            std::ostream & stream ) const  [virtual], [inherited]
```

References CsoundFile::arrangement.

Referenced by CsoundFile::save().

6.2.2.16 exportArrangementForPerformance() [1/2]

```
int CsoundFile::exportArrangementForPerformance (
            std::ostream & stream ) const  [virtual], [inherited]
```

References CsoundFile::arrangement, CsoundFile::exportOrchestra(), CsoundFile::getInstrument(), CsoundFile::getOrcFilename(), CsoundFile::getOrchestraHeader(), and parseInstrument().

6.2.2.17 exportArrangementForPerformance() [2/2]

```
int CsoundFile::exportArrangementForPerformance (
            std::string filename ) const  [virtual], [inherited]
```

References CsoundFile::exportArrangementForPerformance(), and CsoundFile::filename.

Referenced by CsoundFile::exportArrangementForPerformance(), and CsoundFile::exportForPerformance().

6.2.2.18 exportCommand()

```
int CsoundFile::exportCommand (
            std::ostream & stream ) const  [virtual], [inherited]
```

References CsoundFile::command.

Referenced by CsoundFile::save().

6.2.2.19 exportForPerformance()

```
bool CsoundFile::exportForPerformance ( ) const  [virtual], [inherited]
```

References CsoundFile::exportArrangementForPerformance(), CsoundFile::getMidiFilename(), CsoundFile::getOrcFilename(), CsoundFile::getScoFilename(), CsoundFile::midifile, and CsoundFile::save().

6.2.2.20 exportMidifile()

```
int CsoundFile::exportMidifile (
            std::ostream & stream ) const  [virtual], [inherited]
```

References CsoundFile::midifile.

Referenced by CsoundFile::save(), and CsoundFile::save().

6.2.2.21 exportOrchestra()

```
int CsoundFile::exportOrchestra (
            std::ostream & stream ) const  [virtual], [inherited]
```

References CsoundFile::orchestra.

Referenced by CsoundFile::exportArrangementForPerformance(), CsoundFile::save(), and CsoundFile::save().

6.2.2.22 exportScore()

```
int CsoundFile::exportScore (
            std::ostream & stream ) const  [virtual], [inherited]
```

References CsoundFile::score.

Referenced by CsoundFile::save(), and CsoundFile::save().

6.2.2.23 generateFilename()

```
std::string CsoundFile::generateFilename ( )  [virtual], [inherited]
```

References CsoundFile::filename.

6.2.2.24 getArrangement()

```
std::string CsoundFile::getArrangement (
            int index ) const  [virtual], [inherited]
```

References CsoundFile::arrangement.

6.2.2.25 getArrangementCount()

```
int CsoundFile::getArrangementCount ( ) const  [virtual], [inherited]
```

References CsoundFile::arrangement.

6.2.2.26 getCommand()

```
std::string CsoundFile::getCommand ( ) const  [virtual], [inherited]
```

References CsoundFile::command.

Referenced by compile(), csound::MusicModel::getCsoundCommand(), perform(), and csound::MusicModel::perform().

6.2.2.27 getCSD()

```
std::string CsoundFile::getCSD ( ) const  [virtual], [inherited]
```

References CsoundFile::save().

Referenced by csound::MusicModel::perform().

6.2.2.28 getCsound()

```
CSOUND * CppSound::getCsound ( )  [virtual]
```

Referenced by csound::MusicModel::perform().

6.2.2.29 getCsoundFile()

```
CsoundFile * CppSound::getCsoundFile ( )  [virtual]
```

6.2.2.30 getFilename()

```
std::string CsoundFile::getFilename ( ) const  [virtual], [inherited]
```

References CsoundFile::filename.

Referenced by perform().

6.2.2.31 getInstrument() [1/4]

```
std::string CsoundFile::getInstrument (
            int number ) const  [virtual], [inherited]
```

References CsoundFile::getInstrument().

6.2.2.32 getInstrument() [2/4]

```
bool CsoundFile::getInstrument (
            int number,
            std::string & definition ) const  [virtual], [inherited]
```

References findToken(), CsoundFile::orchestra, and parseInstrument().

Referenced by CsoundFile::exportArrangementForPerformance(), CsoundFile::getInstrument(), CsoundFile::getInstrument(), CsoundFile::getInstrumentBody(), and CsoundFile::getInstrumentBody().

6.2.2.33 getInstrument() [3/4]

```
std::string CsoundFile::getInstrument (
            std::string name ) const  [virtual], [inherited]
```

References CsoundFile::getInstrument().

6.2.2.34 getInstrument() [4/4]

```
bool CsoundFile::getInstrument (
            std::string name,
            std::string & definition ) const  [virtual], [inherited]
```

References findToken(), CsoundFile::orchestra, parseInstrument(), and trim().

6.2.2.35 getInstrumentBody() [1/2]

```
std::string CsoundFile::getInstrumentBody (
            int number ) const  [virtual], [inherited]
```

References CsoundFile::getInstrument(), and parseInstrument().

6.2.2.36 getInstrumentBody() [2/2]

```
std::string CsoundFile::getInstrumentBody (
            std::string name ) const  [virtual], [inherited]
```

References CsoundFile::getInstrument(), and parseInstrument().

6.2.2.37 getInstrumentCount()

```
int CsoundFile::getInstrumentCount ( ) const  [virtual], [inherited]
```

References findToken(), CsoundFile::orchestra, and parseInstrument().

6.2.2.38 getInstrumentNames()

```
std::map< int, std::string > CsoundFile::getInstrumentNames ( ) const  [virtual], [inherited]
```

References findToken(), CsoundFile::orchestra, and parseInstrument().

6.2.2.39 getInstrumentNumber()

```
double CsoundFile::getInstrumentNumber (
            std::string name ) const  [virtual], [inherited]
```

References findToken(), CsoundFile::orchestra, parseInstrument(), and trim().

Referenced by csound::MusicModel::arrange(), csound::MusicModel::arrange(), and csound::MusicModel::arrange().

6.2.2.40 getIsCompiled()

```
bool CppSound::getIsCompiled ( ) const  [virtual]
```

6.2.2.41 getIsGo()

```
bool CppSound::getIsGo ( )  [virtual]
```

6.2.2.42 getIsPerforming()

```
bool CppSound::getIsPerforming ( ) const  [virtual]
```

6.2.2.43 getMidiFilename()

```
std::string CsoundFile::getMidiFilename ( ) const  [virtual], [inherited]
```

References CsoundFile::args, CsoundFile::argv, CsoundFile::command, and scatterArgs().

Referenced by CsoundFile::exportForPerformance().

6.2.2.44 getOrcFilename()

```
std::string CsoundFile::getOrcFilename ( ) const  [virtual], [inherited]
```

References CsoundFile::args, CsoundFile::argv, CsoundFile::command, and scatterArgs().

Referenced by CsoundFile::exportArrangementForPerformance(), and CsoundFile::exportForPerformance().

6.2.2.45 getOrchestra()

```
std::string CsoundFile::getOrchestra ( ) const  [virtual], [inherited]
```

References CsoundFile::orchestra.

Referenced by compile(), and csound::MusicModel::getCsoundOrchestra().

6.2.2.46 getOrchestraHeader()

```
std::string CsoundFile::getOrchestraHeader ( ) const  [virtual], [inherited]
```

References findToken(), and CsoundFile::orchestra.

Referenced by CsoundFile::exportArrangementForPerformance().

6.2.2.47 getOutputSoundfileName()

```
std::string CppSound::getOutputSoundfileName ( ) const  [virtual]
```

Reimplemented from CsoundFile.

6.2.2.48 getScoFilename()

```
std::string CsoundFile::getScoFilename ( ) const  [virtual], [inherited]
```

References CsoundFile::args, CsoundFile::argv, CsoundFile::command, and scatterArgs().

Referenced by CsoundFile::exportForPerformance().

6.2.2.49 getScore()

```
std::string CsoundFile::getScore ( ) const  [virtual], [inherited]
```

References CsoundFile::score.

Referenced by compile().

6.2.2.50 getSpoutSize()

```
size_t CppSound::getSpoutSize ( ) const  [virtual]
```

6.2.2.51 getThis()

```
intptr_t CppSound::getThis ( )  [virtual]
```

6.2.2.52 importArrangement()

```
int CsoundFile::importArrangement (
            std::istream & stream )  [virtual], [inherited]
```

References CsoundFile::arrangement, getline(), CsoundFile::removeArrangement(), and trim().

Referenced by CsoundFile::importFile().

6.2.2.53 importCommand()

```
int CsoundFile::importCommand (
            std::istream & stream )  [virtual], [inherited]
```

References CsoundFile::command, and getline().

Referenced by CsoundFile::importFile().

6.2.2.54 importFile() [1/2]

```
int CsoundFile::importFile (
            std::istream & stream )  [virtual], [inherited]
```

References getline(), CsoundFile::importArrangement(), CsoundFile::importCommand(), CsoundFile::importMidifile(), CsoundFile::importOrchestra(), and CsoundFile::importScore().

6.2.2.55 importFile() [2/2]

```
int CsoundFile::importFile (
            std::string filename )  [virtual], [inherited]
```

Imports the indicated file, which can be a Csound unified file (.csd), Csound orchestra (.orc), Csound score (.sco), standard MIDI file (.mid), or MusicXML v2 (.xml) file.

The data that is read replaces existing data of that type, but leaves other types of data untouched.

The MusicXML notes become instrument number + 1, time in seconds, duration in seconds, MIDI key number, and MIDI velocity number.

References CsoundFile::filename, CsoundFile::importFile(), CsoundFile::importMidifile(), CsoundFile::importOrchestra(), CsoundFile::importScore(), and CsoundFile::score.

Referenced by CsoundFile::importFile(), CsoundFile::load(), and CsoundFile::load().

6.2.2.56 importMidifile()

```
int CsoundFile::importMidifile (
            std::istream & stream )  [virtual], [inherited]
```

References getline(), and CsoundFile::midifile.

Referenced by CsoundFile::importFile(), and CsoundFile::importFile().

6.2.2.57 importOrchestra()

```
int CsoundFile::importOrchestra (
            std::istream & stream )  [virtual], [inherited]
```

References getline(), and CsoundFile::orchestra.

Referenced by CsoundFile::importFile(), CsoundFile::importFile(), and CsoundFile::loadOrcLibrary().

6.2.2.58 importScore()

```
int CsoundFile::importScore (
            std::istream & stream )  [virtual], [inherited]
```

References getline(), and CsoundFile::score.

Referenced by CsoundFile::importFile(), and CsoundFile::importFile().

6.2.2.59 inputMessage()

```
void CppSound::inputMessage (
            const char * istatement )  [virtual]
```

6.2.2.60 insertArrangement()

```
void CsoundFile::insertArrangement (
            int index,
            std::string instrument )  [virtual], [inherited]
```

References CsoundFile::arrangement.

6.2.2.61 load() [1/2]

```
int CsoundFile::load (
            std::istream & stream )  [virtual], [inherited]
```

References CsoundFile::importFile(), and CsoundFile::removeAll().

6.2.2.62 load() [2/2]

```
int CsoundFile::load (
            std::string filename )  [virtual], [inherited]
```

Clears all contents of this, then imports the indicated file, which can be a Csound unified file (.csd), Csound orchestra (.orc), Csound score (.sco), standard MIDI file (.mid), or MusicXML v2 (.xml) file.

The MusicXML notes become instrument number + 1, time in seconds, duration in seconds, MIDI key number, and MIDI velocity number.

References CsoundFile::filename, CsoundFile::importFile(), and CsoundFile::removeAll().

Referenced by CsoundFile::setCSD().

6.2.2.63 loadOrcLibrary()

```
bool CsoundFile::loadOrcLibrary (
            const char * filename = 0 )  [virtual], [inherited]
```

References CsoundFile::filename, CsoundFile::importOrchestra(), and CsoundFile::removeOrchestra().

6.2.2.64 perform() [1/2]

```
int CppSound::perform ( )  [virtual]
```

References CsoundFile::args, CsoundFile::argv, CsoundFile::command, CsoundFile::filename, CsoundFile::getCommand(), CsoundFile::getFilename(), perform(), and scatterArgs().

Referenced by perform().

6.2.2.65 perform() [2/2]

```
int CppSound::perform (
            int argc,
            const char ** argv )  [virtual]
```

References cleanup(), and compile().

6.2.2.66 performKsmps()

```
int CppSound::performKsmps ( )  [virtual]
```

6.2.2.67 removeAll()

```
void CsoundFile::removeAll ( )  [virtual], [inherited]
```

References CsoundFile::arrangement, CsoundFile::command, CsoundFile::filename, CsoundFile::orchestra, CsoundFile::removeMidifile(), and CsoundFile::score.

Referenced by CsoundFile::CsoundFile(), CsoundFile::load(), and CsoundFile::load().

6.2.2.68 removeArrangement() [1/2]

```
void CsoundFile::removeArrangement ( )  [virtual], [inherited]
```

References CsoundFile::arrangement.

Referenced by CsoundFile::importArrangement().

6.2.2.69 removeArrangement() [2/2]

```
void CsoundFile::removeArrangement (
            int index )  [virtual], [inherited]
```

References CsoundFile::arrangement.

6.2.2.70 removeCommand()

```
void CsoundFile::removeCommand ( )  [virtual], [inherited]
```

References CsoundFile::command.

6.2.2.71 removeMidifile()

```
void CsoundFile::removeMidifile ( )  [virtual], [inherited]
```

References CsoundFile::midifile.

Referenced by CsoundFile::removeAll().

6.2.2.72 removeOrchestra()

```
void CsoundFile::removeOrchestra ( )  [virtual], [inherited]
```

References CsoundFile::orchestra.

Referenced by CsoundFile::loadOrcLibrary().

6.2.2.73 removeScore()

```
void CsoundFile::removeScore ( )  [virtual], [inherited]
```

References CsoundFile::score.

Referenced by csound::MusicModel::clear(), csound::MusicModel::createCsoundScore(), and csound::MusicModel::generate().

6.2.2.74 save() [1/2]

```
int CsoundFile::save (
            std::ostream & stream ) const  [virtual], [inherited]
```

References CsoundFile::arrangement, CsoundFile::exportArrangement(), CsoundFile::exportCommand(), CsoundFile::exportMidifile(), CsoundFile::exportOrchestra(), CsoundFile::exportScore(), and CsoundFile::midifile.

6.2.2.75 save() [2/2]

```
int CsoundFile::save (
            std::string filename ) const  [virtual], [inherited]
```

References CsoundFile::exportMidifile(), CsoundFile::exportOrchestra(), CsoundFile::exportScore(), CsoundFile::filename, and CsoundFile::save().

Referenced by CsoundFile::exportForPerformance(), CsoundFile::getCSD(), and CsoundFile::save().

6.2.2.76 setArrangement()

```
void CsoundFile::setArrangement (
            int index,
            std::string instrument )  [virtual], [inherited]
```

References CsoundFile::arrangement.

6.2.2.77 setCommand()

```
void CsoundFile::setCommand (
            std::string commandLine )  [virtual], [inherited]
```

References CsoundFile::command.

Referenced by csound::MusicModel::perform(), and csound::MusicModel::setCsoundCommand().

6.2.2.78 setCSD()

```
void CsoundFile::setCSD (
            std::string xml )  [virtual], [inherited]
```

References CsoundFile::load().

6.2.2.79 setFilename()

```
void CsoundFile::setFilename (
            std::string name )  [virtual], [inherited]
```

References CsoundFile::filename.

6.2.2.80 setIsPerforming()

```
void CppSound::setIsPerforming (
            bool isPerforming )  [virtual]
```

6.2.2.81 setOrchestra()

```
void CsoundFile::setOrchestra (
            std::string orchestra )  [virtual], [inherited]
```

References CsoundFile::orchestra.

Referenced by csound::MusicModel::setCsoundOrchestra().

6.2.2.82 setScore()

```
void CsoundFile::setScore (
            std::string score )  [virtual], [inherited]
```

References CsoundFile::score.

6.2.2.83 stop()

```
void CppSound::stop ( )  [virtual]
```

Referenced by csound::MusicModel::stop().

6.2.2.84 write()

```
void CppSound::write (
            const char * text )  [virtual]
```

6.2.3 Field Documentation

6.2.3.1 args

```
std::vector<std::string> CsoundFile::args  [protected], [inherited]
```

Referenced by compile(), CsoundFile::getMidiFilename(), CsoundFile::getOrcFilename(), CsoundFile::getScoFilename(), and perform().

6.2.3.2 argv

```
std::vector<char *> CsoundFile::argv  [protected], [inherited]
```

Referenced by compile(), compile(), CsoundFile::getMidiFilename(), CsoundFile::getOrcFilename(), CsoundFile::getScoFilename(), and perform().

6.2.3.3 arrangement

```
std::vector<std::string> CsoundFile::arrangement  [inherited]
```

Referenced by CsoundFile::addArrangement(), CsoundFile::exportArrangement(), CsoundFile::exportArrangementForPerformance(), CsoundFile::getArrangement(), CsoundFile::getArrangementCount(), CsoundFile::importArrangement(), CsoundFile::insertArrangement() CsoundFile::removeAll(), CsoundFile::removeArrangement(), CsoundFile::removeArrangement(), CsoundFile::save(), and CsoundFile::setArrangement().

6.2.3.4 command

```
std::string CsoundFile::command  [protected], [inherited]
```

CsOptions.

Referenced by CsoundFile::exportCommand(), CsoundFile::getCommand(), CsoundFile::getMidiFilename(), CsoundFile::getOrcFilename CsoundFile::getScoFilename(), CsoundFile::importCommand(), perform(), CsoundFile::removeAll(), CsoundFile::removeCommand(), and CsoundFile::setCommand().

6.2.3.5 filename

```
std::string CsoundFile::filename  [protected], [inherited]
```

What are we storing, anyway?

Referenced by CsoundFile::exportArrangementForPerformance(), CsoundFile::generateFilename(), CsoundFile::getFilename(), CsoundFile::importFile(), CsoundFile::load(), CsoundFile::loadOrcLibrary(), perform(), CsoundFile::removeAll(), CsoundFile::save(), and CsoundFile::setFilename().

6.2.3.6 libraryFilename

```
std::string CsoundFile::libraryFilename  [inherited]
```

Patch library and arrangement.

6.2.3.7 midifile

```
std::vector<unsigned char> CsoundFile::midifile  [protected], [inherited]
```

CsMidi.

Referenced by CsoundFile::exportForPerformance(), CsoundFile::exportMidifile(), CsoundFile::importMidifile(), CsoundFile::removeMidifile(), and CsoundFile::save().

6.2.3.8 orchestra

```
std::string CsoundFile::orchestra  [protected], [inherited]
```

CsInstruments.

Referenced by CsoundFile::exportOrchestra(), CsoundFile::getInstrument(), CsoundFile::getInstrument(), CsoundFile::getInstrumentCoun CsoundFile::getInstrumentNames(), CsoundFile::getInstrumentNumber(), CsoundFile::getOrchestra(), CsoundFile::getOrchestraHeader() CsoundFile::importOrchestra(), CsoundFile::removeAll(), CsoundFile::removeOrchestra(), and CsoundFile::setOrchestra().

6.2.3.9 score

```
std::string CsoundFile::score  [protected], [inherited]
```

CsScore.

Referenced by CsoundFile::addScoreLine(), CsoundFile::exportScore(), CsoundFile::getScore(), CsoundFile::importFile(), CsoundFile::importScore(), CsoundFile::removeAll(), CsoundFile::removeScore(), and CsoundFile::setScore().

6.3 csound::are_cl_objects<... > Struct Template Reference

`#include <Lisp.hpp>`

Static Public Attributes

- static constexpr bool p = true

6.3.1 Field Documentation

6.3.1.1 p

```
template<typename...  >
constexpr bool csound::are_cl_objects<...  >::p = true  [static], [constexpr]
```

6.4 csound::are_cl_objects< Head, Tail... > Struct Template Reference

`#include <Lisp.hpp>`

Static Public Attributes

- static constexpr bool p = is_cl_object<Head>::p && are_cl_objects<Tail...>::p

6.4.1 Field Documentation

6.4.1.1 p

```
template<typename Head , typename...  Tail>
constexpr bool csound::are_cl_objects< Head, Tail...  >::p = is_cl_object<Head>::p && are_cl_objects<Tail...>↩
::p  [static], [constexpr]
```

6.5 csound::AscendingDistanceComparator Struct Reference

Public Member Functions

- double ascendingDistance (double a, double b)
- AscendingDistanceComparator (double origin_, size_t divisionsPerOctave_)
- bool operator() (double a, double b)

Data Fields

- size_t divisionsPerOctave
- double origin

6.5.1 Constructor & Destructor Documentation

6.5.1.1 AscendingDistanceComparator()

```
csound::AscendingDistanceComparator::AscendingDistanceComparator (
            double origin_,
            size_t divisionsPerOctave_ )  [inline]
```

6.5.2 Member Function Documentation

6.5.2.1 ascendingDistance()

```
double csound::AscendingDistanceComparator::ascendingDistance (
            double a,
            double b )  [inline]
```

References divisionsPerOctave, csound::fundamentalDomainByPredicate(), and csound::Voicelead::pc().

Referenced by operator()().

6.5.2.2 operator()()

```
bool csound::AscendingDistanceComparator::operator() (
            double a,
            double b )  [inline]
```

References ascendingDistance(), csound::fundamentalDomainByPredicate(), and origin.

6.5.3 Field Documentation

6.5.3.1 divisionsPerOctave

```
size_t csound::AscendingDistanceComparator::divisionsPerOctave
```

Referenced by ascendingDistance().

6.5.3.2 origin

```
double csound::AscendingDistanceComparator::origin
```

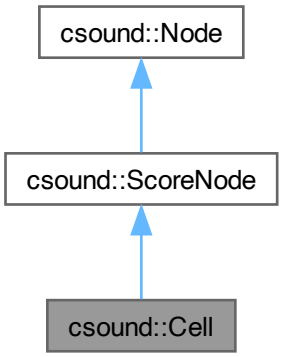
Referenced by operator()().

6.6 csound::Cell Class Reference

Score node that simplifies building up structures of motivic cells, and incrementally transforming them, as in Minimalism.

```
#include <Cell.hpp>
```

Inheritance diagram for csound::Cell:



Public Member Functions

- virtual void addChild (Node *node)
  Adds an immediate child Node to this.
- Cell ()
- virtual size_t childCount () const
  Returns the number of immediate children of this.
- virtual void clear ()
  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()
  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)
  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &collectingScore)

Optionally generate notes into the score.

- virtual Node * getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual double getDurationSeconds () const
- virtual std::string getImportFilename () const
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual bool getRelativeDuration () const
- virtual int getRepeatCount () const
- virtual Score & getScore ()
- virtual void setDurationSeconds (double value)
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void setImportFilename (std::string filename)
- virtual void setRelativeDuration (bool value)
- virtual void setRepeatCount (int count)
- virtual void transform (Score &score)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.
- virtual ~Cell ()

Data Fields

- std::vector< Node * > children

  Child Nodes, if any.
- double duration

  If not 0, the score is rescaled to this duration.
- double durationSeconds

  If relativeDuraton is true, then this time is added to the duration of the Nth repetition in order to obtain the starting time of the N + 1th repetition; if relativeDuration is false, then this time is added to the starting time of the Nth repetition in order to obtain the starting time of the N + 1th repetition.
- std::string importFilename
- bool relativeDuration

  Indicates whether the durationSeconds of this cell is added to the duration of the notes produced by the child nodes of this (true) at each repetition, or is simply the duration of the cell (false), in which case the notes of the Nth repetition may (or may not) overlap the notes of the N + 1th repetition.
- int repeatCount

  The number of times to repeat the notes produced by the child nodes of this.

Protected Attributes

- Eigen::MatrixXd localCoordinates
- Score score

6.6.1 Detailed Description

Score node that simplifies building up structures of motivic cells, and incrementally transforming them, as in Minimalism.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 Cell()

```
csound::Cell::Cell ( )
```

6.6.2.2 ~Cell()

```
csound::Cell::~Cell ( )  [virtual]
```

6.6.3 Member Function Documentation

6.6.3.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.6.3.2 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.6.3.3 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.6.3.4 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.6.3.5 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.6.3.6 generate()

```
void csound::ScoreNode::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented from csound::Node.

Reimplemented in csound::ExternalNode, and csound::MCRM.

References csound::ScoreNode::duration, csound::fundamentalDomainByPredicate(), csound::Score::getCsoundScoreHeader(), csound::ScoreNode::importFilename, csound::Score::load(), csound::Score::process(), csound::ScoreNode::score, csound::Score::setDuration(), and csound::Score::sort().

Referenced by csound::MCRM::generate().

6.6.3.7 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.6.3.8 getDurationSeconds()

```
virtual double csound::Cell::getDurationSeconds ( ) const  [inline], [virtual]
```

6.6.3.9 getImportFilename()

```
virtual std::string csound::Cell::getImportFilename ( ) const  [inline], [virtual]
```

6.6.3.10 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.6.3.11 getRelativeDuration()

```
virtual bool csound::Cell::getRelativeDuration ( ) const  [inline], [virtual]
```

6.6.3.12 getRepeatCount()

```
virtual int csound::Cell::getRepeatCount ( ) const  [inline], [virtual]
```

6.6.3.13 getScore()

```
Score & csound::ScoreNode::getScore ( )  [virtual], [inherited]
```

References csound::ScoreNode::score.

Referenced by main().

6.6.3.14 setDurationSeconds()

```
virtual void csound::Cell::setDurationSeconds (
            double value )  [inline], [virtual]
```

6.6.3.15 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.6.3.16 setImportFilename()

```
virtual void csound::Cell::setImportFilename (
            std::string filename )  [inline], [virtual]
```

6.6.3.17 setRelativeDuration()

```
virtual void csound::Cell::setRelativeDuration (
            bool value )  [inline], [virtual]
```

6.6.3.18 setRepeatCount()

```
virtual void csound::Cell::setRepeatCount (
            int count )  [inline], [virtual]
```

6.6.3.19 transform()

```
void csound::Cell::transform (
            Score & score_from_children )  [virtual]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented from csound::Node.

References durationSeconds, csound::Event::getTime(), csound::System::message(), relativeDuration, repeatCount, csound::ScoreNode::score, csound::Event::setTime(), and csound::Score::sort().

6.6.3.20 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord and csound::Node::transform().

6.6.4 Field Documentation

6.6.4.1 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.6.4.2 duration

```
double csound::ScoreNode::duration  [inherited]
```

If not 0, the score is rescaled to this duration.

Referenced by csound::ScoreNode::generate(), csound::ExternalNode::generateLocally(), and csound::Stack::getDuration().

6.6.4.3 durationSeconds

```
double csound::Cell::durationSeconds
```

If relativeDuraton is true, then this time is added to the duration of the Nth repetition in order to obtain the starting time of the N + 1th repetition; if relativeDuration is false, then this time is added to the starting time of the Nth repetition in order to obtain the starting time of the N + 1th repetition.

Referenced by transform().

6.6.4.4 importFilename

```
std::string csound::ScoreNode::importFilename  [inherited]
```

Referenced by csound::ScoreNode::generate().

6.6.4.5 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.6.4.6 relativeDuration

```
bool csound::Cell::relativeDuration
```

Indicates whether the durationSeconds of this cell is added to the duration of the notes produced by the child nodes of this (true) at each repetition, or is simply the duration of the cell (false), in which case the notes of the Nth repetition may (or may not) overlap the notes of the N + 1th repetition.

Referenced by transform().

6.6.4.7 repeatCount

```
int csound::Cell::repeatCount
```

The number of times to repeat the notes produced by the child nodes of this.

Referenced by transform().

6.6.4.8 score

```
Score csound::ScoreNode::score  [protected], [inherited]
```

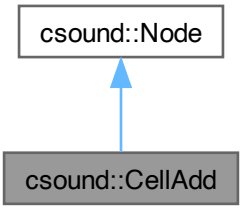
Referenced by csound::StrangeAttractor::evaluateAttractor(), csound::ExternalNode::generate(), csound::ScoreNode::generate(), csound::MCRM::generate(), csound::ExternalNode::generateLocally(), csound::ImageToScore2::generateLocally(), csound::Lindenmayer::generateLocally(), csound::Rescale::getRescale(), csound::ScoreNode::getScore(), csound::Lindenmayer::interpre csound::MCRM::iterate(), csound::StrangeAttractor::iterate_without_rendering(), csound::KMeansMCRM::means_to_notes(), csound::ImageToScore2::pixel_to_event(), csound::StrangeAttractor::render(), csound::Rescale::Rescale(), csound::Rescale::setRescale(transform(), csound::Rescale::transform(), csound::CMaskNode::translate_to_silence(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Lindenmayer::updateActual().

6.7 csound::CellAdd Class Reference

The indicated factor is added to the indicated dimension of each note produced by the child nodes of this, beginning at the start index and proceeding up to but not including the end index, at the specified stride.

```
#include <Cell.hpp>
```

Inheritance diagram for csound::CellAdd:



Public Member Functions

- virtual void add (Event::Dimensions dimension, double value, size_t start, size_t end, size_t stride)
- virtual void addChild (Node *node)

  Adds an immediate child Node to this.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)

Returns a reference to the indicated element of the local transformation of coordinate system.

- virtual void generate (Score &score_from_this)

  Optionally generate notes into the score.
- virtual Node * getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void transform (Score &score)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

Data Fields

- std::vector< Node * > children

  Child Nodes, if any.

Protected Attributes

- Eigen::MatrixXd localCoordinates

6.7.1 Detailed Description

The indicated factor is added to the indicated dimension of each note produced by the child nodes of this, beginning at the start index and proceeding up to but not including the end index, at the specified stride.

Each dimension may have its own factor.

6.7.2 Member Function Documentation

6.7.2.1 add()

```
void csound::CellAdd::add (
            Event::Dimensions dimension,
            double value,
            size_t start,
            size_t end,
            size_t stride )  [virtual]
```

6.7.2.2 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.7.2.3 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.7.2.4 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.7.2.5 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.7.2.6 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.7.2.7 generate()

```
void csound::Node::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented in csound::ExternalNode, csound::ScoreNode, csound::ChordLindenmayer, csound::MCRM, csound::Generator, csound::Random, csound::LispGenerator, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.7.2.8 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.7.2.9 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.7.2.10 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.7.2.11 transform()

```
void csound::CellAdd::transform (
            Score & score_from_children )  [virtual]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented from csound::Node.

6.7.2.12 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord
and csound::Node::transform().

6.7.3 Field Documentation

6.7.3.1 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.7.3.2 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

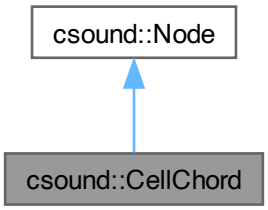
Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.8 csound::CellChord Class Reference

Notes produced by the child nodes of this are conformed to the chord, starting at the indicated start index, up to but not including the end index, at the indicated stride.

```
#include <Cell.hpp>
```

Inheritance diagram for csound::CellChord:



Public Member Functions

- virtual void addChild (Node *node)

  Adds an immediate child Node to this.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void chord (const Chord &chord, size_t start, size_t end, size_t stride)
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &score_from_this)

  Optionally generate notes into the score.
- virtual Node * getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void transform (Score &score)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

Data Fields

- std::vector< Node * > children

  Child Nodes, if any.

Protected Attributes

- Eigen::MatrixXd localCoordinates

6.8.1 Detailed Description

Notes produced by the child nodes of this are conformed to the chord, starting at the indicated start index, up to but not including the end index, at the indicated stride.

6.8.2 Member Function Documentation

6.8.2.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.8.2.2 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.8.2.3 chord()

```
void csound::CellChord::chord (
            const Chord & chord,
            size_t start,
            size_t end,
            size_t stride )  [virtual]
```

6.8.2.4 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.8.2.5 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.8.2.6 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.8.2.7 generate()

```
void csound::Node::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented in csound::ExternalNode, csound::ScoreNode, csound::ChordLindenmayer, csound::MCRM, csound::Generator, csound::Random, csound::LispGenerator, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.8.2.8 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.8.2.9 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.8.2.10 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.8.2.11 transform()

```
void csound::CellChord::transform (
            Score & score_from_children )  [virtual]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented from csound::Node.

References csound::conformToChord().

6.8.2.12 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord
and csound::Node::transform().

6.8.3 Field Documentation

6.8.3.1 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.8.3.2 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

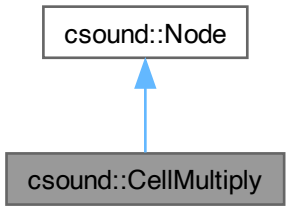
Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.9 csound::CellMultiply Class Reference

The indicated dimension of each note produced by the child nodes of this, beginning at the start index and proceeding up to but not including the end index, at the specified stride, is multiplied by the indicated factor.

```
#include <Cell.hpp>
```

Inheritance diagram for csound::CellMultiply:



Public Member Functions

- virtual void addChild (Node ∗node)

  Adds an immediate child Node to this.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &score_from_this)

  Optionally generate notes into the score.
- virtual Node ∗ getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual void multiply (Event::Dimensions dimension, double value, size_t start, size_t end, size_t stride)
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void transform (Score &score)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

Data Fields

- std::vector< Node ∗ > children

  Child Nodes, if any.

Protected Attributes

- Eigen::MatrixXd localCoordinates

6.9.1 Detailed Description

The indicated dimension of each note produced by the child nodes of this, beginning at the start index and proceeding up to but not including the end index, at the specified stride, is multiplied by the indicated factor.

Each dimension may have its own factor.

6.9.2 Member Function Documentation

6.9.2.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.9.2.2 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.9.2.3 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.9.2.4 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.9.2.5 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.9.2.6 generate()

```
void csound::Node::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented in csound::ExternalNode, csound::ScoreNode, csound::ChordLindenmayer, csound::MCRM, csound::Generator, csound::Random, csound::LispGenerator, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.9.2.7 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.9.2.8 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.9.2.9 multiply()

```
void csound::CellMultiply::multiply (
            Event::Dimensions dimension,
            double value,
            size_t start,
            size_t end,
            size_t stride )  [virtual]
```

6.9.2.10 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.9.2.11 transform()

```
void csound::CellMultiply::transform (
            Score & score_from_children )  [virtual]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented from csound::Node.

6.9.2.12 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord
and csound::Node::transform().

6.9.3 Field Documentation

6.9.3.1 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.9.3.2 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

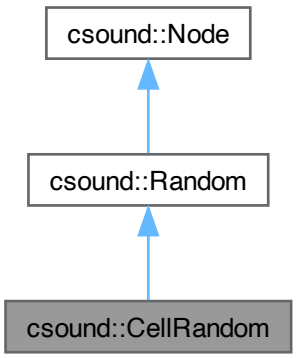
Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.10 csound::CellRandom Class Reference

Notes produced by the child nodes of this, starting at the indicated start index, up to but not including the indicated end index, at the indicated stride, have added to them a random variable from the indicated distribution, rescaled to the indicated minimum and range.

```
#include <Cell.hpp>
```

Inheritance diagram for csound::CellRandom:



Public Member Functions

- virtual void addChild (Node ∗node)

  Adds an immediate child Node to this.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual void createDistribution (std::string distribution)
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &score)

  Optionally generate notes into the score.
- virtual Node ∗ getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual Eigen::MatrixXd getRandomCoordinates ()
- virtual void random (const std::string &distribution, Event::Dimensions dimension, size_t start, size_t end, size_t stride)
- virtual double sample ()
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void transform (Score &score)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

Static Public Member Functions

- static void seed (int s)

Data Fields

- double a
- double b
- double c
- std::vector< Node ∗ > children

  Child Nodes, if any.
- int column
- int eventCount
- bool incrementTime
- double Lambda
- double maximum
- double mean
- double minimum
- double q
- int row
- double sigma

Static Public Attributes

- static std::mt19937 mersenneTwister

Protected Attributes

- std::bernoulli_distribution bernoulli_distribution_generator
- std::exponential_distribution exponential_distribution_generator
- void * generator_
- std::geometric_distribution geometric_distribution_generator
- Eigen::MatrixXd localCoordinates
- std::lognormal_distribution lognormal_distribution_generator
- std::normal_distribution normal_distribution_generator
- std::uniform_int_distribution< std::int64_t > uniform_int_generator
- std::uniform_real_distribution uniform_real_generator
- std::uniform_int_distribution< std::int32_t > uniform_smallint_generator

6.10.1 Detailed Description

Notes produced by the child nodes of this, starting at the indicated start index, up to but not including the indicated end index, at the indicated stride, have added to them a random variable from the indicated distribution, rescaled to the indicated minimum and range.

Parameters for the random variable are set as for the base Random node.

6.10.2 Member Function Documentation

6.10.2.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.10.2.2 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.10.2.3 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.10.2.4 createDistribution()

```
void csound::Random::createDistribution (
            std::string distribution )  [virtual], [inherited]
```

References csound::Random::bernoulli_distribution_generator, csound::Random::distribution, csound::Random::exponential_distribution_ csound::Random::generator_, csound::Random::geometric_distribution_generator, csound::Random::Lambda, csound::Random::lognormal_distribution_generator, csound::Random::maximum, csound::Random::mean, csound::Random::minimum, csound::Random::normal_distribution_generator, csound::Random::q, csound::Random::sigma, csound::Random::uniform_int_generator csound::Random::uniform_real_generator, and csound::Random::uniform_smallint_generator.

Referenced by csound::Random::generate(), csound::StrangeAttractor::StrangeAttractor(), transform(), and csound::Random::transform().

6.10.2.5 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.10.2.6 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.10.2.7 generate()

```
void csound::Random::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented from csound::Node.

References csound::Random::createDistribution(), csound::Random::distribution, csound::Random::eventCount, csound::fundamentalDomainByPredicate(), csound::Random::getRandomCoordinates(), and csound::Random::incrementTime.

6.10.2.8 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.10.2.9 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.10.2.10 getRandomCoordinates()

```
Eigen::MatrixXd csound::Random::getRandomCoordinates ( )  [virtual], [inherited]
```

References csound::fundamentalDomainByPredicate(), csound::Node::getLocalCoordinates(), csound::Event::HOMOGENEITY, and csound::Random::sample().

Referenced by csound::Random::generate(), and csound::Random::transform().

6.10.2.11 random()

```
void csound::CellRandom::random (
            const std::string & distribution,
            Event::Dimensions dimension,
            size_t start,
            size_t end,
            size_t stride )  [virtual]
```

6.10.2.12 sample()

```
double csound::Random::sample ( )  [virtual], [inherited]
```

References csound::Random::bernoulli_distribution_generator, csound::Random::exponential_distribution_generator, csound::Random::generator_, csound::Random::geometric_distribution_generator, csound::Random::lognormal_distribution_generator, csound::Random::mersenneTwister, csound::Random::normal_distribution_generator, csound::Random::uniform_int_generator, csound::Random::uniform_real_generator, and csound::Random::uniform_smallint_generator.

Referenced by csound::StrangeAttractor::calculateFractalDimension(), csound::StrangeAttractor::codeRandomize(), csound::Random::getRandomCoordinates(), csound::StrangeAttractor::render(), csound::StrangeAttractor::shuffleRandomNumbers(), and transform().

6.10.2.13 seed()

```
void csound::Random::seed (
            int s )  [static], [inherited]
```

References csound::Random::mersenneTwister.

6.10.2.14 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.10.2.15 transform()

```
void csound::CellRandom::transform (
            Score & score_from_children )  [virtual]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented from csound::Random.

References csound::Random::createDistribution(), and csound::Random::sample().

6.10.2.16 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord and csound::Node::transform().

6.10.3 Field Documentation

6.10.3.1 a

```
double csound::Random::a  [inherited]
```

6.10.3.2 b

```
double csound::Random::b  [inherited]
```

6.10.3.3 bernoulli_distribution_generator

```
std::bernoulli_distribution csound::Random::bernoulli_distribution_generator  [protected], [inherited]
```

Referenced by csound::Random::createDistribution(), and csound::Random::sample().

6.10.3.4 c

```
double csound::Random::c  [inherited]
```

6.10.3.5 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.10.3.6 column

```
int csound::Random::column  [inherited]
```

6.10.3.7 eventCount

```
int csound::Random::eventCount  [inherited]
```

Referenced by csound::Random::generate(), and csound::Random::transform().

6.10.3.8 exponential_distribution_generator

```
std::exponential_distribution csound::Random::exponential_distribution_generator  [protected],
[inherited]
```

Referenced by csound::Random::createDistribution(), and csound::Random::sample().

6.10.3.9 generator_

```
void* csound::Random::generator_  [protected], [inherited]
```

Referenced by csound::Random::createDistribution(), and csound::Random::sample().

6.10.3.10 geometric_distribution_generator

```
std::geometric_distribution csound::Random::geometric_distribution_generator  [protected], [inherited]
```

Referenced by csound::Random::createDistribution(), and csound::Random::sample().

6.10.3.11 incrementTime

```
bool csound::Random::incrementTime  [inherited]
```

Referenced by csound::Random::generate().

6.10.3.12 Lambda

```
double csound::Random::Lambda  [inherited]
```

Referenced by csound::Random::createDistribution().

6.10.3.13 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.10.3.14 lognormal_distribution_generator

```
std::lognormal_distribution csound::Random::lognormal_distribution_generator  [protected], [inherited]
```

Referenced by csound::Random::createDistribution(), and csound::Random::sample().

6.10.3.15 maximum

```
double csound::Random::maximum  [inherited]
```

Referenced by csound::Random::createDistribution().

6.10.3.16 mean

```
double csound::Random::mean  [inherited]
```

Referenced by csound::Random::createDistribution().

6.10.3.17 mersenneTwister

```
std::mt19937 csound::Random::mersenneTwister  [static], [inherited]
```

Referenced by csound::Random::sample(), csound::Random::seed(), and csound::CellShuffle::transform().

6.10.3.18 minimum

```
double csound::Random::minimum  [inherited]
```

Referenced by csound::Random::createDistribution().

6.10.3.19 normal_distribution_generator

```
std::normal_distribution csound::Random::normal_distribution_generator  [protected], [inherited]
```

Referenced by csound::Random::createDistribution(), and csound::Random::sample().

6.10.3.20 q

```
double csound::Random::q  [inherited]
```

Referenced by csound::Random::createDistribution().

6.10.3.21 row

```
int csound::Random::row  [inherited]
```

6.10.3.22 sigma

```
double csound::Random::sigma  [inherited]
```

Referenced by csound::Random::createDistribution().

6.10.3.23 uniform_int_generator

```
std::uniform_int_distribution<std::int64_t> csound::Random::uniform_int_generator  [protected],
[inherited]
```

Referenced by csound::Random::createDistribution(), and csound::Random::sample().

6.10.3.24 uniform_real_generator

```
std::uniform_real_distribution csound::Random::uniform_real_generator  [protected], [inherited]
```

Referenced by csound::Random::createDistribution(), and csound::Random::sample().

6.10.3.25 uniform_smallint_generator

```
std::uniform_int_distribution<std::int32_t> csound::Random::uniform_smallint_generator  [protected],
[inherited]
```

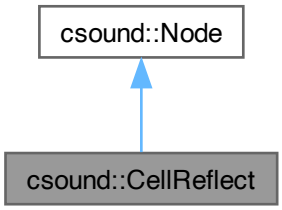
Referenced by csound::Random::createDistribution(), and csound::Random::sample().

6.11 csound::CellReflect Class Reference

The indicated dimension of each note produced by the child nodes of this, beginning at the start index and proceeding up to but not including the end index, at the specified stride, is reflected (i.e.

```
#include <Cell.hpp>
```

Inheritance diagram for csound::CellReflect:



Public Member Functions

- virtual void addChild (Node *node)
  Adds an immediate child Node to this.
- virtual size_t childCount () const
  Returns the number of immediate children of this.
- virtual void clear ()
  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()
  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)
  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &score_from_this)
  Optionally generate notes into the score.
- virtual Node * getChild (size_t index)
  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const

Returns the local transformation of coordinate system.

- virtual void reflect (Event::Dimensions dimension, double value, size_t start, size_t end, size_t stride)
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void transform (Score &score)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

Data Fields

- std::vector< Node * > children

  Child Nodes, if any.

Protected Attributes

- Eigen::MatrixXd localCoordinates

6.11.1 Detailed Description

The indicated dimension of each note produced by the child nodes of this, beginning at the start index and proceeding up to but not including the end index, at the specified stride, is reflected (i.e.

inverted) around the indicated center.

6.11.2 Member Function Documentation

6.11.2.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.11.2.2 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.11.2.3 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.11.2.4 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.11.2.5 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.11.2.6 generate()

```
void csound::Node::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented in csound::ExternalNode, csound::ScoreNode, csound::ChordLindenmayer, csound::MCRM, csound::Generator, csound::Random, csound::LispGenerator, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.11.2.7 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.11.2.8 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.11.2.9 reflect()

```
void csound::CellReflect::reflect (
            Event::Dimensions dimension,
            double value,
            size_t start,
            size_t end,
            size_t stride )  [virtual]
```

6.11.2.10 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.11.2.11 transform()

```
void csound::CellReflect::transform (
            Score & score_from_children )  [virtual]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented from csound::Node.

6.11.2.12 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord
and csound::Node::transform().

6.11.3 Field Documentation

6.11.3.1 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.11.3.2 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

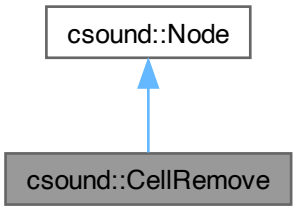
Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.12 csound::CellRemove Class Reference

Notes are removed from the notes produced by the child nodes of this, beginning at the indicated start index, up to but not including the end index, at the indicated stride.

```
#include <Cell.hpp>
```

Inheritance diagram for csound::CellRemove:



Public Member Functions

- virtual void addChild (Node ∗node)

  Adds an immediate child Node to this.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &score_from_this)

  Optionally generate notes into the score.
- virtual Node ∗ getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual void remove (size_t start, size_t end, size_t stride)
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void transform (Score &score)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

Data Fields

- std::vector< Node ∗ > children

  Child Nodes, if any.

Protected Attributes

- Eigen::MatrixXd localCoordinates

6.12.1 Detailed Description

Notes are removed from the notes produced by the child nodes of this, beginning at the indicated start index, up to but not including the end index, at the indicated stride.

The times of the child notes are adjusted to close the gap, i.e. the times of the child notes are rescaled to close gaps resulting from the deleted notes.

6.12.2 Member Function Documentation

6.12.2.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.12.2.2 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.12.2.3 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.12.2.4 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.12.2.5 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.12.2.6 generate()

```
void csound::Node::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented in csound::ExternalNode, csound::ScoreNode, csound::ChordLindenmayer, csound::MCRM, csound::Generator, csound::Random, csound::LispGenerator, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.12.2.7 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.12.2.8 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.12.2.9 remove()

```
void csound::CellRemove::remove (
            size_t start,
            size_t end,
            size_t stride )  [virtual]
```

6.12.2.10 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.12.2.11 transform()

```
void csound::CellRemove::transform (
            Score & score_from_children )  [virtual]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented from csound::Node.

References csound::Event::TIME.

6.12.2.12 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord
and csound::Node::transform().

6.12.3 Field Documentation

6.12.3.1 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.12.3.2 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

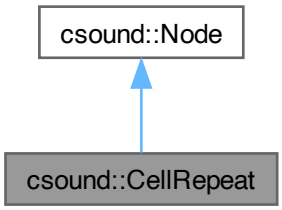
Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.13 csound::CellRepeat Class Reference

All notes produced by child nodes are repeated for the specified number of iterations, beginning at the start index and proceeding up to but not including the end index, at the specified stride.

```
#include <Cell.hpp>
```

Inheritance diagram for csound::CellRepeat:



Public Member Functions

- virtual void addChild (Node ∗node)

  Adds an immediate child Node to this.
- CellRepeat ()

  All notes produced by child nodes are repeated for the specified number of iterations, beginning at the start index and proceeding up to but not including the end index, at the specified stride.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &score_from_this)

  Optionally generate notes into the score.
- virtual Node ∗ getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual void repeat (size_t iterations, double duration, bool absolute_duration, size_t start, size_t end, size_t stride)
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void transform (Score &score)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.
- virtual ∼CellRepeat ()

Data Fields

- std::vector< Node ∗ > children

  Child Nodes, if any.

Protected Attributes

- Eigen::MatrixXd localCoordinates

6.13.1 Detailed Description

All notes produced by child nodes are repeated for the specified number of iterations, beginning at the start index and proceeding up to but not including the end index, at the specified stride.

If absolute_duration is true, then the next repetition occurs after that duration; if false, then the indicated duration is added to the total duration of the repeated notes.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 CellRepeat()

```
csound::CellRepeat::CellRepeat ( )
```

All notes produced by child nodes are repeated for the specified number of iterations, beginning at the start index and proceeding up to but not including the end index, at the specified stride.

If absolute_duration is true, then the next repetition occurs after that duration; if false, then the indicated duration is added to the total duration of the repeated notes.

6.13.2.2 ~CellRepeat()

```
csound::CellRepeat::~CellRepeat ( )  [virtual]
```

6.13.3 Member Function Documentation

6.13.3.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.13.3.2 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.13.3.3 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.13.3.4 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.13.3.5 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.13.3.6 generate()

```
void csound::Node::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented in csound::ExternalNode, csound::ScoreNode, csound::ChordLindenmayer, csound::MCRM, csound::Generator, csound::Random, csound::LispGenerator, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.13.3.7 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.13.3.8 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.13.3.9 repeat()

```
void csound::CellRepeat::repeat (
            size_t iterations,
            double duration,
            bool absolute_duration,
            size_t start,
            size_t end,
            size_t stride )  [virtual]
```

6.13.3.10 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.13.3.11 transform()

```
void csound::CellRepeat::transform (
            Score & score_from_children )  [virtual]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented from csound::Node.

References csound::System::inform(), and csound::Score::sort().

6.13.3.12 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord and csound::Node::transform().

6.13.4 Field Documentation

6.13.4.1 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.13.4.2 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

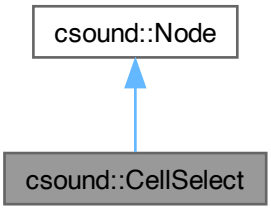
Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.14 csound::CellSelect Class Reference

The notes produced by the child nodes of this are returned as sampled from the indicated start index, up to but not including the indicated end index, at the indicated stride.

`#include <Cell.hpp>`

Inheritance diagram for csound::CellSelect:



Public Member Functions

- virtual void addChild (Node *node)

  Adds an immediate child Node to this.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &score_from_this)

  Optionally generate notes into the score.
- virtual Node * getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual void select (size_t start, size_t end, size_t stride)
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void transform (Score &score)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

Data Fields

- std::vector< Node * > children

  Child Nodes, if any.

Protected Attributes

- Eigen::MatrixXd localCoordinates

6.14.1 Detailed Description

The notes produced by the child nodes of this are returned as sampled from the indicated start index, up to but not including the indicated end index, at the indicated stride.

6.14.2 Member Function Documentation

6.14.2.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.14.2.2 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.14.2.3 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.14.2.4 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.14.2.5 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.14.2.6 generate()

```
void csound::Node::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented in csound::ExternalNode, csound::ScoreNode, csound::ChordLindenmayer, csound::MCRM, csound::Generator, csound::Random, csound::LispGenerator, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.14.2.7 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.14.2.8 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.14.2.9 select()

```
void csound::CellSelect::select (
            size_t start,
            size_t end,
            size_t stride )  [virtual]
```

6.14.2.10 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.14.2.11 transform()

```
void csound::CellSelect::transform (
            Score & score_from_children )  [virtual]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented from csound::Node.

6.14.2.12 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord and csound::Node::transform().

6.14.3 Field Documentation

6.14.3.1 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.14.3.2 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

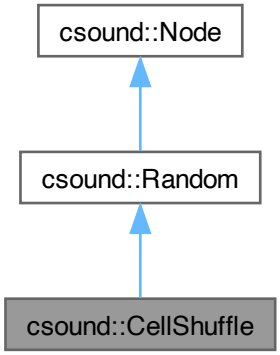
Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.15 csound::CellShuffle Class Reference

Notes produced by the child nodes of this, starting at the indicated start index, up to but not including the indicated end index, at the indicated stride, are randomly shuffled as to time.

`#include <Cell.hpp>`

Inheritance diagram for csound::CellShuffle:



Public Member Functions

- virtual void addChild (Node *node)
  Adds an immediate child Node to this.
- virtual size_t childCount () const
  Returns the number of immediate children of this.
- virtual void clear ()
  Recursively clears all child Nodes of this.
- virtual void createDistribution (std::string distribution)
- virtual Eigen::MatrixXd createTransform ()
  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)
  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &score)
  Optionally generate notes into the score.
- virtual Node * getChild (size_t index)
  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const
  Returns the local transformation of coordinate system.
- virtual Eigen::MatrixXd getRandomCoordinates ()
- virtual double sample ()

- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void shuffle (size_t start, size_t end, size_t stride)
- virtual void transform (Score &score)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

Static Public Member Functions

- static void seed (int s)

Data Fields

- double a
- double b
- double c
- std::vector< Node * > children

  Child Nodes, if any.
- int column
- std::string distribution
- int eventCount
- bool incrementTime
- double Lambda
- double maximum
- double mean
- double minimum
- double q
- int row
- double sigma

Static Public Attributes

- static std::mt19937 mersenneTwister

Protected Attributes

- std::bernoulli_distribution bernoulli_distribution_generator
- std::exponential_distribution exponential_distribution_generator
- void * generator_
- std::geometric_distribution geometric_distribution_generator
- Eigen::MatrixXd localCoordinates
- std::lognormal_distribution lognormal_distribution_generator
- std::normal_distribution normal_distribution_generator
- std::uniform_int_distribution< std::int64_t > uniform_int_generator
- std::uniform_real_distribution uniform_real_generator
- std::uniform_int_distribution< std::int32_t > uniform_smallint_generator

6.15.1 Detailed Description

Notes produced by the child nodes of this, starting at the indicated start index, up to but not including the indicated end index, at the indicated stride, are randomly shuffled as to time.

6.15.2 Member Function Documentation

6.15.2.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.15.2.2 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.15.2.3 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.15.2.4 createDistribution()

```
void csound::Random::createDistribution (
            std::string distribution )  [virtual], [inherited]
```

References csound::Random::bernoulli_distribution_generator, csound::Random::distribution, csound::Random::exponential_distribution_, csound::Random::generator_, csound::Random::geometric_distribution_generator, csound::Random::Lambda, csound::Random::lognormal_distribution_generator, csound::Random::maximum, csound::Random::mean, csound::Random::minimum, csound::Random::normal_distribution_generator, csound::Random::q, csound::Random::sigma, csound::Random::uniform_int_generator, csound::Random::uniform_real_generator, and csound::Random::uniform_smallint_generator.

Referenced by csound::Random::generate(), csound::StrangeAttractor::StrangeAttractor(), csound::CellRandom::transform(), and csound::Random::transform().

6.15.2.5 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.15.2.6 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.15.2.7 generate()

```
void csound::Random::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented from csound::Node.

References csound::Random::createDistribution(), csound::Random::distribution, csound::Random::eventCount, csound::fundamentalDomainByPredicate(), csound::Random::getRandomCoordinates(), and csound::Random::incrementTime.

6.15.2.8 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.15.2.9 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.15.2.10 getRandomCoordinates()

```
Eigen::MatrixXd csound::Random::getRandomCoordinates ( )  [virtual], [inherited]
```

References csound::fundamentalDomainByPredicate(), csound::Node::getLocalCoordinates(), csound::Event::HOMOGENEITY, and csound::Random::sample().

Referenced by csound::Random::generate(), and csound::Random::transform().

6.15.2.11 sample()

```
double csound::Random::sample ( )  [virtual], [inherited]
```

References csound::Random::bernoulli_distribution_generator, csound::Random::exponential_distribution_generator, csound::Random::generator_, csound::Random::geometric_distribution_generator, csound::Random::lognormal_distribution_generator, csound::Random::mersenneTwister, csound::Random::normal_distribution_generator, csound::Random::uniform_int_generator, csound::Random::uniform_real_generator, and csound::Random::uniform_smallint_generator.

Referenced by csound::StrangeAttractor::calculateFractalDimension(), csound::StrangeAttractor::codeRandomize(), csound::Random::getRandomCoordinates(), csound::StrangeAttractor::render(), csound::StrangeAttractor::shuffleRandomNumbers(), and csound::CellRandom::transform().

6.15.2.12 seed()

```
void csound::Random::seed (
            int s )  [static], [inherited]
```

References csound::Random::mersenneTwister.

6.15.2.13 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.15.2.14 shuffle()

```
void csound::CellShuffle::shuffle (
            size_t start,
            size_t end,
            size_t stride )  [virtual]
```

6.15.2.15 transform()

```
void csound::CellShuffle::transform (
            Score & score_from_children )  [virtual]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented from csound::Random.

References csound::Random::mersenneTwister, and csound::Score::setDuration().

6.15.2.16 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord and csound::Node::transform().

6.15.3 Field Documentation

6.15.3.1 a

```
double csound::Random::a  [inherited]
```

6.15.3.2 b

```
double csound::Random::b  [inherited]
```

6.15.3.3 bernoulli_distribution_generator

```
std::bernoulli_distribution csound::Random::bernoulli_distribution_generator  [protected], [inherited]
```

Referenced by csound::Random::createDistribution(), and csound::Random::sample().

6.15.3.4 c

```
double csound::Random::c  [inherited]
```

6.15.3.5 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.15.3.6 column

```
int csound::Random::column  [inherited]
```

6.15.3.7 distribution

```
std::string csound::Random::distribution  [inherited]
```

Referenced by csound::Random::createDistribution(), csound::Random::generate(), csound::Random::Random(), and csound::Random::transform().

6.15.3.8 eventCount

```
int csound::Random::eventCount  [inherited]
```

Referenced by csound::Random::generate(), and csound::Random::transform().

6.15.3.9 exponential_distribution_generator

```
std::exponential_distribution csound::Random::exponential_distribution_generator  [protected],
[inherited]
```

Referenced by csound::Random::createDistribution(), and csound::Random::sample().

6.15.3.10 generator_

```
void* csound::Random::generator_  [protected], [inherited]
```

Referenced by csound::Random::createDistribution(), and csound::Random::sample().

6.15.3.11 geometric_distribution_generator

```
std::geometric_distribution csound::Random::geometric_distribution_generator  [protected], [inherited]
```

Referenced by csound::Random::createDistribution(), and csound::Random::sample().

6.15.3.12 incrementTime

```
bool csound::Random::incrementTime  [inherited]
```

Referenced by csound::Random::generate().

6.15.3.13 Lambda

```
double csound::Random::Lambda  [inherited]
```

Referenced by csound::Random::createDistribution().

6.15.3.14 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.15.3.15 lognormal_distribution_generator

```
std::lognormal_distribution csound::Random::lognormal_distribution_generator  [protected], [inherited]
```

Referenced by csound::Random::createDistribution(), and csound::Random::sample().

6.15.3.16 maximum

```
double csound::Random::maximum  [inherited]
```

Referenced by csound::Random::createDistribution().

6.15.3.17 mean

```
double csound::Random::mean  [inherited]
```

Referenced by csound::Random::createDistribution().

6.15.3.18 mersenneTwister

```
std::mt19937 csound::Random::mersenneTwister  [static], [inherited]
```

Referenced by csound::Random::sample(), csound::Random::seed(), and transform().

6.15.3.19 minimum

```
double csound::Random::minimum  [inherited]
```

Referenced by csound::Random::createDistribution().

6.15.3.20 normal_distribution_generator

```
std::normal_distribution csound::Random::normal_distribution_generator  [protected], [inherited]
```

Referenced by csound::Random::createDistribution(), and csound::Random::sample().

6.15.3.21 q

```
double csound::Random::q  [inherited]
```

Referenced by csound::Random::createDistribution().

6.15.3.22 row

```
int csound::Random::row  [inherited]
```

6.15.3.23 sigma

```
double csound::Random::sigma  [inherited]
```

Referenced by csound::Random::createDistribution().

6.15.3.24 uniform_int_generator

```
std::uniform_int_distribution<std::int64_t> csound::Random::uniform_int_generator  [protected],
[inherited]
```

Referenced by csound::Random::createDistribution(), and csound::Random::sample().

6.15.3.25 uniform_real_generator

```
std::uniform_real_distribution csound::Random::uniform_real_generator  [protected], [inherited]
```

Referenced by csound::Random::createDistribution(), and csound::Random::sample().

6.15.3.26 uniform_smallint_generator

```
std::uniform_int_distribution<std::int32_t> csound::Random::uniform_smallint_generator  [protected],
[inherited]
```

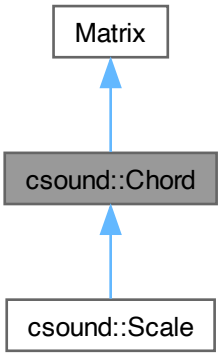
Referenced by csound::Random::createDistribution(), and csound::Random::sample().

6.16 csound::Chord Class Reference

Chords consist of simultaneously sounding pitches.

`#include <ChordSpaceBase.hpp>`

Inheritance diagram for csound::Chord:



Public Types

- enum {
  PITCH = 0 , DURATION = 1 , LOUDNESS = 2 , INSTRUMENT = 3 ,
  PAN = 4 , COUNT = 5 }

Public Member Functions

- virtual Chord a (int arpeggiation, double &resultPitch, int &resultVoice) const

  Returns the ith arpeggiation, current voice, and corresponding revoicing of the chord.
- virtual Chord ceiling (double g=1.) const

  Returns a new chord whose pitches are the ceilings of this chord's pitches, with respect to the generator of transposition g, which defaults to 1 semitone.
- virtual Chord center () const

  Returns the maximally even chord in the chord's space, e.g.
- Chord ()
- Chord (const Chord &other)
- Chord (const std::vector< double > &other)
- Chord (int size)
- virtual void clamp (double g=1.)

  Rounds the pitches in this chord to the nearest integer multiple of g, the generator of transposition.
- virtual Chord clone () const

- virtual bool contains (double pitch_) const

  Returns whether or not the chord contains the pitch.
- virtual size_t count (double pitch) const

  Returns the number of voices in this chord, the same as the number of dimensions in this chord space.
- virtual Chord cycle (int stride=1) const

  Returns a copy of the chord cyclically permuted by a stride, by default 1.
- virtual double distanceToOrigin () const

  Returns the Euclidean distance of this chord from its space's origin.
- virtual double distanceToUnisonDiagonal () const

  Returns the Euclidean distance from this chord to the unison diagonal of its chord space.
- virtual Chord eI (int opt_sector=0) const

  Returns the equivalent of the chord within a fundamental domain of inversional equivalence.
- virtual Chord eO () const

  Returns the equivalent of the chord within the representative fundamental domain of octave equivalence.
- virtual Chord eOP () const

  Returns the equivalent of the chord within the representative fundamental domain of octave and permutational equivalence.
- virtual Chord eOPI (int opt_sector=0) const

  Returns the equivalent of the chord within a fundamental domain of octave, permutational, and inversional equivalence.
- virtual Chord eOPT (int opt_sector=0) const

  Returns the equivalent of the chord within a fundamental domain of octave, permutational, and transpositional equivalence.
- virtual Chord eOPTI (int opt_sector=0) const

  Returns the equivalent of the chord within a fundamental domain of range, permutational, transpositional, and inversional equivalence.
- virtual Chord eOPTT (double g=1., int opt_sector=0) const

  Returns the equivalent of the chord within a fundamental domain of octave, permutational, and transpositional equivalence but in the equal temperament generated by g.
- virtual Chord eOPTTI (double g=1., int opt_sector=0) const

  Returns the equivalent of the chord within a fundamental domain of range, permutational, transpositional, and inversional equivalence but in the equal temperament generated by g.
- virtual Chord eOT () const

  Returns the equivalent of the chord within the representative fundamental domain of octave and transpositional equivalence.
- virtual Chord eOTT (double g=1.) const

  Returns the equivalent of the chord within a fundamental domain of octave and transpositional equivalence but in the equal temperament generated by g.
- virtual Chord eP () const

  Returns the equivalent of the chord within the representative fundamental domain of permutational equivalence.
- virtual Chord epcs () const

  Returns the equivalent of the chord under pitch-class equivalence, i.e.
- virtual Chord eppcs () const

  Returns the equivalent of the chord under pitch-class equivalence, i.e.
- virtual bool equals (const Chord &other) const

  Returns whether the voices of this chord equal the voices of the other.
- virtual Chord eR (double range) const

  Returns the equivalent of the chord within the representative fundamental domain of a range equivalence.
- virtual Chord eRP (double range) const

  Returns the equivalent of the chord within the representative fundamental domain of range and permutational equivalence.

- virtual Chord eRPI (double range, int opt_sector=0) const

  Returns the equivalent of the chord within a fundamental domain of range, permutational, and inversional equivalence.
- virtual Chord eRPT (double range, int opt_sector=0) const

  Returns the equivalent of the chord within a fundamental domain of range, permutational, and transpositional equivalence.
- virtual Chord eRPTI (double range, int opt_sector=0) const

  Returns the equivalent of the chord within the representative fundamental domain of range, permutational, transpositional, and inversional equivalence.
- virtual std::vector< Chord > eRPTs (double range=OCTAVE()) const

  Returns all equivalents of the chord within all fundamental domains of range, permutational, and transpositional equivalence.
- virtual Chord eRPTT (double range, double g=1., int opt_sector=0) const

  Returns the equivalent of the chord within a fundamental domain of range, permutational, and transpositional equivalence, in the equal temperament generated by g; the same as chord type.
- virtual Chord eRPTTI (double range, double g=1., int opt_sector=0) const

  Returns the equivalent of the chord within a fundamental domain of range, permutational, transpositional, and inversional equivalence.
- virtual std::vector< Chord > eRPTTs (double range, double g=1.) const

  Returns all equivalents of the chord within all fundamental domains of range, permutational, and transpositional equivalence in the equal temperament generated by g; equivalent to all inversions of the chord in the musician's sense.
- virtual Chord eT () const

  Returns the equivalent of the chord within a fundamental domain of range, permutational, transpositional, and inversional equivalence in the equal temperament generated by g; the same as set class.
- virtual Chord et () const

  Returns the equivalent of the chord within the fundamental domain of transposition to 0.
- virtual Chord eTT (double g=1.) const

  Returns the equivalent of the chord within the representative fundamental domain of transpositonal equivalence and the equal temperament generated by g, i.e., returns the chord transposed such that its layer is 0 or, under transposition, the positive layer closest to 0.
- virtual Chord floor () const

  Returns a new chord whose pitches are the floors of this chord's pitches.
- virtual void fromString (std::string text)

  Rebuilds the chord's pitches (only) from a line of text.
- virtual double getDuration (int voice=0) const
- virtual double getInstrument (int voice=0) const
- virtual double getLoudness (int voice=0) const
- virtual double getPan (int voice=0) const
- virtual double getPitch (int voice) const
- virtual double & getPitchReference (int voice)
- virtual bool greater (const Chord &other) const

  Returns whether the voices of this chord are greater than the voices of the other.
- virtual bool greater_equals (const Chord &other) const

  Returns whether the voices of this chord are greater than or equal to the voices of the other.
- virtual HyperplaneEquation hyperplane_equation (int opt_sector) const

  Returns the hyperplane equation for the inversion flat that evenly divides the fundamental domain in the indicated sector of the OPT cyclical region.
- virtual Chord I (double center=0.0) const

  Inverts the chord by another chord that is on the unison diagonal, by default the origin.
- virtual bool Iform (const Chord &Y, double g=1.) const

  Returns whether the chord is an inversional form of Y with interval size g.

- virtual std::string information () const

  Print much information about the chord including whether it is within important equivalence classes, or what its equivalents would be.
- virtual std::string information_debug (int opt_sector) const

  Print much information about the chord including whether it is within important equivalence classes, or what its equivalents would be.
- virtual std::string information_sector (int opt_sector) const

  Print much information about the chord including whether it is within important equivalence classes, or what its equivalents would be.
- virtual void initialize_sectors ()

  Initializes the fundamental domains (sectors) of the cyclical regions of OPT equivalence and OPTI equivalence, as well as the hyperplane equations that define the inversion flat in each OPT sector.
- virtual Chord inverse_prime_form () const

  Returns this chord as the inverse standard "prime form.".
- virtual bool is_compact (double range=12.) const

  Returns whether this chord has a compact voicing.
- virtual bool is_minor () const

  Returns whether this chord is "minor" in the sense of having the smallest "wraparound interval" of all its voicings.
- virtual bool is_opt_sector (int opt_sector=0) const

  Returns whether or not this chord lies within the indicated sector of the cyclical region of OPT fundamental domains.
- virtual bool is_opti_sector (int opti_sector=0) const

  Returns whether or not this chord lies within the indicated sector of the cyclical region of OPTI fundamental domains.
- virtual bool iseI (int opt_sector=0) const
- virtual bool iseI_chord (Chord *inverse, int opt_sector=0) const

  Returns whether the chord is within a fundamental domain of inversional equivalence.
- virtual bool iseO () const

  Returns whether the chord is within the representative fundamental domain of octave equivalence.
- virtual bool iseOP () const

  Returns whether the chord is within the representative fundamental domain of octave and permutational equivalence.
- virtual bool iseOPI (int opt_sector=0) const

  Returns whether the chord is within a fundamental domain of octave, permutational, and inversional equivalence.
- virtual bool iseOPT (int opt_sector=0) const

  Returns whether the chord is within a fundamental domainof octave, permutational, and transpositional equivalence.
- virtual bool iseOPTI (int opt_sector=0) const

  Returns whether the chord is within a fundamental domain of octave, permutational, transpositional, and inversional equivalence.
- virtual bool iseOPTT (double g=1., int opt_sector=0) const

  Returns whether the chord is within a fundamental domain of octave, permutational, and transpositional equivalence in the equal temperament generated by g.
- virtual bool iseOPTTI (double g=1., int opt_sector=0) const

  Returns whether the chord is within a fundamental domain of octave, permutational, transpositional, and inversional equivalence in the equal temperament generated by g.
- virtual bool iseOT () const

  Returns whether the chord is within the representative fundamental domain of octave and transpositional equivalence.
- virtual bool iseOTT (double g=1.) const

  Returns whether the chord is within the representative fundamental domain of octave and translational equivalence in the equal temperament generated by g.
- virtual bool iseP () const

  Returns whether the chord is within the representative fundamental domain of permutational equivalence.

- virtual bool isepcs () const

  Returns whether the chord is within the fundamental domain of pitch-class equivalence, i.e.
- virtual bool iseR (double range_) const

  Returns whether the chord is within the representative fundamental domain of the indicated range equivalence.
- virtual bool iseRP (double range) const

  Returns whether the chord is within the representative fundamental domain of range and permutational equivalence.
- virtual bool iseRPI (double range, int opt_sector=0) const

  Returns whether the chord is within a fundamental domain of range, permutational, and inversional equivalence.
- virtual bool iseRPT (double range, int opt_sector=0) const

  Returns whether the chord is within a fundamental domain of range, permutational, and transpositional equivalence.
- virtual bool iseRPTI (double range, int opt_sector=0) const

  Returns whether the chord is within a fundamental domain of range, permutational, transpositional, and inversional equivalence.
- virtual bool iseRPTT (double range, double g=1., int opt_sector=0) const

  Returns whether the chord is within a fundamental domain of range, permutational, and transpositional equivalence in the equal temperament generated by g.
- virtual bool iseRPTTI (double range, double g=1., int opt_sector=0) const

  Returns whether the chord is within a fundamental domain of range, permutational, transpositional, and inversional equivalence in the ' equal temperament generated by g.
- virtual bool iseRT (double range) const

  Returns whether the chord is within the representative fundamental domain of range and transpositional equivalence.
- virtual bool iseRTT (double range, double g=1.) const

  Returns whether the chord is within a fundamental domain of range and transpositional equivalence in the equal temperament generated by g.
- virtual bool iseT () const

  Returns whether the chord is within the representative fundamental domain of transpositional equivalence.
- virtual bool iset () const

  Returns whether the chord is within the fundamental domain of transposition to 0.
- virtual bool iseTT (double g=1.) const

  Returns whether the chord is within the representative fundamental domain of transpositional equivalence in the equal temperament generated by g.
- virtual Chord K () const

  Returns the chord inverted by the sum of its first two voices.
- virtual Chord K_range (double range) const
- virtual double layer () const

  Returns the sum of the pitches in the chord.
- virtual bool lesser (const Chord &other) const

  Returns whether the voices of this chord are less than the voices of the other.
- virtual bool lesser_equals (const Chord &other) const

  Returns whether the voices of this chord are less than or equal to the voices of the other.
- virtual std::vector< double > max () const

  Returns the highest pitch in the chord, and also the voice index of that pitch.
- virtual double maximumInterval () const

  Returns the maximum interval within the chord.
- virtual std::vector< double > min () const

  Returns the lowest pitch in the chord, and also the voice index of that pitch.
- virtual double minimumInterval () const

  Returns the minimum interval within the chord.

- virtual Chord move (int voice, double interval) const

  Move 1 voice of the chord.
- virtual std::string name () const

  Return the jazz-style name of the chord, if possible, or else a human-readable list of the voices in the chord.
- virtual Chord normal_form () const

  Returns this chord as its standard "normal form.".
- virtual Chord normal_order () const

  Returns this chord in standard "normal order." For a very clear explanation, see: https://www.↩ mta.ca/pc-set/pc-set_new/pages/page04/page04.html and http://openmusictheory.↩ com/normalOrder.html/.
- virtual Chord nrD () const

  Performs the dominant transformation (which is not a neo-Reimannian transformation).
- virtual Chord nrH () const

  Performs the neo-Riemannian hexatonic pole transformation.
- virtual Chord nrL () const

  Performs the neo-Riemannian Lettonwechsel transformation.
- virtual Chord nrN () const

  Performs the neo-Riemannian Nebenverwandt transformation.
- virtual Chord nrP () const

  Performs the neo-Riemannian parallel transformation.
- virtual Chord nrR () const

  Performs the neo-Riemannian parallel transformation.
- virtual Chord nrS () const

  Performs the neo-Riemannian Slide transformation.
- virtual operator std::vector< double > () const
- virtual Chord & operator= (const Chord &other)
- virtual Chord & operator= (const std::vector< double > &other)
- virtual std::vector< Chord > opt_domain (int sector) const

  Returns the vertices of the OPT fundamental domain for the indicated sector of the cyclical region.
- virtual std::vector< int > opt_domain_sectors () const

  Returns the zero-based index(s) of the sector(s) within the cyclical region of OPT fundamental domains to which the chord belongs.
- virtual std::vector< Chord > opti_domain (int sector) const

  Returns the vertices of the OPTI fundamental domain for the indicated sector of the cyclical region.
- virtual std::vector< int > opti_domain_sectors () const

  Returns the zero-based index(s) of the sector(s) within the cyclical region of OPTI fundamental domains to which the chord belongs.
- virtual Chord origin () const

  Returns the origin of the chord's space.
- virtual std::vector< Chord > permutations () const

  Returns the permutations of the pitches in a chord.
- virtual Chord prime_form () const

  Returns this chord as its standard "prime form.".
- virtual Chord Q (double x, const Chord &m, double g=1.) const

  Returns the contextual transposition of the chord by x with respect to m with minimum interval size g.
- virtual Chord reflect (int opt_sector) const

  Reflects the chord in the inversion flat of the indicated OPT domain sector.
- virtual void resize (size_t voiceN)

- virtual bool self_inverse (int opt_sector=0) const

  Returns whether or not this chord is invariant under reflection in the inversion flat of the indicated OPT sector.
- virtual void setDuration (double value, int voice=-1)
- virtual void setInstrument (double value, int voice=-1)
- virtual void setLoudness (double value, int voice=-1)
- virtual void setPan (double value, int voice=-1)
- virtual void setPitch (int voice, double value)
- virtual Chord T (double interval) const

  Transposes the chord by the indicated interval (may be a fraction).
- virtual Chord T_voiceleading (const Chord &voiceleading)

  Transposes the chord by the indicated voiceleading (passed as a Chord of directed intervals).
- virtual bool test (const char *caption="") const

  Tests the internal consistency of the predicates ("iseX") and transformations ("eX") of this chord, and prints a report.
- virtual bool Tform (const Chord &Y, double g=1.) const

  Returns whether the chord is a transpositional form of Y with interval size g.
- virtual std::string toString () const

  Returns a string representation of the chord's pitches (only).
- virtual Chord v (int direction=1) const

  Returns a copy of the chord 'inverted' in the musician's sense, i.e.
- virtual Chord voiceleading (const Chord &destination) const

  Returns the transpositions (as a Chord of directed intervals) that takes this chord to the destination chord.
- virtual size_t voices () const

  Returns the number of voices in this chord; that is, the number of dimensions in the chord space for this chord.
- virtual std::vector< Chord > voicings () const

  Returns all the 'inversions' (in the musician's sense) or octavewise revoicings of the chord.
- virtual ~Chord ()

Static Public Member Functions

- static std::map< int, std::vector< Chord > > & cyclical_regions_for_dimensionalities ()

  For each chord space of dimensions 3 <= n <= 12, there is one cyclical region of n fundamental domains of OPT equivalence.
- static std::map< int, std::vector< HyperplaneEquation > > & hyperplane_equations_for_opt_sectors ()

  For each chord space of dimensions 3 <= n <= 12, there are n fundamental domains (sectors) of OPT equivalence.
- static std::map< int, std::vector< std::vector< Chord > > > & opt_sectors_for_dimensionalities ()

  For each chord space of dimensions 3 <= n <= 12, there are n fundamental domains (sectors) of OPT equivalence.
- static std::map< int, std::vector< std::vector< Chord > > > & opt_simplexes_for_dimensionalities ()

  Returns a collection of vertices for the OPT fundamental domains; each has an added vertex to make a simplex for chord location.
- static std::map< int, std::vector< std::vector< Chord > > > & opti_sectors_for_dimensionalities ()

  For each chord space of dimensions 3 <= n <= 12, there are n fundamental domains (sectors) of OPTI equivalence.
- static std::map< int, std::vector< std::vector< Chord > > > & opti_simplexes_for_dimensionalities ()

  Returns a collection of vertices for the OPTI fundamental domains that have an added vertex to make a simplex for chord location.
- static double rownd (double x, int places=12)

  Rounds the value of x to the specified number of decimal places.

6.16.1 Detailed Description

Chords consist of simultaneously sounding pitches.

The pitches are represented as semitones with 0 at the origin and middle C as 60. Each voice also has a duration, velocity, channel, and pan. Eigen matrices are accessed (row, column) and stored as column vectors, so a Chord is accessed (voice (same as row), attribute).

6.16.2 Member Enumeration Documentation

6.16.2.1 anonymous enum

```
anonymous enum
```

Enumerator

PITCH	
DURATION	
LOUDNESS	
INSTRUMENT	
PAN	
COUNT	

6.16.3 Constructor & Destructor Documentation

6.16.3.1 Chord() [1/4]

```
csound::Chord::Chord ( )  [inline]
```

6.16.3.2 Chord() [2/4]

```
csound::Chord::Chord (
            int size )  [inline]
```

6.16.3.3 Chord() [3/4]

```
csound::Chord::Chord (
            const Chord & other )  [inline]
```

6.16.3.4 Chord() [4/4]

```
csound::Chord::Chord (
            const std::vector< double > & other )  [inline]
```

6.16.3.5 ~Chord()

```
csound::Chord::~Chord ( )  [inline], [virtual]
```

6.16.4 Member Function Documentation

6.16.4.1 a()

```
Chord csound::Chord::a (
            int arpeggiation,
            double & resultPitch,
            int & resultVoice ) const  [inline], [virtual]
```

Returns the ith arpeggiation, current voice, and corresponding revoicing of the chord.

Positive arpeggiations start with the lowest voice of the chord and revoice up; negative arpeggiations start with the highest voice of the chord and revoice down.

References getPitch(), and voices().

6.16.4.2 ceiling()

```
Chord csound::Chord::ceiling (
            double g = 1.  )  const  [inline], [virtual]
```

Returns a new chord whose pitches are the ceilings of this chord's pitches, with respect to the generator of transposition g, which defaults to 1 semitone.

References CHORD_SPACE_DEBUG, csound::print_chord(), and setPitch().

Referenced by csound::equate< EQUIVALENCE_RELATION_Tg >(), main(), and csound::predicate< EQUIVALENCE_RELATION_Tg >

6.16.4.3 center()

```
Chord csound::Chord::center ( ) const  [inline], [virtual]
```

Returns the maximally even chord in the chord's space, e.g.

the augmented triad for 3 dimensions.

References csound::OCTAVE(), and setPitch().

Referenced by csound::hyperplane_equation_from_random_inversion_flat(), initialize_sectors(), main(), csound::reflect_by_householder(), csound::reflect_in_central_diagonal(), and csound::reflect_in_central_point().

6.16.4.4 clamp()

```
void csound::Chord::clamp (
            double g = 1.  )  [inline], [virtual]
```

Rounds the pitches in this chord to the nearest integer multiple of g, the generator of transposition.

This is valid only if g goes evenly into 12 (the octave), i.e. in 12/g tone equal temperament.

References csound::OCTAVE().

Referenced by csound::PITV::initialize().

6.16.4.5 clone()

```
virtual Chord csound::Chord::clone ( ) const  [inline], [virtual]
```

6.16.4.6 contains()

```
bool csound::Chord::contains (
            double pitch_ ) const  [inline], [virtual]
```

Returns whether or not the chord contains the pitch.

References csound::eq_tolerance().

Referenced by main().

6.16.4.7 count()

```
size_t csound::Chord::count (
            double pitch ) const  [inline], [virtual]
```

Returns the number of voices in this chord, the same as the number of dimensions in this chord space.

References csound::eq_tolerance().

Referenced by main(), csound::parallelFifth(), and csound::voiceleadingSimpler().

6.16.4.8 cycle()

```
Chord csound::Chord::cycle (
            int stride = 1 ) const  [inline], [virtual]
```

Returns a copy of the chord cyclically permuted by a stride, by default 1.

The direction of rotation is by default the same as musicians' first inversion, second inversion, and so on; but negative sign will reverse the direction of rotation.

- 1 is pop the front and push it on the back, shifting the middle down. 0 1 2 3 4 => 1 2 3 4 0
- 1 is pop the back and push it on the front, shifting the middle up. 0 1 2 3 4 => 4 0 1 2 3

Referenced by main(), permutations(), and v().

6.16.4.9 cyclical_regions_for_dimensionalities()

```
std::map< int, std::vector< Chord > > & csound::Chord::cyclical_regions_for_dimensionalities ( )
[inline], [static]
```

For each chord space of dimensions 3 <= n <= 12, there is one cyclical region of n fundamental domains of OPT equivalence.

The vertices of the cyclical region consist of the n octavewise revoicings of the origin. This function returns a global collection of these cyclical regions.

6.16.4.10 distanceToOrigin()

```
double csound::Chord::distanceToOrigin ( ) const  [inline], [virtual]
```

Returns the Euclidean distance of this chord from its space's origin.

References csound::euclidean().

Referenced by main().

6.16.4.11 distanceToUnisonDiagonal()

```
double csound::Chord::distanceToUnisonDiagonal ( ) const  [inline], [virtual]
```

Returns the Euclidean distance from this chord to the unison diagonal of its chord space.

References csound::euclidean(), and setPitch().

Referenced by main().

6.16.4.12 eI()

```
Chord csound::Chord::eI (
            int opt_sector = 0 ) const  [inline], [virtual]
```

Returns the equivalent of the chord within a fundamental domain of inversional equivalence.

References csound::equate< EQUIVALENCE_RELATION_I >(), and csound::OCTAVE().

Referenced by csound::equate< EQUIVALENCE_RELATION_RPI >(), and main().

6.16.4.13 eO()

```
Chord csound::Chord::eO ( ) const  [inline], [virtual]
```

Returns the equivalent of the chord within the representative fundamental domain of octave equivalence.

References csound::OCTAVE().

Referenced by main().

6.16.4.14 eOP()

```
Chord csound::Chord::eOP ( ) const  [inline], [virtual]
```

Returns the equivalent of the chord within the representative fundamental domain of octave and permutational equivalence.

References csound::OCTAVE().

Referenced by csound::addVoice(), csound::ChordLindenmayer::chordOperation(), csound::Scale::degree(), csound::fill(), csound::PITV::fromChord(), is_k_dual(), csound::PITV::list(), csound::ChordLindenmayer::modalityOperation(), csound::octavewiseRevoicings(), csound::compare_by_op::operator()(), csound::removeVoice(), test_pitv(), csound::transpose_degrees(), and csound::voiceleadingClosestRange().

6.16.4.15 eOPI()

```
Chord csound::Chord::eOPI (
            int opt_sector = 0 ) const  [inline], [virtual]
```

Returns the equivalent of the chord within a fundamental domain of octave, permutational, and inversional equivalence.

References csound::OCTAVE().

6.16.4.16 eOPT()

```
Chord csound::Chord::eOPT (
            int opt_sector = 0 ) const  [inline], [virtual]
```

Returns the equivalent of the chord within a fundamental domain of octave, permutational, and transpositional equivalence.

References csound::OCTAVE().

6.16.4.17 eOPTI()

```
Chord csound::Chord::eOPTI (
            int opt_sector = 0 ) const  [inline], [virtual]
```

Returns the equivalent of the chord within a fundamental domain of range, permutational, transpositional, and inversional equivalence.

References csound::OCTAVE().

6.16.4.18 eOPTT()

```
Chord csound::Chord::eOPTT (
            double g = 1.,
            int opt_sector = 0 ) const  [inline], [virtual]
```

Returns the equivalent of the chord within a fundamental domain of octave, permutational, and transpositional equivalence but in the equal temperament generated by g.

References csound::OCTAVE().

Referenced by is_k_dual(), csound::PITV::list(), and setDifference().

6.16.4.19 eOPTTI()

```
Chord csound::Chord::eOPTTI (
            double g = 1.,
            int opt_sector = 0 ) const  [inline], [virtual]
```

Returns the equivalent of the chord within a fundamental domain of range, permutational, transpositional, and inversional equivalence but in the equal temperament generated by g.

References csound::OCTAVE().

Referenced by csound::PITV::list(), and test_pitv().

6.16.4.20 eOT()

```
Chord csound::Chord::eOT ( ) const  [inline], [virtual]
```

Returns the equivalent of the chord within the representative fundamental domain of octave and transpositional equivalence.

6.16.4.21 eOTT()

```
Chord csound::Chord::eOTT (
            double g = 1.  )  const  [inline], [virtual]
```

Returns the equivalent of the chord within a fundamental domain of octave and transpositional equivalence but in the equal temperament generated by g.

6.16.4.22 eP()

```
Chord csound::Chord::eP ( ) const  [inline], [virtual]
```

Returns the equivalent of the chord within the representative fundamental domain of permutational equivalence.

References csound::equate< EQUIVALENCE_RELATION_P >(), and csound::OCTAVE().

Referenced by eppcs(), csound::hyperplane_equation_from_random_inversion_flat(), Iform(), main(), and Tform().

6.16.4.23 epcs()

```
Chord csound::Chord::epcs ( ) const  [inline], [virtual]
```

Returns the equivalent of the chord under pitch-class equivalence, i.e.

the pitch-class set of the chord.

References csound::chord(), csound::epc(), and setPitch().

Referenced by csound::conformToChord_equivalence(), Iform(), main(), and Tform().

6.16.4.24 eppcs()

```
Chord csound::Chord::eppcs ( ) const  [inline], [virtual]
```

Returns the equivalent of the chord under pitch-class equivalence, i.e.

the pitch-class set of the chord, sorted by pitch-class.

References csound::chord(), eP(), csound::epc(), and setPitch().

Referenced by csound::PITV::fromChord().

6.16.4.25 equals()

```
bool csound::Chord::equals (
            const Chord & other ) const  [inline], [virtual]
```

Returns whether the voices of this chord equal the voices of the other.

Referenced by test_pitv().

6.16.4.26 eR()

```
Chord csound::Chord::eR (
            double range ) const  [inline], [virtual]
```

Returns the equivalent of the chord within the representative fundamental domain of a range equivalence.

References csound::equate< EQUIVALENCE_RELATION_R >().

6.16.4.27 eRP()

```
Chord csound::Chord::eRP (
            double range ) const  [inline], [virtual]
```

Returns the equivalent of the chord within the representative fundamental domain of range and permutational equivalence.

References csound::equate< EQUIVALENCE_RELATION_RP >().

Referenced by csound::equate< EQUIVALENCE_RELATION_RPI >(), and K_range().

6.16.4.28 eRPI()

```
Chord csound::Chord::eRPI (
            double range,
            int opt_sector = 0 ) const  [inline], [virtual]
```

Returns the equivalent of the chord within a fundamental domain of range, permutational, and inversional equivalence.

References csound::equate< EQUIVALENCE_RELATION_RPI >().

6.16.4.29 eRPT()

```
Chord csound::Chord::eRPT (
            double range,
            int opt_sector = 0 ) const  [inline], [virtual]
```

Returns the equivalent of the chord within a fundamental domain of range, permutational, and transpositional equivalence.

References csound::equate< EQUIVALENCE_RELATION_RPT >().

6.16.4.30 eRPTI()

```
Chord csound::Chord::eRPTI (
            double range,
            int opt_sector = 0 ) const  [inline], [virtual]
```

Returns the equivalent of the chord within the representative fundamental domain of range, permutational, transpositional, and inversional equivalence.

References csound::equate< EQUIVALENCE_RELATION_RPTI >().

6.16.4.31 eRPTs()

```
std::vector< Chord > csound::Chord::eRPTs (
            double range = OCTAVE() ) const  [inline], [virtual]
```

Returns all equivalents of the chord within all fundamental domains of range, permutational, and transpositional equivalence.

Referenced by csound::equate< EQUIVALENCE_RELATION_RPT >().

6.16.4.32 eRPTT()

```
Chord csound::Chord::eRPTT (
            double range,
            double g = 1.,
            int opt_sector = 0 ) const  [inline], [virtual]
```

Returns the equivalent of the chord within a fundamental domain of range, permutational, and transpositional equivalence, in the equal temperament generated by g; the same as chord type.

References csound::equate< EQUIVALENCE_RELATION_RPTg >().

6.16.4.33 eRPTTI()

```
Chord csound::Chord::eRPTTI (
            double range,
            double g = 1.,
            int opt_sector = 0 ) const  [inline], [virtual]
```

Returns the equivalent of the chord within a fundamental domain of range, permutational, transpositional, and inversional equivalence.

References csound::equate< EQUIVALENCE_RELATION_RPTgI >().

6.16.4.34 eRPTTs()

```
std::vector< Chord > csound::Chord::eRPTTs (
            double range,
            double g = 1.  )  const  [inline], [virtual]
```

Returns all equivalents of the chord within all fundamental domains of range, permutational, and transpositional equivalence in the equal temperament generated by g; equivalent to all inversions of the chord in the musician's sense.

Referenced by csound::equate< EQUIVALENCE_RELATION_RPTg >().

6.16.4.35 eT()

```
Chord csound::Chord::eT ( ) const  [inline], [virtual]
```

Returns the equivalent of the chord within a fundamental domain of range, permutational, transpositional, and inversional equivalence in the equal temperament generated by g; the same as set class.

References csound::equate< EQUIVALENCE_RELATION_T >(), and csound::OCTAVE().

Referenced by csound::equate< EQUIVALENCE_RELATION_Tg >(), csound::hyperplane_equation_from_random_inversion_flat(), initialize_sectors(), main(), csound::predicate< EQUIVALENCE_RELATION_Tg >(), and csound::reflect_by_householder().

6.16.4.36 et()

```
Chord csound::Chord::et ( ) const  [inline], [virtual]
```

Returns the equivalent of the chord within the fundamental domain of transposition to 0.

References csound::T().

Referenced by main().

6.16.4.37 eTT()

```
Chord csound::Chord::eTT (
            double g = 1.  )  const  [inline], [virtual]
```

Returns the equivalent of the chord within the representative fundamental domain of transpositonal equivalence and the equal temperament generated by g, i.e., returns the chord transposed such that its layer is 0 or, under transposition, the positive layer closest to 0.

NOTE: Does NOT return the result under any other equivalence class.

References csound::equate< EQUIVALENCE_RELATION_Tg >(), and csound::OCTAVE().

Referenced by main().

6.16.4.38 floor()

```
Chord csound::Chord::floor ( ) const  [inline], [virtual]
```

Returns a new chord whose pitches are the floors of this chord's pitches.

References setPitch().

Referenced by main().

6.16.4.39 fromString()

```
void csound::Chord::fromString (
            std::string text )  [inline], [virtual]
```

Rebuilds the chord's pitches (only) from a line of text.

6.16.4.40 getDuration()

```
double csound::Chord::getDuration (
            int voice = 0 ) const  [inline], [virtual]
```

Referenced by csound::note(), and csound::toScore().

6.16.4.41 getInstrument()

```
double csound::Chord::getInstrument (
            int voice = 0 ) const  [inline], [virtual]
```

Referenced by csound::note(), and csound::toScore().

6.16.4.42 getLoudness()

```
double csound::Chord::getLoudness (
            int voice = 0 ) const  [inline], [virtual]
```

Referenced by csound::note(), and csound::toScore().

6.16.4.43 getPan()

```
double csound::Chord::getPan (
            int voice = 0 ) const  [inline], [virtual]
```

Referenced by csound::note(), and csound::toScore().

6.16.4.44 getPitch()

```
double csound::Chord::getPitch (
            int voice ) const  [inline], [virtual]
```

Referenced by a(), csound::addVoice(), csound::ChordLindenmayer::arithmetic(), csound::chord(), csound::closestPitch(), csound::equate< EQUIVALENCE_RELATION_P >(), csound::equate< EQUIVALENCE_RELATION_r >(), csound::euclidean(), csound::hyperplane_equation_from_random_inversion_flat(), K(), csound::midpoint(), csound::next(), csound::note(), csound::operator<(), csound::operator==(), csound::operator>(), csound::predicate< EQUIVALENCE_RELATION_P >(), csound::predicate< EQUIVALENCE_RELATION_r >(), csound::reflect_in_central_diagonal(), csound::reflect_in_central_point(), csound::reflect_in_unison_diagonal(), csound::scale(), csound::Scale::Scale(), T_voiceleading(), test_pitv(), csound::toScore(), csound::Scale::transpose(), v(), csound::voiceleading(), voiceleading(), csound::voiceleadingClosestRange(), and csound::voiceleadingSmoothness().

6.16.4.45 getPitchReference()

```
double & csound::Chord::getPitchReference (
            int voice )  [inline], [virtual]
```

6.16.4.46 greater()

```
bool csound::Chord::greater (
            const Chord & other ) const  [inline], [virtual]
```

Returns whether the voices of this chord are greater than the voices of the other.

6.16.4.47 greater_equals()

```
bool csound::Chord::greater_equals (
            const Chord & other ) const  [inline], [virtual]
```

Returns whether the voices of this chord are greater than or equal to the voices of the other.

6.16.4.48 hyperplane_equation()

```
HyperplaneEquation csound::Chord::hyperplane_equation (
            int opt_sector ) const  [inline], [virtual]
```

Returns the hyperplane equation for the inversion flat that evenly divides the fundamental domain in the indicated sector of the OPT cyclical region.

Referenced by csound::reflect_by_householder(), and csound::reflect_in_inversion_flat().

6.16.4.49 hyperplane_equations_for_opt_sectors()

```
std::map< int, std::vector< HyperplaneEquation > > & csound::Chord::hyperplane_equations_for_↩
opt_sectors ( )  [inline], [static]
```

For each chord space of dimensions 3 <= n <= 12, there are n fundamental domains (sectors) of OPT equivalence.

For each OPT fundamental domain, there is a inversion flat that evenly divides the OPT fundamental domain into 2 OPTI fundamental domains. This function returns a global collection of the hyperplane equations that define these inversion flats.

6.16.4.50 I()

```
Chord csound::Chord::I (
            double center = 0.0 ) const  [inline], [virtual]
```

Inverts the chord by another chord that is on the unison diagonal, by default the origin.

NOTE: Does NOT return an equivalent under any requivalence relation.

References csound::I(), and setPitch().

Referenced by csound::ChordLindenmayer::chordOperation(), Iform(), main(), csound::ChordLindenmayer::modalityOperation(), and test_pitv().

6.16.4.51 Iform()

```
bool csound::Chord::Iform (
            const Chord & Y,
            double g = 1.  )  const  [inline], [virtual]
```

Returns whether the chord is an inversional form of Y with interval size g.

Only works in equal temperament.

References eP(), epcs(), I(), and csound::OCTAVE().

6.16.4.52 information()

```
std::string csound::Chord::information ( ) const  [inline], [virtual]
```

Print much information about the chord including whether it is within important equivalence classes, or what its equivalents would be.

Referenced by main(), test_pitv(), test_pitv(), csound::Scale::transpose(), and csound::transpose_degrees().

6.16.4.53 information_debug()

```
std::string csound::Chord::information_debug (
            int opt_sector ) const  [inline], [virtual]
```

Print much information about the chord including whether it is within important equivalence classes, or what its equivalents would be.

The printout first enables then restores debugging diagnostics.

Referenced by main().

6.16.4.54 information_sector()

```
std::string csound::Chord::information_sector (
            int opt_sector ) const  [inline], [virtual]
```

Print much information about the chord including whether it is within important equivalence classes, or what its equivalents would be.

References csound::print_chord(), csound::print_opti_sectors(), csound::reflect_in_inversion_flat(), and csound::toString().

6.16.4.55 initialize_sectors()

```
void csound::Chord::initialize_sectors ( )  [inline], [virtual]
```

Initializes the fundamental domains (sectors) of the cyclical regions of OPT equivalence and OPTI equivalence, as well as the hyperplane equations that define the inversion flat in each OPT sector.

The cyclical region C of OPT for n voices is the (n-1)-simplicial region of R^n / T with n vertices at A_i = [0^(n - i), 12^n)]_T, for 0 <= i < n. These are the n octavewise revoicings of the origin.

(1) To obtain the fundamental regions of OPT in C, for dimensions 0 <= d < n, replace C[(d+n-1)n] with the center of C c to give OPT_d.

(2) To obtain the fundamental regions for OPTI in C for dimensions 0 <= d < n, replace OPT_d[(d+n-2)n] with the midpoint of OPT_d[(d+n)n] => OPT_d[(d+n-2)n] to give OPTI_d_0, and replace OPT_d[(d+n)n] with the midpoint of OPT_d[(d+n)n] => OPT_d[(d+n-2)n] to give OPTI_d_1.

(3) A vector that is normal to the inversion flat in OPT_d is then OPT_d[(d+n)n] => OPT_d[d+n-2)n]. Normalizing this vector gives the unit normal vector u for the inversion flat. Then the hyperplane equation for the inversion flat is u and its constant term is u dot c.

NOTE:

In this code, sector vertices are NOT permuted.

The reason for starting with C[n-1] is to include the origin in the 0th fundamental domain, because we regard OPT sector 0 as the *representative* fundamental domain of OPT.

This code is based on the construction of Noam Elkies described in the *Generalized Chord Spaces* draft by Callender, Quinn, and Tymoczko.

References center(), CHORD_SPACE_DEBUG, csound::HyperplaneEquation::constant_term, eT(), csound::midpoint(), setPitch(), T(), toString(), csound::toString(), and csound::HyperplaneEquation::unit_normal_vector.

6.16.4.56 inverse_prime_form()

```
Chord csound::Chord::inverse_prime_form ( ) const  [inline], [virtual]
```

Returns this chord as the inverse standard "prime form.".

NOTE: The code here does NOT remove duplicate pitch-classes.

References csound::I(), and csound::inverse_prime_forms_for_chords().

Referenced by csound::PITV::fromChord(), csound::PITV::initialize(), and csound::PITV::list().

6.16.4.57 is_compact()

```
bool csound::Chord::is_compact (
            double range = 12.  )  const  [inline], [virtual]
```

Returns whether this chord has a compact voicing.

This identifies whether the chord belongs to the representative fundamental domain of the OPT equivalence class. In Tymoczko's 1-based notation: x[1] + 12 - x[N] <= x[i + 1] - x[i], 1 <= i < N - 1 In 0-based notation: x[0] + 12 - x[N-1] <= x[i + 1] - x[i], 0 <= i < N - 2

References csound::le_tolerance().

6.16.4.58 is_minor()

```
bool csound::Chord::is_minor ( ) const  [inline], [virtual]
```

Returns whether this chord is "minor" in the sense of having the smallest "wraparound interval" of all its voicings.

References csound::gt_tolerance(), and csound::lt_tolerance().

6.16.4.59 is_opt_sector()

```
bool csound::Chord::is_opt_sector (
            int opt_sector = 0 ) const  [inline], [virtual]
```

Returns whether or not this chord lies within the indicated sector of the cyclical region of OPT fundamental domains.

Referenced by csound::predicate< EQUIVALENCE_RELATION_RPT >(), csound::predicate< EQUIVALENCE_RELATION_RPTg >(), csound::predicate< EQUIVALENCE_RELATION_RPTgI >(), and csound::predicate< EQUIVALENCE_RELATION_RPTI >().

6.16.4.60 is_opti_sector()

```
bool csound::Chord::is_opti_sector (
            int opti_sector = 0 ) const  [inline], [virtual]
```

Returns whether or not this chord lies within the indicated sector of the cyclical region of OPTI fundamental domains.

Referenced by csound::predicate< EQUIVALENCE_RELATION_I >().

6.16.4.61 isel()

```
bool csound::Chord::iseI (
            int opt_sector = 0 ) const  [inline], [virtual]
```

Referenced by main().

6.16.4.62 isel_chord()

```
bool csound::Chord::iseI_chord (
            Chord * inverse,
            int opt_sector = 0 ) const  [inline], [virtual]
```

Returns whether the chord is within a fundamental domain of inversional equivalence.

References csound::OCTAVE().

6.16.4.63 iseO()

```
bool csound::Chord::iseO ( ) const  [inline], [virtual]
```

Returns whether the chord is within the representative fundamental domain of octave equivalence.

References csound::OCTAVE().

Referenced by main().

6.16.4.64 iseOP()

```
bool csound::Chord::iseOP ( ) const  [inline], [virtual]
```

Returns whether the chord is within the representative fundamental domain of octave and permutational equivalence.

References csound::OCTAVE().

6.16.4.65 iseOPI()

```
bool csound::Chord::iseOPI (
            int opt_sector = 0 ) const  [inline], [virtual]
```

Returns whether the chord is within a fundamental domain of octave, permutational, and inversional equivalence.

References csound::OCTAVE().

6.16.4.66 iseOPT()

```
bool csound::Chord::iseOPT (
            int opt_sector = 0 ) const  [inline], [virtual]
```

Returns whether the chord is within a fundamental domainof octave, permutational, and transpositional equivalence.

References csound::OCTAVE().

6.16.4.67 iseOPTI()

```
bool csound::Chord::iseOPTI (
            int opt_sector = 0 ) const  [inline], [virtual]
```

Returns whether the chord is within a fundamental domain of octave, permutational, transpositional, and inversional equivalence.

References csound::OCTAVE().

6.16.4.68 iseOPTT()

```
bool csound::Chord::iseOPTT (
            double g = 1.,
            int opt_sector = 0 ) const  [inline], [virtual]
```

Returns whether the chord is within a fundamental domain of octave, permutational, and transpositional equivalence in the equal temperament generated by g.

References csound::OCTAVE().

6.16.4.69 iseOPTTI()

```
bool csound::Chord::iseOPTTI (
            double g = 1.,
            int opt_sector = 0 ) const  [inline], [virtual]
```

Returns whether the chord is within a fundamental domain of octave, permutational, transpositional, and inversional equivalence in the equal temperament generated by g.

References csound::OCTAVE().

6.16.4.70 iseOT()

```
virtual bool csound::Chord::iseOT ( ) const  [inline], [virtual]
```

Returns whether the chord is within the representative fundamental domain of octave and transpositional equivalence.

6.16.4.71 iseOTT()

```
virtual bool csound::Chord::iseOTT (
            double g = 1.  )  const  [inline], [virtual]
```

Returns whether the chord is within the representative fundamental domain of octave and translational equivalence in the equal temperament generated by g.

6.16.4.72 iseP()

```
bool csound::Chord::iseP ( ) const  [inline], [virtual]
```

Returns whether the chord is within the representative fundamental domain of permutational equivalence.

References csound::OCTAVE().

Referenced by main().

6.16.4.73 isepcs()

```
bool csound::Chord::isepcs ( ) const  [inline], [virtual]
```

Returns whether the chord is within the fundamental domain of pitch-class equivalence, i.e.

is a pitch-class set.

References csound::epc(), and csound::eq_tolerance().

Referenced by main().

6.16.4.74 iseR()

```
bool csound::Chord::iseR (
            double range_ ) const  [inline], [virtual]
```

Returns whether the chord is within the representative fundamental domain of the indicated range equivalence.

Referenced by csound::equate< EQUIVALENCE_RELATION_R >().

6.16.4.75 iseRP()

```
bool csound::Chord::iseRP (
            double range ) const  [inline], [virtual]
```

Returns whether the chord is within the representative fundamental domain of range and permutational equivalence.

6.16.4.76 iseRPI()

```
bool csound::Chord::iseRPI (
            double range,
            int opt_sector = 0 ) const  [inline], [virtual]
```

Returns whether the chord is within a fundamental domain of range, permutational, and inversional equivalence.

6.16.4.77 iseRPT()

```
bool csound::Chord::iseRPT (
            double range,
            int opt_sector = 0 ) const  [inline], [virtual]
```

Returns whether the chord is within a fundamental domain of range, permutational, and transpositional equivalence.

6.16.4.78 iseRPTI()

```
bool csound::Chord::iseRPTI (
            double range,
            int opt_sector = 0 ) const  [inline], [virtual]
```

Returns whether the chord is within a fundamental domain of range, permutational, transpositional, and inversional equivalence.

6.16.4.79 iseRPTT()

```
bool csound::Chord::iseRPTT (
            double range,
            double g = 1.,
            int opt_sector = 0 ) const  [inline], [virtual]
```

Returns whether the chord is within a fundamental domain of range, permutational, and transpositional equivalence in the equal temperament generated by g.

6.16.4.80 iseRPTTI()

```
bool csound::Chord::iseRPTTI (
            double range,
            double g = 1.,
            int opt_sector = 0 ) const  [inline], [virtual]
```

Returns whether the chord is within a fundamental domain of range, permutational, transpositional, and inversional equivalence in the ' equal temperament generated by g.

6.16.4.81 iseRT()

```
virtual bool csound::Chord::iseRT (
            double range ) const  [inline], [virtual]
```

Returns whether the chord is within the representative fundamental domain of range and transpositional equivalence.

6.16.4.82 iseRTT()

```
virtual bool csound::Chord::iseRTT (
            double range,
            double g = 1.  )  const  [inline], [virtual]
```

Returns whether the chord is within a fundamental domain of range and transpositional equivalence in the equal temperament generated by g.

6.16.4.83 iseT()

```
bool csound::Chord::iseT ( ) const  [inline], [virtual]
```

Returns whether the chord is within the representative fundamental domain of transpositional equivalence.

References csound::OCTAVE().

Referenced by main().

6.16.4.84 iset()

```
bool csound::Chord::iset ( ) const  [inline], [virtual]
```

Returns whether the chord is within the fundamental domain of transposition to 0.

Referenced by main().

6.16.4.85 iseTT()

```
bool csound::Chord::iseTT (
            double g = 1.  )  const  [inline], [virtual]
```

Returns whether the chord is within the representative fundamental domain of transpositional equivalence in the equal temperament generated by g.

References csound::OCTAVE().

Referenced by main().

6.16.4.86 K()

```
Chord csound::Chord::K ( ) const  [inline], [virtual]
```

Returns the chord inverted by the sum of its first two voices.

References csound::chord(), csound::epc(), getPitch(), setPitch(), and voices().

Referenced by csound::ChordLindenmayer::chordOperation(), is_k_dual(), and csound::ChordLindenmayer::modalityOperation().

6.16.4.87 K_range()

```
Chord csound::Chord::K_range (
            double range ) const  [inline], [virtual]
```

References csound::chord(), and eRP().

6.16.4.88 layer()

```
double csound::Chord::layer ( ) const  [inline], [virtual]
```

Returns the sum of the pitches in the chord.

Referenced by csound::equate< EQUIVALENCE_RELATION_R >(), csound::equate< EQUIVALENCE_RELATION_T >(), csound::predicate< EQUIVALENCE_RELATION_R >(), csound::predicate< EQUIVALENCE_RELATION_T >(), csound::predicate< EQUIVALENCE_RELATION_Tg >(), csound::reflect_in_central_diagonal(), and csound::reflect_in_unison_diagonal(

6.16.4.89 lesser()

```
bool csound::Chord::lesser (
            const Chord & other ) const  [inline], [virtual]
```

Returns whether the voices of this chord are less than the voices of the other.

6.16.4.90 lesser_equals()

```
bool csound::Chord::lesser_equals (
            const Chord & other ) const  [inline], [virtual]
```

Returns whether the voices of this chord are less than or equal to the voices of the other.

6.16.4.91 max()

```
std::vector< double > csound::Chord::max ( ) const  [inline], [virtual]
```

Returns the highest pitch in the chord, and also the voice index of that pitch.

References csound::gt_tolerance().

Referenced by csound::equate< EQUIVALENCE_RELATION_R >(), and csound::predicate< EQUIVALENCE_RELATION_R >().

6.16.4.92 maximumInterval()

```
double csound::Chord::maximumInterval ( ) const  [inline], [virtual]
```

Returns the maximum interval within the chord.

References csound::gt_tolerance().

Referenced by main().

6.16.4.93 min()

```
std::vector< double > csound::Chord::min ( ) const  [inline], [virtual]
```

Returns the lowest pitch in the chord, and also the voice index of that pitch.

References csound::lt_tolerance().

Referenced by main(), csound::next(), and csound::predicate< EQUIVALENCE_RELATION_R >().

6.16.4.94 minimumInterval()

```
double csound::Chord::minimumInterval ( ) const  [inline], [virtual]
```

Returns the minimum interval within the chord.

References csound::lt_tolerance().

Referenced by main().

6.16.4.95 move()

```
Chord csound::Chord::move (
            int voice,
            double interval ) const  [inline], [virtual]
```

Move 1 voice of the chord.

NOTE: Does NOT return an equivalent under any requivalence relation.

References csound::chord(), setPitch(), and csound::T().

6.16.4.96 name()

```
std::string csound::Chord::name ( ) const  [inline], [virtual]
```

Return the jazz-style name of the chord, if possible, or else a human-readable list of the voices in the chord.

Reimplemented in csound::Scale.

References csound::nameForChord().

Referenced by is_k_dual(), csound::Scale::relative_tonicizations_for_scale_types(), and csound::Scale::tonicizations().

6.16.4.97 normal_form()

```
Chord csound::Chord::normal_form ( ) const  [inline], [virtual]
```

Returns this chord as its standard "normal form.".

NOTE: The code here does NOT remove duplicate pitch-classes.

References csound::normal_forms_for_chords().

Referenced by csound::PITV::fromChord(), csound::PITV::initialize(), csound::compare_by_normal_form::operator()(), and setDifference().

6.16.4.98 normal_order()

```
Chord csound::Chord::normal_order ( ) const  [inline], [virtual]
```

Returns this chord in standard "normal order." For a very clear explanation, see: https://www.↩mta.ca/pc-set/pc-set_new/pages/page04/page04.html and http://openmusictheory.↩com/normalOrder.html/.

NOTE: The code here does NOT remove duplicate pitch-classes. "Normal order" is the most compact ordering to the left of pitch-classes in a chord, measured by pitch-class interval.

References csound::lt_tolerance(), and csound::OCTAVE().

Referenced by csound::compare_by_normal_order::operator()().

6.16.4.99 nrD()

```
Chord csound::Chord::nrD ( ) const  [inline], [virtual]
```

Performs the dominant transformation (which is not a neo-Reimannian transformation).

The result is returned in OP.

References csound::T().

6.16.4.100 nrH()

```
Chord csound::Chord::nrH ( ) const  [inline], [virtual]
```

Performs the neo-Riemannian hexatonic pole transformation.

The result is returned in OP.

References nrL(), and nrP().

6.16.4.101 nrL()

```
Chord csound::Chord::nrL ( ) const  [inline], [virtual]
```

Performs the neo-Riemannian Lettonwechsel transformation.

The result is returned in OP.

Referenced by nrH(), and nrN().

6.16.4.102 nrN()

```
Chord csound::Chord::nrN ( ) const  [inline], [virtual]
```

Performs the neo-Riemannian Nebenverwandt transformation.

The result is returned in NP.

References nrL(), and nrP().

6.16.4.103 nrP()

```
Chord csound::Chord::nrP ( ) const  [inline], [virtual]
```

Performs the neo-Riemannian parallel transformation.

The result is returned in OP.

Referenced by nrH(), nrN(), and nrS().

6.16.4.104 nrR()

```
Chord csound::Chord::nrR ( ) const  [inline], [virtual]
```

Performs the neo-Riemannian parallel transformation.

Referenced by nrS().

6.16.4.105 nrS()

```
Chord csound::Chord::nrS ( ) const  [inline], [virtual]
```

Performs the neo-Riemannian Slide transformation.

The result is returned in OP.

References nrP(), and nrR().

6.16.4.106 operator std::vector< double >()

```
csound::Chord::operator std::vector< double > ( ) const  [inline], [virtual]
```

6.16.4.107 operator=() [1/2]

```
Chord & csound::Chord::operator= (
            const Chord & other )  [inline], [virtual]
```

6.16.4.108 operator=() [2/2]

```
Chord & csound::Chord::operator= (
            const std::vector< double > & other )  [inline], [virtual]
```

6.16.4.109 opt_domain()

```
std::vector< Chord > csound::Chord::opt_domain (
            int sector ) const  [inline], [virtual]
```

Returns the vertices of the OPT fundamental domain for the indicated sector of the cyclical region.

6.16.4.110 opt_domain_sectors()

```
std::vector< int > csound::Chord::opt_domain_sectors ( ) const  [inline], [virtual]
```

Returns the zero-based index(s) of the sector(s) within the cyclical region of OPT fundamental domains to which the chord belongs.

A chord on a vertex, edge, or facet shared by more than one sector belongs to each of them; the center of the cyclical region belongs to all of the sectors. Sectors are generated by rotation of a fundamental domain around the central axis (equivalently, by the octavewise revoicing of chords) and correspond to "chord inversion" in the musician's sense.

Referenced by csound::PITV::list(), and csound::reflect_by_householder().

6.16.4.111 opt_sectors_for_dimensionalities()

```
std::map< int, std::vector< std::vector< Chord > > > & csound::Chord::opt_sectors_for_dimensionalities
( )  [inline], [static]
```

For each chord space of dimensions 3 <= n <= 12, there are n fundamental domains (sectors) of OPT equivalence.

This function returns a global collection of these sectors.

6.16.4.112 opt_simplexes_for_dimensionalities()

```
std::map< int, std::vector< std::vector< Chord > > > & csound::Chord::opt_simplexes_for_dimensionalities
( )  [inline], [static]
```

Returns a collection of vertices for the OPT fundamental domains; each has an added vertex to make a simplex for chord location.

6.16.4.113 opti_domain()

```
std::vector< Chord > csound::Chord::opti_domain (
            int sector ) const  [inline], [virtual]
```

Returns the vertices of the OPTI fundamental domain for the indicated sector of the cyclical region.

6.16.4.114 opti_domain_sectors()

```
std::vector< int > csound::Chord::opti_domain_sectors ( ) const  [inline], [virtual]
```

Returns the zero-based index(s) of the sector(s) within the cyclical region of OPTI fundamental domains to which the chord belongs.

A chord on a vertex, edge, or facet shared by more than one sector belongs to each them; the center of the cyclical region belongs to all of the sectors. Sectors are generated by rotation of a fundamental domain (equivalently, by the octavewise revoicing of chords) and correspond to "chord inversion" in the musician's ordinary sense. SCOPED_DEBUGGING debug;

References CHORD_SPACE_DEBUG, csound::distance_to_points(), csound::lt_tolerance(), and csound::toString().

Referenced by csound::print_chord(), and csound::print_opti_sectors().

6.16.4.115 opti_sectors_for_dimensionalities()

```
std::map< int, std::vector< std::vector< Chord > > > & csound::Chord::opti_sectors_for_dimensionalities
( )  [inline], [static]
```

For each chord space of dimensions 3 <= n <= 12, there are n fundamental domains (sectors) of OPTI equivalence.

This function returns a global collection of these sectors.

6.16.4.116 opti_simplexes_for_dimensionalities()

```
std::map< int, std::vector< std::vector< Chord > > > & csound::Chord::opti_simplexes_for_↩
dimensionalities ( )  [inline], [static]
```

Returns a collection of vertices for the OPTI fundamental domains that have an added vertex to make a simplex for chord location.

6.16.4.117 origin()

```
Chord csound::Chord::origin ( ) const  [inline], [virtual]
```

Returns the origin of the chord's space.

References resize().

Referenced by csound::reflect_in_unison_diagonal().

6.16.4.118 permutations()

```
std::vector< Chord > csound::Chord::permutations ( ) const  [inline], [virtual]
```

Returns the permutations of the pitches in a chord.

The permutations are always returned in the same order.

References cycle().

Referenced by main().

6.16.4.119 prime_form()

```
Chord csound::Chord::prime_form ( ) const  [inline], [virtual]
```

Returns this chord as its standard "prime form.".

NOTE: The code here does NOT remove duplicate pitch-classes.

References csound::I(), and csound::prime_forms_for_chords().

Referenced by csound::PITV::fromChord(), csound::PITV::initialize(), and csound::PITV::list().

6.16.4.120 Q()

```
Chord csound::Chord::Q (
            double x,
            const Chord & m,
            double g = 1. )  const  [inline], [virtual]
```

Returns the contextual transposition of the chord by x with respect to m with minimum interval size g.

NOTE: Does NOT return an equivalent under any requivalence relation.

References csound::T().

Referenced by csound::ChordLindenmayer::chordOperation().

6.16.4.121 reflect()

```
Chord csound::Chord::reflect (
            int opt_sector ) const  [inline], [virtual]
```

Reflects the chord in the inversion flat of the indicated OPT domain sector.

References csound::reflect_in_inversion_flat().

Referenced by main().

6.16.4.122 resize()

```
void csound::Chord::resize (
            size_t voiceN )  [inline], [virtual]
```

Referenced by csound::addVoice(), csound::chord(), csound::chordForName(), csound::fill(), csound::gather(), csound::iterator(), main(), origin(), csound::PITV::preinitialize(), csound::removeVoice(), csound::scaleForName(), csound::Scale::transpose(), and csound::transpose_degrees().

6.16.4.123 rownd()

```
double csound::Chord::rownd (
            double x,
            int places = 12 )  [inline], [static]
```

Rounds the value of x to the specified number of decimal places.

6.16.4.124 self_inverse()

```
bool csound::Chord::self_inverse (
            int opt_sector = 0 ) const  [inline], [virtual]
```

Returns whether or not this chord is invariant under reflection in the inversion flat of the indicated OPT sector.

Such are the shared vertices, edges, and facets of those fundamental domains that involve inversional equivalence.

References csound::reflect_in_inversion_flat().

Referenced by csound::predicate< EQUIVALENCE_RELATION_I >().

6.16.4.125 setDuration()

```
void csound::Chord::setDuration (
            double value,
            int voice = -1 )  [inline], [virtual]
```

6.16.4.126 setInstrument()

```
void csound::Chord::setInstrument (
            double value,
            int voice = -1 )  [inline], [virtual]
```

6.16.4.127 setLoudness()

```
void csound::Chord::setLoudness (
            double value,
            int voice = -1 )  [inline], [virtual]
```

6.16.4.128 setPan()

```
void csound::Chord::setPan (
            double value,
            int voice = -1 )  [inline], [virtual]
```

6.16.4.129 setPitch()

```
void csound::Chord::setPitch (
            int voice,
            double value )  [inline], [virtual]
```

Referenced by csound::addVoice(), csound::ChordLindenmayer::arithmetic(), ceiling(), center(), csound::chord(), distanceToUnisonDiagonal(), epcs(), eppcs(), csound::equate< EQUIVALENCE_RELATION_r >(), csound::equate< EQUIVALENCE_RE csound::fill(), floor(), csound::gather(), csound::hyperplane_equation_from_random_inversion_flat(), I(), initialize_sectors(), csound::iterator(), K(), main(), csound::midpoint(), move(), csound::next(), csound::reflect_by_householder(), csound::reflect_in_inversion_flat(), T(), T_voiceleading(), test_pitv(), csound::Scale::transpose(), v(), csound::voiceleading(), voiceleading(), and csound::voiceleadingClosestRange().

6.16.4.130 T()

```
Chord csound::Chord::T (
            double interval ) const  [inline], [virtual]
```

Transposes the chord by the indicated interval (may be a fraction).

NOTE: Does NOT return an equivalent under any requivalence relation.

References setPitch(), and csound::T().

Referenced by csound::ChordLindenmayer::chordOperation(), csound::equate< EQUIVALENCE_RELATION_T >(), csound::equate< EQUIVALENCE_RELATION_Tg >(), csound::fill(), initialize_sectors(), main(), csound::ChordLindenmayer::modalityOpe csound::predicate< EQUIVALENCE_RELATION_Tg >(), csound::reflect_in_central_diagonal(), csound::reflect_in_unison_diagonal(), Tform(), and csound::Scale::transpose().

6.16.4.131 T_voiceleading()

```
Chord csound::Chord::T_voiceleading (
            const Chord & voiceleading )  [inline], [virtual]
```

Transposes the chord by the indicated voiceleading (passed as a Chord of directed intervals).

NOTE: Does NOT return an equivalent under any equivalence relation.

References getPitch(), setPitch(), and csound::voiceleading().

6.16.4.132 test()

```
bool csound::Chord::test (
            const char * caption = "" ) const  [inline], [virtual]
```

Tests the internal consistency of the predicates ("iseX") and transformations ("eX") of this chord, and prints a report.

References csound::toString().

6.16.4.133 Tform()

```
bool csound::Chord::Tform (
            const Chord & Y,
            double g = 1.  )  const  [inline], [virtual]
```

Returns whether the chord is a transpositional form of Y with interval size g.

Only works in equal temperament.

References eP(), epcs(), csound::OCTAVE(), and T().

6.16.4.134 toString()

```
std::string csound::Chord::toString ( ) const  [inline], [virtual]
```

Returns a string representation of the chord's pitches (only).

Quadratic complexity, but short enough not to matter.

Referenced by csound::HarmonyIFS::add_interpolation_point_as_chord(), csound::HarmonyIFS2::add_interpolation_point_as_chord(), csound::ChordLindenmayer::chordOperation(), csound::equate< EQUIVALENCE_RELATION_R >(), csound::fill(), csound::fundamentalDomainByPredicate(), csound::fundamentalDomainByTransformation(), csound::indexForOctavewiseRevoicing(), initialize_sectors(), is_k_dual(), csound::HarmonyIFS2::iterate(), csound::PITV::list(), main(), csound::octavewiseRevoicing(), csound::octavewiseRevoicings(), csound::predicate< EQUIVALENCE_RELATION_I >(), csound::predicate< EQUIVALENCE_RELATION
csound::print_chord(), csound::reflect_by_householder(), csound::Scale::relative_tonicizations_for_scale_types(), csound::scale(), csound::ChordLindenmayer::scaleDegreeOperation(), csound::ChordLindenmayer::scaleOperation(), csound::ChordLindenmayer::scoreOperation(), setDifference(), csound::Scale::tonicizations(), csound::Scale::transpose(), and csound::transpose_degrees().

6.16.4.135 v()

```
Chord csound::Chord::v (
            int direction = 1 ) const  [inline], [virtual]
```

Returns a copy of the chord 'inverted' in the musician's sense, i.e.

revoiced by cyclically permuting the chord and adding (or subtracting) an octave to the highest (or lowest) voice. The revoicing will move the chord up or down in pitch. A positive direction is the same as a musician's first inversion, second inversion, etc.

References csound::chord(), cycle(), getPitch(), csound::OCTAVE(), and setPitch().

Referenced by csound::scale(), and voicings().

6.16.4.136 voiceleading()

```
Chord csound::Chord::voiceleading (
            const Chord & destination ) const  [inline], [virtual]
```

Returns the transpositions (as a Chord of directed intervals) that takes this chord to the destination chord.

NOTE: Makes no assumption that both chords are in the same equivalence class.

References getPitch(), and setPitch().

6.16.4.137 voices()

```
size_t csound::Chord::voices ( ) const  [inline], [virtual]
```

Returns the number of voices in this chord; that is, the number of dimensions in the chord space for this chord.

Referenced by a(), csound::addVoice(), csound::ChordLindenmayer::arithmetic(), csound::chord(), csound::closestPitch(), csound::Scale::degree(), csound::equate< EQUIVALENCE_RELATION_P >(), csound::equate< EQUIVALENCE_RELATION_r >(), csound::equate< EQUIVALENCE_RELATION_T >(), csound::equivalentDegree(), csound::euclidean(), K(), csound::midpoint(), csound::Scale::modulations(), csound::Scale::modulations_for_scale_types(), csound::next(), csound::notes(), csound::operator<(), csound::operator==(), csound::operator>(), csound::predicate< EQUIVALENCE_RELATION_P >(), csound::predicate< EQUIVALENCE_RELATION_r >(), csound::reflect_by_householder(), csound::reflect_in_central_diagonal(), csound::reflect_in_central_point(), csound::reflect_in_inversion_flat(), csound::reflect_in_unison_diagonal(), csound::Scale::relative_tonici csound::removeVoice(), csound::Scale::Scale(), csound::Scale::Scale(), csound::Scale::secondary(), csound::Scale::tonicizations(), csound::toScore(), csound::transpose_degrees(), csound::voiceleading(), csound::voiceleadingClosestRange(), and csound::voiceleadingSmoothness().

6.16.4.138 voicings()

```
std::vector< Chord > csound::Chord::voicings ( ) const  [inline], [virtual]
```

Returns all the 'inversions' (in the musician's sense) or octavewise revoicings of the chord.

The first voice is transposed up by one octave, and all voices are then rotated "left" so the transposed voice becomes the last voice.

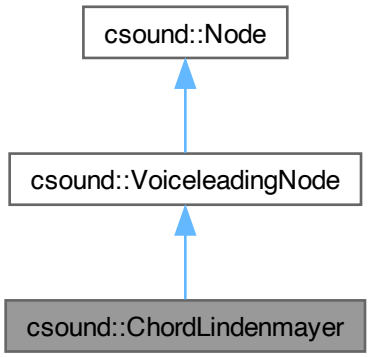
References csound::chord(), and v().

6.17 csound::ChordLindenmayer Class Reference

A Lindenmayer system consists of a turtle representing a position in musical space, that is, a note; commands for moving the turtle or writing its state into a musical score; an axiom or initial set of commands; and zero or more rules for replacing commands with arbitrary sequences of commands.

```
#include <ChordLindenmayer.hpp>
```

Inheritance diagram for csound::ChordLindenmayer:



Public Member Functions

- virtual void addChild (Node *node)

  Adds an immediate child Node to this.
- virtual void addRule (std::string command, std::string replacement)
- virtual void apply (Score &score, const VoiceleadingOperation &priorOperation, const VoiceleadingOperation &currentOperation)

  Apply the current voice-leading operation to the score, within the specified range of notes.
- void C (double time, double C_)

  Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified prime chord and transposition.
- void C_name (double time, std::string C_)

  Same as C, except the chord can be specified by jazz-type name (e.g.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- void chord (const csound::Chord &chord, double time)

  Apply the specified chord to the current segement.
- ChordLindenmayer ()
- void chordVoiceleading (const csound::Chord &chord, double time, bool avoid_parallels)

  Apply the specified chord to the current segement, using the closest voice-leading from the pitches of the previous segment.
- void CL (double time, double C_, bool avoidParallels=true)

  Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified chord; the voicing of the chord will be the smoothest voice-leading from the pitches of the previous chord.
- void CL_name (double time, std::string C_, bool avoidParallels=true)

  Same as CL, except the chord is specified by jazz-type name (e.g.
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- void CV (double time, double C_, double V_)

  Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified prime chord, transpostion, and voicing.
- void CV_name (double time, std::string C_, double V_)

  Same as CV, except the chord is specified by jazz-type name (e.g.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &score)

  Optionally generate notes into the score.
- virtual void generateLocally ()

  Scores are generated as follows:
- virtual double getAngle () const
- virtual std::string getAxiom () const
- virtual Node * getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual int getIterationCount () const
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.

- virtual std::vector< double > getModality () const
- virtual std::string getReplacement (std::string command)
- virtual Chord getTurtleChord () const
- virtual Chord getTurtleModality () const
- virtual Scale getTurtleScale () const
- virtual int getTurtleScaleDegree () const
- virtual void initialize ()
- void K (double time)

  Find the C of the previous segment, and contextually invert it; apply the resulting C to the current segment.
- void KL (double time, bool avoidParallels=true)

  Find the C of the previous segment, and contextually invert it; apply the resulting C to the current segment, using the closest voiceleading from the pitches of the previous segment.
- void KV (double time, double V_)

  Find the C of the previous segment, and contextually invert it; apply the resulting C to the current segment with voicing V.
- void L (double time, bool avoidParallels=true)

  Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the smoothest voice-leading from the pitches of the previous segment.
- void PT (double time, double P_, double T)

  Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified prime chord and transposition.
- void PTL (double time, double P_, double T, bool avoidParallels=true)

  Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified chord; the voicing of the chord will be the smoothest voice-leading from the pitches of the previous chord.
- void PTV (double time, double P_, double T, double V_)

  Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified prime chord, transpostion, and voicing.
- void Q (double time, double Q_)

  Find the C of the previous segment, and contextually transpose it; apply the resulting C to the current segment.
- void QL (double time, double Q_, bool avoidParallels=true)

  Find the C of the previous segment, and contextually transpose it; apply the resulting C to the current segment, using the specified octavewise revoicing.
- void QV (double time, double Q_, double V_)

  Find the C of the previous segment, and contextually transpose it; apply the resulting C to the current segment with voicing V.
- virtual void setAngle (double angle)
- virtual void setAxiom (std::string axiom)
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void setIterationCount (int count)
- virtual void setModality (const std::vector< double > &pcs)
- virtual void setTurtleChord (const csound::Chord &chord)
- virtual void setTurtleModality (const csound::Chord &chord)
- virtual void setTurtleScale (const csound::Scale &scale)
- virtual void setTurtleScaleDegree (int degree)
- virtual void transform (Score &score)

  Apply all of the voice-leading operations stored within this node to the score.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

- void V (double time, double V_)

  Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified voicing of the chord.

- virtual ~ChordLindenmayer ()

Data Fields

- double angle
- bool avoidParallels

  If true (the default), voice-leadings will avoid parallel fifths.

- std::string axiom
- double base

  The lowest pitch of the range of voicings, as a MIDI key number (default = 36).

- clock_t beganAt
- std::vector< Node * > children

  Child Nodes, if any.

- size_t divisionsPerOctave

  The number of equally tempered divisions of the octave (default = 12).

- clock_t elapsed
- clock_t endedAt
- int iterationCount
- std::vector< double > modality

  Context for the K and Q operations; must have the same cardinality as the pitch-classes in use.

- std::map< double, VoiceleadingOperation > operations

  Voice-leading operations stored in order of starting time.

- std::string production
- double range

  The range of voicings, from the lowest to the highest pitch, as a MIDI key number (default = 60).

- bool rescaleTimes
- std::map< std::string, std::string > rules
- Score score
- Turtle turtle
- std::stack< Turtle > turtleStack

Protected Member Functions

- virtual void applyVoiceleadingOperations ()
- virtual void arithmetic (Chord &target, const std::string &operation, const std::string &targetString, const std↩ ::vector< std::string > &command)
- virtual double arithmetic (const double &target, const std::string &operation, const std::string &targetString, const std::vector< std::string > &command)
- virtual double arithmetic (const double &target, const std::string &operation, const std::string &targetString, double p1, double p2, double p3, double p4, double p5)
- virtual void arithmetic (Event &target, const std::string &operation, const std::string &targetString, const std↩ ::vector< std::string > &command)

- virtual void chordOperation (const std::string &operation, const std::string &target, const std::vector< std::string > &command)
- virtual Eigen::MatrixXd createRotation (int dimension1, int dimension2, double angle) const
- virtual double equivalence (const double &value, const std::string &equivalenceClass) const
- virtual void fixStatus ()
- virtual void generateLindenmayerSystem ()

  Iterates the replacement rules on the axiom and subsequent productions to produce the final production, a possibly long string of turtle commands.
- virtual void interpret (std::vector< std::string > command)

  The first element of the command is always the operation, the second element is always the target.
- virtual void modalityOperation (const std::string &operation, const std::string &target, const std::vector< std::string > &command)
- virtual void noteOperation (const std::string &operation, const std::string &target, const std::vector< std::string > &command)
- virtual void noteOrientationOperation (const std::string &operation, const std::string &target, const std::vector< std::string > &command)
- virtual void noteStepOperation (const std::string &operation, const std::string &target, const std::vector< std::string > &command)
- virtual void scaleDegreeOperation (const std::string &operation, const std::string &target, const std::vector< std::string > &command)
- virtual void scaleOperation (const std::string &operation, const std::string &target, const std::vector< std::string > &command)
- virtual void scoreOperation (const std::string &operation, const std::string &target, const std::vector< std::string > &command)
- virtual void tieOverlappingNotes ()
- virtual void turtleOperation (const std::string &operation, const std::string &target, const std::vector< std::string > &command)
- virtual void voicingOperation (const std::string &operation, const std::string &target, const std::vector< std::string > &command)
- virtual void writeScore ()

  Parses the final production into commands, each a tuple of strings, and interprets each command to write notes and chord progressions into the score.

Protected Attributes

- Eigen::MatrixXd localCoordinates

6.17.1 Detailed Description

A Lindenmayer system consists of a turtle representing a position in musical space, that is, a note; commands for moving the turtle or writing its state into a musical score; an axiom or initial set of commands; and zero or more rules for replacing commands with arbitrary sequences of commands.

The turtle T represents the current state of the Lindenmayer system: a note vector N that represents a position in score space, a step size S, an orientation O, a chord C, a chord that defines modality M, an octavewise chord revoicing V, a scale Sc, a scale degree Sd, and a range Ra. Vectors are given as `{el0,el2,...}` **without any spaces**.

The turtle commands are defined `(operation target ...)` or `(operation target[dimension] ...)`, as follows:

```
([ T)        Push the current turtle state on a stack (start a branch).
```

```
(] T)          Pop the current turtle state from the stack (return to start).
(W N e)        Write the current turtle note N into the score under
               equivalence class e.
(F N x e)      Move the turtle position N "forward" x steps S along its
               current orientation O) under equivalence class e.
(o N[d] x e)   Apply arithmetic operation o to dimension d of the turtle
               position N with parameter x under equivalence class e.
(o S[d] x e)   Apply arithmetic operation o to dimension d  of the turtle
               step size S with parameter x under equivalence class e.
(R O a b w)    Rotate the turtle orientation O in the plane of dimensions
               a and b by angle w radians.
(W C e)        Write the current turtle chord C with octavewise revoicing
               from OP order V to the score under equivalence class e. Chord
               voices default to the same time and other dimensions as the
               current turtle note N.
(o C v e)      Apply arithmetic operation o to the turtle chord C as a whole
               with parameter v (a vector or chord name) under equivalence
               class e.
(o C[i] x e)   Apply arithmetic operation o to voice i of the turtle chord C
               with parameter x under equivalence class e.
(T C x)        Transpose the turtle chord C by x under equivalence class e.
(I C x)        Invert the turtle chord C by reflecting around pitch-class x.
(K C)          Apply Neo-Riemannian inversion by exchange to the turtle
               chord C.
(Q C x)        Apply Neo-Riemannian contextual transposition by x
               (by reference to the turtle's modality M) to turtle chord C.
(++ C)         Add a voice (doubling the first pitch in OP order) to the
               turtle chord C.
(-- C)         Remove a voice (the uppermost pitch in OP order) from the
               turtle chord C.
(o M v e)      Apply arithmetic operation o to the turtle modality M as a
               whole with parameter v (a vector or chord name) under
               equivalence class e.
(o M[i] x e)   Apply arithmetic operation o to voice i of the turtle modality
               M with parameter x under equivalence class e.
(o V x)        Apply arithmetic operation o to the voicing index of the
               turtle chord with parameter x. Of necessity the
               equivalence class is the range of the score.
(= Sc n v)     Assign the vector of pitches v to the turtle scale Sc with
               name n.
(C Sc n m)     Obtain the turtle chord C with m voices at the nth degree
               of the current turtle scale Sc.
(M Sc n k)     Modulate the turtle scale Sc to a new scale Sc with the the
               current turtle chord C as the common chord but with n voices;
               if more than one scale exists with that common chord,
               choose the kth scale.
(o Sd x)       Apply arithmetic operation o to the turtle scale degree Sd,
               with parameter x.
(C Sd m)       Obtain the turtle chord C of m voices as the current scale
               degree Sd of the turtle scale Sc.
(C P)          Apply the current turtle chord C to the score, starting at
               the current time and continuing until the next application.
(Cl P)         Apply the current turtle chord C to the score, using the
               closest voice-leading from the previous chord (if any),
               starting at the current time and continuing to the next
               application.
(Sc P)         Apply the current turtle scale Sc to the score, starting
               at the current time and continuing until the next application.
(0 P)          End the scope of the previous application of a chord or scale.
(= P n)        Assign the range n to the size of the score, i.e. define
               range equivalence.
(seed P x)     Seed the static random generator used by all random
               distributions with x.
```

An arithmetic operation may also consist of sampling a random distribution, e.g. `(u N[k] minimum maximum)`; all parameters of the distribution must be given. The complete set of operations is:

```
Assignment           = x e
Addition             + x e
Subtraction          - x e
Multiplication       * x e
Division             / x e
Uniform              uni min max
Normal (Gaussian)    nor mean sigma
Binomial             bin p k
Negative binomial    nbi p k
Poisson              poi mean
Exponential          exp lambda
Gamma                gam alpha beta
Weibull              wei a b
Extreme value        ext a b
```

```
Log normal          log mean sigma
Chi squared         chi n
Cauchy              cau a b
Fisher              fis m n
Student             stu n
```

Dimensions are:

```
Time                t
Duration            d
MIDI status         s
Instrument          i
MIDI key            k
MIDI velocity       v
Phase in radians    p
Pan                 x
Depth               y
Height              z
Pitch-class set     m
```

Equivalence classes are:

```
None                0
The octave          O
Range of the score  R
```

PLEASE NOTE: Scale commands take precedence over chord commands. Not all commands are implemented. Unimplemented commands silently perform no operation, but may still be used to define replacement rules.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 ChordLindenmayer()

```
csound::ChordLindenmayer::ChordLindenmayer ( )
```

6.17.2.2 ~ChordLindenmayer()

```
csound::ChordLindenmayer::~ChordLindenmayer ( )  [virtual]
```

6.17.3 Member Function Documentation

6.17.3.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.17.3.2 addRule()

```
void csound::ChordLindenmayer::addRule (
            std::string command,
            std::string replacement )  [virtual]
```

References rules.

6.17.3.3 apply()

```
void csound::VoiceleadingNode::apply (
            Score & score,
            const VoiceleadingOperation & priorOperation,
            const VoiceleadingOperation & currentOperation )  [virtual], [inherited]
```

Apply the current voice-leading operation to the score, within the specified range of notes.

If voice-leading proper is to be performed, the prior voice-leading operation is used to determine how to lead the voices.

References csound::VoiceleadingNode::avoidParallels, csound::VoiceleadingNode::base, csound::Voicelead::cToM(), csound::VoiceleadingNode::divisionsPerOctave, csound::fundamentalDomainByPredicate(), csound::Score::getDuration(), csound::System::getMessageLevel(), csound::Score::getPTV(), csound::System::inform(), csound::System::INFORMATION_LEVEL, csound::VoiceleadingNode::modality, csound::Voicelead::mToPitchClassSet(), csound::Voicelead::pitchClassSetToPandT(), csound::printChord(), csound::VoiceleadingNode::range, csound::Score::setK(), csound::Score::setKL(), csound::Score::setKV(), csound::Score::setPitchClassSet(), csound::Score::setPT(), csound::Score::setPTV(), csound::Score::setQ(), csound::Score::setQL(), csound::Score::setQV(), and csound::Score::voicelead().

Referenced by csound::VoiceleadingNode::transform().

6.17.3.4 applyVoiceleadingOperations()

```
void csound::ChordLindenmayer::applyVoiceleadingOperations ( )  [protected], [virtual]
```

References score, and csound::VoiceleadingNode::transform().

Referenced by generateLocally().

6.17.3.5 arithmetic() [1/4]

```
void csound::ChordLindenmayer::arithmetic (
            Chord & target,
            const std::string & operation,
            const std::string & targetString,
            const std::vector< std::string > & command )  [protected], [virtual]
```

References arithmetic(), equivalence(), csound::Chord::getPitch(), csound::parseIndex(), csound::parseVector(), csound::real(), csound::Chord::setPitch(), and csound::Chord::voices().

Referenced by arithmetic(), arithmetic(), arithmetic(), chordOperation(), modalityOperation(), noteOperation(), noteStepOperation(), scaleDegreeOperation(), and voicingOperation().

6.17.3.6 arithmetic() [2/4]

```
double csound::ChordLindenmayer::arithmetic (
            const double & target,
            const std::string & operation,
            const std::string & targetString,
            const std::vector< std::string > & command )  [protected], [virtual]
```

References arithmetic(), and csound::real().

6.17.3.7 arithmetic() [3/4]

```
double csound::ChordLindenmayer::arithmetic (
            const double & target,
            const std::string & operation,
            const std::string & targetString,
            double p1,
            double p2,
            double p3,
            double p4,
            double p5 )  [protected], [virtual]
```

References csound::System::debug(), and csound::twister.

6.17.3.8 arithmetic() [4/4]

```
void csound::ChordLindenmayer::arithmetic (
            Event & target,
            const std::string & operation,
            const std::string & targetString,
            const std::vector< std::string > & command )  [protected], [virtual]
```

References arithmetic(), csound::System::debug(), equivalence(), csound::parseIndex(), csound::parseVector(), and csound::Event::toString().

6.17.3.9 C()

```
void csound::VoiceleadingNode::C (
            double time,
            double C_ )  [inherited]
```

Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified prime chord and transposition.

Note that C (equivalent to PT) specifies what musicians normally call a chord.

References csound::VoiceleadingNode::operations.

Referenced by csound::VoiceleadingNode::C_name().

6.17.3.10 C_name()

```
void csound::VoiceleadingNode::C_name (
            double time,
            std::string C_ )  [inherited]
```

Same as C, except the chord can be specified by jazz-type name (e.g.

EbM7) instead of C number.

References csound::VoiceleadingNode::C(), csound::VoiceleadingNode::divisionsPerOctave, and csound::Voicelead::nameToC().

6.17.3.11 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.17.3.12 chord()

```
void csound::VoiceleadingNode::chord (
            const csound::Chord & chord,
            double time )  [inherited]
```

Apply the specified chord to the current segement.

References csound::VoiceleadingNode::chord(), and csound::VoiceleadingNode::operations.

Referenced by csound::VoiceleadingNode::chord(), chordOperation(), csound::VoiceleadingNode::chordVoiceleading(), scaleOperation(), scoreOperation(), and setTurtleChord().

6.17.3.13 chordOperation()

```
void csound::ChordLindenmayer::chordOperation (
            const std::string & operation,
            const std::string & target,
            const std::vector< std::string > & command )  [protected], [virtual]
```

References csound::addVoice(), csound::Score::append(), arithmetic(), csound::Turtle::chord, csound::VoiceleadingNode::chord(), csound::System::debug(), csound::Chord::eOP(), csound::Chord::I(), csound::Chord::K(), csound::Turtle::modality, csound::Turtle::note, csound::octavewiseRevoicing(), csound::Chord::Q(), csound::Turtle::rangeSize, csound::real(), csound::removeVoice(), score, csound::Event::setKey(), csound::Chord::T(), csound::Chord::toString(), turtle, and csound::Turtle::voicing.

Referenced by interpret().

6.17.3.14 chordVoiceleading()

```
void csound::VoiceleadingNode::chordVoiceleading (
            const csound::Chord & chord,
            double time,
            bool avoid_parallels = true )  [inherited]
```

Apply the specified chord to the current segement, using the closest voice-leading from the pitches of the previous segment.

References csound::VoiceleadingNode::chord(), csound::fundamentalDomainByPredicate(), and csound::VoiceleadingNode::operations.

Referenced by scoreOperation().

6.17.3.15 CL()

```
void csound::VoiceleadingNode::CL (
            double time,
            double C_,
            bool avoidParallels = true )  [inherited]
```

Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified chord; the voicing of the chord will be the smoothest voice-leading from the pitches of the previous chord.

Optionally, parallel fifths can be avoided. Note that CL (equivalent to PTL) specifies what musicians normally call the voice-leading of a chord.

References csound::VoiceleadingNode::avoidParallels, and csound::VoiceleadingNode::operations.

Referenced by csound::VoiceleadingNode::CL_name().

6.17.3.16 CL_name()

```
void csound::VoiceleadingNode::CL_name (
            double time,
            std::string C_,
            bool avoidParallels = true )  [inherited]
```

Same as CL, except the chord is specified by jazz-type name (e.g.

EbM7) instead of C number.

References csound::VoiceleadingNode::avoidParallels, csound::VoiceleadingNode::CL(), csound::VoiceleadingNode::divisionsPerOctave, and csound::Voicelead::nameToC().

6.17.3.17 clear()

```
void csound::ChordLindenmayer::clear ( )  [virtual]
```

Recursively clears all child Nodes of this.

Reimplemented from csound::Node.

References rules, score, and turtleStack.

6.17.3.18 createRotation()

```
Eigen::MatrixXd csound::ChordLindenmayer::createRotation (
            int dimension1,
            int dimension2,
            double angle ) const  [protected], [virtual]
```

References angle, and csound::Event::ELEMENT_COUNT.

Referenced by noteOrientationOperation().

6.17.3.19 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.17.3.20 CV()

```
void csound::VoiceleadingNode::CV (
            double time,
            double C_,
            double V_ )  [inherited]
```

Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified prime chord, transpostion, and voicing.

Note that CV (equivalent to PTV) specifies what musicians normally call the voicing, or octavewise inversion, of a chord.

References csound::VoiceleadingNode::operations.

Referenced by csound::VoiceleadingNode::CV_name().

6.17.3.21 CV_name()

```
void csound::VoiceleadingNode::CV_name (
            double time,
            std::string C_,
            double V_ )  [inherited]
```

Same as CV, except the chord is specified by jazz-type name (e.g.

EbM7) instead of C number.

References csound::VoiceleadingNode::CV(), csound::VoiceleadingNode::divisionsPerOctave, and csound::Voicelead::nameToC().

6.17.3.22 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.17.3.23 equivalence()

```
double csound::ChordLindenmayer::equivalence (
            const double & value,
            const std::string & equivalenceClass ) const  [protected], [virtual]
```

References csound::Conversions::modulus(), csound::Turtle::rangeSize, and turtle.

Referenced by arithmetic(), and arithmetic().

6.17.3.24 fixStatus()

```
void csound::ChordLindenmayer::fixStatus ( )  [protected], [virtual]
```

References score.

Referenced by generateLocally().

6.17.3.25 generate()

```
void csound::ChordLindenmayer::generate (
            Score & score_from_this )  [virtual]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented from csound::Node.

References generateLocally(), and score.

6.17.3.26 generateLindenmayerSystem()

```
void csound::ChordLindenmayer::generateLindenmayerSystem ( )  [protected], [virtual]
```

Iterates the replacement rules on the axiom and subsequent productions to produce the final production, a possibly long string of turtle commands.

References axiom, csound::System::debug(), csound::System::inform(), iterationCount, production, and rules.

Referenced by generateLocally().

6.17.3.27 generateLocally()

```
void csound::ChordLindenmayer::generateLocally ( )  [virtual]
```

Scores are generated as follows:

1. The initial value of the turtle is set by the Lindenmayer system.<
   - >
   - The Lindenmayer system is rewritten by taking the axiom, parsing it into words, and replacing each word with the product of a rewriting rule, if one exists, or itself, if there is no rule. This procedure is iterated for a specified number of times.
   - The finished, rewritten Lindenmayer system is interpreted as a series of commands for moving a turtle around in various music spaces to write a score.
     (a) Notes (N operations) are written directly into the score.
     (b) Chords (C operations) are written into the score as notes.
     (c) Score operations are written into the score as voice-leading operations, to be applied after all notes have been generated.
   - Overlapping and directly abutting notes in the score are joined.
   - The chord and scale operations are actually applied to the score.
   - Overlapping and abutting notes in the score are again joined.

References applyVoiceleadingOperations(), fixStatus(), generateLindenmayerSystem(), csound::System::inform(), initialize(), score, tieOverlappingNotes(), and writeScore().

Referenced by generate().

6.17.3.28 getAngle()

```
double csound::ChordLindenmayer::getAngle ( ) const  [virtual]
```

References angle.

6.17.3.29 getAxiom()

```
std::string csound::ChordLindenmayer::getAxiom ( ) const  [virtual]
```

References axiom.

6.17.3.30 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.17.3.31 getIterationCount()

```
int csound::ChordLindenmayer::getIterationCount ( ) const  [virtual]
```

References iterationCount.

6.17.3.32 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.17.3.33 getModality()

```
std::vector< double > csound::VoiceleadingNode::getModality ( ) const  [virtual], [inherited]
```

References csound::VoiceleadingNode::modality.

6.17.3.34 getReplacement()

```
std::string csound::ChordLindenmayer::getReplacement (
            std::string command )  [virtual]
```

References rules.

6.17.3.35 getTurtleChord()

```
Chord csound::ChordLindenmayer::getTurtleChord ( ) const  [virtual]
```

References csound::Turtle::chord, and turtle.

6.17.3.36 getTurtleModality()

```
Chord csound::ChordLindenmayer::getTurtleModality ( ) const  [virtual]
```

References csound::Turtle::modality, and turtle.

6.17.3.37 getTurtleScale()

```
Scale csound::ChordLindenmayer::getTurtleScale ( ) const  [virtual]
```

References csound::Turtle::scale, and turtle.

6.17.3.38 getTurtleScaleDegree()

```
int csound::ChordLindenmayer::getTurtleScaleDegree ( ) const  [virtual]
```

References csound::Turtle::scaleDegree, and turtle.

6.17.3.39 initialize()

```
void csound::ChordLindenmayer::initialize ( )  [virtual]
```

References csound::Turtle::initialize(), score, turtle, and turtleStack.

Referenced by generateLocally().

6.17.3.40 interpret()

```
void csound::ChordLindenmayer::interpret (
            std::vector< std::string > tokens )  [protected], [virtual]
```

The first element of the command is always the operation, the second element is always the target.

Not all operations are defined for all targets, and not all operations have the same parameters. The logic switches first on target, then on operation, then on any remaining parameters of the command.

References chordOperation(), modalityOperation(), noteOperation(), noteOrientationOperation(), noteStepOperation(), scaleDegreeOperation(), scaleOperation(), scoreOperation(), turtleOperation(), and voicingOperation().

Referenced by writeScore().

6.17.3.41 K()

```
void csound::VoiceleadingNode::K (
            double time )  [inherited]
```

Find the C of the previous segment, and contextually invert it; apply the resulting C to the current segment.

Contextual inversion is that inversion of C in which the first two pitch-classes are exchanged. If the chords are major or minor triads, produces the relative minor or major.

References csound::VoiceleadingNode::operations.

6.17.3.42 KL()

```
void csound::VoiceleadingNode::KL (
            double time,
            bool avoidParallels = true )  [inherited]
```

Find the C of the previous segment, and contextually invert it; apply the resulting C to the current segment, using the closest voiceleading from the pitches of the previous segment.

Contextual inversion is that inversion of C in which the first two pitch-classes are exchanged.

References csound::VoiceleadingNode::avoidParallels, and csound::VoiceleadingNode::operations.

6.17.3.43 KV()

```
void csound::VoiceleadingNode::KV (
            double time,
            double V_ )  [inherited]
```

Find the C of the previous segment, and contextually invert it; apply the resulting C to the current segment with voicing V.

Contextual inversion is that inversion of C in which the first two pitch-classes are exchanged.

References csound::VoiceleadingNode::operations.

6.17.3.44 L()

```
void csound::VoiceleadingNode::L (
            double time,
            bool avoidParallels = true )  [inherited]
```

Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the smoothest voice-leading from the pitches of the previous segment.

Optionally, parallel fifths can be avoided. Note that L specifies what musicians normally call voice-leading.

References csound::VoiceleadingNode::avoidParallels, and csound::VoiceleadingNode::operations.

6.17.3.45 modalityOperation()

```
void csound::ChordLindenmayer::modalityOperation (
            const std::string & operation,
            const std::string & target,
            const std::vector< std::string > & command )  [protected], [virtual]
```

References csound::addVoice(), arithmetic(), csound::System::debug(), csound::Chord::eOP(), csound::Chord::I(), csound::Chord::K(), csound::Turtle::modality, csound::real(), csound::removeVoice(), csound::Chord::T(), and turtle.

Referenced by interpret().

6.17.3.46 noteOperation()

```
void csound::ChordLindenmayer::noteOperation (
            const std::string & operation,
            const std::string & target,
            const std::vector< std::string > & command )  [protected], [virtual]
```

References csound::Score::append(), arithmetic(), csound::System::debug(), csound::Turtle::note, csound::Turtle::orientation, csound::real(), score, csound::Turtle::step, and turtle.

Referenced by interpret().

6.17.3.47 noteOrientationOperation()

```
void csound::ChordLindenmayer::noteOrientationOperation (
            const std::string & operation,
            const std::string & target,
            const std::vector< std::string > & command )  [protected], [virtual]
```

References angle, createRotation(), csound::System::debug(), csound::getIndex(), csound::Turtle::orientation, csound::real(), and turtle.

Referenced by interpret().

6.17.3.48 noteStepOperation()

```
void csound::ChordLindenmayer::noteStepOperation (
            const std::string & operation,
            const std::string & target,
            const std::vector< std::string > & command )  [protected], [virtual]
```

References arithmetic(), csound::System::debug(), csound::Turtle::step, and turtle.

Referenced by interpret().

6.17.3.49 PT()

```
void csound::VoiceleadingNode::PT (
            double time,
            double P_,
            double T )  [inherited]
```

Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified prime chord and transposition.

Note that PT specifies what musicians normally call a chord, e.g. "E flat major ninth." However, chords do not have to be in twelve tone equal temperament.

References csound::VoiceleadingNode::operations.

6.17.3.50 PTL()

```
void csound::VoiceleadingNode::PTL (
            double time,
            double P_,
            double T,
            bool avoidParallels = true )  [inherited]
```

Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified chord; the voicing of the chord will be the smoothest voice-leading from the pitches of the previous chord.

Optionally, parallel fifths can be avoided. Note that PTL specifies what musicians normally call the voice-leading of a chord.

References csound::VoiceleadingNode::avoidParallels, and csound::VoiceleadingNode::operations.

6.17.3.51 PTV()

```
void csound::VoiceleadingNode::PTV (
            double time,
            double P_,
            double T,
            double V_ )  [inherited]
```

Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified prime chord, transpostion, and voicing.

Note that PTV specifies what musicians normally call the voicing, or octavewise inversion, of a chord.

References csound::VoiceleadingNode::operations.

6.17.3.52 Q()

```
void csound::VoiceleadingNode::Q (
            double time,
            double Q_ )  [inherited]
```

Find the C of the previous segment, and contextually transpose it; apply the resulting C to the current segment.

Contextual transposition transposes C up by Q if C is an I-form, and down by Q if C is a T-form.

References csound::VoiceleadingNode::operations.

6.17.3.53 QL()

```
void csound::VoiceleadingNode::QL (
            double time,
            double Q_,
            bool avoidParallels = true )  [inherited]
```

Find the C of the previous segment, and contextually transpose it; apply the resulting C to the current segment, using the specified octavewise revoicing.

Contextual transposition transposes C up by Q if C is an I-form, and down by Q if C is a T-form.

References csound::VoiceleadingNode::avoidParallels, and csound::VoiceleadingNode::operations.

6.17.3.54 QV()

```
void csound::VoiceleadingNode::QV (
            double time,
            double Q_,
            double V_ )  [inherited]
```

Find the C of the previous segment, and contextually transpose it; apply the resulting C to the current segment with voicing V.

Contextual transposition transposes C up by Q if C is an I-form, and down by Q if C is a T-form.

References csound::VoiceleadingNode::operations.

6.17.3.55 scaleDegreeOperation()

```
void csound::ChordLindenmayer::scaleDegreeOperation (
            const std::string & operation,
            const std::string & target,
            const std::vector< std::string > & command )  [protected], [virtual]
```

References arithmetic(), csound::Turtle::chord, csound::Scale::chord(), csound::System::debug(), csound::equivalentDegree(), csound::real(), csound::Turtle::scale, csound::Turtle::scaleDegree, csound::Chord::toString(), and turtle.

Referenced by interpret().

6.17.3.56 scaleOperation()

```
void csound::ChordLindenmayer::scaleOperation (
            const std::string & operation,
            const std::string & target,
            const std::vector< std::string > & command )  [protected], [virtual]
```

References csound::Turtle::chord, csound::VoiceleadingNode::chord(), csound::Scale::chord(), csound::System::debug(), csound::Scale::degree(), csound::equivalentDegree(), csound::Scale::modulations_for_voices(), csound::parseVector(), csound::real(), csound::Turtle::scale, csound::Scale, csound::Chord::toString(), and turtle.

Referenced by interpret().

6.17.3.57 scoreOperation()

```
void csound::ChordLindenmayer::scoreOperation (
            const std::string & operation,
            const std::string & target,
            const std::vector< std::string > & command )  [protected], [virtual]
```

References csound::Turtle::chord, csound::VoiceleadingNode::chord(), csound::VoiceleadingNode::chordVoiceleading(), csound::System::debug(), csound::Event::getTime(), csound::Turtle::note, csound::VoiceleadingNode::operations, csound::real(), csound::Turtle::scale, csound::Chord::toString(), turtle, and csound::twister.

Referenced by interpret().

6.17.3.58 setAngle()

```
void csound::ChordLindenmayer::setAngle (
            double angle )  [virtual]
```

References angle.

6.17.3.59 setAxiom()

```
void csound::ChordLindenmayer::setAxiom (
            std::string axiom )  [virtual]
```

References axiom.

6.17.3.60 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.17.3.61 setIterationCount()

```
void csound::ChordLindenmayer::setIterationCount (
            int count )  [virtual]
```

References iterationCount.

6.17.3.62 setModality()

```
void csound::VoiceleadingNode::setModality (
            const std::vector< double > & pcs )  [virtual], [inherited]
```

References csound::VoiceleadingNode::modality.

6.17.3.63 setTurtleChord()

```
void csound::ChordLindenmayer::setTurtleChord (
            const csound::Chord & chord )  [virtual]
```

References csound::Turtle::chord, csound::VoiceleadingNode::chord(), and turtle.

6.17.3.64 setTurtleModality()

```
void csound::ChordLindenmayer::setTurtleModality (
            const csound::Chord & chord )  [virtual]
```

References csound::Turtle::modality, csound::VoiceleadingNode::modality, and turtle.

6.17.3.65 setTurtleScale()

```
void csound::ChordLindenmayer::setTurtleScale (
            const csound::Scale & scale )  [virtual]
```

References csound::Turtle::scale, csound::scale(), and turtle.

6.17.3.66 setTurtleScaleDegree()

```
void csound::ChordLindenmayer::setTurtleScaleDegree (
            int degree )  [virtual]
```

References csound::Turtle::scaleDegree, and turtle.

6.17.3.67 tieOverlappingNotes()

```
void csound::ChordLindenmayer::tieOverlappingNotes ( )  [protected], [virtual]
```

References score, and csound::Score::tieOverlappingNotes().

Referenced by generateLocally().

6.17.3.68 transform()

```
void csound::VoiceleadingNode::transform (
            Score & score )  [virtual], [inherited]
```

Apply all of the voice-leading operations stored within this node to the score.

Enables voice-leading operations to be used outside the context of a music graph.

Reimplemented from csound::Node.

References csound::VoiceleadingNode::apply(), csound::Score::findScale(), csound::fundamentalDomainByPredicate(), csound::Score::getDuration(), csound::Event::getTime(), csound::Score::indexAfterTime(), csound::Score::indexAtTime(), csound::System::inform(), csound::VoiceleadingNode::operations, csound::VoiceleadingOperation::rescaledBeginTime, csound::VoiceleadingNode::rescaleTimes, csound::Score::scaleActualMinima, and csound::Score::sort().

Referenced by applyVoiceleadingOperations().

6.17.3.69 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord and csound::Node::transform().

6.17.3.70 turtleOperation()

```
void csound::ChordLindenmayer::turtleOperation (
            const std::string & operation,
            const std::string & target,
            const std::vector< std::string > & command )  [protected], [virtual]
```

References csound::System::debug(), turtle, and turtleStack.

Referenced by interpret().

6.17.3.71 V()

```
void csound::VoiceleadingNode::V (
            double time,
            double V_ )  [inherited]
```

Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified voicing of the chord.

Note that V specifies what musicians normally call the voicing or octavewise inversion of the chord.

References csound::VoiceleadingNode::operations.

6.17.3.72 voicingOperation()

```
void csound::ChordLindenmayer::voicingOperation (
            const std::string & operation,
            const std::string & target,
            const std::vector< std::string > & command )  [protected], [virtual]
```

References arithmetic(), csound::System::debug(), turtle, and csound::Turtle::voicing.

Referenced by interpret().

6.17.3.73 writeScore()

```
void csound::ChordLindenmayer::writeScore ( )  [protected], [virtual]
```

Parses the final production into commands, each a tuple of strings, and interprets each command to write notes and chord progressions into the score.

References interpret(), and production.

Referenced by generateLocally().

6.17.4 Field Documentation

6.17.4.1 angle

```
double csound::ChordLindenmayer::angle
```

Referenced by createRotation(), getAngle(), noteOrientationOperation(), and setAngle().

6.17.4.2 avoidParallels

```
bool csound::VoiceleadingNode::avoidParallels  [inherited]
```

If true (the default), voice-leadings will avoid parallel fifths.

Referenced by csound::VoiceleadingNode::apply(), csound::VoiceleadingNode::CL(), csound::VoiceleadingNode::CL_name(), csound::VoiceleadingNode::KL(), csound::VoiceleadingNode::L(), csound::VoiceleadingNode::PTL(), and csound::VoiceleadingNode::QL()

6.17.4.3 axiom

```
std::string csound::ChordLindenmayer::axiom
```

Referenced by generateLindenmayerSystem(), getAxiom(), and setAxiom().

6.17.4.4 base

```
double csound::VoiceleadingNode::base  [inherited]
```

The lowest pitch of the range of voicings, as a MIDI key number (default = 36).

Referenced by csound::VoiceleadingNode::apply().

6.17.4.5 beganAt

```
clock_t csound::ChordLindenmayer::beganAt
```

6.17.4.6 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.17.4.7 divisionsPerOctave

```
size_t csound::VoiceleadingNode::divisionsPerOctave  [inherited]
```

The number of equally tempered divisions of the octave (default = 12).

Note that the octave is always size 12. The size of a division of the octave is then 1 in 12-tone equal temperament, 0.5 in 24-tone equal temperament, 1.33333 in 9-tone equal temperament, and so on.

Referenced by csound::VoiceleadingNode::apply(), csound::VoiceleadingNode::C_name(), csound::VoiceleadingNode::CL_name(), and csound::VoiceleadingNode::CV_name().

6.17.4.8 elapsed

```
clock_t csound::ChordLindenmayer::elapsed
```

6.17.4.9 endedAt

```
clock_t csound::ChordLindenmayer::endedAt
```

6.17.4.10 iterationCount

```
int csound::ChordLindenmayer::iterationCount
```

Referenced by generateLindenmayerSystem(), getIterationCount(), and setIterationCount().

6.17.4.11 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.17.4.12 modality

```
std::vector<double> csound::VoiceleadingNode::modality  [inherited]
```

Context for the K and Q operations; must have the same cardinality as the pitch-classes in use.

Referenced by csound::VoiceleadingNode::apply(), csound::VoiceleadingNode::getModality(), csound::VoiceleadingNode::setModality(), and setTurtleModality().

6.17.4.13 operations

```
std::map<double, VoiceleadingOperation> csound::VoiceleadingNode::operations  [inherited]
```

Voice-leading operations stored in order of starting time.

Referenced by csound::VoiceleadingNode::C(), csound::VoiceleadingNode::chord(), csound::VoiceleadingNode::chordVoiceleading(), csound::VoiceleadingNode::CL(), csound::VoiceleadingNode::CV(), csound::VoiceleadingNode::K(), csound::VoiceleadingNode::KL(), csound::VoiceleadingNode::KV(), csound::VoiceleadingNode::L(), csound::VoiceleadingNode::PT(), csound::VoiceleadingNode::PTL(), csound::VoiceleadingNode::PTV(), csound::VoiceleadingNode::Q(), csound::VoiceleadingNode::QL(), csound::VoiceleadingNode::QV(), scoreOperation(), csound::VoiceleadingNode::transform(), and csound::VoiceleadingNode::V().

6.17.4.14 production

```
std::string csound::ChordLindenmayer::production
```

Referenced by generateLindenmayerSystem(), and writeScore().

6.17.4.15 range

```
double csound::VoiceleadingNode::range  [inherited]
```

The range of voicings, from the lowest to the highest pitch, as a MIDI key number (default = 60).

Referenced by csound::VoiceleadingNode::apply().

6.17.4.16 rescaleTimes

```
bool csound::VoiceleadingNode::rescaleTimes  [inherited]
```

Referenced by csound::VoiceleadingNode::transform().

6.17.4.17 rules

```
std::map<std::string, std::string> csound::ChordLindenmayer::rules
```

Referenced by addRule(), clear(), generateLindenmayerSystem(), and getReplacement().

6.17.4.18 score

```
Score csound::ChordLindenmayer::score
```

Referenced by applyVoiceleadingOperations(), chordOperation(), clear(), fixStatus(), generate(), generateLocally(), initialize(), noteOperation(), and tieOverlappingNotes().

6.17.4.19 turtle

```
Turtle csound::ChordLindenmayer::turtle
```

Referenced by chordOperation(), equivalence(), getTurtleChord(), getTurtleModality(), getTurtleScale(), getTurtleScaleDegree(), initialize(), modalityOperation(), noteOperation(), noteOrientationOperation(), noteStepOperation(), scaleDegreeOperation(), scaleOperation(), scoreOperation(), setTurtleChord(), setTurtleModality(), setTurtleScale(), setTurtleScaleDegree(), turtleOperation(), and voicingOperation().

6.17.4.20 turtleStack

```
std::stack<Turtle> csound::ChordLindenmayer::turtleStack
```

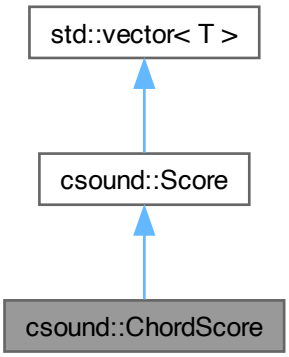
Referenced by clear(), initialize(), and turtleOperation().

6.18 csound::ChordScore Class Reference

Score equipped with chords.

```
#include <ChordSpace.hpp>
```

Inheritance diagram for csound::ChordScore:



Public Member Functions

- virtual void add (double time, double duration, double status, double instrument, double key, double velocity, double phase=0, double pan=0, double depth=0, double height=0, double pitches=4095)
- virtual void append (double time, double duration, double status, double instrument, double key, double velocity, double phase=0, double pan=0, double depth=0, double height=0, double pitches=4095)
- virtual void append (Event event)
- virtual void append_event (Event event)
- virtual void append_note (double time, double duration, double status, double instrument, double key, double velocity, double phase=0, double pan=0, double depth=0, double height=0, double pitches=4095)
- virtual void appendToCsoundScoreHeader (const std::string &text)
- virtual void arrange (int oldInstrumentNumber, int newInstrumentNumber)

  Re-assign instrument number for export to Csound score.
- virtual void arrange (int oldInstrumentNumber, int newInstrumentNumber, double gain)

  Re-assign instrument number and adjust gain for export to Csound score.
- virtual void arrange (int oldInstrumentNumber, int newInstrumentNumber, double gain, double pan)

  Re-assign instrument number, adjust gain, and change pan for export to Csound score.
- virtual void arrange_all (int oldInstrumentNumber, int newInstrumentNumber, double gain, double pan)
- virtual void conformToChords (bool tie_overlaps, bool octave_equivalence)

  Conforms the pitch-classes of the events in this to the closest pitch-class of the chord, if any, that obtains at that time.
- virtual void dump (std::ostream &stream)
- virtual void findScale ()
- virtual std::string getBlueScore (double tonesPerOctave=12.0, bool conformPitches=false)

  Translate the Silence events in this to a Csound score for blue, that is, to a list of i statements with with inso, time, duration, dbsp, pch, pan.
- virtual Chord * getChord (double time_)

  Returns a pointer to the first chord that starts at or after the specified time.
- virtual std::string getCsoundScore (double tonesPerOctave=12.0, bool conformPitches=false)

  Translate the Silence events in this to a Csound score, that is, to a list of i statements.
- virtual std::string getCsoundScoreHeader () const
- virtual double getDuration ()

  Returns the time from the first event to the last event.
- virtual double getDurationFromZero () const

  Returns the time from 0 to the final off time; this assumes that no events start before time 0.
- virtual std::vector< double > getPitches (size_t begin, size_t end, size_t divisionsPerOctave=12) const

  Return a vector containing the MIDI key numbers in the specified segment of the score.
- virtual std::vector< double > getPT (size_t begin, size_t end, double lowest, double range, size_t divisionsPer↩ Octave=12) const

  For the specified segment of the score, return the indexes for the prime chord and its transposition, within the specified range.
- virtual std::vector< double > getPTV (size_t begin, size_t end, double lowest, double range, size_t divisions↩ PerOctave=12) const

  For the specified segment of the score, return the indexes for the prime chord, its transposition, and their voicing within the specified range.
- virtual std::vector< bool > & getRescaleMinima ()
- virtual std::vector< bool > & getRescaleRanges ()
- void getScale (std::vector< Event > &score, int dimension, size_t beginAt, size_t endAt, double &minimum, double &range)
- virtual const Event & getScaleActualMinima () const
- virtual const Event & getScaleActualRanges () const

- virtual Event & getScaleTargetMinima ()
- virtual Event & getScaleTargetRanges ()
- virtual std::vector< double > getVoicing (size_t begin, size_t end, size_t divisionsPerOctave=12) const

  Iterate over each note from the beginning to end of the segment; sort the unique pitches; return those unique pitches which also have unique pitch-class sets, in order from lowest to highest in pitch; this has the effect of returning the "inversion" or "voicing", in the musician's informal sense, of the pitches in that segment of the score.
- virtual int indexAfterTime (double time)

  Return the index of the first event after the specified time, that is return "end" for the time; if the time is not found, return the size of the score.
- virtual int indexAtTime (double time)

  Return the index of the first event at or after the specified time, that is, return "begin" for the time; if the time is not found, return the size of the score.
- virtual double indexToTime (size_t index)

  Return the time of the first event at or after the specified index; if the index is not found, return DBL_MAX.
- void initialize ()
- virtual void insertChord (double tyme, const Chord chord)
- virtual void load (std::istream &stream)
- virtual void load (std::string filename)

  Loads score data from a MIDI (.mid) file, or a MusicXML (.xml) file.
- virtual void load_filename (std::string filename)
- virtual void process ()

  Calls Event::process on all Events in this.
- virtual void remove (size_t index)
- virtual void removeArrangement ()

  Remove instrument number, gain, and pan assignments.
- virtual void rescale ()
- virtual void rescale (Event &event)
- virtual void rescale (int dimension, bool rescaleMinimum, double minimum, bool rescaleRange=false, double range=0.0)
- virtual void rescale_event (Event &event)
- virtual void save (std::ostream &stream)

  Save as a MIDI file, format 1.
- virtual void save (std::string filename)

  Save as a MIDI file, format 1 (.mid) file, or as a partwise MusicXML (.xml) file, or as a Fomus music notation (.fms) file.
- virtual void save_filename (std::string filename)
- virtual void setCsoundScoreHeader (const std::string &text)
- virtual void setDuration (double targetDuration)

  Multiply existing times and durations by (targetDuration / getDuration()), i.e.
- virtual void setDurationFromZero (double targetDuration)
- virtual void setK (size_t priorBegin, size_t begin, size_t end, double base, double range)

  Find the non-unique pitch-class set of the prior segment; invert the set such that the inversion's first two pitch-classes are exchanged from the origina; conform the pitches of the current segment to that inversion.
- virtual void setKL (size_t priorBegin, size_t begin, size_t end, double base, double range, bool avoidParallels=true)

  Find the non-unique pitch-class set of the prior segment; invert the set such that the inversion's first two pitch-classes are exchanged from the original; conform the pitches of the current segment to that inversion, using the closest voice-leading from the pitches of the prior segment, optionally avoiding parallel fifths.
- virtual void setKV (size_t priorBegin, size_t begin, size_t end, double V, double base, double range)

  Find the non-unique pitch-class set of the prior segment; invert the set such that the inversion's first two pitch-classes are exchanged from the original; conform the pitches of the current segment to that inversion, with voicing V.

- virtual void setPitchClassSet (size_t begin, size_t end, const std::vector< double > &pcs, size_t divisionsPer↩ Octave=12)

  Set the pitches of the specified segment of the score to the specified pitch-class set.
- virtual void setPitches (size_t begin, size_t end, const std::vector< double > &pitches)

  Set the pitches of the specified segment of the score to the specified pitches.
- virtual void setPT (size_t begin, size_t end, double prime, double transposition, double lowest, double range, size_t divisionsPerOctave=12)

  For the specified segment of the score, adjust the pitches to match the specified indexes for the prime chord and its transposition within the specified range.
- virtual void setPTV (size_t begin, size_t end, double prime, double transposition, double voicing, double lowest, double range, size_t divisionsPerOctave=12)

  For the specified segment of the score, adjust the pitches to match the specified indexes for the prime chord, its transposition, and their voicing within the specified range.
- virtual void setQ (size_t priorBegin, size_t begin, size_t end, double Q, const std::vector< double > &context, double base, double range)

  Find the non-unique pitch-class set of the prior segment; transpose the set up by Q if the set is a T-form of the context, or down by Q if the set is an I-form of the context; then conform the pitches of the current segment to that set.
- virtual void setQL (size_t priorBegin, size_t begin, size_t end, double Q, const std::vector< double > &context, double base, double range, bool avoidParallels=true)

  Find the non-unique pitch-class set of the prior segment; transpose the set up by Q if the set is a T-form of the context, or down by Q if the set is an I-form of the context; then conform the pitches of the segment to that set, using the closest voice-leading from the pitches of the prior segment, optionally avoiding parallel fifths.
- virtual void setQV (size_t priorBegin, size_t begin, size_t end, double Q, const std::vector< double > &context, double V, double base, double range)

  Find the non-unique pitch-class set of the prior segment; transpose the set up by Q if the set is a T-form of the context, or down by Q if the set is an I-form of the context; then conform the pitches of the current segment to that set, with the voicing V.
- void setScale (std::vector< Event > &score, int dimension, bool rescaleMinimum, bool rescaleRange, size_t beginAt, size_t endAt, double targetMinimum, double targetRange)
- virtual void setVoicing (size_t begin, size_t end, const std::vector< double > &voicing, double range, size_t divisionsPerOctave=12)

  Move the pitches in the segment as little as possible to make them have the same ordering of pitch-class sets as the voicing, from the bottom to the top of the range.
- virtual void sort ()

  Sort all events in the score by time, instrument number, pitch, duration, loudness, and other dimensions as given by Event::SORT_ORDER.
- virtual void temper (double tonesPerOctave=12.0)

  Confirm pitches in this score to the closest pitch in the indicated system of equal temperament.
- virtual void tieOverlappingNotes (bool considerInstrumentNumber=false)

  If the score contains two notes of the same pitch and loudness greater than 0 that overlap in time, extend the earlier note and discard the later note.
- virtual std::string toJson ()

  Translates most of this Score to JSON:
- virtual std::string toString ()
- virtual void transform (const Eigen::MatrixXd &transformation)

  Multiply each event in this by the transformation.
- virtual void voicelead (size_t beginSource, size_t endSource, size_t beginTarget, size_t endTarget, const std↩ ::vector< double > &targetPitches, double lowest, double range, bool avoidParallelFifths, size_t divisionsPer↩ Octave=12)

  Performs voice-leading between the specified segments of the score within the specified range, using the specified target pitches.

- virtual void voicelead (size_t beginSource, size_t endSource, size_t beginTarget, size_t endTarget, double lowest, double range, bool avoidParallelFifths, size_t divisionsPerOctave=12)

  Performs voice-leading between the specified segments of the score within the specified range.
- virtual void voicelead_pitches (size_t beginSource, size_t endSource, size_t beginTarget, size_t endTarget, const std::vector< double > &targetPitches, double lowest, double range, bool avoidParallelFifths, size_t divisions↩ PerOctave=12)
- virtual void voicelead_segments (size_t beginSource, size_t endSource, size_t beginTarget, size_t endTarget, double lowest, double range, bool avoidParallelFifths, size_t divisionsPerOctave=12)

Data Fields

- std::map< double, Chord > chords_for_times
- std::string csound_score_header

  Arbitrary text that is prepended to the Csound score.
- T elements

  STL member.
- std::map< int, double > gains
- MidiFile midifile
- std::map< int, double > pans
- std::map< int, double > reassignments
- std::vector< bool > rescaleMinima
- std::vector< bool > rescaleRanges
- Event scaleActualMaxima
- Event scaleActualMinima
- Event scaleActualRanges
- Event scaleTargetMinima
- Event scaleTargetRanges

Protected Member Functions

- void createMusicModel ()

6.18.1 Detailed Description

Score equipped with chords.

The notes in the score may be conformed to the chord that obtains at the time of the notes. The times and durations of notes and chords are rescaled together. This is done by finding minimum and maximum times by counting both note times and chord times.

6.18.2 Member Function Documentation

6.18.2.1 add()

```
void csound::Score::add (
            double time,
            double duration,
            double status,
            double instrument,
            double key,
            double velocity,
            double phase = 0,
            double pan = 0,
            double depth = 0,
            double height = 0,
            double pitches = 4095 )  [virtual], [inherited]
```

References csound::fundamentalDomainByPredicate(), and csound::Event::setTime().

6.18.2.2 append() [1/2]

```
void csound::Score::append (
            double time,
            double duration,
            double status,
            double instrument,
            double key,
            double velocity,
            double phase = 0,
            double pan = 0,
            double depth = 0,
            double height = 0,
            double pitches = 4095 )  [virtual], [inherited]
```

References csound::fundamentalDomainByPredicate(), and csound::Event::setTime().

6.18.2.3 append() [2/2]

```
void csound::Score::append (
            Event event )  [virtual], [inherited]
```

References csound::fundamentalDomainByPredicate().

Referenced by csound::ChordLindenmayer::chordOperation(), csound::ImageToScore2::generateLocally(), csound::Score::load(), csound::KMeansMCRM::means_to_notes(), csound::ChordLindenmayer::noteOperation(), csound::notes(), csound::StrangeAttractor::ren csound::seqToScore(), csound::toScore(), csound::CounterpointNode::transform(), csound::CMaskNode::translate_to_silence(), csound::Intercut::traverse(), csound::Stack::traverse(), and csound::Koch::traverse().

6.18.2.4 append_event()

```
void csound::Score::append_event (
            Event event )  [virtual], [inherited]
```

References csound::fundamentalDomainByPredicate().

6.18.2.5 append_note()

```
void csound::Score::append_note (
            double time,
            double duration,
            double status,
            double instrument,
            double key,
            double velocity,
            double phase = 0,
            double pan = 0,
            double depth = 0,
            double height = 0,
            double pitches = 4095 )  [virtual], [inherited]
```

References csound::fundamentalDomainByPredicate(), and csound::Event::setTime().

6.18.2.6 appendToCsoundScoreHeader()

```
void csound::Score::appendToCsoundScoreHeader (
            const std::string & text )  [virtual], [inherited]
```

References csound::Score::csound_score_header, and csound::fundamentalDomainByPredicate().

Referenced by csound::CMaskNode::translate_to_silence().

6.18.2.7 arrange() [1/3]

```
void csound::Score::arrange (
            int oldInstrumentNumber,
            int newInstrumentNumber )  [virtual], [inherited]
```

Re-assign instrument number for export to Csound score.

References csound::fundamentalDomainByPredicate(), and csound::Score::reassignments.

Referenced by csound::MusicModel::arrange(), csound::MusicModel::arrange(), and csound::MusicModel::arrange().

6.18.2.8 arrange() [2/3]

```
void csound::Score::arrange (
            int oldInstrumentNumber,
            int newInstrumentNumber,
            double gain )  [virtual], [inherited]
```

Re-assign instrument number and adjust gain for export to Csound score.

References csound::fundamentalDomainByPredicate(), csound::Score::gains, and csound::Score::reassignments.

6.18.2.9 arrange() [3/3]

```
void csound::Score::arrange (
            int oldInstrumentNumber,
            int newInstrumentNumber,
            double gain,
            double pan )  [virtual], [inherited]
```

Re-assign instrument number, adjust gain, and change pan for export to Csound score.

References csound::fundamentalDomainByPredicate(), csound::Score::gains, csound::Score::pans, and csound::Score::reassignments.

6.18.2.10 arrange_all()

```
void csound::Score::arrange_all (
            int oldInstrumentNumber,
            int newInstrumentNumber,
            double gain,
            double pan )  [virtual], [inherited]
```

References csound::fundamentalDomainByPredicate(), csound::Score::gains, csound::Score::pans, and csound::Score::reassignments.

6.18.2.11 conformToChords()

```
SILENCE_PUBLIC void csound::ChordScore::conformToChords (
            bool tie_overlaps,
            bool octave_equivalence )  [virtual]
```

Conforms the pitch-classes of the events in this to the closest pitch-class of the chord, if any, that obtains at that time.

References csound::conformToChord_equivalence(), and csound::sort().

6.18.2.12 createMusicModel()

```
void csound::Score::createMusicModel ( )  [protected], [inherited]
```

6.18.2.13 dump()

```
void csound::Score::dump (
            std::ostream & stream )  [virtual], [inherited]
```

References csound::fundamentalDomainByPredicate().

Referenced by csound::Score::toString().

6.18.2.14 findScale()

```
void csound::Score::findScale ( )  [virtual], [inherited]
```

References csound::Event::ELEMENT_COUNT, csound::fundamentalDomainByPredicate(), csound::Score::getScale(), csound::Score::scaleActualMaxima, csound::Score::scaleActualMinima, csound::Score::scaleActualRanges, and csound::Score::sort().

Referenced by csound::Score::toJson(), csound::VoiceleadingNode::transform(), and csound::Koch::traverse().

6.18.2.15 getBlueScore()

```
std::string csound::Score::getBlueScore (
            double tonesPerOctave = 12.0,
            bool conformPitches = false )  [virtual], [inherited]
```

Translate the Silence events in this to a Csound score for blue, that is, to a list of i statements with with inso, time, duration, dbsp, pch, pan.

References csound::Score::csound_score_header, csound::fundamentalDomainByPredicate(), csound::Score::gains, csound::Score::pans, csound::Score::reassignments, and csound::Score::sort().

6.18.2.16 getChord()

```
SILENCE_PUBLIC Chord * csound::ChordScore::getChord (
            double time_ )  [virtual]
```

Returns a pointer to the first chord that starts at or after the specified time.

If there is no such chord, a null pointer is returned.

6.18.2.17 getCsoundScore()

```
std::string csound::Score::getCsoundScore (
            double tonesPerOctave = 12.0,
            bool conformPitches = false )  [virtual], [inherited]
```

Translate the Silence events in this to a Csound score, that is, to a list of i statements.

The Silence events are rounded off to the nearest equally tempered pitch by the specified number of tones per octave; if this argument is zero, the pitch is not tempered. The Silence events are conformed to the nearest pitch-class set in the pitch-class set dimension of the event, if the conform pitches argument is true; otherwise, the pitches are not conformed.

References csound::Score::csound_score_header, csound::fundamentalDomainByPredicate(), csound::Score::gains, csound::Score::pans, csound::Score::reassignments, and csound::Score::sort().

Referenced by csound::MusicModel::createCsoundScore(), and main().

6.18.2.18 getCsoundScoreHeader()

```
std::string csound::Score::getCsoundScoreHeader ( ) const  [virtual], [inherited]
```

References csound::Score::csound_score_header.

Referenced by csound::ScoreNode::generate().

6.18.2.19 getDuration()

```
SILENCE_PUBLIC double csound::ChordScore::getDuration ( )  [virtual]
```

Returns the time from the first event to the last event.

Reimplemented from csound::Score.

References csound::sort().

6.18.2.20 getDurationFromZero()

```
double csound::Score::getDurationFromZero ( ) const  [virtual], [inherited]
```

Returns the time from 0 to the final off time; this assumes that no events start before time 0.

sort();

References csound::fundamentalDomainByPredicate().

Referenced by csound::Score::setDurationFromZero().

6.18.2.21 getPitches()

```
std::vector< double > csound::Score::getPitches (
            size_t begin,
            size_t end,
            size_t divisionsPerOctave = 12 ) const  [virtual], [inherited]
```

Return a vector containing the MIDI key numbers in the specified segment of the score.

References csound::chord(), csound::fundamentalDomainByPredicate(), csound::Event::getKey_tempered(), csound::System::inform(), and csound::printChord().

Referenced by csound::Score::getPT(), csound::Score::getPTV(), csound::Score::getVoicing(), csound::Score::setK(), csound::Score::setKL(), csound::Score::setKV(), csound::Score::setPT(), csound::Score::setQ(), csound::Score::setQL(), csound::Score::setQV(), csound::Score::voicelead(), and csound::Score::voicelead().

6.18.2.22 getPT()

```
std::vector< double > csound::Score::getPT (
            size_t begin,
            size_t end,
            double lowest,
            double range,
            size_t divisionsPerOctave = 12 ) const  [virtual], [inherited]
```

For the specified segment of the score, return the indexes for the prime chord and its transposition, within the specified range.

See: http://ruccas.org/pub/Gogins/music_atoms.pdf

References csound::chord(), csound::fundamentalDomainByPredicate(), csound::Score::getPitches(), csound::Voicelead::pitchClassSetTo and csound::Voicelead::uniquePcs().

6.18.2.23 getPTV()

```
std::vector< double > csound::Score::getPTV (
            size_t begin,
            size_t end,
            double lowest,
            double range,
            size_t divisionsPerOctave = 12 ) const  [virtual], [inherited]
```

For the specified segment of the score, return the indexes for the prime chord, its transposition, and their voicing within the specified range.

Each of these indexes forms an additive cyclic group.

See: http://ruccas.org/pub/Gogins/music_atoms.pdf

References csound::chord(), csound::Voicelead::chordToPTV(), csound::fundamentalDomainByPredicate(), and csound::Score::getPitches().

Referenced by csound::VoiceleadingNode::apply().

6.18.2.24 getRescaleMinima()

```
std::vector< bool > & csound::Score::getRescaleMinima ( )  [virtual], [inherited]
```

References csound::Score::rescaleMinima.

6.18.2.25 getRescaleRanges()

```
std::vector< bool > & csound::Score::getRescaleRanges ( )  [virtual], [inherited]
```

References csound::Score::rescaleRanges.

6.18.2.26 getScale()

```
SILENCE_PUBLIC void csound::ChordScore::getScale (
            std::vector< Event > & score,
            int dimension,
            size_t beginAt,
            size_t endAt,
            double & minimum,
            double & range )
```

References csound::Event::getDuration(), csound::Event::getTime(), and csound::sort().

6.18.2.27 getScaleActualMinima()

```
const Event & csound::Score::getScaleActualMinima ( ) const  [virtual], [inherited]
```

References csound::Score::scaleActualMinima.

6.18.2.28 getScaleActualRanges()

```
const Event & csound::Score::getScaleActualRanges ( ) const  [virtual], [inherited]
```

References csound::Score::scaleActualRanges.

6.18.2.29 getScaleTargetMinima()

```
Event & csound::Score::getScaleTargetMinima ( )  [virtual], [inherited]
```

References csound::Score::scaleTargetMinima.

6.18.2.30 getScaleTargetRanges()

```
Event & csound::Score::getScaleTargetRanges ( )  [virtual], [inherited]
```

References csound::Score::scaleTargetRanges.

6.18.2.31 getVoicing()

```
std::vector< double > csound::Score::getVoicing (
            size_t begin,
            size_t end,
            size_t divisionsPerOctave = 12 ) const  [virtual], [inherited]
```

Iterate over each note from the beginning to end of the segment; sort the unique pitches; return those unique pitches which also have unique pitch-class sets, in order from lowest to highest in pitch; this has the effect of returning the "inversion" or "voicing", in the musician's informal sense, of the pitches in that segment of the score.

References csound::fundamentalDomainByPredicate(), csound::Score::getPitches(), csound::System::inform(), csound::Voicelead::pc(), csound::printChord(), and csound::Voicelead::uniquePcs().

Referenced by csound::Score::voicelead(), and csound::Score::voicelead().

6.18.2.32 indexAfterTime()

```
int csound::Score::indexAfterTime (
            double time )  [virtual], [inherited]
```

Return the index of the first event after the specified time, that is return "end" for the time; if the time is not found, return the size of the score.

Iterating from indexAtTime(t1) to indexAfterTime(t2) is guaranteed to iterate over all and only those events included from and including t1 and up to but not including t2.

References csound::fundamentalDomainByPredicate().

Referenced by csound::VoiceleadingNode::transform().

6.18.2.33 indexAtTime()

```
int csound::Score::indexAtTime (
            double time )  [virtual], [inherited]
```

Return the index of the first event at or after the specified time, that is, return "begin" for the time; if the time is not found, return the size of the score.

Iterating from indexAtTime(t1) to indexAfterTime(t2) is guaranteed to iterate over all and only those events included between t1 and t2.

References csound::fundamentalDomainByPredicate().

Referenced by csound::VoiceleadingNode::transform().

6.18.2.34 indexToTime()

```
double csound::Score::indexToTime (
            size_t index )  [virtual], [inherited]
```

Return the time of the first event at or after the specified index; if the index is not found, return DBL_MAX.

References csound::fundamentalDomainByPredicate().

6.18.2.35 initialize()

```
void csound::Score::initialize ( )  [inherited]
```

References csound::Score::csound_score_header, csound::Event::DEPTH, csound::Event::DURATION, csound::Event::HEIGHT, csound::Event::HOMOGENEITY, csound::Event::INSTRUMENT, csound::Event::KEY, csound::Event::PAN, csound::Event::PHASE, csound::Event::PITCHES, csound::Score::rescaleMinima, csound::Score::rescaleRanges, csound::Score::scaleTargetMinima, csound::Score::scaleTargetRanges, csound::Event::STATUS, csound::Event::TIME, and csound::Event::VELOCITY.

Referenced by csound::Score::Score().

6.18.2.36 insertChord()

```
SILENCE_PUBLIC void csound::ChordScore::insertChord (
            double tyme,
            const Chord chord )  [virtual]
```

References csound::chord().

6.18.2.37 load() [1/2]

```
void csound::Score::load (
            std::istream & stream )  [virtual], [inherited]
```

References csound::Score::append(), csound::fundamentalDomainByPredicate(), and csound::iterator().

6.18.2.38 load() [2/2]

```
void csound::Score::load (
            std::string filename )  [virtual], [inherited]
```

Loads score data from a MIDI (.mid) file, or a MusicXML (.xml) file.

Non-sounding data is ignored.

References csound::System::error(), csound::fundamentalDomainByPredicate(), csound::System::inform(), and csound::Score::load().

Referenced by csound::ScoreNode::generate(), csound::Score::load(), and csound::Score::load_filename().

6.18.2.39 load_filename()

```
void csound::Score::load_filename (
            std::string filename )  [virtual], [inherited]
```

References csound::Score::load().

6.18.2.40 process()

```
void csound::Score::process ( )  [virtual], [inherited]
```

Calls Event::process on all Events in this.

References csound::fundamentalDomainByPredicate(), csound::System::inform(), and csound::Score::sort().

Referenced by csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::ExternalNode::generate(), and csound::ScoreNode::generate().

6.18.2.41 remove()

```
void csound::Score::remove (
            size_t index )  [virtual], [inherited]
```

References csound::fundamentalDomainByPredicate().

6.18.2.42 removeArrangement()

```
void csound::Score::removeArrangement ( )  [virtual], [inherited]
```

Remove instrument number, gain, and pan assignments.

References csound::Score::gains, csound::Score::pans, and csound::Score::reassignments.

Referenced by csound::MusicModel::removeArrangement().

6.18.2.43 rescale() [1/3]

```
void csound::Score::rescale ( )  [virtual], [inherited]
```

References csound::Event::ELEMENT_COUNT, csound::fundamentalDomainByPredicate(), csound::Score::rescaleMinima, csound::Score::rescaleRanges, csound::Score::scaleTargetMinima, csound::Score::scaleTargetRanges, csound::Score::setScale(), and csound::Score::sort().

Referenced by csound::Score::rescale_event().

6.18.2.44 rescale() [2/3]

```
void csound::Score::rescale (
            Event & event )  [virtual], [inherited]
```

References csound::Event::HOMOGENEITY, csound::Score::rescaleMinima, csound::Score::rescaleRanges, csound::scale(), csound::Score::scaleActualMinima, csound::Score::scaleActualRanges, csound::Score::scaleTargetMinima, and csound::Score::scaleTargetRanges.

6.18.2.45 rescale() [3/3]

```
void csound::Score::rescale (
            int dimension,
            bool rescaleMinimum,
            double minimum,
            bool rescaleRange = false,
            double range = 0.0 )  [virtual], [inherited]
```

References csound::fundamentalDomainByPredicate(), and csound::Score::setScale().

6.18.2.46 rescale_event()

```
void csound::Score::rescale_event (
            Event & event )  [virtual], [inherited]
```

References csound::fundamentalDomainByPredicate(), and csound::Score::rescale().

6.18.2.47 save() [1/2]

```
void csound::Score::save (
            std::ostream & stream )  [virtual], [inherited]
```

Save as a MIDI file, format 1.

References csound::fundamentalDomainByPredicate(), csound::note(), and csound::Score::sort().

6.18.2.48 save() [2/2]

```
void csound::Score::save (
            std::string filename )  [virtual], [inherited]
```

Save as a MIDI file, format 1 (.mid) file, or as a partwise MusicXML (.xml) file, or as a Fomus music notation (.fms) file.

Only sounding data is saved.

References csound::System::error(), csound::fundamentalDomainByPredicate(), csound::System::inform(), csound::Score::save(), and csound::Score::sort().

Referenced by csound::MusicModel::csoundArgv(), csound::MusicModel::generate(), csound::MusicModel::processArgs(), csound::Score::save(), and csound::Score::save_filename().

6.18.2.49 save_filename()

```
void csound::Score::save_filename (
            std::string filename )  [virtual], [inherited]
```

References csound::Score::save().

6.18.2.50 setCsoundScoreHeader()

```
void csound::Score::setCsoundScoreHeader (
            const std::string & text )  [virtual], [inherited]
```

References csound::Score::csound_score_header, and csound::fundamentalDomainByPredicate().

6.18.2.51 setDuration()

```
SILENCE_PUBLIC void csound::ChordScore::setDuration (
            double targetDuration )  [virtual]
```

Multiply existing times and durations by (targetDuration / getDuration()), i.e.

stretch or shrink musical time.

Reimplemented from csound::Score.

References csound::chord(), csound::Event::getTime(), and csound::sort().

6.18.2.52 setDurationFromZero()

```
void csound::Score::setDurationFromZero (
            double targetDuration )  [virtual], [inherited]
```

References csound::fundamentalDomainByPredicate(), csound::Score::getDurationFromZero(), and csound::Event::getTime().

6.18.2.53 setK()

```
void csound::Score::setK (
            size_t priorBegin,
            size_t begin,
            size_t end,
            double base,
            double range )  [virtual], [inherited]
```

Find the non-unique pitch-class set of the prior segment; invert the set such that the inversion's first two pitch-classes are exchanged from the origina; conform the pitches of the current segment to that inversion.

References csound::fundamentalDomainByPredicate(), csound::Score::getPitches(), csound::Voicelead::K(), csound::printChord(), csound::Score::setPitchClassSet(), and csound::Voicelead::uniquePcs().

Referenced by csound::VoiceleadingNode::apply().

6.18.2.54 setKL()

```
void csound::Score::setKL (
            size_t priorBegin,
            size_t begin,
            size_t end,
            double base,
            double range,
            bool avoidParallels = true )  [virtual], [inherited]
```

Find the non-unique pitch-class set of the prior segment; invert the set such that the inversion's first two pitch-classes are exchanged from the original; conform the pitches of the current segment to that inversion, using the closest voice-leading from the pitches of the prior segment, optionally avoiding parallel fifths.

References csound::fundamentalDomainByPredicate(), csound::Score::getPitches(), csound::Voicelead::K(), csound::Voicelead::uniquePc and csound::Score::voicelead().

Referenced by csound::VoiceleadingNode::apply().

6.18.2.55 setKV()

```
void csound::Score::setKV (
            size_t priorBegin,
            size_t begin,
            size_t end,
            double V,
            double base,
            double range )  [virtual], [inherited]
```

Find the non-unique pitch-class set of the prior segment; invert the set such that the inversion's first two pitch-classes are exchanged from the original; conform the pitches of the current segment to that inversion, with voicing V.

References csound::fundamentalDomainByPredicate(), csound::Score::getPitches(), csound::Voicelead::K(), csound::Voicelead::pitchClas csound::Score::setPTV(), and csound::Voicelead::uniquePcs().

Referenced by csound::VoiceleadingNode::apply().

6.18.2.56 setPitchClassSet()

```
void csound::Score::setPitchClassSet (
            size_t begin,
            size_t end,
            const std::vector< double > & pcs,
            size_t divisionsPerOctave = 12 )  [virtual], [inherited]
```

Set the pitches of the specified segment of the score to the specified pitch-class set.

Each pitch in the score is moved to the closest pitch-class in the specified set.

References csound::Voicelead::conformToPitchClassSet(), csound::fundamentalDomainByPredicate(), and csound::Event::setKey().

Referenced by csound::VoiceleadingNode::apply(), csound::Score::setK(), csound::Score::setPT(), csound::Score::setQ(), and csound::Score::voicelead().

6.18.2.57 setPitches()

```
void csound::Score::setPitches (
            size_t begin,
            size_t end,
            const std::vector< double > & pitches )  [virtual], [inherited]
```

Set the pitches of the specified segment of the score to the specified pitches.

Each pitch in the score is moved to the closest pitch in the specified pitches.

References csound::Voicelead::closestPitch(), and csound::fundamentalDomainByPredicate().

Referenced by csound::Score::setPTV(), csound::Score::voicelead(), and csound::Score::voicelead().

6.18.2.58 setPT()

```
void csound::Score::setPT (
            size_t begin,
            size_t end,
            double prime,
            double transposition,
            double lowest,
            double range,
            size_t divisionsPerOctave = 12 )  [virtual], [inherited]
```

For the specified segment of the score, adjust the pitches to match the specified indexes for the prime chord and its transposition within the specified range.

See: http://ruccas.org/pub/Gogins/music_atoms.pdf

References csound::fundamentalDomainByPredicate(), csound::Score::getPitches(), csound::System::inform(), csound::Voicelead::pAndTtoPitchClassSet(), csound::printChord(), csound::Score::setPitchClassSet(), csound::T(), and csound::Voicelead::uniquePcs().

Referenced by csound::VoiceleadingNode::apply().

6.18.2.59 setPTV()

```
void csound::Score::setPTV (
            size_t begin,
            size_t end,
            double prime,
            double transposition,
            double voicing,
            double lowest,
            double range,
            size_t divisionsPerOctave = 12 )  [virtual], [inherited]
```

For the specified segment of the score, adjust the pitches to match the specified indexes for the prime chord, its transposition, and their voicing within the specified range.

Each of these indexes forms an additive cyclic group.

See: http://ruccas.org/pub/Gogins/music_atoms.pdf

References csound::fundamentalDomainByPredicate(), csound::System::inform(), csound::printChord(), csound::Voicelead::ptvToChord(), csound::Score::setPitches(), csound::T(), and csound::Voicelead::uniquePcs().

Referenced by csound::VoiceleadingNode::apply(), csound::Score::setKV(), and csound::Score::setQV().

6.18.2.60 setQ()

```
void csound::Score::setQ (
            size_t priorBegin,
            size_t begin,
            size_t end,
            double Q,
            const std::vector< double > & context,
            double base,
            double range )  [virtual], [inherited]
```

Find the non-unique pitch-class set of the prior segment; transpose the set up by Q if the set is a T-form of the context, or down by Q if the set is an I-form of the context; then conform the pitches of the current segment to that set.

The context will be reduced or doubled as required to match the cardinality of the set.

References csound::fundamentalDomainByPredicate(), csound::Score::getPitches(), csound::System::inform(), csound::matchContextSize(), csound::printChord(), csound::Voicelead::Q(), csound::Score::setPitchClassSet(), and csound::Voicelead::uniquePcs().

Referenced by csound::VoiceleadingNode::apply().

6.18.2.61 setQL()

```
void csound::Score::setQL (
            size_t priorBegin,
            size_t begin,
            size_t end,
            double Q,
            const std::vector< double > & context,
            double base,
            double range,
            bool avoidParallels = true )  [virtual], [inherited]
```

Find the non-unique pitch-class set of the prior segment; transpose the set up by Q if the set is a T-form of the context, or down by Q if the set is an I-form of the context; then conform the pitches of the segment to that set, using the closest voice-leading from the pitches of the prior segment, optionally avoiding parallel fifths.

The context will be reduced or doubled as required to match the cardinality of the set.

References csound::fundamentalDomainByPredicate(), csound::Score::getPitches(), csound::matchContextSize(), csound::printChord(), csound::Voicelead::Q(), csound::Voicelead::uniquePcs(), and csound::Score::voicelead().

Referenced by csound::VoiceleadingNode::apply().

6.18.2.62 setQV()

```
void csound::Score::setQV (
            size_t priorBegin,
            size_t begin,
            size_t end,
            double Q,
            const std::vector< double > & context,
            double V,
            double base,
            double range )  [virtual], [inherited]
```

Find the non-unique pitch-class set of the prior segment; transpose the set up by Q if the set is a T-form of the context, or down by Q if the set is an I-form of the context; then conform the pitches of the current segment to that set, with the voicing V.

The context will be reduced or doubled as required to match the cardinality of the set.

References csound::fundamentalDomainByPredicate(), csound::Score::getPitches(), csound::matchContextSize(), csound::Voicelead::pitchClassSetToPandT(), csound::printChord(), csound::Voicelead::Q(), csound::Score::setPTV(), and csound::Voicelead::uniquePcs().

Referenced by csound::VoiceleadingNode::apply().

6.18.2.63 setScale()

```
SILENCE_PUBLIC void csound::ChordScore::setScale (
            std::vector< Event > & score,
            int dimension,
            bool rescaleMinimum,
            bool rescaleRange,
            size_t beginAt,
            size_t endAt,
            double targetMinimum,
            double targetRange )
```

References csound::chord(), csound::scale(), and csound::sort().

6.18.2.64 setVoicing()

```
void csound::Score::setVoicing (
            size_t begin,
            size_t end,
            const std::vector< double > & voicing,
            double range,
            size_t divisionsPerOctave = 12 )  [virtual], [inherited]
```

Move the pitches in the segment as little as possible to make them have the same ordering of pitch-class sets as the voicing, from the bottom to the top of the range.

This has the effect of "inverting" or "re-voicing", in the musician's informal sense, the pitches in that segment of the score.

References csound::Voicelead::conformToPitchClassSet(), csound::fundamentalDomainByPredicate(), csound::Voicelead::pc(), and csound::Voicelead::pcs().

6.18.2.65 sort()

```
void csound::Score::sort ( )  [virtual], [inherited]
```

Sort all events in the score by time, instrument number, pitch, duration, loudness, and other dimensions as given by Event::SORT_ORDER.

References csound::fundamentalDomainByPredicate().

Referenced by csound::Score::findScale(), csound::ScoreModel::generate(), csound::ScoreNode::generate(), csound::ExternalNode::generateLocally(), csound::ImageToScore2::generateLocally(), csound::Score::getBlueScore(), csound::Score::getCsoundScore(), csound::Score::getDuration(), csound::Score::process(), csound::Score::rescale(), csound::Score::save(), csound::Score::save(), csound::Score::tieOverlappingNotes(), csound::Score::toJson(), csound::Cell::transform(), csound::CellRepeat::transform(), csound::CounterpointNode::transform(), csound::VoiceleadingNode::transform csound::Composition::translateToNotation(), and csound::Koch::traverse().

6.18.2.66 temper()

```
void csound::Score::temper (
            double tonesPerOctave = 12.0 )  [virtual], [inherited]
```

Confirm pitches in this score to the closest pitch in the indicated system of equal temperament.

References csound::fundamentalDomainByPredicate().

Referenced by csound::ScoreModel::generate().

6.18.2.67 tieOverlappingNotes()

```
void csound::Score::tieOverlappingNotes (
            bool considerInstrumentNumber = false )  [virtual], [inherited]
```

If the score contains two notes of the same pitch and loudness greater than 0 that overlap in time, extend the earlier note and discard the later note.

References csound::fundamentalDomainByPredicate(), csound::Event::setOffTime(), and csound::Score::sort().

Referenced by csound::ScoreModel::generate(), and csound::ChordLindenmayer::tieOverlappingNotes().

6.18.2.68 toJson()

```
std::string csound::Score::toJson ( )  [virtual], [inherited]
```

Translates most of this Score to JSON:

1. The vector of Events, sorted and otherwise massaged.
2. The actual minima, maxima, and ranges. The JSON schema is: { events: [[],...], minima: [],' maxima: [], ranges: [] }; This is useful, e.g., for sending a complete score to the JavaScript context of a Web page for display using WebGL or Three.js.

References csound::Score::findScale(), csound::fundamentalDomainByPredicate(), csound::Score::scaleActualMaxima, csound::Score::scaleActualMinima, csound::Score::scaleActualRanges, and csound::Score::sort().

6.18.2.69 toString()

```
std::string csound::Score::toString ( )  [virtual], [inherited]
```

References csound::Score::dump(), and csound::fundamentalDomainByPredicate().

6.18.2.70 transform()

```
void csound::Score::transform (
            const Eigen::MatrixXd & transformation )  [virtual], [inherited]
```

Multiply each event in this by the transformation.

References csound::fundamentalDomainByPredicate().

6.18.2.71 voicelead() [1/2]

```
void csound::Score::voicelead (
            size_t beginSource,
            size_t endSource,
            size_t beginTarget,
            size_t endTarget,
            const std::vector< double > & targetPitches,
            double lowest,
            double range,
            bool avoidParallelFifths,
            size_t divisionsPerOctave = 12 )  [virtual], [inherited]
```

Performs voice-leading between the specified segments of the score within the specified range, using the specified target pitches.

The voice-leading is first the closest by taxicab norm, and then the simplest in motion, optionally avoiding parallel fifths. Only the pitches of the target notes are affected. If necessary, the number of pitches in the target chord is adjusted to match the source.

See: http://ruccas.org/pub/Gogins/music_atoms.pdf

References csound::fundamentalDomainByPredicate(), csound::System::getMessageLevel(), csound::Score::getPitches(), csound::Score::getVoicing(), csound::System::inform(), csound::System::INFORMATION_LEVEL, csound::Voicelead::nonBijectiveVoicelea csound::Voicelead::pcs(), csound::printChord(), csound::Score::setPitchClassSet(), csound::Score::setPitches(), and csound::Voicelead::uniquePcs().

6.18.2.72 voicelead() [2/2]

```
void csound::Score::voicelead (
            size_t beginSource,
            size_t endSource,
            size_t beginTarget,
            size_t endTarget,
            double lowest,
            double range,
            bool avoidParallelFifths,
            size_t divisionsPerOctave = 12 )  [virtual], [inherited]
```

Performs voice-leading between the specified segments of the score within the specified range.

The voice-leading is first the closest by taxicab norm, and then the simplest in motion, optionally avoiding parallel fifths. Only the pitches of the target notes are affected. If necessary, the number of pitches in the target chord is adjusted to match the source.

See: http://ruccas.org/pub/Gogins/music_atoms.pdf

References csound::fundamentalDomainByPredicate(), csound::System::getMessageLevel(), csound::Score::getPitches(), csound::Score::getVoicing(), csound::System::inform(), csound::System::INFORMATION_LEVEL, csound::Voicelead::nonBijectiveVoicele csound::Voicelead::pcs(), csound::printChord(), csound::Score::setPitches(), and csound::Voicelead::uniquePcs().

Referenced by csound::VoiceleadingNode::apply(), csound::Score::setKL(), csound::Score::setQL(), csound::Score::voicelead_pitches(), and csound::Score::voicelead_segments().

6.18.2.73 voicelead_pitches()

```
void csound::Score::voicelead_pitches (
            size_t beginSource,
            size_t endSource,
            size_t beginTarget,
            size_t endTarget,
            const std::vector< double > & targetPitches,
            double lowest,
            double range,
            bool avoidParallelFifths,
            size_t divisionsPerOctave = 12 )  [virtual], [inherited]
```

References csound::fundamentalDomainByPredicate(), and csound::Score::voicelead().

6.18.2.74 voicelead_segments()

```
void csound::Score::voicelead_segments (
            size_t beginSource,
            size_t endSource,
            size_t beginTarget,
            size_t endTarget,
            double lowest,
            double range,
            bool avoidParallelFifths,
            size_t divisionsPerOctave = 12 )  [virtual], [inherited]
```

References csound::fundamentalDomainByPredicate(), and csound::Score::voicelead().

6.18.3 Field Documentation

6.18.3.1 chords_for_times

```
std::map<double, Chord> csound::ChordScore::chords_for_times
```

6.18.3.2 csound_score_header

```
std::string csound::Score::csound_score_header  [inherited]
```

Arbitrary text that is prepended to the Csound score.

Should normally be Csound comments or "f" statements, or pre-composed Csound events.

Referenced by csound::Score::appendToCsoundScoreHeader(), csound::Score::getBlueScore(), csound::Score::getCsoundScore(), csound::Score::getCsoundScoreHeader(), csound::Score::initialize(), and csound::Score::setCsoundScoreHeader().

6.18.3.3 elements

```
T std::vector< T >::elements  [inherited]
```

STL member.

6.18.3.4 gains

```
std::map<int, double> csound::Score::gains  [inherited]
```

Referenced by csound::Score::arrange(), csound::Score::arrange(), csound::Score::arrange_all(), csound::Score::getBlueScore(), csound::Score::getCsoundScore(), and csound::Score::removeArrangement().

6.18.3.5 midifile

```
MidiFile csound::Score::midifile  [inherited]
```

6.18.3.6 pans

```
std::map<int, double> csound::Score::pans  [inherited]
```

Referenced by csound::Score::arrange(), csound::Score::arrange_all(), csound::Score::getBlueScore(), csound::Score::getCsoundScore() and csound::Score::removeArrangement().

6.18.3.7 reassignments

```
std::map<int, double> csound::Score::reassignments  [inherited]
```

Referenced by csound::Score::arrange(), csound::Score::arrange(), csound::Score::arrange(), csound::Score::arrange_all(), csound::Score::getBlueScore(), csound::Score::getCsoundScore(), and csound::Score::removeArrangement().

6.18.3.8 rescaleMinima

```
std::vector<bool> csound::Score::rescaleMinima  [inherited]
```

Referenced by csound::Rescale::getRescale(), csound::Score::getRescaleMinima(), csound::Score::initialize(), csound::Rescale::Rescale(), csound::Score::rescale(), csound::Score::rescale(), csound::Rescale::setRescale(), and csound::Rescale::transform().

6.18.3.9 rescaleRanges

```
std::vector<bool> csound::Score::rescaleRanges  [inherited]
```

Referenced by csound::Rescale::getRescale(), csound::Score::getRescaleRanges(), csound::Score::initialize(), csound::Rescale::Rescale(), csound::Score::rescale(), csound::Score::rescale(), csound::Rescale::setRescale(), and csound::Rescale::transform().

6.18.3.10 scaleActualMaxima

```
Event csound::Score::scaleActualMaxima  [inherited]
```

Referenced by csound::Score::findScale(), and csound::Score::toJson().

6.18.3.11 scaleActualMinima

```
Event csound::Score::scaleActualMinima  [inherited]
```

Referenced by csound::Score::findScale(), csound::Lindenmayer::generateLocally(), csound::Score::getScaleActualMinima(), csound::Score::rescale(), csound::Score::toJson(), csound::VoiceleadingNode::transform(), csound::Koch::traverse(), and csound::Lindenmayer::updateActual().

6.18.3.12 scaleActualRanges

```
Event csound::Score::scaleActualRanges  [inherited]
```

Referenced by csound::Score::findScale(), csound::Lindenmayer::generateLocally(), csound::Score::getScaleActualRanges(), csound::Score::rescale(), csound::Score::toJson(), and csound::Lindenmayer::updateActual().

6.18.3.13 scaleTargetMinima

```
Event csound::Score::scaleTargetMinima  [inherited]
```

Referenced by csound::Rescale::getRescale(), csound::Score::getScaleTargetMinima(), csound::Score::initialize(), csound::ImageToScore2::pixel_to_event(), csound::Score::rescale(), csound::Score::rescale(), csound::Rescale::setRescale(), and csound::Rescale::transform().

6.18.3.14 scaleTargetRanges

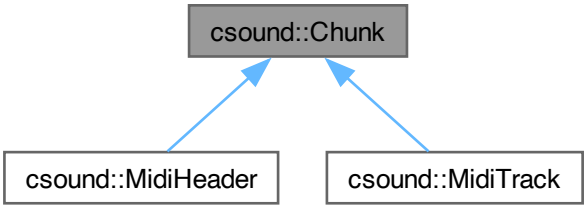
```
Event csound::Score::scaleTargetRanges  [inherited]
```

Referenced by csound::Rescale::getRescale(), csound::Score::getScaleTargetRanges(), csound::Score::initialize(), csound::ImageToScore2::pixel_to_event(), csound::Score::rescale(), csound::Score::rescale(), csound::Rescale::setRescale(), and csound::Rescale::transform().

6.19 csound::Chunk Class Reference

```
#include <Midifile.hpp>
```

Inheritance diagram for csound::Chunk:



Public Member Functions

- Chunk ()
- Chunk (const char *_id)
- Chunk (const Chunk &a)
- virtual void markChunkEnd (std::ostream &stream)
- virtual void markChunkSize (std::ostream &stream)
- virtual void markChunkStart (std::ostream &stream)
- Chunk & operator= (const Chunk &a)
- virtual void read (std::istream &stream)
- virtual void write (std::ostream &stream)
- virtual ~Chunk ()

Data Fields

- int chunkEnd
- int chunkSize
- int chunkSizePosition
- int chunkStart
- int id

6.19.1 Constructor & Destructor Documentation

6.19.1.1 Chunk() [1/3]

```
csound::Chunk::Chunk ( )
```

6.19.1.2 Chunk() [2/3]

```
csound::Chunk::Chunk (
            const char * _id )
```

References csound::MidiFile::chunkName(), and csound::fundamentalDomainByPredicate().

6.19.1.3 Chunk() [3/3]

```
csound::Chunk::Chunk (
            const Chunk & a )
```

6.19.1.4 ~Chunk()

```
csound::Chunk::~Chunk ( )  [virtual]
```

6.19.2 Member Function Documentation

6.19.2.1 markChunkEnd()

```
void csound::Chunk::markChunkEnd (
            std::ostream & stream )  [virtual]
```

References chunkEnd, chunkSize, chunkSizePosition, chunkStart, csound::fundamentalDomainByPredicate(), and csound::MidiFile::writeInt().

Referenced by csound::MidiHeader::write(), and csound::MidiTrack::writeOut().

6.19.2.2 markChunkSize()

```
void csound::Chunk::markChunkSize (
            std::ostream & stream )  [virtual]
```

References chunkSizePosition, and csound::fundamentalDomainByPredicate().

Referenced by write().

6.19.2.3 markChunkStart()

```
void csound::Chunk::markChunkStart (
            std::ostream & stream )  [virtual]
```

References chunkStart, and csound::fundamentalDomainByPredicate().

Referenced by write().

6.19.2.4 operator=()

```
Chunk & csound::Chunk::operator= (
            const Chunk & a )
```

References chunkEnd, chunkSize, chunkSizePosition, chunkStart, and id.

6.19.2.5 read()

```
void csound::Chunk::read (
            std::istream & stream )  [virtual]
```

Reimplemented in csound::MidiHeader.

References chunkSize, csound::fundamentalDomainByPredicate(), csound::System::inform(), csound::MidiFile::readInt(), and csound::System::warn().

Referenced by csound::MidiHeader::read(), and csound::MidiTrack::readIn().

6.19.2.6 write()

```
void csound::Chunk::write (
            std::ostream & stream )  [virtual]
```

Reimplemented in csound::MidiHeader.

References chunkSize, csound::fundamentalDomainByPredicate(), markChunkSize(), markChunkStart(), and csound::MidiFile::writeInt().

Referenced by csound::MidiHeader::write(), and csound::MidiTrack::writeOut().

6.19.3 Field Documentation

6.19.3.1 chunkEnd

```
int csound::Chunk::chunkEnd
```

Referenced by markChunkEnd(), operator=(), csound::MidiHeader::operator=(), and csound::MidiTrack::operator=().

6.19.3.2 chunkSize

```
int csound::Chunk::chunkSize
```

Referenced by markChunkEnd(), operator=(), csound::MidiHeader::operator=(), csound::MidiTrack::operator=(), read(), and write().

6.19.3.3 chunkSizePosition

```
int csound::Chunk::chunkSizePosition
```

Referenced by markChunkEnd(), markChunkSize(), operator=(), csound::MidiHeader::operator=(), and csound::MidiTrack::operator=().

6.19.3.4 chunkStart

```
int csound::Chunk::chunkStart
```

Referenced by markChunkEnd(), markChunkStart(), operator=(), csound::MidiHeader::operator=(), and csound::MidiTrack::operator=().

6.19.3.5 id

```
int csound::Chunk::id
```

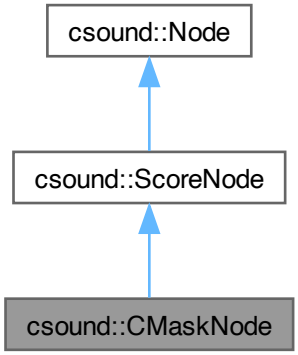
Referenced by csound::MidiFile::dump(), operator=(), csound::MidiHeader::operator=(), and csound::MidiTrack::operator=().

6.20 csound::CMaskNode Class Reference

Uses the CMask library for tendency masks to generate events as a Csound score in the format determined by the CMask parameters text.

`#include <CMaskNode.hpp>`

Inheritance diagram for csound::CMaskNode:



Public Member Functions

- virtual void addChild (Node *node)
  Adds an immediate child Node to this.
- virtual size_t childCount () const
  Returns the number of immediate children of this.
- virtual void clear ()
  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()
  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)
  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &collectingScore)
  Optionally generate notes into the score.
- virtual void generateLocally ()
- virtual Node * getChild (size_t index)
  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const
  Returns the local transformation of coordinate system.
- virtual std::string getParametersText () const
- virtual Score & getScore ()

- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void setParametersText (const std::string &parameters_text_)
- virtual void transform (Score &score_from_children)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void translate_to_silence ()

  Maps Silence score fields to Csound "i" statement pfields.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

Data Fields

- std::vector< Node * > children

  Child Nodes, if any.
- double duration

  If not 0, the score is rescaled to this duration.
- std::string importFilename

Protected Attributes

- Eigen::MatrixXd localCoordinates
- std::string parameters_text
- Score score
- std::string score_text

6.20.1 Detailed Description

Uses the CMask library for tendency masks to generate events as a Csound score in the format determined by the CMask parameters text.

The generated Csound score is also translated to a Silence score in the Silence music graph. This only works if at least the first 5 pfields in the Csound score (instrument number, time, duration, MIDI key, MIDI velocity) follow the Silence conventions declared in Event.hpp.

Score literals in the CMask parameters are simply appended to the Csound score header in the Score object, and likewise are copied directly and without any processing to the translated Csound Score.

For documentation and examples for CMask, see `Andre Bartetzki's original documentation`.

6.20.2 Member Function Documentation

6.20.2.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.20.2.2 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.20.2.3 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.20.2.4 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.20.2.5 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.20.2.6 generate()

```
void csound::ScoreNode::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented from csound::Node.

Reimplemented in csound::ExternalNode, and csound::MCRM.

References csound::ScoreNode::duration, csound::fundamentalDomainByPredicate(), csound::Score::getCsoundScoreHeader(), csound::ScoreNode::importFilename, csound::Score::load(), csound::Score::process(), csound::ScoreNode::score, csound::Score::setDuration(), and csound::Score::sort().

Referenced by csound::MCRM::generate().

6.20.2.7 generateLocally()

```
virtual void csound::CMaskNode::generateLocally ( )  [inline], [virtual]
```

References csound::fundamentalDomainByPredicate(), parameters_text, score_text, and translate_to_silence().

6.20.2.8 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.20.2.9 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.20.2.10 getParametersText()

```
virtual std::string csound::CMaskNode::getParametersText ( ) const  [inline], [virtual]
```

References parameters_text.

6.20.2.11 getScore()

```
Score & csound::ScoreNode::getScore ( )  [virtual], [inherited]
```

References csound::ScoreNode::score.

Referenced by main().

6.20.2.12 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.20.2.13 setParametersText()

```
virtual void csound::CMaskNode::setParametersText (
            const std::string & parameters_text_ )  [inline], [virtual]
```

References csound::fundamentalDomainByPredicate(), and parameters_text.

6.20.2.14 transform()

```
void csound::Node::transform (
            Score & score_from_children )  [virtual], [inherited]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented in csound::Cell, csound::CellRepeat, csound::CellAdd, csound::CellMultiply, csound::CellReflect, csound::CellSelect, csound::CellRemove, csound::CellChord, csound::CellRandom, csound::CellShuffle, csound::CounterpointNode, csound::RemoveDuplicates, csound::Transformer, csound::Random, csound::Rescale, csound::VoiceleadingNode, csound::LispTransformer, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.20.2.15 translate_to_silence()

```
virtual void csound::CMaskNode::translate_to_silence ( )  [inline], [virtual]
```

Maps Silence score fields to Csound "i" statement pfields.

As noted above, the CMask fields *must* be configured to match the Silence conventions for Csound scores. The first 5 fields are required. No more than 10 fields are used.

i_instrument = p1 i_time = p2 i_duration = p3 i_midi_key = p4 i_midi_velocity = p5 k_space_front_to_back = p6 ; Ambisonic X k_space_left_to_right = p7 ; Ambisonic Y k_space_bottom_to_top = p8; Ambisonic Z i_phase = p9 i_pitches = p10 ; Mason number

References csound::Score::append(), csound::Score::appendToCsoundScoreHeader(), csound::fundamentalDomainByPredicate(), csound::ScoreNode::score, score_text, and csound::Event::setStatus().

Referenced by generateLocally().

6.20.2.16 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord and csound::Node::transform().

6.20.3 Field Documentation

6.20.3.1 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.20.3.2 duration

```
double csound::ScoreNode::duration  [inherited]
```

If not 0, the score is rescaled to this duration.

Referenced by csound::ScoreNode::generate(), csound::ExternalNode::generateLocally(), and csound::Stack::getDuration().

6.20.3.3 importFilename

```
std::string csound::ScoreNode::importFilename  [inherited]
```

Referenced by csound::ScoreNode::generate().

6.20.3.4 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.20.3.5 parameters_text

```
std::string csound::CMaskNode::parameters_text  [protected]
```

Referenced by generateLocally(), getParametersText(), and setParametersText().

6.20.3.6 score

```
Score csound::ScoreNode::score  [protected], [inherited]
```

Referenced by csound::StrangeAttractor::evaluateAttractor(), csound::ExternalNode::generate(), csound::ScoreNode::generate(), csound::MCRM::generate(), csound::ExternalNode::generateLocally(), csound::ImageToScore2::generateLocally(), csound::Lindenmayer::generateLocally(), csound::Rescale::getRescale(), csound::ScoreNode::getScore(), csound::Lindenmayer::interpre csound::MCRM::iterate(), csound::StrangeAttractor::iterate_without_rendering(), csound::KMeansMCRM::means_to_notes(), csound::ImageToScore2::pixel_to_event(), csound::StrangeAttractor::render(), csound::Rescale::Rescale(), csound::Rescale::setRescale(csound::Cell::transform(), csound::Rescale::transform(), translate_to_silence(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Lindenmayer::updateActual().

6.20.3.7 score_text

```
std::string csound::CMaskNode::score_text  [protected]
```

Referenced by generateLocally(), and translate_to_silence().

6.21 csound::compare_by_normal_form Struct Reference

```
#include <ChordSpaceBase.hpp>
```

Public Member Functions

- bool operator() (const Chord &a, const Chord &b) const

6.21.1 Member Function Documentation

6.21.1.1 operator()()

```
bool csound::compare_by_normal_form::operator() (
            const Chord & a,
            const Chord & b ) const  [inline]
```

References csound::Chord::normal_form().

6.22 csound::compare_by_normal_order Struct Reference

```
#include <ChordSpaceBase.hpp>
```

Public Member Functions

- bool operator() (const Chord &a, const Chord &b) const

6.22.1 Member Function Documentation

6.22.1.1 operator()()

```
bool csound::compare_by_normal_order::operator() (
            const Chord & a,
            const Chord & b ) const  [inline]
```

References csound::Chord::normal_order().

6.23 csound::compare_by_op Struct Reference

```
#include <ChordSpaceBase.hpp>
```

Public Member Functions

- bool operator() (const Chord &a, const Chord &b) const

6.23.1 Member Function Documentation

6.23.1.1 operator()()

```
bool csound::compare_by_op::operator() (
            const Chord & a,
            const Chord & b ) const  [inline]
```

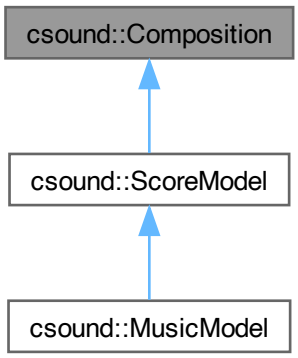
References csound::Chord::eOP().

6.24 csound::Composition Class Reference

Base class for user-defined musical compositions.

```
#include <Composition.hpp>
```

Inheritance diagram for csound::Composition:



Public Member Functions

- virtual void clear ()

  Clear all contents of this.
- virtual void clearOutputSoundfileName ()
- Composition ()
- virtual int generate ()

  Generate performance events and store them in the score.
- virtual void generateAllNames ()

  Generates all filenames and other text based on required stem, output_directory, filename extension, and metadata.
- virtual std::string getAlbum () const
- virtual std::string getArtist () const
- virtual std::string getAuthor () const
- virtual std::string getBasename () const

  Returns the complete basename of the file, i.e., the output directory plus the stem.
- virtual std::string getCdSoundfileFilepath () const

  Returns a soundfile name for a CD audio track based on the filename of this, by appending ".cd.wav" to the filename.
- virtual bool getConformPitches () const

  Returns whether or not the pitches in generated scores will be conformed to the nearest equally tempered pitch.
- virtual std::string getCopyright () const
- virtual double getDuration () const

  Returns the duration to which all times and durations of all events will be rescaled.
- virtual std::string getFileFilepath () const

  Returns the complete basename of the file, i.e., the output directory plus the filename.
- virtual std::string getFilename () const

  Returns the stem of this, which is used as a base for derived filenames (soundfile, MIDI file, etc.).
- virtual std::string getFomusfileFilepath () const

  Returns a MusicXML filename based on the filename of this, by appending ".fms" to the filename.

- virtual std::string getLicense () const
- virtual std::string getLilypondfileFilepath () const

  Returns a MusicXML filename based on the filename of this, by appending ".ly" to the filename.
- virtual std::string getMidifileFilepath () const

  Returns a MIDI filename based on the filename of this, by appending ".mid" to the filename.
- virtual std::string getMp3SoundfileFilepath () const

  Returns a soundfile name for an MP3 file based on the filename of this, by appending ".mp3" to the filename.
- virtual std::string getMusicXmlfileFilepath () const

  Returns a MusicXML filename based on the filename of this, by appending ".xml" to the filename.
- virtual std::string getNormalizedSoundfileFilepath () const

  Returns a soundfile name based on the filename of this, by appending ".norm.wav" to the filename.
- virtual std::string getOutputDirectory () const

  Returns the directory in which to place the output files of this.
- virtual std::string getOutputSoundfileFilepath () const

  Returns a soundfile name based on the filename of this, by appending ".wav" to the filename, which is the default, or a non-default ouput name which need not be a file but must be set using setOutputSoundfileName().
- virtual std::string getPerformanceRightsOrganization () const
- virtual Score & getScore ()

  Return the self-contained Score.
- virtual bool getTieOverlappingNotes () const

  Returns whether or not overlapping notes in generated scores are replaced by one note.
- virtual std::string getTimestamp () const

  Returns the time the score was generated.
- virtual std::string getTitle () const
- virtual double getTonesPerOctave () const

  Returns the number of equally tempered intervals per octave (the default is 12, 0 means non-equally tempered).
- virtual std::string getYear () const
- virtual int normalizeOutputSoundfile (double levelDb=-3.0)

  Assuming the score has been rendered, uses sox to translate the output soundfile to a normalized soundfile.
- virtual int perform ()

  Performs the current score to create an output soundfile, which should be tagged with author, timestamp, copyright, title, and optionally album.
- virtual int performAll ()

  Convenience function that calls performMaster(), and translateMaster().
- virtual int performMaster ()

  Convenience function that calls saveMidi(), saveMusicXML(), and perform().
- virtual int processArgs (const std::vector< std::string > &args)

  Pass the invoking program's command-line arguments to processArgs() and it will perform with possibly back-end-dependent options.
- virtual int processArgv (int argc, const char **argv)

  Pass the invoking program's command-line arguments to processArgs() and it will perform with possibly back-end-dependent options.
- virtual int render ()

  Convenience function that calls clear(), generate(), perform().
- virtual int renderAll ()

  Convenience function that calls clear(), generate(), performAll().
- virtual void setAlbum (std::string value)
- virtual void setArtist (std::string value)

- virtual void setAuthor (std::string value)
- virtual void setConformPitches (bool conformPitches)

  Sets whether or not the pitches in generated scores will be conformed to the nearest equally tempered pitch.
- virtual void setCopyright (std::string value)
- virtual void setDuration (double seconds)

  At the end of processing, if the defined duration is not zero, the times and durations of all events are rescaled to the defined duration.
- virtual void setFilename (std::string filename)

  Sets the filename of this – basically, the title of the composition.
- virtual void setLicense (std::string value)
- virtual void setOutputDirectory (std::string directory)

  Sets the directory in which to place the output files of this.
- virtual void setOutputSoundfileName (std::string name)

  Sets a non-default output name (could be an audio device not a file).
- virtual void setPerformanceRightsOrganization (std::string value)
- virtual void setScore (Score &score)

  Sets the score in this to the indicated score.
- virtual void setTieOverlappingNotes (bool tieOverlappingNotes)

  Sets whether or not overlapping notes in generated scores are replaced by one note.
- virtual void setTitle (std::string value)
- virtual void setTonesPerOctave (double tonesPerOctave)

  Sets the number of equally tempered intervals per octave (the default is 12, 0 means non-equally tempered).
- virtual void setYear (std::string value)
- virtual int tagFile (std::string filename) const
- virtual int translateMaster ()

  Convenience function that calls rescaleOutputSoundfile(), translateToCdAudio(), and translateToMp3().
- virtual int translateToCdAudio (double levelDb=-3.0)

  Assuming the score has been rendered, uses sox to translate the output soundfile to normalized CD-audio format.
- virtual int translateToMp3 (double bitrate=256.01, double levelDb=-3.0)

  Assuming the score has been rendered, uses sox and LAME to translate the output soundfile to normalized MP3 format.
- virtual int translateToMp4 ()

  Assuming the score has been rendered, uses sox and ffmpeg to translate the output soundfile to a normalized mp4 video suitable for uploading to YouTube.
- virtual int translateToNotation (const std::vector< std::string > partNames=std::vector< std::string >(), std::string header="")

  Saves the generated score in Fomus format and uses Fomus and Lilypond to translate that to a PDF of music notation.
- virtual void write (const char *text)

  Write as if to stderr.
- virtual ~Composition ()

Static Public Member Functions

- static std::string generateFilename ()

  Generates a versioned filename.
- static std::string makeTimestamp ()

  Returns the current locale time as a string.

Protected Attributes

- std::string album

  Optional metadata.
- std::string artist

  Required metadata.
- std::string author

  Required metadata.
- std::string base_filepath

  Generated.
- Score baseScore
- std::string bext_description

  Generated.
- std::string bext_orig_ref

  Generated.
- std::string bext_originator

  Generated.
- std::string cd_quality_filepath

  Generated.
- bool conformPitches
- std::string copyright

  Required metadata.
- double duration
- std::string flac_filepath

  Generated.
- std::string label

  Generated.
- std::string license

  Required metadata.
- std::string master_filepath

  Generated.
- std::string midi_filepath

  Generated.
- std::string mp3_filepath

  Generated.
- std::string mp4_filepath

  Generated.
- std::string normalized_master_filepath

  Generated.
- std::string notes

  Optional metadata, defaults to "Electroacoustic Music.".
- std::string output_directory

  Required.
- std::string output_filename
- std::string performance_rights_organization

  Optional metadata.
- Score & score
- std::string spectrogram_filepath

Generated.

- std::string stem

  Required.
- bool tieOverlappingNotes
- std::string timestamp

  Generated.
- double tonesPerOctave
- std::string track

  Optional metadata.
- std::string year

  Required metadata.

6.24.1 Detailed Description

Base class for user-defined musical compositions.

Contains a Score object for collecting generated Events such as notes and control messages.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 Composition()

```
csound::Composition::Composition ( )
```

6.24.2.2 ~Composition()

```
csound::Composition::~Composition ( )  [virtual]
```

6.24.3 Member Function Documentation

6.24.3.1 clear()

```
void csound::Composition::clear ( )  [virtual]
```

Clear all contents of this.

Probably should be overridden in derived classes.

Reimplemented in csound::MusicModel, and csound::ScoreModel.

References score.

Referenced by csound::MusicModel::clear(), csound::ScoreModel::clear(), render(), and renderAll().

6.24.3.2 clearOutputSoundfileName()

```
void csound::Composition::clearOutputSoundfileName ( )  [virtual]
```

References output_filename.

6.24.3.3 generate()

```
int csound::Composition::generate ( )  [virtual]
```

Generate performance events and store them in the score.

Must be overidden in derived classes.

Reimplemented in csound::MusicModel, and csound::ScoreModel.

Referenced by render(), and renderAll().

6.24.3.4 generateAllNames()

```
void csound::Composition::generateAllNames ( )  [virtual]
```

Generates all filenames and other text based on required stem, output_directory, filename extension, and metadata.

References album, artist, author, base_filepath, bext_description, bext_orig_ref, bext_originator, cd_quality_filepath, copyright, flac_filepath, csound::fundamentalDomainByPredicate(), getLicense(), getOutputDirectory(), getTitle(), csound::System::inform(), label, makeTimestamp(), master_filepath, midi_filepath, mp3_filepath, mp4_filepath, normalized_master_filepath, notes, output_directory, performance_rights_organization, spectrogram_filepath, stem, timestamp, track, and year.

Referenced by csound::MusicModel::csoundArgv(), processArgs(), and csound::MusicModel::processArgs().

6.24.3.5 generateFilename()

```
std::string csound::Composition::generateFilename ( )  [static]
```

Generates a versioned filename.

References csound::fundamentalDomainByPredicate(), and makeTimestamp().

6.24.3.6 getAlbum()

```
std::string csound::Composition::getAlbum ( ) const  [virtual]
```

References album.

Referenced by tagFile(), and translateToMp3().

6.24.3.7 getArtist()

```
std::string csound::Composition::getArtist ( ) const  [virtual]
```

References artist.

Referenced by tagFile(), and translateToNotation().

6.24.3.8 getAuthor()

```
std::string csound::Composition::getAuthor ( ) const  [virtual]
```

References author.

Referenced by tagFile(), and translateToMp3().

6.24.3.9 getBasename()

```
std::string csound::Composition::getBasename ( ) const  [virtual]
```

Returns the complete basename of the file, i.e., the output directory plus the stem.

References base_filepath.

Referenced by getFomusfileFilepath(), and getLilypondfileFilepath().

6.24.3.10 getCdSoundfileFilepath()

```
std::string csound::Composition::getCdSoundfileFilepath ( ) const  [virtual]
```

Returns a soundfile name for a CD audio track based on the filename of this, by appending ".cd.wav" to the filename.

References cd_quality_filepath.

Referenced by translateToCdAudio(), and translateToMp3().

6.24.3.11 getConformPitches()

```
bool csound::Composition::getConformPitches ( ) const  [virtual]
```

Returns whether or not the pitches in generated scores will be conformed to the nearest equally tempered pitch.

References conformPitches.

Referenced by csound::ScoreModel::generate().

6.24.3.12 getCopyright()

```
std::string csound::Composition::getCopyright ( ) const  [virtual]
```

References copyright.

Referenced by tagFile(), translateToMp3(), and translateToMp4().

6.24.3.13 getDuration()

```
double csound::Composition::getDuration ( ) const  [virtual]
```

Returns the duration to which all times and durations of all events will be rescaled.

If the duration is 0, no rescaling is performed.

References duration.

6.24.3.14 getFileFilepath()

```
std::string csound::Composition::getFileFilepath ( ) const  [virtual]
```

Returns the complete basename of the file, i.e., the output directory plus the filename.

References base_filepath.

6.24.3.15 getFilename()

```
std::string csound::Composition::getFilename ( ) const  [virtual]
```

Returns the stem of this, which is used as a base for derived filenames (soundfile, MIDI file, etc.).

References stem.

6.24.3.16 getFomusfileFilepath()

```
std::string csound::Composition::getFomusfileFilepath ( ) const  [virtual]
```

Returns a MusicXML filename based on the filename of this, by appending ".fms" to the filename.

References getBasename().

Referenced by translateToNotation().

6.24.3.17 getLicense()

```
std::string csound::Composition::getLicense ( ) const  [virtual]
```

References license.

Referenced by generateAllNames(), and tagFile().

6.24.3.18 getLilypondfileFilepath()

```
std::string csound::Composition::getLilypondfileFilepath ( ) const  [virtual]
```

Returns a MusicXML filename based on the filename of this, by appending ".ly" to the filename.

References getBasename().

6.24.3.19 getMidifileFilepath()

```
std::string csound::Composition::getMidifileFilepath ( ) const  [virtual]
```

Returns a MIDI filename based on the filename of this, by appending ".mid" to the filename.

References midi_filepath.

Referenced by csound::MusicModel::csoundArgv(), csound::MusicModel::generate(), and csound::MusicModel::processArgs().

6.24.3.20 getMp3SoundfileFilepath()

```
std::string csound::Composition::getMp3SoundfileFilepath ( ) const  [virtual]
```

Returns a soundfile name for an MP3 file based on the filename of this, by appending ".mp3" to the filename.

References mp3_filepath.

Referenced by translateToMp3().

6.24.3.21 getMusicXmlfileFilepath()

```
std::string csound::Composition::getMusicXmlfileFilepath ( ) const  [virtual]
```

Returns a MusicXML filename based on the filename of this, by appending ".xml" to the filename.

References base_filepath.

6.24.3.22 getNormalizedSoundfileFilepath()

```
std::string csound::Composition::getNormalizedSoundfileFilepath ( ) const  [virtual]
```

Returns a soundfile name based on the filename of this, by appending ".norm.wav" to the filename.

References normalized_master_filepath.

Referenced by normalizeOutputSoundfile(), and csound::MusicModel::processArgs().

6.24.3.23 getOutputDirectory()

```
std::string csound::Composition::getOutputDirectory ( ) const  [virtual]
```

Returns the directory in which to place the output files of this.

References csound::fundamentalDomainByPredicate(), and output_directory.

Referenced by generateAllNames().

6.24.3.24 getOutputSoundfileFilepath()

```
std::string csound::Composition::getOutputSoundfileFilepath ( ) const  [virtual]
```

Returns a soundfile name based on the filename of this, by appending ".wav" to the filename, which is the default, or a non-default ouput name which need not be a file but must be set using setOutputSoundfileName().

References master_filepath, and output_filename.

Referenced by csound::MusicModel::getCsoundCommand(), normalizeOutputSoundfile(), csound::MusicModel::perform(), csound::MusicModel::processArgs(), translateMaster(), and translateToCdAudio().

6.24.3.25 getPerformanceRightsOrganization()

```
std::string csound::Composition::getPerformanceRightsOrganization ( ) const  [virtual]
```

References performance_rights_organization.

6.24.3.26 getScore()

```
Score & csound::Composition::getScore ( )  [virtual]
```

Return the self-contained Score.

References score.

Referenced by csound::MusicModel::csoundArgv(), and csound::MusicModel::processArgs().

6.24.3.27 getTieOverlappingNotes()

```
bool csound::Composition::getTieOverlappingNotes ( ) const  [virtual]
```

Returns whether or not overlapping notes in generated scores are replaced by one note.

References tieOverlappingNotes.

Referenced by csound::ScoreModel::generate().

6.24.3.28 getTimestamp()

```
std::string csound::Composition::getTimestamp ( ) const  [virtual]
```

Returns the time the score was generated.

References timestamp.

Referenced by tagFile().

6.24.3.29 getTitle()

```
std::string csound::Composition::getTitle ( ) const  [virtual]
```

References stem.

Referenced by generateAllNames(), tagFile(), translateToMp3(), translateToMp4(), and translateToNotation().

6.24.3.30 getTonesPerOctave()

```
double csound::Composition::getTonesPerOctave ( ) const  [virtual]
```

Returns the number of equally tempered intervals per octave (the default is 12, 0 means non-equally tempered).

References tonesPerOctave.

Referenced by csound::ScoreModel::generate().

6.24.3.31 getYear()

```
std::string csound::Composition::getYear ( ) const  [virtual]
```

References year.

6.24.3.32 makeTimestamp()

```
std::string csound::Composition::makeTimestamp ( )  [static]
```

Returns the current locale time as a string.

References csound::fundamentalDomainByPredicate().

Referenced by generateAllNames(), and generateFilename().

6.24.3.33 normalizeOutputSoundfile()

```
int csound::Composition::normalizeOutputSoundfile (
            double levelDb = -3.0 )  [virtual]
```

Assuming the score has been rendered, uses sox to translate the output soundfile to a normalized soundfile.

References csound::fundamentalDomainByPredicate(), getNormalizedSoundfileFilepath(), getOutputSoundfileFilepath(), csound::System::inform(), and tagFile().

Referenced by translateMaster().

6.24.3.34 perform()

```
int csound::Composition::perform ( )  [virtual]
```

Performs the current score to create an output soundfile, which should be tagged with author, timestamp, copyright, title, and optionally album.

The default implementation does nothing. Must be overridden in derived classes.

Reimplemented in csound::MusicModel.

Referenced by performMaster(), and render().

6.24.3.35 performAll()

```
int csound::Composition::performAll ( )  [virtual]
```

Convenience function that calls performMaster(), and translateMaster().

References csound::fundamentalDomainByPredicate(), csound::System::inform(), csound::System::message(), performMaster(), csound::System::startTiming(), csound::System::stopTiming(), and translateMaster().

Referenced by renderAll().

6.24.3.36 performMaster()

```
int csound::Composition::performMaster ( )  [virtual]
```

Convenience function that calls saveMidi(), saveMusicXML(), and perform().

References csound::fundamentalDomainByPredicate(), csound::System::inform(), and perform().

Referenced by performAll().

6.24.3.37 processArgs()

```
int csound::Composition::processArgs (
            const std::vector< std::string > & args )  [virtual]
```

Pass the invoking program's command-line arguments to processArgs() and it will perform with possibly back-end-dependent options.

Additional arguments can be added to the args before the call. Default implementation calls renderAll().

Reimplemented in csound::MusicModel.

References generateAllNames(), and renderAll().

Referenced by processArgv().

6.24.3.38 processArgv()

```
int csound::Composition::processArgv (
            int argc,
            const char ** argv )  [virtual]
```

Pass the invoking program's command-line arguments to processArgs() and it will perform with possibly back-end-dependent options.

Default implementation calls the std::string overload.

References csound::fundamentalDomainByPredicate(), and processArgs().

Referenced by main().

6.24.3.39 render()

```
int csound::Composition::render ( )  [virtual]
```

Convenience function that calls clear(), generate(), perform().

timestamp = makeTimestamp();

Reimplemented in csound::MusicModel.

References clear(), csound::fundamentalDomainByPredicate(), generate(), and perform().

6.24.3.40 renderAll()

```
int csound::Composition::renderAll ( )  [virtual]
```

Convenience function that calls clear(), generate(), performAll().

References clear(), csound::fundamentalDomainByPredicate(), generate(), and performAll().

Referenced by processArgs().

6.24.3.41 setAlbum()

```
void csound::Composition::setAlbum (
            std::string value )  [virtual]
```

References album.

Referenced by main().

6.24.3.42 setArtist()

```
void csound::Composition::setArtist (
            std::string value )  [virtual]
```

References artist.

6.24.3.43 setAuthor()

```
void csound::Composition::setAuthor (
            std::string value )  [virtual]
```

References author.

Referenced by main().

6.24.3.44 setConformPitches()

```
void csound::Composition::setConformPitches (
            bool conformPitches )  [virtual]
```

Sets whether or not the pitches in generated scores will be conformed to the nearest equally tempered pitch.

References conformPitches.

6.24.3.45 setCopyright()

```
void csound::Composition::setCopyright (
            std::string value )  [virtual]
```

References copyright.

6.24.3.46 setDuration()

```
void csound::Composition::setDuration (
            double seconds )  [virtual]
```

At the end of processing, if the defined duration is not zero, the times and durations of all events are rescaled to the defined duration.

References duration, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.24.3.47 setFilename()

```
void csound::Composition::setFilename (
            std::string filename )  [virtual]
```

Sets the filename of this – basically, the title of the composition.

References stem.

6.24.3.48 setLicense()

```
void csound::Composition::setLicense (
            std::string value )  [virtual]
```

References license.

6.24.3.49 setOutputDirectory()

```
void csound::Composition::setOutputDirectory (
            std::string directory )  [virtual]
```

Sets the directory in which to place the output files of this.

The directory name must end with a directory separator.

References csound::fundamentalDomainByPredicate(), and output_directory.

Referenced by csound::MusicModel::processArgs().

6.24.3.50 setOutputSoundfileName()

```
void csound::Composition::setOutputSoundfileName (
            std::string name )  [virtual]
```

Sets a non-default output name (could be an audio device not a file).

References output_filename.

6.24.3.51 setPerformanceRightsOrganization()

```
void csound::Composition::setPerformanceRightsOrganization (
            std::string value )  [virtual]
```

References performance_rights_organization.

Referenced by main().

6.24.3.52 setScore()

```
void csound::Composition::setScore (
            Score & score )  [virtual]
```

Sets the score in this to the indicated score.

References csound::fundamentalDomainByPredicate(), and score.

6.24.3.53 setTieOverlappingNotes()

```
void csound::Composition::setTieOverlappingNotes (
            bool tieOverlappingNotes )  [virtual]
```

Sets whether or not overlapping notes in generated scores are replaced by one note.

References csound::fundamentalDomainByPredicate(), and tieOverlappingNotes.

Referenced by main().

6.24.3.54 setTitle()

```
void csound::Composition::setTitle (
            std::string value )  [virtual]
```

References stem.

Referenced by main().

6.24.3.55 setTonesPerOctave()

```
void csound::Composition::setTonesPerOctave (
            double tonesPerOctave )  [virtual]
```

Sets the number of equally tempered intervals per octave (the default is 12, 0 means non-equally tempered).

References tonesPerOctave.

6.24.3.56 setYear()

```
void csound::Composition::setYear (
            std::string value )  [virtual]
```

References year.

Referenced by main().

6.24.3.57 tagFile()

```
int csound::Composition::tagFile (
            std::string filename ) const  [virtual]
```

References csound::fundamentalDomainByPredicate(), getAlbum(), getArtist(), getAuthor(), getCopyright(), getLicense(), getTimestamp(), getTitle(), and csound::System::inform().

Referenced by normalizeOutputSoundfile(), translateMaster(), and translateToCdAudio().

6.24.3.58 translateMaster()

```
int csound::Composition::translateMaster ( )  [virtual]
```

Convenience function that calls rescaleOutputSoundfile(), translateToCdAudio(), and translateToMp3().

References csound::fundamentalDomainByPredicate(), getOutputSoundfileFilepath(), csound::System::inform(), csound::System::message(), normalizeOutputSoundfile(), csound::System::startTiming(), csound::System::stopTiming(), tagFile(), translateToCdAudio(), translateToMp3(), and translateToMp4().

Referenced by performAll(), and csound::MusicModel::processArgs().

6.24.3.59 translateToCdAudio()

```
int csound::Composition::translateToCdAudio (
            double levelDb = -3.0 )  [virtual]
```

Assuming the score has been rendered, uses sox to translate the output soundfile to normalized CD-audio format.

References csound::fundamentalDomainByPredicate(), getCdSoundfileFilepath(), getOutputSoundfileFilepath(), csound::System::inform(), and tagFile().

Referenced by translateMaster().

6.24.3.60 translateToMp3()

```
int csound::Composition::translateToMp3 (
            double bitrate = 256.01,
            double levelDb = -3.0 )  [virtual]
```

Assuming the score has been rendered, uses sox and LAME to translate the output soundfile to normalized MP3 format.

References author, csound::fundamentalDomainByPredicate(), getAlbum(), getAuthor(), getCdSoundfileFilepath(), getCopyright(), getMp3SoundfileFilepath(), getTitle(), csound::System::inform(), and year.

Referenced by translateMaster().

6.24.3.61 translateToMp4()

```
int csound::Composition::translateToMp4 ( )  [virtual]
```

Assuming the score has been rendered, uses sox and ffmpeg to translate the output soundfile to a normalized mp4 video suitable for uploading to YouTube.

References album, artist, author, cd_quality_filepath, copyright, csound::fundamentalDomainByPredicate(), getCopyright(), getTitle(), csound::System::inform(), master_filepath, mp4_filepath, notes, performance_rights_organization, spectrogram_filepath, stem, track, and year.

Referenced by translateMaster().

6.24.3.62 translateToNotation()

```
int csound::Composition::translateToNotation (
            const std::vector< std::string > partNames = std::vector<std::string>(),
            std::string header = "" )  [virtual]
```

Saves the generated score in Fomus format and uses Fomus and Lilypond to translate that to a PDF of music notation.

A meter of 4/4 and a tempo of MM 120 is assumed. A vector of part names may be supplied.

References duration, csound::fundamentalDomainByPredicate(), getArtist(), csound::Score::getDuration(), getFomusfileFilepath(), getTitle(), csound::iterator(), csound::Conversions::round(), score, and csound::Score::sort().

Referenced by csound::MusicModel::processArgs().

6.24.3.63 write()

```
void csound::Composition::write (
            const char * text )  [virtual]
```

Write as if to stderr.

References csound::fundamentalDomainByPredicate(), and csound::System::message().

6.24.4 Field Documentation

6.24.4.1 album

```
std::string csound::Composition::album  [protected]
```

Optional metadata.

Referenced by generateAllNames(), getAlbum(), setAlbum(), and translateToMp4().

6.24.4.2 artist

```
std::string csound::Composition::artist  [protected]
```

Required metadata.

Allows for performer, etc. to differ from author. Defaults to author.

Referenced by generateAllNames(), getArtist(), setArtist(), and translateToMp4().

6.24.4.3 author

```
std::string csound::Composition::author  [protected]
```

Required metadata.

Referenced by generateAllNames(), getAuthor(), setAuthor(), translateToMp3(), and translateToMp4().

6.24.4.4 base_filepath

```
std::string csound::Composition::base_filepath  [protected]
```

Generated.

The dirname and stem of the output files.

Referenced by generateAllNames(), getBasename(), getFileFilepath(), and getMusicXmlfileFilepath().

6.24.4.5 baseScore

```
Score csound::Composition::baseScore  [protected]
```

6.24.4.6 bext_description

```
std::string csound::Composition::bext_description  [protected]
```

Generated.

Referenced by generateAllNames().

6.24.4.7 bext_orig_ref

```
std::string csound::Composition::bext_orig_ref  [protected]
```

Generated.

Referenced by generateAllNames().

6.24.4.8 bext_originator

```
std::string csound::Composition::bext_originator  [protected]
```

Generated.

Referenced by generateAllNames().

6.24.4.9 cd_quality_filepath

```
std::string csound::Composition::cd_quality_filepath  [protected]
```

Generated.

Referenced by generateAllNames(), getCdSoundfileFilepath(), and translateToMp4().

6.24.4.10 conformPitches

```
bool csound::Composition::conformPitches  [protected]
```

Referenced by csound::MusicModel::createCsoundScore(), getConformPitches(), and setConformPitches().

6.24.4.11 copyright

```
std::string csound::Composition::copyright  [protected]
```

Required metadata.

Referenced by generateAllNames(), getCopyright(), setCopyright(), and translateToMp4().

6.24.4.12 duration

```
double csound::Composition::duration  [protected]
```

Referenced by csound::MusicModel::generate(), csound::ScoreModel::generate(), getDuration(), setDuration(), and translateToNotation().

6.24.4.13 flac_filepath

```
std::string csound::Composition::flac_filepath  [protected]
```

Generated.

Referenced by generateAllNames().

6.24.4.14 label

```
std::string csound::Composition::label  [protected]
```

Generated.

Referenced by generateAllNames().

6.24.4.15 license

```
std::string csound::Composition::license  [protected]
```

Required metadata.

Defaults to Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International.

Referenced by getLicense(), and setLicense().

6.24.4.16 master_filepath

```
std::string csound::Composition::master_filepath  [protected]
```

Generated.

Referenced by generateAllNames(), getOutputSoundfileFilepath(), and translateToMp4().

6.24.4.17 midi_filepath

```
std::string csound::Composition::midi_filepath  [protected]
```

Generated.

Referenced by generateAllNames(), and getMidifileFilepath().

6.24.4.18 mp3_filepath

```
std::string csound::Composition::mp3_filepath  [protected]
```

Generated.

Referenced by generateAllNames(), and getMp3SoundfileFilepath().

6.24.4.19 mp4_filepath

```
std::string csound::Composition::mp4_filepath  [protected]
```

Generated.

Referenced by generateAllNames(), and translateToMp4().

6.24.4.20 normalized_master_filepath

```
std::string csound::Composition::normalized_master_filepath  [protected]
```

Generated.

Referenced by generateAllNames(), and getNormalizedSoundfileFilepath().

6.24.4.21 notes

```
std::string csound::Composition::notes  [protected]
```

Optional metadata, defaults to "Electroacoustic Music.".

Referenced by generateAllNames(), and translateToMp4().

6.24.4.22 output_directory

```
std::string csound::Composition::output_directory  [protected]
```

Required.

The target directory of the output files. Defaults to the current working directory.

Referenced by generateAllNames(), getOutputDirectory(), and setOutputDirectory().

6.24.4.23 output_filename

```
std::string csound::Composition::output_filename  [protected]
```

Referenced by clearOutputSoundfileName(), getOutputSoundfileFilepath(), and setOutputSoundfileName().

6.24.4.24 performance_rights_organization

```
std::string csound::Composition::performance_rights_organization  [protected]
```

Optional metadata.

Referenced by generateAllNames(), getPerformanceRightsOrganization(), setPerformanceRightsOrganization(), and translateToMp4().

6.24.4.25 score

```
Score& csound::Composition::score  [protected]
```

Referenced by csound::MusicModel::arrange(), csound::MusicModel::arrange(), csound::MusicModel::arrange(), clear(), csound::MusicModel::createCsoundScore(), csound::MusicModel::generate(), csound::ScoreModel::generate(), getScore(), csound::MusicModel::removeArrangement(), setScore(), and translateToNotation().

6.24.4.26 spectrogram_filepath

```
std::string csound::Composition::spectrogram_filepath  [protected]
```

Generated.

Referenced by generateAllNames(), and translateToMp4().

6.24.4.27 stem

```
std::string csound::Composition::stem  [protected]
```

Required.

The stem must be a valid filename and also represents the title. All other names, text, and commands are generated from directory, stem, filename extensions, and required metadata.

Referenced by generateAllNames(), getFilename(), getTitle(), setFilename(), setTitle(), and translateToMp4().

6.24.4.28 tieOverlappingNotes

```
bool csound::Composition::tieOverlappingNotes  [protected]
```

Referenced by getTieOverlappingNotes(), and setTieOverlappingNotes().

6.24.4.29 timestamp

```
std::string csound::Composition::timestamp  [protected]
```

Generated.

Referenced by generateAllNames(), and getTimestamp().

6.24.4.30 tonesPerOctave

```
double csound::Composition::tonesPerOctave  [protected]
```

Referenced by csound::MusicModel::createCsoundScore(), getTonesPerOctave(), and setTonesPerOctave().

6.24.4.31 track

```
std::string csound::Composition::track  [protected]
```

Optional metadata.

Referenced by generateAllNames(), and translateToMp4().

6.24.4.32 year

```
std::string csound::Composition::year  [protected]
```

Required metadata.

Referenced by generateAllNames(), getYear(), setYear(), translateToMp3(), and translateToMp4().

6.25 csound::Conversions Class Reference

Conversions to and from various music and signal processing units.

```
#include <Conversions.hpp>
```

Static Public Member Functions

- static double amplitudeToDecibels (double amplitude)
- static double amplitudeToGain (double Amplitude)
- static double amplitudeToMidi (double Amplitude)
- static std::string boolToString (bool value)
- static double decibelsToAmplitude (double decibels)
- static double decibelsToMidi (double decibels)
- static std::string doubleToString (double value)
- static char * dupstr (const char *string)

  Return a new copy of a "C" string allocated on the heap.
- static double findClosestPitchClass (double M, double pitchClass, double tones=12.0)

  Given the pitch-class set number M = sum over pitch-classes of (2 ^ pitch-class), return the pitch-class in the set that is closest to the argumen pitch-class.
- static double gainToAmplitude (double Gain)
- static double gainToDb (double inputDb, double gain, bool odbfs=false)

  Return a new value in dB that represents the input value in dB adjusted by the specified gain.
- static double get2PI ()
- static double getMaximumAmplitude (int size)

  Returns the maximum soundfile amplitude for the sample size, assuming either float or twos' complement integer samples.
- static double getMaximumDynamicRange ()
- static double getMiddleCHz ()
- static double getNORM_7 ()
- static double getPI ()
- static int getSampleSize ()

  Returns the maximum soundfile amplitude for the sample size.
- static double hzToMidi (double Hz, bool rounded)
- static double hzToOctave (double Hz)
- static double hzToSamplingIncrement (double Hz, double SR, double SamplesPerCycle)
- static bool initialize ()
- static std::string intToString (int value)
- static double leftPan (double x)
- static double midiToAmplitude (double Midi)
- static double midiToDecibels (double Midi)
- static double midiToGain (double Midi)
- static double midiToHz (double Midi)
- static double midiToOctave (double Midi)
- static double midiToPitchClass (double midiKey)
- static double midiToPitchClassSet (double midiKey)
- static double midiToRoundedOctave (double midiKey)
- static double midiToSamplingIncrement (double Midi, double SR, double SamplesPerCycle)
- static double modulus (double a, double b)

  True modulus accounting for sign.
- static std::string mToName (double pitchClassSet)

  Return the jazz-style scale or chord name for the pitch-class set number M = sum over pitch-classes of (2 ^ pitch-class) These numbers form a multiplicative monoid for all pitch-class sets in a system of equal temperament.
- static double nameToM (std::string name)

  Return the pitch-class set number M = sum over pitch-classes of (2 ^ pitch-class) for the jazz-style scale or chord name.
- static std::vector< double > nameToPitches (std::string name)

  Return the pitches for a chord name.

- static double octaveToHz (double Octave)
- static double octaveToMidi (double Octave, bool rounded)
- static double octaveToSamplingIncrement (double Octave, double SR, double SamplesPerCycle)
- static double phaseToTableLengths (double Phase, double TableSampleCount)
- static double pitchClassSetToMidi (double pitchClassSet)
- static double pitchClassToMidi (double pitchClass)
- static double rightPan (double x)
- static double round (double value)
- static bool stringToBool (std::string value, bool default_=false)

  Translate the string value to a boolean value, returning the default if the string value is empty.
- static double stringToDouble (std::string value, double default_=0.0)

  Translate the string value to a double-precision value, returning the default if the string value is empty.
- static int stringToInt (std::string value, int default_=0)

  Translate the string value to an integer value, returning the default if the string value is empty.
- static void stringToVector (const std::string &text, std::vector< double > &vector)

  Parses text in the format "n,..,n" to a vector of doubles.
- static int swapInt (int Source)
- static short swapShort (short Source)
- static double temper (double octave, double tonesPerOctave)
- static std::string & trim (std::string &value)
- static std::string & trimQuotes (std::string &value)

6.25.1 Detailed Description

Conversions to and from various music and signal processing units.

Note that: silence::Event represents loudness in MIDI units (0 to 127). silence::Orchestra represents loudness in gain (0 to 1). silence::WaveSoundfileOut represents loudness in amplitude (0 to 1 for float samples, 0 to 32767 for short samples). Loudness can also be represented in positive decibels (0 to 84 for short samples, 0 to whatever for float samples). For float samples, decibels are assumed to be equivalent to MIDI velocity; otherwise, MIDI velocity is rescaled according to the maximum dynamic range supported by the sample size. All loudness conversions are driven by sample word size, which must be set before use; the default is 4 (float samples).

6.25.2 Member Function Documentation

6.25.2.1 amplitudeToDecibels()

```
double csound::Conversions::amplitudeToDecibels (
            double amplitude )  [static]
```

References csound::fundamentalDomainByPredicate().

Referenced by amplitudeToMidi().

6.25.2.2 amplitudeToGain()

```
double csound::Conversions::amplitudeToGain (
            double Amplitude )  [static]
```

References csound::fundamentalDomainByPredicate(), and getMaximumAmplitude().

6.25.2.3 amplitudeToMidi()

```
double csound::Conversions::amplitudeToMidi (
            double Amplitude )  [static]
```

References amplitudeToDecibels(), decibelsToMidi(), and csound::fundamentalDomainByPredicate().

Referenced by csound::Event::setAmplitude().

6.25.2.4 boolToString()

```
std::string csound::Conversions::boolToString (
            bool value )  [static]
```

6.25.2.5 decibelsToAmplitude()

```
double csound::Conversions::decibelsToAmplitude (
            double decibels )  [static]
```

References csound::fundamentalDomainByPredicate().

Referenced by midiToAmplitude().

6.25.2.6 decibelsToMidi()

```
double csound::Conversions::decibelsToMidi (
            double decibels )  [static]
```

References getMaximumDynamicRange().

Referenced by amplitudeToMidi().

6.25.2.7 doubleToString()

```
std::string csound::Conversions::doubleToString (
            double value )  [static]
```

References csound::fundamentalDomainByPredicate().

6.25.2.8 dupstr()

```
char * csound::Conversions::dupstr (
            const char * string )  [static]
```

Return a new copy of a "C" string allocated on the heap.

The user is responsible for freeing the copy.

References csound::fundamentalDomainByPredicate().

6.25.2.9 findClosestPitchClass()

```
double csound::Conversions::findClosestPitchClass (
            double M,
            double pitchClass,
            double tones = 12.0 )  [static]
```

Given the pitch-class set number M = sum over pitch-classes of (2 ^ pitch-class), return the pitch-class in the set that is closest to the argumen pitch-class.

References csound::fundamentalDomainByPredicate(), midiToPitchClass(), midiToPitchClassSet(), and round().

Referenced by csound::Event::conformToPitchClassSet().

6.25.2.10 gainToAmplitude()

```
double csound::Conversions::gainToAmplitude (
            double Gain )  [static]
```

References csound::fundamentalDomainByPredicate(), and getMaximumAmplitude().

6.25.2.11 gainToDb()

```
double csound::Conversions::gainToDb (
            double inputDb,
            double gain,
            bool odbfs = false )  [static]
```

Return a new value in dB that represents the input value in dB adjusted by the specified gain.

If odbfs is false (the default), then 0 dB is the threshold of hearing; otherwise, 0 dB is full scale.

References csound::fundamentalDomainByPredicate().

6.25.2.12 get2PI()

```
double csound::Conversions::get2PI ( )  [static]
```

Referenced by csound::Soundfile::cosineGrain(), csound::Soundfile::jonesParksGrain(), and csound::StrangeAttractor::specialFunctions().

6.25.2.13 getMaximumAmplitude()

```
double csound::Conversions::getMaximumAmplitude (
            int size )  [static]
```

Returns the maximum soundfile amplitude for the sample size, assuming either float or twos' complement integer samples.

References csound::fundamentalDomainByPredicate().

Referenced by amplitudeToGain(), gainToAmplitude(), and getMaximumDynamicRange().

6.25.2.14 getMaximumDynamicRange()

```
double csound::Conversions::getMaximumDynamicRange ( )  [static]
```

References csound::fundamentalDomainByPredicate(), and getMaximumAmplitude().

Referenced by decibelsToMidi(), and midiToDecibels().

6.25.2.15 getMiddleCHz()

```
double csound::Conversions::getMiddleCHz ( )  [static]
```

6.25.2.16 getNORM_7()

```
double csound::Conversions::getNORM_7 ( )  [static]
```

6.25.2.17 getPI()

```
double csound::Conversions::getPI ( )  [static]
```

6.25.2.18 getSampleSize()

```
int csound::Conversions::getSampleSize ( )  [static]
```

Returns the maximum soundfile amplitude for the sample size.

6.25.2.19 hzToMidi()

```
double csound::Conversions::hzToMidi (
            double Hz,
            bool rounded )  [static]
```

References csound::fundamentalDomainByPredicate(), hzToOctave(), and octaveToMidi().

Referenced by csound::Event::setFrequency().

6.25.2.20 hzToOctave()

```
double csound::Conversions::hzToOctave (
            double Hz )  [static]
```

References csound::fundamentalDomainByPredicate().

Referenced by hzToMidi().

6.25.2.21 hzToSamplingIncrement()

```
double csound::Conversions::hzToSamplingIncrement (
            double Hz,
            double SR,
            double SamplesPerCycle )  [static]
```

References csound::fundamentalDomainByPredicate().

Referenced by midiToSamplingIncrement(), and octaveToSamplingIncrement().

6.25.2.22 initialize()

```
bool csound::Conversions::initialize ( )  [static]
```

References csound::fundamentalDomainByPredicate(), and csound::iterator().

6.25.2.23 intToString()

```
std::string csound::Conversions::intToString (
            int value )  [static]
```

References csound::fundamentalDomainByPredicate().

6.25.2.24 leftPan()

```
double csound::Conversions::leftPan (
            double x )  [static]
```

References csound::fundamentalDomainByPredicate().

Referenced by csound::Soundfile::cosineGrain(), csound::Event::getLeftGain(), and csound::Soundfile::jonesParksGrain().

6.25.2.25 midiToAmplitude()

```
double csound::Conversions::midiToAmplitude (
            double Midi )  [static]
```

References decibelsToAmplitude(), csound::fundamentalDomainByPredicate(), and midiToDecibels().

Referenced by csound::Event::getAmplitude().

6.25.2.26 midiToDecibels()

```
double csound::Conversions::midiToDecibels (
            double Midi )  [static]
```

References csound::fundamentalDomainByPredicate(), and getMaximumDynamicRange().

Referenced by midiToAmplitude().

6.25.2.27 midiToGain()

```
double csound::Conversions::midiToGain (
            double Midi )  [static]
```

References csound::fundamentalDomainByPredicate().

Referenced by csound::Event::getGain().

6.25.2.28 midiToHz()

```
double csound::Conversions::midiToHz (
            double Midi )  [static]
```

References csound::fundamentalDomainByPredicate(), midiToOctave(), and octaveToHz().

Referenced by csound::Event::getFrequency(), and midiToSamplingIncrement().

6.25.2.29 midiToOctave()

```
double csound::Conversions::midiToOctave (
            double Midi )  [static]
```

References csound::fundamentalDomainByPredicate().

Referenced by csound::Event::getKey_tempered(), midiToHz(), midiToRoundedOctave(), csound::Event::temper(), csound::Event::toBlueIStatement(), csound::Event::toCsoundIStatementHeld(), and csound::Event::toCsoundIStatementRelease().

6.25.2.30 midiToPitchClass()

```
double csound::Conversions::midiToPitchClass (
            double midiKey )  [static]
```

References csound::fundamentalDomainByPredicate(), and round().

Referenced by csound::Event::conformToPitchClassSet(), findClosestPitchClass(), midiToPitchClassSet(), and csound::CounterpointNode::transform().

6.25.2.31 midiToPitchClassSet()

```
double csound::Conversions::midiToPitchClassSet (
            double midiKey )  [static]
```

References csound::fundamentalDomainByPredicate(), and midiToPitchClass().

Referenced by findClosestPitchClass().

6.25.2.32 midiToRoundedOctave()

```
double csound::Conversions::midiToRoundedOctave (
            double midiKey )  [static]
```

References csound::fundamentalDomainByPredicate(), midiToOctave(), and round().

Referenced by csound::Event::conformToPitchClassSet().

6.25.2.33 midiToSamplingIncrement()

```
double csound::Conversions::midiToSamplingIncrement (
            double Midi,
            double SR,
            double SamplesPerCycle )  [static]
```

References csound::fundamentalDomainByPredicate(), hzToSamplingIncrement(), and midiToHz().

6.25.2.34 modulus()

```
double csound::Conversions::modulus (
            double a,
            double b )  [static]
```

True modulus accounting for sign.

Referenced by csound::ChordLindenmayer::equivalence(), and csound::Lindenmayer::interpret().

6.25.2.35 mToName()

```
std::string csound::Conversions::mToName (
            double pitchClassSet )  [static]
```

Return the jazz-style scale or chord name for the pitch-class set number M = sum over pitch-classes of (2 ^ pitch-class) These numbers form a multiplicative monoid for all pitch-class sets in a system of equal temperament.

6.25.2.36 nameToM()

```
double csound::Conversions::nameToM (
            std::string name )  [static]
```

Return the pitch-class set number M = sum over pitch-classes of (2 ^ pitch-class) for the jazz-style scale or chord name.

These numbers form a multiplicative monoid for all pitch-class sets in a system of equal temperament.

Referenced by csound::Voicelead::nameToC(), and nameToPitches().

6.25.2.37 nameToPitches()

```
std::vector< double > csound::Conversions::nameToPitches (
            std::string name )  [static]
```

Return the pitches for a chord name.

References csound::fundamentalDomainByPredicate(), and nameToM().

6.25.2.38 octaveToHz()

```
double csound::Conversions::octaveToHz (
            double Octave )  [static]
```

References csound::fundamentalDomainByPredicate().

Referenced by midiToHz(), and octaveToSamplingIncrement().

6.25.2.39 octaveToMidi()

```
double csound::Conversions::octaveToMidi (
            double Octave,
            bool rounded )  [static]
```

References csound::fundamentalDomainByPredicate(), and round().

Referenced by csound::Event::conformToPitchClassSet(), csound::Event::getKey_tempered(), hzToMidi(), csound::StrangeAttractor::rende
csound::Event::temper(), csound::Event::toCsoundIStatementHeld(), and csound::Event::toCsoundIStatementRelease().

6.25.2.40 octaveToSamplingIncrement()

```
double csound::Conversions::octaveToSamplingIncrement (
            double Octave,
            double SR,
            double SamplesPerCycle )  [static]
```

References csound::fundamentalDomainByPredicate(), hzToSamplingIncrement(), and octaveToHz().

6.25.2.41 phaseToTableLengths()

```
double csound::Conversions::phaseToTableLengths (
            double Phase,
            double TableSampleCount )  [static]
```

References csound::fundamentalDomainByPredicate().

6.25.2.42 pitchClassSetToMidi()

```
double csound::Conversions::pitchClassSetToMidi (
            double pitchClassSet )  [static]
```

References csound::fundamentalDomainByPredicate().

6.25.2.43 pitchClassToMidi()

```
double csound::Conversions::pitchClassToMidi (
            double pitchClass )  [static]
```

References csound::fundamentalDomainByPredicate(), and round().

Referenced by csound::Event::conformToPitchClassSet().

6.25.2.44 rightPan()

```
double csound::Conversions::rightPan (
            double x )  [static]
```

References csound::fundamentalDomainByPredicate().

Referenced by csound::Soundfile::cosineGrain(), csound::Event::getRightGain(), and csound::Soundfile::jonesParksGrain().

6.25.2.45 round()

```
double csound::Conversions::round (
            double value )  [static]
```

Referenced by csound::Event::conformToPitchClassSet(), csound::Soundfile::cosineGrain(), findClosestPitchClass(), csound::Event::getChannel(), csound::Event::getKey_tempered(), csound::Event::getKeyNumber(), csound::Event::getStatusNumber(), csound::Event::getVelocityNumber(), csound::Event::isMatchingEvent(), csound::Event::isMatchingNoteOff(), csound::Event::isNoteOff(), csound::Event::isNoteOn(), midiToPitchClass(), midiToRoundedOctave(), octaveToMidi(), pitchClassToMidi(), temper(), csound::Event::toCsoundIStatementHeld(), csound::Event::toCsoundIStatementRelease(), and csound::Composition::translateToNotation

6.25.2.46 stringToBool()

```
bool csound::Conversions::stringToBool (
            std::string value,
            bool default_ = false )  [static]
```

Translate the string value to a boolean value, returning the default if the string value is empty.

References csound::fundamentalDomainByPredicate().

6.25.2.47 stringToDouble()

```
double csound::Conversions::stringToDouble (
            std::string value,
            double default_ = 0.0 )  [static]
```

Translate the string value to a double-precision value, returning the default if the string value is empty.

References csound::fundamentalDomainByPredicate().

Referenced by csound::Lindenmayer::interpret().

6.25.2.48 stringToInt()

```
int csound::Conversions::stringToInt (
            std::string value,
            int default_ = 0 )  [static]
```

Translate the string value to an integer value, returning the default if the string value is empty.

References csound::fundamentalDomainByPredicate().

6.25.2.49 stringToVector()

```
void csound::Conversions::stringToVector (
            const std::string & text,
            std::vector< double > & vector )  [static]
```

Parses text in the format "n,..,n" to a vector of doubles.

References csound::fundamentalDomainByPredicate().

6.25.2.50 swapInt()

```
int csound::Conversions::swapInt (
            int Source )  [static]
```

References csound::fundamentalDomainByPredicate().

6.25.2.51 swapShort()

```
short csound::Conversions::swapShort (
            short Source )  [static]
```

References csound::fundamentalDomainByPredicate().

6.25.2.52 temper()

```
double csound::Conversions::temper (
            double octave,
            double tonesPerOctave )  [static]
```

References round().

Referenced by csound::Event::temper(), csound::Event::toCsoundIStatementHeld(), and csound::Event::toCsoundIStatementRelease().

6.25.2.53 trim()

```
std::string & csound::Conversions::trim (
            std::string & value )  [static]
```

Referenced by csound::Lindenmayer::interpret().

6.25.2.54 trimQuotes()

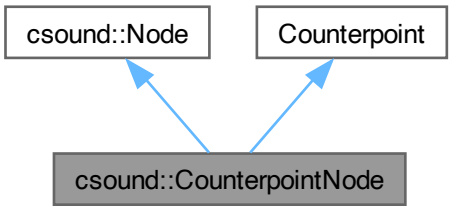
```
std::string & csound::Conversions::trimQuotes (
            std::string & value )  [static]
```

6.26 csound::CounterpointNode Class Reference

Uses Bill Schottstaedt's species counterpoint generator code to either (a) generate a counterpoint in species 1, 2, or 3 for a cantus firmus selected from notes generated by child nodes, or (b) attempt to correct the voice leading for species 1, 2, or 3 counterpoint in notes generated by child nodes.

`#include <CounterpointNode.hpp>`

Inheritance diagram for csound::CounterpointNode:



Public Types

- enum { MostNotes_ = 128 , MostVoices_ = 12 }
- enum { GenerateCounterpoint = 0 , CorrectCounterpoint = 1 }
- enum {
  Unison = 0 , MinorSecond = 1 , MajorSecond = 2 , MinorThird = 3 ,
  MajorThird = 4 , Fourth = 5 , Tritone = 6 , Fifth = 7 ,
  MinorSixth = 8 , MajorSixth = 9 , MinorSeventh = 10 , MajorSeventh = 11 ,
  Octave = 12 }
- enum {
  Aeolian = 1 , Dorian = 2 , Phrygian = 3 , Lydian = 4 ,
  Mixolydian = 5 , Ionian = 6 , Locrian = 7 }
- enum { DirectMotion = 1 , ContraryMotion = 2 , ObliqueMotion = 3 , NoMotion = 4 }
- enum {
  WholeNote = 8 , HalfNote = 4 , DottedHalfNote = 6 , QuarterNote = 2 ,
  DottedQuarterNote = 3 , EighthNote = 1 }
- enum {
  One = 0 , Two = 2 , Three = 3 , Four = 4 ,
  Five = 5 , Six = 6 , Eight = 8 }
- enum { infinity = 1000000 , Bad = 100 , RealBad = 200 }
- enum { INTERVALS_WITH_BASS_SIZE = 8 }
- enum { NumFields = 16 , Field = (MostVoices_+1) , EndF = (Field∗NumFields) }

Public Member Functions

- int ABS (int i)
- virtual void addChild (Node *node)

  Adds an immediate child Node to this.
- void AddInterval (int n)
- int ADissonance (int Interval, int Cn, int Cp, int v, int Species)
- int AnOctave (int Interval)
- void AnySpecies (int OurMode, int *StartPitches, int CurV, int CantusFirmusLength, int Species)
- void ARRBLT (int *dest, int *source, int num)
- int ASeventh (int Interval)
- int ASkip (int Interval)
- int AStep (int Interval)
- int ATenth (int Interval)
- int AThird (int Interval)
- int BadMelody (int Intv)
- int Bass (int Cn, int v)
- int Beat8 (int n)
- void BestFitFirst (int CurTime, int CurrentPenalty, int NumParts, int Species, int BrLim)
- int Cantus (int n, int v)
- int Check (int Cn, int Cp, int v, int NumParts, int Species, int CurLim)
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- void CleanRhy ()
- virtual void clear ()
- virtual void clear ()

  Recursively clears all child Nodes of this.
- int ConsecutiveSkipsInSameDirection (int Pitch1, int Pitch2, int Pitch3)
- void counterpoint (int OurMode, int *StartPitches, int CurV, int CantusFirmusLength, int Species, int *cantus)
- CounterpointNode ()
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- int CurRhy (int n)
- int DirectMotionToPerfectConsonance (int Pitch1, int Pitch2, int Pitch3, int Pitch4)
- int Doubled (int Pitch, int Cn, int v)
- int DownBeat (int n, int v)
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- int ExtremeRange (int Pitch)
- void fillCantus (int c0, int c1, int c2, int c3, int c4, int c5, int c6, int c7, int c8, int c9, int c10, int c11, int c12, int c13, int c14)
- void FillRhyPat ()
- int FirstNote (int n, int v)
- virtual void generate (Score &score_from_this)

  Optionally generate notes into the score.
- virtual Node * getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual int getGenerationMode () const
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.

- virtual int getMusicMode () const
- virtual double getSecondsPerPulse () const
- virtual int getSpecies () const
- virtual std::vector< int > & getVoiceBeginnings ()
- virtual size_t getVoices () const
- int GoodRhy ()
- virtual void initialize (int mostnotes, int mostvoices)
- int InMode (int Pitch, int Mode)
- int LastNote (int n, int v)
- int Look (int CurPen, int CurVoice, int NumParts, int Species, int Lim, int *Pens, int *Is, int *CurNotes)
- int MAX (int a, int b)
- void message (const char *format, va_list valist)
- void message (const char *format,...)
- int MIN (int a, int b)
- int MotionType (int Pitch1, int Pitch2, int Pitch3, int Pitch4)
- int NextToLastNote (int n, int v)
- int Other (int Cn, int v, int v1)
- int OtherVoiceCheck (int Cn, int Cp, int v, int NumParts, int Species, int CurLim)
- int OutOfRange (int Pitch)
- int PitchRepeats (int Cn, int Cp, int v)
- float RANDOM (float amp)
- int SaveIndx (int indx, int *Sp)
- void SaveResults (int CurrentPenalty, int Penalty, int v1, int Species)
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void setGenerationMode (int value)
- virtual void setMusicMode (int value)
- virtual void setSecondsPerPulse (double value)
- virtual void setSpecies (int value)
- void SetUs (int n, int p, int v)
- virtual void setVoiceBeginnings (const std::vector< int > &value)
- virtual void setVoices (size_t value)
- int Size (int MelInt)
- int SpecialSpeciesCheck (int Cn, int Cp, int v, int Other0, int Other1, int Other2, int NumParts, int Species, int MelInt, int Interval, int ActInt, int LastIntClass, int Pitch, int LastMelInt, int CurLim)
- void toCsoundScore (std::string filename, double secondsPerPulse)
- int TooMuchOfInterval (int Cn, int Cp, int v)
- int TotalRange (int Cn, int Cp, int v)
- virtual void transform (Score &score)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.
- int UpBeat (int n, int v)
- int Us (int n, int v)
- void UsedRhy (int n)
- int VIndex (int Time, int VNum)
- void winners (int v1, int *data, int *best, int *best1, int *best2, int *durs)
- virtual ~CounterpointNode ()

Data Fields

- int AllDone
- int AllVoicesSkipPenalty
- int AscendingSixthPenalty
- int AugmentedIntervalPenalty
- int BadCadencePenalty
- int BadMelodyPenalty
- int BasePitch
- Eigen::MatrixXi BestFit
- Eigen::MatrixXi BestFit1
- Eigen::MatrixXi BestFit2
- int BestFitPenalty
- int Branches
- std::vector< Node * > children

  Child Nodes, if any.
- int CompoundPenalty
- int CrossAboveCantusPenalty
- int CrossBelowBassPenalty
- Eigen::MatrixXi Ctrpt
- int DirectMotionPenalty
- int DirectPerfectOnDownbeatPenalty
- int DirectToFifthPenalty
- int DirectToOctavePenalty
- int DirectToTritonePenalty
- int DissonanceNotFillingThirdPenalty
- int DissonancePenalty
- int DoubledFifthPenalty
- int DoubledLeadingTonePenalty
- int DoubledSixthPenalty
- int DownBeatUnisonPenalty
- Eigen::MatrixXi Dur
- int EighthJumpPenalty
- int EndOnPerfectPenalty
- int ExtremeRangePenalty
- int FifthFollowedBySameDirectionPenalty
- int FifthPrecededBySameDirectionPenalty
- int Fits [3]
- int FourRepeatedNotesPenalty
- int generationMode
- int HalfUntiedPenalty
- int HighestSemitone
- int InnerVoicesInDirectToPerfectPenalty
- int InnerVoicesInDirectToTritonePenalty
- int IntervalsWithBass [INTERVALS_WITH_BASS_SIZE]
- int LeapAtCadencePenalty
- int LesserLigaturePenalty
- int LowerNeighborPenalty
- int LowestSemitone
- int LydianCadentialTritonePenalty
- int MaxPenalty

- int MelodicBoredomPenalty
- int MelodicTritonePenalty
- int Mode
- int MostNotes
- int MostVoices
- int musicMode
- int NoLeadingTonePenalty
- int NoMotionAgainstOctavePenalty
- int NotaCambiataPenalty
- int NotaLigaturePenalty
- int NotBestCadencePenalty
- int NotContraryToOthersPenalty
- int NoTimeForaLigaturePenalty
- int NotTriadPenalty
- int OctaveLeapPenalty
- Eigen::MatrixXi Onset
- int OutOfModePenalty
- int OutOfRangePenalty
- int OverOctavePenalty
- int OverTwelfthPenalty
- int ParallelFifthPenalty
- int ParallelUnisonPenalty
- float PenaltyRatio
- int PerfectConsonancePenalty
- long randx
- int RepeatedPitchPenalty
- int RepetitionOnUpbeatPenalty
- Eigen::VectorXi RhyNotes
- Eigen::MatrixXi RhyPat
- double secondsPerPulse
- int SixFiveChordPenalty
- int SixthFollowedBySameDirectionPenalty
- int SixthLeapPenalty
- int SixthPrecededBySameDirectionPenalty
- int SkipFollowedBySameDirectionPenalty
- int SkipFromUnisonPenalty
- int SkipPrecededBySameDirectionPenalty
- int SkipTo8vePenalty
- int SkipToDownBeatPenalty
- int species
- int TenthToOctavePenalty
- int ThirdDoubledPenalty
- int ThreeRepeatedNotesPenalty
- int ThreeSkipsPenalty
- Eigen::VectorXi TotalNotes
- int TotalTime
- int TripledBassPenalty
- int TwoRepeatedNotesPenalty
- int TwoSkipsNotInTriadPenalty
- int TwoSkipsPenalty
- std::normal_distribution uniform_real_generator

- int UnisonDownbeatPenalty
- int UnisonOnBeat4Penalty
- int UnisonPenalty
- int UnisonUpbeatPenalty
- int UnpreparedSixFivePenalty
- int UnresolvedLeadingTonePenalty
- int UnresolvedLigaturePenalty
- int UnresolvedSixFivePenalty
- int UpperNeighborPenalty
- int UpperVoicesTooFarApartPenalty
- Eigen::VectorXi vbs
- int VerticalTritonePenalty
- std::vector< int > voiceBeginnings
- size_t voices

Static Public Attributes

- static int _Aeolian [12] = {1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0}
- static int _Dorian [12] = {1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0}
- static int _Ionian [12] = {1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1}
- static int _Locrian [12] = {1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0}
- static int _Lydian [12] = {1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1}
- static int _Mixolydian [12] = {1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0}
- static int _Phrygian [12] = {1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0}
- static int BadMelodyInterval [13] = {0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0}
- static int Dissonance [13] = {0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0}
- static int ImperfectConsonance [13] = {0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0}
- static int Indx [17] = {0, 1, -1, 2, -2, 3, -3, 0, 4, -4, 5, 7, -5, 8, 12, -7,-12}
- static std::mt19937 mersenneTwister
- static int PerfectConsonance [13] = {1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1}

Protected Attributes

- Eigen::MatrixXd localCoordinates

6.26.1 Detailed Description

Uses Bill Schottstaedt's species counterpoint generator code to either (a) generate a counterpoint in species 1, 2, or 3 for a cantus firmus selected from notes generated by child nodes, or (b) attempt to correct the voice leading for species 1, 2, or 3 counterpoint in notes generated by child nodes.

6.26.2 Member Enumeration Documentation

6.26.2.1 anonymous enum

```
anonymous enum  [inherited]
```

Enumerator

Most↩ Notes_	
Most↩ Voices_	

6.26.2.2 anonymous enum

```
anonymous enum
```

Enumerator

GenerateCounterpoint	
CorrectCounterpoint	

6.26.2.3 anonymous enum

```
anonymous enum [inherited]
```

Enumerator

Unison	
MinorSecond	
MajorSecond	
MinorThird	
MajorThird	
Fourth	
Tritone	
Fifth	
MinorSixth	
MajorSixth	
MinorSeventh	
MajorSeventh	
Octave	

6.26.2.4 anonymous enum

```
anonymous enum [inherited]
```

Enumerator

Aeolian	

Enumerator

Dorian	
Phrygian	
Lydian	
Mixolydian	
Ionian	
Locrian	

6.26.2.5 anonymous enum

`anonymous enum [inherited]`

Enumerator

DirectMotion	
ContraryMotion	
ObliqueMotion	
NoMotion	

6.26.2.6 anonymous enum

`anonymous enum [inherited]`

Enumerator

WholeNote	
HalfNote	
DottedHalfNote	
QuarterNote	
DottedQuarterNote	
EighthNote	

6.26.2.7 anonymous enum

`anonymous enum [inherited]`

Enumerator

One	
Two	
Three	
Four	

Enumerator

Five	
Six	
Eight	

6.26.2.8 anonymous enum

```
anonymous enum  [inherited]
```

Enumerator

infinity	
Bad	
RealBad	

6.26.2.9 anonymous enum

```
anonymous enum  [inherited]
```

Enumerator

INTERVALS_WITH_BASS_SIZE	

6.26.2.10 anonymous enum

```
anonymous enum  [inherited]
```

Enumerator

NumFields	
Field	
EndF	

6.26.3 Constructor & Destructor Documentation

6.26.3.1 CounterpointNode()

```
csound::CounterpointNode::CounterpointNode ( )
```

References Counterpoint::FillRhyPat().

6.26.3.2 ~CounterpointNode()

```
csound::CounterpointNode::~CounterpointNode ( )  [virtual]
```

6.26.4 Member Function Documentation

6.26.4.1 ABS()

```
int Counterpoint::ABS (
            int i )  [inherited]
```

Referenced by Counterpoint::AnOctave(), Counterpoint::ASkip(), Counterpoint::AStep(), Counterpoint::ATenth(), Counterpoint::BadMelody(), Counterpoint::Check(), Counterpoint::DirectMotionToPerfectConsonance(), Counterpoint::OtherVoiceCheck() Counterpoint::SaveResults(), Counterpoint::Size(), and Counterpoint::SpecialSpeciesCheck().

6.26.4.2 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.26.4.3 AddInterval()

```
void Counterpoint::AddInterval (
            int n )  [inherited]
```

References Counterpoint::IntervalsWithBass.

Referenced by Counterpoint::OtherVoiceCheck().

6.26.4.4 ADissonance()

```
int Counterpoint::ADissonance (
            int Interval,
            int Cn,
            int Cp,
            int v,
            int Species )  [inherited]
```

References Counterpoint::AStep(), Counterpoint::Beat8(), Counterpoint::Dissonance, Counterpoint::DownBeat(), Counterpoint::Dur, Counterpoint::FirstNote(), Counterpoint::LastNote(), Counterpoint::Onset, Counterpoint::UpBeat(), Counterpoint::Us(), and Counterpoint::WholeNote.

Referenced by Counterpoint::Check().

6.26.4.5 AnOctave()

```
int Counterpoint::AnOctave (
            int Interval )  [inherited]
```

References Counterpoint::ABS(), and Counterpoint::Unison.

Referenced by Counterpoint::Check().

6.26.4.6 AnySpecies()

```
void Counterpoint::AnySpecies (
            int OurMode,
            int * StartPitches,
            int CurV,
            int CantusFirmusLength,
            int Species )  [inherited]
```

References Counterpoint::AllDone, Counterpoint::BasePitch, Counterpoint::BestFit, Counterpoint::BestFitFirst(), Counterpoint::BestFitPenalty, Counterpoint::Branches, Counterpoint::CleanRhy(), Counterpoint::Ctrpt, Counterpoint::Dur, Counterpoint::GoodRhy(), Counterpoint::HalfNote, Counterpoint::infinity, Counterpoint::MaxPenalty, Counterpoint::Mode, Counterpoint::MostNotes, Counterpoint::MostVoices, Counterpoint::Onset, Counterpoint::PenaltyRatio, Counterpoint::QuarterNote, Counterpoint::RealBad, Counterpoint::RhyNotes, Counterpoint::RhyPat, Counterpoint::TotalNotes, Counterpoint::TotalTime, Counterpoint::UsedRhy(), and Counterpoint::WholeNote.

Referenced by Counterpoint::counterpoint(), and main().

6.26.4.7 ARRBLT()

```
void Counterpoint::ARRBLT (
            int * dest,
            int * source,
            int num )  [inherited]
```

Referenced by Counterpoint::SaveIndx().

6.26.4.8 ASeventh()

```
int Counterpoint::ASeventh (
            int Interval )  [inherited]
```

References Counterpoint::MajorSeventh, and Counterpoint::MinorSeventh.

Referenced by Counterpoint::SpecialSpeciesCheck().

6.26.4.9 ASkip()

```
int Counterpoint::ASkip (
            int Interval )  [inherited]
```

References Counterpoint::ABS(), and Counterpoint::MajorSecond.

Referenced by Counterpoint::Check(), Counterpoint::ConsecutiveSkipsInSameDirection(), Counterpoint::OtherVoiceCheck(), Counterpoint::SaveResults(), and Counterpoint::SpecialSpeciesCheck().

6.26.4.10 AStep()

```
int Counterpoint::AStep (
            int Interval )  [inherited]
```

References Counterpoint::ABS(), Counterpoint::MajorSecond, and Counterpoint::MinorSecond.

Referenced by Counterpoint::ADissonance(), Counterpoint::Check(), and Counterpoint::SpecialSpeciesCheck().

6.26.4.11 ATenth()

```
int Counterpoint::ATenth (
            int Interval )  [inherited]
```

References Counterpoint::ABS(), and Counterpoint::AThird().

Referenced by Counterpoint::Check().

6.26.4.12 AThird()

```
int Counterpoint::AThird (
            int Interval )  [inherited]
```

References Counterpoint::MajorThird, and Counterpoint::MinorThird.

Referenced by Counterpoint::ATenth(), and Counterpoint::SpecialSpeciesCheck().

6.26.4.13 BadMelody()

```
int Counterpoint::BadMelody (
            int Intv )  [inherited]
```

References Counterpoint::ABS(), Counterpoint::BadMelodyInterval, Counterpoint::MinorSixth, and Counterpoint::Octave.

Referenced by Counterpoint::Check().

6.26.4.14 Bass()

```
int Counterpoint::Bass (
            int Cn,
            int v )  [inherited]
```

References Counterpoint::Cantus(), Counterpoint::MIN(), and Counterpoint::Other().

Referenced by Counterpoint::Check(), Counterpoint::OtherVoiceCheck(), and Counterpoint::SpecialSpeciesCheck().

6.26.4.15 Beat8()

```
int Counterpoint::Beat8 (
            int n )  [inherited]
```

Referenced by Counterpoint::ADissonance(), Counterpoint::DownBeat(), and Counterpoint::SpecialSpeciesCheck().

6.26.4.16 BestFitFirst()

```
void Counterpoint::BestFitFirst (
            int CurTime,
            int CurrentPenalty,
            int NumParts,
            int Species,
            int BrLim )  [inherited]
```

References Counterpoint::AllDone, Counterpoint::BestFitFirst(), Counterpoint::BestFitPenalty, Counterpoint::Branches, Counterpoint::EndF, Counterpoint::Field, Counterpoint::Indx, Counterpoint::infinity, Counterpoint::Look(), Counterpoint::MaxPenalty, Counterpoint::MIN(), Counterpoint::MostVoices, Counterpoint::NumFields, Counterpoint::Onset, Counterpoint::PenaltyRatio, Counterpoint::SaveResults(), Counterpoint::SetUs(), Counterpoint::TotalTime, Counterpoint::Us(), and Counterpoint::VIndex().

Referenced by Counterpoint::AnySpecies(), and Counterpoint::BestFitFirst().

6.26.4.17 Cantus()

```
int Counterpoint::Cantus (
            int n,
            int v )  [inherited]
```

References Counterpoint::Ctrpt, and Counterpoint::Onset.

Referenced by Counterpoint::Bass(), and Counterpoint::Check().

6.26.4.18 Check()

```
int Counterpoint::Check (
            int Cn,
            int Cp,
            int v,
            int NumParts,
            int Species,
            int CurLim )  [inherited]
```

References Counterpoint::ABS(), Counterpoint::ADissonance(), Counterpoint::Aeolian, Counterpoint::AnOctave(), Counterpoint::ASkip(), Counterpoint::AStep(), Counterpoint::ATenth(), Counterpoint::BadCadencePenalty, Counterpoint::BadMelody(), Counterpoint::BadMelodyPenalty, Counterpoint::BasePitch, Counterpoint::Bass(), Counterpoint::Cantus(), Counterpoint::CompoundPenal Counterpoint::ConsecutiveSkipsInSameDirection(), Counterpoint::CrossAboveCantusPenalty, Counterpoint::DirectMotion, Counterpoint::DirectMotionPenalty, Counterpoint::DirectMotionToPerfectConsonance(), Counterpoint::DirectPerfectOnDownbeatPenalty, Counterpoint::DirectToFifthPenalty, Counterpoint::DirectToOctavePenalty, Counterpoint::Dissonance, Counterpoint::DissonanceNotFilling Counterpoint::DissonancePenalty, Counterpoint::Doubled(), Counterpoint::DoubledLeadingTonePenalty, Counterpoint::DownBeat(), Counterpoint::EndOnPerfectPenalty, Counterpoint::ExtremeRange(), Counterpoint::ExtremeRangePenalty, Counterpoint::Fifth, Counterpoint::FifthFollowedBySameDirectionPenalty, Counterpoint::FifthPrecededBySameDirectionPenalty, Counterpoint::FirstNote(), Counterpoint::FourRepeatedNotesPenalty, Counterpoint::Fourth, Counterpoint::InMode(), Counterpoint::LastNote(), Counterpoint::LeapAtCadencePenalty, Counterpoint::LowerNeighborPenalty, Counterpoint::Lydian, Counterpoint::LydianCadentialTritone Counterpoint::MajorSixth, Counterpoint::MajorThird, Counterpoint::MAX(), Counterpoint::MelodicBoredomPenalty, Counterpoint::MelodicTritonePenalty, Counterpoint::MinorSecond, Counterpoint::MinorSixth, Counterpoint::Mode, Counterpoint::MotionType(), Counterpoint::NextToLastNote(), Counterpoint::NoLeadingTonePenalty, Counterpoint::NoMotionAgainstOcta Counterpoint::Octave, Counterpoint::OctaveLeapPenalty, Counterpoint::OtherVoiceCheck(), Counterpoint::OutOfModePenalty, Counterpoint::OutOfRange(), Counterpoint::OutOfRangePenalty, Counterpoint::OverOctavePenalty, Counterpoint::OverTwelfthPenalty, Counterpoint::ParallelFifthPenalty, Counterpoint::ParallelUnisonPenalty, Counterpoint::PerfectConsonance, Counterpoint::PerfectConsona Counterpoint::Phrygian, Counterpoint::PitchRepeats(), Counterpoint::RepetitionOnUpbeatPenalty, Counterpoint::SixthFollowedBySameD Counterpoint::SixthLeapPenalty, Counterpoint::SixthPrecededBySameDirectionPenalty, Counterpoint::SkipFollowedBySameDirectionPen Counterpoint::SkipFromUnisonPenalty, Counterpoint::SkipPrecededBySameDirectionPenalty, Counterpoint::SkipTo8vePenalty, Counterpoint::SpecialSpeciesCheck(), Counterpoint::TenthToOctavePenalty, Counterpoint::ThreeRepeatedNotesPenalty, Counterpoint::ThreeSkipsPenalty, Counterpoint::TooMuchOfInterval(), Counterpoint::TotalNotes, Counterpoint::TotalRange(), Counterpoint::Tritone, Counterpoint::TwoRepeatedNotesPenalty, Counterpoint::TwoSkipsNotInTriadPenalty, Counterpoint::TwoSkipsPenal Counterpoint::Unison, Counterpoint::UnisonDownbeatPenalty, Counterpoint::UnisonPenalty, Counterpoint::UnresolvedLeadingTonePenal Counterpoint::UpBeat(), Counterpoint::UpperNeighborPenalty, Counterpoint::Us(), and Counterpoint::VerticalTritonePenalty.

Referenced by Counterpoint::Look().

6.26.4.19 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.26.4.20 CleanRhy()

```
void Counterpoint::CleanRhy ( )  [inherited]
```

References Counterpoint::RhyPat.

Referenced by Counterpoint::AnySpecies().

6.26.4.21 clear() [1/2]

```
void Counterpoint::clear ( )  [virtual], [inherited]
```

References Counterpoint::BestFit, Counterpoint::BestFit1, Counterpoint::BestFit2, Counterpoint::Ctrpt, Counterpoint::Dur, Counterpoint::MostVoices, Counterpoint::Onset, Counterpoint::RhyNotes, Counterpoint::RhyPat, Counterpoint::TotalNotes, and Counterpoint::vbs.

Referenced by transform().

6.26.4.22 clear() [2/2]

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.26.4.23 ConsecutiveSkipsInSameDirection()

```
int Counterpoint::ConsecutiveSkipsInSameDirection (
            int Pitch1,
            int Pitch2,
            int Pitch3 )  [inherited]
```

References Counterpoint::ASkip().

Referenced by Counterpoint::Check().

6.26.4.24 counterpoint()

```
void Counterpoint::counterpoint (
            int OurMode,
            int * StartPitches,
            int CurV,
            int CantusFirmusLength,
            int Species,
            int * cantus )  [inherited]
```

References Counterpoint::AnySpecies(), Counterpoint::Ctrpt, Counterpoint::Fits, Counterpoint::initialize(), and Counterpoint::vbs.

Referenced by main(), and transform().

6.26.4.25 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.26.4.26 CurRhy()

```
int Counterpoint::CurRhy (
            int n )  [inherited]
```

References Counterpoint::RhyPat.

Referenced by Counterpoint::GoodRhy().

6.26.4.27 DirectMotionToPerfectConsonance()

```
int Counterpoint::DirectMotionToPerfectConsonance (
            int Pitch1,
            int Pitch2,
            int Pitch3,
            int Pitch4 )  [inherited]
```

References Counterpoint::ABS(), Counterpoint::DirectMotion, Counterpoint::MotionType(), and Counterpoint::PerfectConsonance.

Referenced by Counterpoint::Check(), and Counterpoint::OtherVoiceCheck().

6.26.4.28 Doubled()

```
int Counterpoint::Doubled (
            int Pitch,
            int Cn,
            int v )  [inherited]
```

References Counterpoint::Other().

Referenced by Counterpoint::Check().

6.26.4.29 DownBeat()

```
int Counterpoint::DownBeat (
            int n,
            int v )  [inherited]
```

References Counterpoint::Beat8(), and Counterpoint::Onset.

Referenced by Counterpoint::ADissonance(), Counterpoint::Check(), Counterpoint::SpecialSpeciesCheck(), and Counterpoint::UpBeat().

6.26.4.30 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.26.4.31 ExtremeRange()

```
int Counterpoint::ExtremeRange (
            int Pitch )  [inherited]
```

References Counterpoint::HighestSemitone, and Counterpoint::LowestSemitone.

Referenced by Counterpoint::Check().

6.26.4.32 fillCantus()

```
void Counterpoint::fillCantus (
            int c0,
            int c1,
            int c2,
            int c3,
            int c4,
            int c5,
            int c6,
            int c7,
            int c8,
            int c9,
            int c10,
            int c11,
            int c12,
            int c13,
            int c14 )  [inherited]
```

References Counterpoint::Ctrpt.

Referenced by main().

6.26.4.33 FillRhyPat()

```
void Counterpoint::FillRhyPat ( )  [inherited]
```

References Counterpoint::EighthNote, Counterpoint::HalfNote, Counterpoint::QuarterNote, Counterpoint::RhyNotes, Counterpoint::RhyPat, and Counterpoint::WholeNote.

Referenced by CounterpointNode(), and main().

6.26.4.34 FirstNote()

```
int Counterpoint::FirstNote (
            int n,
            int v )  [inherited]
```

Referenced by Counterpoint::ADissonance(), and Counterpoint::Check().

6.26.4.35 generate()

```
void csound::Node::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented in csound::ExternalNode, csound::ScoreNode, csound::ChordLindenmayer, csound::MCRM, csound::Generator, csound::Random, csound::LispGenerator, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.26.4.36 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.26.4.37 getGenerationMode()

```
virtual int csound::CounterpointNode::getGenerationMode ( ) const  [inline], [virtual]
```

6.26.4.38 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.26.4.39 getMusicMode()

```
virtual int csound::CounterpointNode::getMusicMode ( ) const  [inline], [virtual]
```

6.26.4.40 getSecondsPerPulse()

```
virtual double csound::CounterpointNode::getSecondsPerPulse ( ) const  [inline], [virtual]
```

6.26.4.41 getSpecies()

```
virtual int csound::CounterpointNode::getSpecies ( ) const  [inline], [virtual]
```

6.26.4.42 getVoiceBeginnings()

```
virtual std::vector< int > & csound::CounterpointNode::getVoiceBeginnings ( )  [inline], [virtual]
```

6.26.4.43 getVoices()

```
virtual size_t csound::CounterpointNode::getVoices ( ) const  [inline], [virtual]
```

6.26.4.44 GoodRhy()

```
int Counterpoint::GoodRhy ( )  [inherited]
```

References Counterpoint::CurRhy(), Counterpoint::MAX(), Counterpoint::MIN(), and Counterpoint::RANDOM().

Referenced by Counterpoint::AnySpecies().

6.26.4.45 initialize()

```
void Counterpoint::initialize (
            int mostnotes,
            int mostvoices )  [virtual], [inherited]
```

References Counterpoint::BestFit, Counterpoint::BestFit1, Counterpoint::BestFit2, Counterpoint::Ctrpt, Counterpoint::Dur, Counterpoint::MostNotes, Counterpoint::MostVoices, Counterpoint::Onset, Counterpoint::randx, Counterpoint::RhyNotes, Counterpoint::RhyPat, Counterpoint::TotalNotes, and Counterpoint::vbs.

Referenced by Counterpoint::Counterpoint(), and Counterpoint::counterpoint().

6.26.4.46 InMode()

```
int Counterpoint::InMode (
            int Pitch,
            int Mode )  [inherited]
```

References Counterpoint::_Aeolian, Counterpoint::_Dorian, Counterpoint::_Ionian, Counterpoint::_Locrian, Counterpoint::_Lydian, Counterpoint::_Mixolydian, Counterpoint::_Phrygian, Counterpoint::Aeolian, Counterpoint::Dorian, Counterpoint::Ionian, Counterpoint::Locrian, Counterpoint::Lydian, Counterpoint::Mixolydian, Counterpoint::Mode, and Counterpoint::Phrygian.

Referenced by Counterpoint::Check(), Counterpoint::OtherVoiceCheck(), and Counterpoint::SaveResults().

6.26.4.47 LastNote()

```
int Counterpoint::LastNote (
            int n,
            int v )  [inherited]
```

References Counterpoint::TotalNotes.

Referenced by Counterpoint::ADissonance(), Counterpoint::Check(), and Counterpoint::OtherVoiceCheck().

6.26.4.48 Look()

```
int Counterpoint::Look (
            int CurPen,
            int CurVoice,
            int NumParts,
            int Species,
            int Lim,
            int * Pens,
            int * Is,
            int * CurNotes )  [inherited]
```

References Counterpoint::Check(), Counterpoint::Ctrpt, Counterpoint::Indx, Counterpoint::Look(), Counterpoint::MIN(), Counterpoint::SaveIndx(), and Counterpoint::SetUs().

Referenced by Counterpoint::BestFitFirst(), and Counterpoint::Look().

6.26.4.49 MAX()

```
int Counterpoint::MAX (
            int a,
            int b )  [inherited]
```

Referenced by Counterpoint::Check(), Counterpoint::GoodRhy(), and Counterpoint::TotalRange().

6.26.4.50 message() [1/2]

```
void Counterpoint::message (
            const char * format,
            va_list valist )  [inherited]
```

References csound::System::message().

6.26.4.51 message() [2/2]

```
void Counterpoint::message (
            const char * format,
             ...  )  [inherited]
```

References Counterpoint::message().

Referenced by Counterpoint::message(), and Counterpoint::SaveResults().

6.26.4.52 MIN()

```
int Counterpoint::MIN (
            int a,
            int b )  [inherited]
```

Referenced by Counterpoint::Bass(), Counterpoint::BestFitFirst(), Counterpoint::GoodRhy(), Counterpoint::Look(), Counterpoint::SaveResults(), and Counterpoint::TotalRange().

6.26.4.53 MotionType()

```
int Counterpoint::MotionType (
            int Pitch1,
            int Pitch2,
            int Pitch3,
            int Pitch4 )  [inherited]
```

References Counterpoint::ContraryMotion, Counterpoint::DirectMotion, Counterpoint::NoMotion, and Counterpoint::ObliqueMotion.

Referenced by Counterpoint::Check(), Counterpoint::DirectMotionToPerfectConsonance(), and Counterpoint::OtherVoiceCheck().

6.26.4.54 NextToLastNote()

```
int Counterpoint::NextToLastNote (
            int n,
            int v )  [inherited]
```

References Counterpoint::TotalNotes.

Referenced by Counterpoint::Check(), and Counterpoint::SpecialSpeciesCheck().

6.26.4.55 Other()

```
int Counterpoint::Other (
            int Cn,
            int v,
            int v1 )  [inherited]
```

References Counterpoint::Ctrpt, Counterpoint::Onset, and Counterpoint::VIndex().

Referenced by Counterpoint::Bass(), Counterpoint::Doubled(), Counterpoint::OtherVoiceCheck(), and Counterpoint::SaveResults().

6.26.4.56 OtherVoiceCheck()

```
int Counterpoint::OtherVoiceCheck (
            int Cn,
            int Cp,
            int v,
            int NumParts,
            int Species,
            int CurLim )  [inherited]
```

References Counterpoint::ABS(), Counterpoint::AddInterval(), Counterpoint::AllVoicesSkipPenalty, Counterpoint::ASkip(), Counterpoint::AugmentedIntervalPenalty, Counterpoint::Bass(), Counterpoint::ContraryMotion, Counterpoint::CrossBelowBassPenalty, Counterpoint::DirectMotion, Counterpoint::DirectMotionToPerfectConsonance(), Counterpoint::Dissonance, Counterpoint::DoubledFifthPe Counterpoint::DoubledLeadingTonePenalty, Counterpoint::DoubledSixthPenalty, Counterpoint::Fifth, Counterpoint::Fourth, Counterpoint::InMode(), Counterpoint::InnerVoicesInDirectToPerfectPenalty, Counterpoint::InnerVoicesInDirectToTritonePenalty, Counterpoint::INTERVALS_WITH_BASS_SIZE, Counterpoint::IntervalsWithBass, Counterpoint::LastNote(), Counterpoint::MajorThird, Counterpoint::Mode, Counterpoint::MotionType(), Counterpoint::NotContraryToOthersPenalty, Counterpoint::NotTriadPenalty, Counterpoint::Octave, Counterpoint::Other(), Counterpoint::ParallelFifthPenalty, Counterpoint::ParallelUnisonPenalty, Counterpoint::SixFiveChordPenalty, Counterpoint::ThirdDoubledPenalty, Counterpoint::TripledBassPenalty, Counterpoint::Tritone, Counterpoint::Unison, Counterpoint::UnisonPenalty, Counterpoint::UnpreparedSixFivePenalty, Counterpoint::UnresolvedSixFivePenalty, Counterpoint::UpperVoicesTooFarApartPenalty, Counterpoint::Us(), and Counterpoint::VerticalTritonePenalty.

Referenced by Counterpoint::Check().

6.26.4.57 OutOfRange()

```
int Counterpoint::OutOfRange (
            int Pitch )  [inherited]
```

References Counterpoint::HighestSemitone, and Counterpoint::LowestSemitone.

Referenced by Counterpoint::Check().

6.26.4.58 PitchRepeats()

```
int Counterpoint::PitchRepeats (
            int Cn,
            int Cp,
            int v )  [inherited]
```

References Counterpoint::Us().

Referenced by Counterpoint::Check().

6.26.4.59 RANDOM()

```
float Counterpoint::RANDOM (
            float amp )  [inherited]
```

References Counterpoint::mersenneTwister, and Counterpoint::uniform_real_generator.

Referenced by Counterpoint::GoodRhy().

6.26.4.60 SaveIndx()

```
int Counterpoint::SaveIndx (
            int indx,
            int * Sp )  [inherited]
```

References Counterpoint::ARRBLT(), Counterpoint::EndF, and Counterpoint::Field.

Referenced by Counterpoint::Look().

6.26.4.61 SaveResults()

```
void Counterpoint::SaveResults (
            int CurrentPenalty,
            int Penalty,
            int v1,
            int Species )  [inherited]
```

References Counterpoint::ABS(), Counterpoint::ASkip(), Counterpoint::BasePitch, Counterpoint::BestFit, Counterpoint::BestFit1, Counterpoint::BestFit2, Counterpoint::BestFitPenalty, Counterpoint::Ctrpt, Counterpoint::Fifth, Counterpoint::Fits, Counterpoint::Fourth, Counterpoint::InMode(), Counterpoint::MaxPenalty, Counterpoint::message(), Counterpoint::MIN(), Counterpoint::MinorSecond, Counterpoint::MinorThird, Counterpoint::Mode, Counterpoint::Octave, Counterpoint::Other(), Counterpoint::PenaltyRatio, Counterpoint::SetUs(), Counterpoint::TotalNotes, Counterpoint::Unison, and Counterpoint::Us().

Referenced by Counterpoint::BestFitFirst().

6.26.4.62 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.26.4.63 setGenerationMode()

```
virtual void csound::CounterpointNode::setGenerationMode (
            int value )  [inline], [virtual]
```

6.26.4.64 setMusicMode()

```
virtual void csound::CounterpointNode::setMusicMode (
            int value )  [inline], [virtual]
```

6.26.4.65 setSecondsPerPulse()

```
virtual void csound::CounterpointNode::setSecondsPerPulse (
            double value )  [inline], [virtual]
```

6.26.4.66 setSpecies()

```
virtual void csound::CounterpointNode::setSpecies (
            int value )  [inline], [virtual]
```

6.26.4.67 SetUs()

```
void Counterpoint::SetUs (
            int n,
            int p,
            int v )  [inherited]
```

References Counterpoint::Ctrpt.

Referenced by Counterpoint::BestFitFirst(), Counterpoint::Look(), and Counterpoint::SaveResults().

6.26.4.68 setVoiceBeginnings()

```
virtual void csound::CounterpointNode::setVoiceBeginnings (
            const std::vector< int > & value )  [inline], [virtual]
```

6.26.4.69 setVoices()

```
virtual void csound::CounterpointNode::setVoices (
            size_t value )  [inline], [virtual]
```

6.26.4.70 Size()

```
int Counterpoint::Size (
            int MelInt )  [inherited]
```

References Counterpoint::ABS(), Counterpoint::Eight, Counterpoint::Fifth, Counterpoint::Five, Counterpoint::Four, Counterpoint::Fourth, Counterpoint::MajorSecond, Counterpoint::MajorThird, Counterpoint::MinorSecond, Counterpoint::MinorSixth, Counterpoint::MinorThird, Counterpoint::Octave, Counterpoint::One, Counterpoint::Six, Counterpoint::Three, Counterpoint::Two, and Counterpoint::Unison.

Referenced by Counterpoint::TooMuchOfInterval().

6.26.4.71 SpecialSpeciesCheck()

```
int Counterpoint::SpecialSpeciesCheck (
            int Cn,
            int Cp,
            int v,
            int Other0,
            int Other1,
            int Other2,
            int NumParts,
            int Species,
            int MelInt,
            int Interval,
            int ActInt,
            int LastIntClass,
            int Pitch,
            int LastMelInt,
            int CurLim )  [inherited]
```

References Counterpoint::ABS(), Counterpoint::ASeventh(), Counterpoint::ASkip(), Counterpoint::AStep(), Counterpoint::AThird(), Counterpoint::BadCadencePenalty, Counterpoint::Bass(), Counterpoint::Beat8(), Counterpoint::Dissonance, Counterpoint::DissonancePe Counterpoint::DownBeat(), Counterpoint::DownBeatUnisonPenalty, Counterpoint::Dur, Counterpoint::EighthJumpPenalty, Counterpoint::EighthNote, Counterpoint::Fifth, Counterpoint::Fourth, Counterpoint::HalfNote, Counterpoint::HalfUntiedPenalty, Counterpoint::LesserLigaturePenalty, Counterpoint::MajorSecond, Counterpoint::MajorThird, Counterpoint::MinorSecond, Counterpoint::MinorSixth, Counterpoint::Mode, Counterpoint::NextToLastNote(), Counterpoint::NotaCambiataPenalty, Counterpoint::NotaLigaturePenalty, Counterpoint::NoTimeForaLigaturePenalty, Counterpoint::Onset, Counterpoint::Phrygian, Counterpoint::QuarterNote, Counterpoint::SkipToDownBeatPenalty, Counterpoint::Tritone, Counterpoint::Unison, Counterpoint::UnisonOnBeat4Penalty, Counterpoint::UnisonUpbeatPenalty, Counterpoint::UnresolvedLigaturePenalty, Counterpoint::UpBeat(), and Counterpoint::Us().

Referenced by Counterpoint::Check().

6.26.4.72 toCsoundScore()

```
void Counterpoint::toCsoundScore (
            std::string filename,
            double secondsPerPulse )  [inherited]
```

References Counterpoint::Ctrpt, Counterpoint::Dur, csound::System::inform(), Counterpoint::Onset, and Counterpoint::TotalNotes.

Referenced by main().

6.26.4.73 TooMuchOfInterval()

```
int Counterpoint::TooMuchOfInterval (
            int Cn,
            int Cp,
            int v )  [inherited]
```

References Counterpoint::Ctrpt, and Counterpoint::Size().

Referenced by Counterpoint::Check().

6.26.4.74 TotalRange()

```
int Counterpoint::TotalRange (
            int Cn,
            int Cp,
            int v )  [inherited]
```

References Counterpoint::MAX(), Counterpoint::MIN(), and Counterpoint::Us().

Referenced by Counterpoint::Check().

6.26.4.75 transform()

```
void csound::CounterpointNode::transform (
            Score & score_from_children )  [virtual]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented from csound::Node.

References csound::Score::append(), Counterpoint::clear(), Counterpoint::counterpoint(), Counterpoint::Ctrpt, Counterpoint::Dur, csound::fundamentalDomainByPredicate(), Counterpoint::HighestSemitone, Counterpoint::LowestSemitone, csound::System::message(), csound::Conversions::midiToPitchClass(), musicMode, csound::note(), Counterpoint::Onset, secondsPerPulse, csound::Score::sort(), species, Counterpoint::TotalNotes, voiceBeginnings, and voices.

6.26.4.76 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord and csound::Node::transform().

6.26.4.77 UpBeat()

```
int Counterpoint::UpBeat (
            int n,
            int v )  [inherited]
```

References Counterpoint::DownBeat().

Referenced by Counterpoint::ADissonance(), Counterpoint::Check(), and Counterpoint::SpecialSpeciesCheck().

6.26.4.78 Us()

```
int Counterpoint::Us (
            int n,
            int v )  [inherited]
```

References Counterpoint::Ctrpt.

Referenced by Counterpoint::ADissonance(), Counterpoint::BestFitFirst(), Counterpoint::Check(), Counterpoint::OtherVoiceCheck(), Counterpoint::PitchRepeats(), Counterpoint::SaveResults(), Counterpoint::SpecialSpeciesCheck(), and Counterpoint::TotalRange().

6.26.4.79 UsedRhy()

```
void Counterpoint::UsedRhy (
            int n )  [inherited]
```

References Counterpoint::RhyPat.

Referenced by Counterpoint::AnySpecies().

6.26.4.80 VIndex()

```
int Counterpoint::VIndex (
            int Time,
            int VNum )  [inherited]
```

References Counterpoint::Dur, Counterpoint::Onset, and Counterpoint::TotalNotes.

Referenced by Counterpoint::BestFitFirst(), and Counterpoint::Other().

6.26.4.81 winners()

```
void Counterpoint::winners (
            int v1,
            int * data,
            int * best,
            int * best1,
            int * best2,
            int * durs )  [inherited]
```

References Counterpoint::BestFit, Counterpoint::BestFit1, Counterpoint::BestFit2, Counterpoint::Dur, Counterpoint::Fits, Counterpoint::MostNotes, and Counterpoint::TotalNotes.

6.26.5 Field Documentation

6.26.5.1 _Aeolian

```
int Counterpoint::_Aeolian = {1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0}  [static], [inherited]
```

Referenced by Counterpoint::InMode().

6.26.5.2 _Dorian

```
int Counterpoint::_Dorian = {1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0}  [static], [inherited]
```

Referenced by Counterpoint::InMode().

6.26.5.3 _Ionian

```
int Counterpoint::_Ionian = {1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1}  [static], [inherited]
```

Referenced by Counterpoint::InMode().

6.26.5.4 _Locrian

```
int Counterpoint::_Locrian = {1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0}  [static], [inherited]
```

Referenced by Counterpoint::InMode().

6.26.5.5 _Lydian

```
int Counterpoint::_Lydian = {1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1}  [static], [inherited]
```

Referenced by Counterpoint::InMode().

6.26.5.6 _Mixolydian

```
int Counterpoint::_Mixolydian = {1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0}  [static], [inherited]
```

Referenced by Counterpoint::InMode().

6.26.5.7 _Phrygian

```
int Counterpoint::_Phrygian = {1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0}  [static], [inherited]
```

Referenced by Counterpoint::InMode().

6.26.5.8 AllDone

```
int Counterpoint::AllDone  [inherited]
```

Referenced by Counterpoint::AnySpecies(), and Counterpoint::BestFitFirst().

6.26.5.9 AllVoicesSkipPenalty

```
int Counterpoint::AllVoicesSkipPenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::OtherVoiceCheck().

6.26.5.10 AscendingSixthPenalty

```
int Counterpoint::AscendingSixthPenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint().

6.26.5.11 AugmentedIntervalPenalty

```
int Counterpoint::AugmentedIntervalPenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::OtherVoiceCheck().

6.26.5.12 BadCadencePenalty

```
int Counterpoint::BadCadencePenalty  [inherited]
```

Referenced by Counterpoint::Check(), Counterpoint::Counterpoint(), and Counterpoint::SpecialSpeciesCheck().

6.26.5.13 BadMelodyInterval

```
int Counterpoint::BadMelodyInterval = {0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0}  [static], [inherited]
```

Referenced by Counterpoint::BadMelody().

6.26.5.14 BadMelodyPenalty

```
int Counterpoint::BadMelodyPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.15 BasePitch

```
int Counterpoint::BasePitch  [inherited]
```

Referenced by Counterpoint::AnySpecies(), Counterpoint::Check(), and Counterpoint::SaveResults().

6.26.5.16 BestFit

```
Eigen::MatrixXi Counterpoint::BestFit  [inherited]
```

Referenced by Counterpoint::AnySpecies(), Counterpoint::clear(), Counterpoint::initialize(), Counterpoint::SaveResults(), and Counterpoint::winners().

6.26.5.17 BestFit1

```
Eigen::MatrixXi Counterpoint::BestFit1  [inherited]
```

Referenced by Counterpoint::clear(), Counterpoint::initialize(), Counterpoint::SaveResults(), and Counterpoint::winners().

6.26.5.18 BestFit2

```
Eigen::MatrixXi Counterpoint::BestFit2  [inherited]
```

Referenced by Counterpoint::clear(), Counterpoint::initialize(), Counterpoint::SaveResults(), and Counterpoint::winners().

6.26.5.19 BestFitPenalty

```
int Counterpoint::BestFitPenalty  [inherited]
```

Referenced by Counterpoint::AnySpecies(), Counterpoint::BestFitFirst(), and Counterpoint::SaveResults().

6.26.5.20 Branches

```
int Counterpoint::Branches  [inherited]
```

Referenced by Counterpoint::AnySpecies(), and Counterpoint::BestFitFirst().

6.26.5.21 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.26.5.22 CompoundPenalty

```
int Counterpoint::CompoundPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.23 CrossAboveCantusPenalty

```
int Counterpoint::CrossAboveCantusPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.24 CrossBelowBassPenalty

```
int Counterpoint::CrossBelowBassPenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::OtherVoiceCheck().

6.26.5.25 Ctrpt

```
Eigen::MatrixXi Counterpoint::Ctrpt  [inherited]
```

Referenced by Counterpoint::AnySpecies(), Counterpoint::Cantus(), Counterpoint::clear(), Counterpoint::counterpoint(), Counterpoint::fillCantus(), Counterpoint::initialize(), Counterpoint::Look(), Counterpoint::Other(), Counterpoint::SaveResults(), Counterpoint::SetUs(), Counterpoint::toCsoundScore(), Counterpoint::TooMuchOfInterval(), transform(), and Counterpoint::Us().

6.26.5.26 DirectMotionPenalty

```
int Counterpoint::DirectMotionPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.27 DirectPerfectOnDownbeatPenalty

```
int Counterpoint::DirectPerfectOnDownbeatPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.28 DirectToFifthPenalty

```
int Counterpoint::DirectToFifthPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.29 DirectToOctavePenalty

```
int Counterpoint::DirectToOctavePenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.30 DirectToTritonePenalty

```
int Counterpoint::DirectToTritonePenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint().

6.26.5.31 Dissonance

```
int Counterpoint::Dissonance = {0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0}  [static], [inherited]
```

Referenced by Counterpoint::ADissonance(), Counterpoint::Check(), Counterpoint::OtherVoiceCheck(), and Counterpoint::SpecialSpeciesCheck().

6.26.5.32 DissonanceNotFillingThirdPenalty

```
int Counterpoint::DissonanceNotFillingThirdPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.33 DissonancePenalty

```
int Counterpoint::DissonancePenalty  [inherited]
```

Referenced by Counterpoint::Check(), Counterpoint::Counterpoint(), and Counterpoint::SpecialSpeciesCheck().

6.26.5.34 DoubledFifthPenalty

```
int Counterpoint::DoubledFifthPenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::OtherVoiceCheck().

6.26.5.35 DoubledLeadingTonePenalty

```
int Counterpoint::DoubledLeadingTonePenalty  [inherited]
```

Referenced by Counterpoint::Check(), Counterpoint::Counterpoint(), and Counterpoint::OtherVoiceCheck().

6.26.5.36 DoubledSixthPenalty

```
int Counterpoint::DoubledSixthPenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::OtherVoiceCheck().

6.26.5.37 DownBeatUnisonPenalty

```
int Counterpoint::DownBeatUnisonPenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::SpecialSpeciesCheck().

6.26.5.38 Dur

```
Eigen::MatrixXi Counterpoint::Dur  [inherited]
```

Referenced by Counterpoint::ADissonance(), Counterpoint::AnySpecies(), Counterpoint::clear(), Counterpoint::initialize(), Counterpoint::SpecialSpeciesCheck(), Counterpoint::toCsoundScore(), transform(), Counterpoint::VIndex(), and Counterpoint::winners().

6.26.5.39 EighthJumpPenalty

```
int Counterpoint::EighthJumpPenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::SpecialSpeciesCheck().

6.26.5.40 EndOnPerfectPenalty

```
int Counterpoint::EndOnPerfectPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.41 ExtremeRangePenalty

```
int Counterpoint::ExtremeRangePenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.42 FifthFollowedBySameDirectionPenalty

```
int Counterpoint::FifthFollowedBySameDirectionPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.43 FifthPrecededBySameDirectionPenalty

```
int Counterpoint::FifthPrecededBySameDirectionPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.44 Fits

```
int Counterpoint::Fits[3]  [inherited]
```

Referenced by Counterpoint::counterpoint(), Counterpoint::SaveResults(), and Counterpoint::winners().

6.26.5.45 FourRepeatedNotesPenalty

```
int Counterpoint::FourRepeatedNotesPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.46 generationMode

```
int csound::CounterpointNode::generationMode
```

6.26.5.47 HalfUntiedPenalty

```
int Counterpoint::HalfUntiedPenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::SpecialSpeciesCheck().

6.26.5.48 HighestSemitone

```
int Counterpoint::HighestSemitone  [inherited]
```

Referenced by Counterpoint::ExtremeRange(), Counterpoint::OutOfRange(), and transform().

6.26.5.49 ImperfectConsonance

```
int Counterpoint::ImperfectConsonance = {0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0}  [static], [inherited]
```

6.26.5.50 Indx

```
int Counterpoint::Indx = {0, 1, -1, 2, -2, 3, -3, 0, 4, -4, 5, 7, -5, 8, 12, -7,-12}  [static],
[inherited]
```

Referenced by Counterpoint::BestFitFirst(), and Counterpoint::Look().

6.26.5.51 InnerVoicesInDirectToPerfectPenalty

```
int Counterpoint::InnerVoicesInDirectToPerfectPenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::OtherVoiceCheck().

6.26.5.52 InnerVoicesInDirectToTritonePenalty

```
int Counterpoint::InnerVoicesInDirectToTritonePenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::OtherVoiceCheck().

6.26.5.53 IntervalsWithBass

```
int Counterpoint::IntervalsWithBass[INTERVALS_WITH_BASS_SIZE]  [inherited]
```

Referenced by Counterpoint::AddInterval(), and Counterpoint::OtherVoiceCheck().

6.26.5.54 LeapAtCadencePenalty

```
int Counterpoint::LeapAtCadencePenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.55 LesserLigaturePenalty

```
int Counterpoint::LesserLigaturePenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::SpecialSpeciesCheck().

6.26.5.56 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.26.5.57 LowerNeighborPenalty

```
int Counterpoint::LowerNeighborPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.58 LowestSemitone

```
int Counterpoint::LowestSemitone  [inherited]
```

Referenced by Counterpoint::ExtremeRange(), Counterpoint::OutOfRange(), and transform().

6.26.5.59 LydianCadentialTritonePenalty

```
int Counterpoint::LydianCadentialTritonePenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.60 MaxPenalty

```
int Counterpoint::MaxPenalty  [inherited]
```

Referenced by Counterpoint::AnySpecies(), Counterpoint::BestFitFirst(), and Counterpoint::SaveResults().

6.26.5.61 MelodicBoredomPenalty

```
int Counterpoint::MelodicBoredomPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.62 MelodicTritonePenalty

```
int Counterpoint::MelodicTritonePenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.63 mersenneTwister

```
std::mt19937 Counterpoint::mersenneTwister  [static], [inherited]
```

Referenced by Counterpoint::RANDOM().

6.26.5.64 Mode

```
int Counterpoint::Mode  [inherited]
```

Referenced by Counterpoint::AnySpecies(), Counterpoint::Check(), Counterpoint::InMode(), Counterpoint::OtherVoiceCheck(), Counterpoint::SaveResults(), and Counterpoint::SpecialSpeciesCheck().

6.26.5.65 MostNotes

```
int Counterpoint::MostNotes  [inherited]
```

Referenced by Counterpoint::AnySpecies(), Counterpoint::initialize(), and Counterpoint::winners().

6.26.5.66 MostVoices

```
int Counterpoint::MostVoices  [inherited]
```

Referenced by Counterpoint::AnySpecies(), Counterpoint::BestFitFirst(), Counterpoint::clear(), and Counterpoint::initialize().

6.26.5.67 musicMode

```
int csound::CounterpointNode::musicMode
```

Referenced by transform().

6.26.5.68 NoLeadingTonePenalty

```
int Counterpoint::NoLeadingTonePenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.69 NoMotionAgainstOctavePenalty

```
int Counterpoint::NoMotionAgainstOctavePenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.70 NotaCambiataPenalty

```
int Counterpoint::NotaCambiataPenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::SpecialSpeciesCheck().

6.26.5.71 NotaLigaturePenalty

```
int Counterpoint::NotaLigaturePenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::SpecialSpeciesCheck().

6.26.5.72 NotBestCadencePenalty

```
int Counterpoint::NotBestCadencePenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint().

6.26.5.73 NotContraryToOthersPenalty

```
int Counterpoint::NotContraryToOthersPenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::OtherVoiceCheck().

6.26.5.74 NoTimeForaLigaturePenalty

```
int Counterpoint::NoTimeForaLigaturePenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::SpecialSpeciesCheck().

6.26.5.75 NotTriadPenalty

```
int Counterpoint::NotTriadPenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::OtherVoiceCheck().

6.26.5.76 OctaveLeapPenalty

```
int Counterpoint::OctaveLeapPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.77 Onset

```
Eigen::MatrixXi Counterpoint::Onset  [inherited]
```

Referenced by Counterpoint::ADissonance(), Counterpoint::AnySpecies(), Counterpoint::BestFitFirst(), Counterpoint::Cantus(), Counterpoint::clear(), Counterpoint::DownBeat(), Counterpoint::initialize(), Counterpoint::Other(), Counterpoint::SpecialSpeciesCheck(), Counterpoint::toCsoundScore(), transform(), and Counterpoint::VIndex().

6.26.5.78 OutOfModePenalty

```
int Counterpoint::OutOfModePenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.79 OutOfRangePenalty

```
int Counterpoint::OutOfRangePenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.80 OverOctavePenalty

```
int Counterpoint::OverOctavePenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.81 OverTwelfthPenalty

```
int Counterpoint::OverTwelfthPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.82 ParallelFifthPenalty

```
int Counterpoint::ParallelFifthPenalty  [inherited]
```

Referenced by Counterpoint::Check(), Counterpoint::Counterpoint(), and Counterpoint::OtherVoiceCheck().

6.26.5.83 ParallelUnisonPenalty

```
int Counterpoint::ParallelUnisonPenalty  [inherited]
```

Referenced by Counterpoint::Check(), Counterpoint::Counterpoint(), and Counterpoint::OtherVoiceCheck().

6.26.5.84 PenaltyRatio

```
float Counterpoint::PenaltyRatio  [inherited]
```

Referenced by Counterpoint::AnySpecies(), Counterpoint::BestFitFirst(), and Counterpoint::SaveResults().

6.26.5.85 PerfectConsonance

```
int Counterpoint::PerfectConsonance = {1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1}  [static], [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::DirectMotionToPerfectConsonance().

6.26.5.86 PerfectConsonancePenalty

```
int Counterpoint::PerfectConsonancePenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.87 randx

```
long Counterpoint::randx  [inherited]
```

Referenced by Counterpoint::initialize().

6.26.5.88 RepeatedPitchPenalty

```
int Counterpoint::RepeatedPitchPenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint().

6.26.5.89 RepetitionOnUpbeatPenalty

```
int Counterpoint::RepetitionOnUpbeatPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.90 RhyNotes

```
Eigen::VectorXi Counterpoint::RhyNotes  [inherited]
```

Referenced by Counterpoint::AnySpecies(), Counterpoint::clear(), Counterpoint::FillRhyPat(), and Counterpoint::initialize().

6.26.5.91 RhyPat

```
Eigen::MatrixXi Counterpoint::RhyPat  [inherited]
```

Referenced by Counterpoint::AnySpecies(), Counterpoint::CleanRhy(), Counterpoint::clear(), Counterpoint::CurRhy(), Counterpoint::FillRhyPat(), Counterpoint::initialize(), and Counterpoint::UsedRhy().

6.26.5.92 secondsPerPulse

```
double csound::CounterpointNode::secondsPerPulse
```

Referenced by transform().

6.26.5.93 SixFiveChordPenalty

```
int Counterpoint::SixFiveChordPenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::OtherVoiceCheck().

6.26.5.94 SixthFollowedBySameDirectionPenalty

```
int Counterpoint::SixthFollowedBySameDirectionPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.95 SixthLeapPenalty

```
int Counterpoint::SixthLeapPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.96 SixthPrecededBySameDirectionPenalty

```
int Counterpoint::SixthPrecededBySameDirectionPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.97 SkipFollowedBySameDirectionPenalty

```
int Counterpoint::SkipFollowedBySameDirectionPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.98 SkipFromUnisonPenalty

```
int Counterpoint::SkipFromUnisonPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.99 SkipPrecededBySameDirectionPenalty

```
int Counterpoint::SkipPrecededBySameDirectionPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.100 SkipTo8vePenalty

```
int Counterpoint::SkipTo8vePenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.101 SkipToDownBeatPenalty

```
int Counterpoint::SkipToDownBeatPenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::SpecialSpeciesCheck().

6.26.5.102 species

```
int csound::CounterpointNode::species
```

Referenced by transform().

6.26.5.103 TenthToOctavePenalty

```
int Counterpoint::TenthToOctavePenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.104 ThirdDoubledPenalty

```
int Counterpoint::ThirdDoubledPenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::OtherVoiceCheck().

6.26.5.105 ThreeRepeatedNotesPenalty

```
int Counterpoint::ThreeRepeatedNotesPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.106 ThreeSkipsPenalty

```
int Counterpoint::ThreeSkipsPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.107 TotalNotes

```
Eigen::VectorXi Counterpoint::TotalNotes  [inherited]
```

Referenced by Counterpoint::AnySpecies(), Counterpoint::Check(), Counterpoint::clear(), Counterpoint::initialize(), Counterpoint::LastNote(), Counterpoint::NextToLastNote(), Counterpoint::SaveResults(), Counterpoint::toCsoundScore(), transform(), Counterpoint::VIndex(), and Counterpoint::winners().

6.26.5.108 TotalTime

```
int Counterpoint::TotalTime  [inherited]
```

Referenced by Counterpoint::AnySpecies(), and Counterpoint::BestFitFirst().

6.26.5.109 TripledBassPenalty

```
int Counterpoint::TripledBassPenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::OtherVoiceCheck().

6.26.5.110 TwoRepeatedNotesPenalty

```
int Counterpoint::TwoRepeatedNotesPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.111 TwoSkipsNotInTriadPenalty

```
int Counterpoint::TwoSkipsNotInTriadPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.112 TwoSkipsPenalty

```
int Counterpoint::TwoSkipsPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.113 uniform_real_generator

```
std::normal_distribution Counterpoint::uniform_real_generator  [inherited]
```

Referenced by Counterpoint::RANDOM().

6.26.5.114 UnisonDownbeatPenalty

```
int Counterpoint::UnisonDownbeatPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.115 UnisonOnBeat4Penalty

```
int Counterpoint::UnisonOnBeat4Penalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::SpecialSpeciesCheck().

6.26.5.116 UnisonPenalty

```
int Counterpoint::UnisonPenalty  [inherited]
```

Referenced by Counterpoint::Check(), Counterpoint::Counterpoint(), and Counterpoint::OtherVoiceCheck().

6.26.5.117 UnisonUpbeatPenalty

```
int Counterpoint::UnisonUpbeatPenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::SpecialSpeciesCheck().

6.26.5.118 UnpreparedSixFivePenalty

```
int Counterpoint::UnpreparedSixFivePenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::OtherVoiceCheck().

6.26.5.119 UnresolvedLeadingTonePenalty

```
int Counterpoint::UnresolvedLeadingTonePenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.120 UnresolvedLigaturePenalty

```
int Counterpoint::UnresolvedLigaturePenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::SpecialSpeciesCheck().

6.26.5.121 UnresolvedSixFivePenalty

```
int Counterpoint::UnresolvedSixFivePenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::OtherVoiceCheck().

6.26.5.122 UpperNeighborPenalty

```
int Counterpoint::UpperNeighborPenalty  [inherited]
```

Referenced by Counterpoint::Check(), and Counterpoint::Counterpoint().

6.26.5.123 UpperVoicesTooFarApartPenalty

```
int Counterpoint::UpperVoicesTooFarApartPenalty  [inherited]
```

Referenced by Counterpoint::Counterpoint(), and Counterpoint::OtherVoiceCheck().

6.26.5.124 vbs

```
Eigen::VectorXi Counterpoint::vbs  [inherited]
```

Referenced by Counterpoint::clear(), Counterpoint::counterpoint(), Counterpoint::initialize(), and main().

6.26.5.125 VerticalTritonePenalty

```
int Counterpoint::VerticalTritonePenalty  [inherited]
```

Referenced by Counterpoint::Check(), Counterpoint::Counterpoint(), and Counterpoint::OtherVoiceCheck().

6.26.5.126 voiceBeginnings

```
std::vector<int> csound::CounterpointNode::voiceBeginnings
```

Referenced by transform().

6.26.5.127 voices

```
size_t csound::CounterpointNode::voices
```

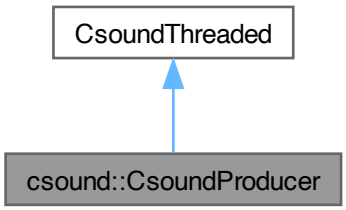
Referenced by transform().

6.27 csound::CsoundProducer Class Reference

Optionally adds metadata, performs post-processing, translates to various soundfile formats as automatic steps in the Csound rendering of a composition to a soundfile.

```
#include <CsoundProducer.hpp>
```

Inheritance diagram for csound::CsoundProducer:



Public Member Functions

- CsoundProducer ()
- virtual bool GetDoGitCommit () const
- virtual std::string GetFilenameBase ()

  Returns the base used for all filenames, which is formed from author-title[-git_hash] with all spaces replaced by underscores.
- virtual std::string GetGitCommitHash ()

  If Git commit is enabled, assumes that the code embedding this piece is within a Git repository and returns the current Git hash of HEAD, to facilitate ensuring a consistent history of revisions of the piece.
- virtual std::string GetMetadata (std::string tag) const

  Returns the value of the metadata for the tag, or an empty string if the tag does not exist.
- virtual void GitCommit ()

  If enabled, assumes that the code embedding this piece is within a Git repository, and commits the repository before rendering the piece to ensure a consistent history of revisions of the piece.
- virtual void Join ()

  Causes the calling thread to wait for the end of the performance thread routine.
- virtual int PerformAndPostProcess ()

  Like PerformAndReset, but performs post-processing, translation, and tagging after rendering, so that these things are all done in the rendering thread.
- virtual int PerformAndPostProcessRoutine ()
- virtual void SetDoGitCommit (bool do_git_commit_)
- virtual void SetMetadata (std::string tag, std::string value)

  Sets the value of a metadata tag.
- virtual void SetOutput (const char *name, const char *type, const char *format)

  Override to not only set but also save type and format.
- virtual int Start ()

  Override to set output filename from metadata in this.
- virtual clock_t startTiming ()
- virtual double stopTiming (clock_t beganAt)
- virtual ~CsoundProducer ()

Protected Attributes

- bool do_git_commit = false
- std::string git_hash
- std::string output_format = "float"
- std::string output_type = "wav"
- std::map< std::string, std::string > tags

6.27.1 Detailed Description

Optionally adds metadata, performs post-processing, translates to various soundfile formats as automatic steps in the Csound rendering of a composition to a soundfile.

Also enables running scripts that can interact with Csound.

6.27.2 Constructor & Destructor Documentation

6.27.2.1 CsoundProducer()

```
csound::CsoundProducer::CsoundProducer ( )  [inline]
```

6.27.2.2 ~CsoundProducer()

```
virtual csound::CsoundProducer::~CsoundProducer ( )  [inline], [virtual]
```

6.27.3 Member Function Documentation

6.27.3.1 GetDoGitCommit()

```
virtual bool csound::CsoundProducer::GetDoGitCommit ( ) const  [inline], [virtual]
```

References do_git_commit.

6.27.3.2 GetFilenameBase()

```
virtual std::string csound::CsoundProducer::GetFilenameBase ( )  [inline], [virtual]
```

Returns the base used for all filenames, which is formed from author-title[-git_hash] with all spaces replaced by underscores.

References do_git_commit, csound::fundamentalDomainByPredicate(), GetGitCommitHash(), and GetMetadata().

Referenced by PerformAndPostProcessRoutine(), and Start().

6.27.3.3 GetGitCommitHash()

```
virtual std::string csound::CsoundProducer::GetGitCommitHash ( )  [inline], [virtual]
```

If Git commit is enabled, assumes that the code embedding this piece is within a Git repository and returns the current Git hash of HEAD, to facilitate ensuring a consistent history of revisions of the piece.

Otherwise, returns an empty string.

References do_git_commit, and csound::fundamentalDomainByPredicate().

Referenced by GetFilenameBase().

6.27.3.4 GetMetadata()

```
virtual std::string csound::CsoundProducer::GetMetadata (
            std::string tag ) const  [inline], [virtual]
```

Returns the value of the metadata for the tag, or an empty string if the tag does not exist.

References csound::fundamentalDomainByPredicate(), and tags.

Referenced by GetFilenameBase().

6.27.3.5 GitCommit()

```
virtual void csound::CsoundProducer::GitCommit ( )  [inline], [virtual]
```

If enabled, assumes that the code embedding this piece is within a Git repository, and commits the repository before rendering the piece to ensure a consistent history of revisions of the piece.

References do_git_commit, and csound::fundamentalDomainByPredicate().

6.27.3.6 Join()

```
virtual void csound::CsoundProducer::Join ( )  [inline], [virtual]
```

Causes the calling thread to wait for the end of the performance thread routine.

References csound::fundamentalDomainByPredicate().

6.27.3.7 PerformAndPostProcess()

```
virtual int csound::CsoundProducer::PerformAndPostProcess ( )  [inline], [virtual]
```

Like PerformAndReset, but performs post-processing, translation, and tagging after rendering, so that these things are all done in the rendering thread.

References csound::fundamentalDomainByPredicate(), and PerformAndPostProcessRoutine().

6.27.3.8 PerformAndPostProcessRoutine()

```
virtual int csound::CsoundProducer::PerformAndPostProcessRoutine ( )  [inline], [virtual]
```

References csound::fundamentalDomainByPredicate(), GetFilenameBase(), output_type, csound::PostProcess(), startTiming(), stopTiming(), and tags.

Referenced by PerformAndPostProcess().

6.27.3.9 SetDoGitCommit()

```
virtual void csound::CsoundProducer::SetDoGitCommit (
            bool do_git_commit_ )  [inline], [virtual]
```

References do_git_commit, and csound::fundamentalDomainByPredicate().

6.27.3.10 SetMetadata()

```
virtual void csound::CsoundProducer::SetMetadata (
            std::string tag,
            std::string value )  [inline], [virtual]
```

Sets the value of a metadata tag.

See: https://www.ffmpeg.org/doxygen/trunk/group__metadata__api.html Other and even user-defined tags may also be used.

References csound::fundamentalDomainByPredicate(), and tags.

6.27.3.11 SetOutput()

```
virtual void csound::CsoundProducer::SetOutput (
            const char * name,
            const char * type,
            const char * format )  [inline], [virtual]
```

Override to not only set but also save type and format.

References csound::fundamentalDomainByPredicate(), output_format, and output_type.

6.27.3.12 Start()

```
virtual int csound::CsoundProducer::Start ( )  [inline], [virtual]
```

Override to set output filename from metadata in this.

Not implemented for real-time rendering.

References csound::fundamentalDomainByPredicate(), GetFilenameBase(), output_format, and output_type.

6.27.3.13 startTiming()

```
virtual clock_t csound::CsoundProducer::startTiming ( )  [inline], [virtual]
```

References csound::fundamentalDomainByPredicate().

Referenced by PerformAndPostProcessRoutine().

6.27.3.14 stopTiming()

```
virtual double csound::CsoundProducer::stopTiming (
            clock_t beganAt )  [inline], [virtual]
```

References csound::fundamentalDomainByPredicate().

Referenced by PerformAndPostProcessRoutine().

6.27.4 Field Documentation

6.27.4.1 do_git_commit

```
bool csound::CsoundProducer::do_git_commit = false  [protected]
```

Referenced by GetDoGitCommit(), GetFilenameBase(), GetGitCommitHash(), GitCommit(), and SetDoGitCommit().

6.27.4.2 git_hash

```
std::string csound::CsoundProducer::git_hash  [protected]
```

6.27.4.3 output_format

```
std::string csound::CsoundProducer::output_format = "float"  [protected]
```

Referenced by SetOutput(), and Start().

6.27.4.4 output_type

```
std::string csound::CsoundProducer::output_type = "wav"  [protected]
```

Referenced by PerformAndPostProcessRoutine(), SetOutput(), and Start().

6.27.4.5 tags

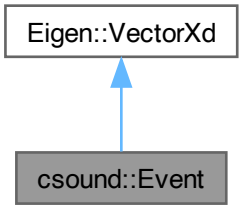
```
std::map<std::string, std::string> csound::CsoundProducer::tags  [protected]
```

Referenced by GetMetadata(), PerformAndPostProcessRoutine(), and SetMetadata().

6.28 csound::Event Class Reference

`#include <Event.hpp>`

Inheritance diagram for csound::Event:



Public Types

- enum { INDEFINITE = 16384 }
- enum Dimensions {
  TIME = 0 , DURATION , STATUS , INSTRUMENT ,
  KEY , VELOCITY , PHASE , PAN ,
  DEPTH , HEIGHT , PITCHES , HOMOGENEITY ,
  ELEMENT_COUNT }

Public Member Functions

- virtual void clearProperties ()
- virtual void conformToPitchClassSet ()
- virtual void correct_negative_duration ()

  If the duration of this Event is negative, first adds it to the Event's onset time, i.e.
- virtual void createNoteOffEvent (Event &event) const
- virtual void dump (std::ostream &stream)
- Event ()
- Event (const Eigen::VectorXd &a)
- Event (const Event &a)
- Event (const std::vector< double > &v)
- Event (double time, double duration, double status, double instrument, double key, double velocity, double phase, double pan, double depth, double height, double pitches)
- Event (std::string text)
- virtual double getAmplitude () const
- virtual int getChannel () const

  MIDI channel numbers are 0-based, Csound instrument numbers are 1-based.
- virtual double getDepth () const

- virtual double getDuration () const
- virtual double getFrequency () const
- virtual double getGain () const
- virtual double getHeight () const
- virtual double getInstrument () const

  MIDI channel numbers are 0-based, Csound instrument numbers are 1-based.
- virtual double getKey () const
- virtual double getKey_tempered (double tonesPerOctave) const
- virtual int getKeyNumber () const
- virtual double getLeftGain () const
- virtual int getMidiStatus () const
- virtual double getOffTime () const
- virtual double getPan () const
- virtual double getPhase () const
- virtual double getPitches () const
- virtual std::string getProperties () const

  Returns any properties of this as a string consisting of "key"="value" pairs in CSV format.
- virtual std::string getProperty (std::string name)
- virtual double getRightGain () const
- virtual double getStatus () const
- virtual int getStatusNumber () const
- virtual double getTime () const
- virtual double getVelocity () const
- virtual int getVelocityNumber () const
- virtual void initialize ()
- virtual bool isMatchingEvent (const Event &event) const
- virtual bool isMatchingNoteOff (const Event &event) const
- virtual bool isMidiEvent () const
- virtual bool isNote () const
- virtual bool isNoteOff () const
- virtual bool isNoteOn () const
- virtual Event & operator= (const Eigen::VectorXd &a)
- virtual Event & operator= (const Event &a)
- virtual void removeProperty (std::string nameO)
- virtual void set (double time, double duration, double status, double instrument, double key, double velocity, double phase=0, double pan=0, double depth=0, double height=0, double pitches=4095)
- virtual void setAmplitude (double amplitude)
- virtual void setChannel (int channel)

  MIDI channel numbers are 0-based, Csound instrument numbers are 1-based.
- virtual void setDepth (double depth)
- virtual void setDuration (double duration)
- virtual void setFrequency (double frequency)
- virtual void setHeight (double height)
- virtual void setInstrument (double instrument)

  MIDI channel numbers are 0-based, Csound instrument numbers are 1-based.
- virtual void setKey (double key)
- virtual void setMidi (double time, char status, char key, char velocity)
- virtual void setOffTime (double offTime)
- virtual void setPan (double pan)
- virtual void setPhase (double phase)

- virtual void setPitches (double pitches)
- virtual void setProperty (std::string name, std::string value)
- virtual void setStatus (double status)
- virtual void setTime (double time)
- virtual void setVelocity (double velocity)
- virtual void temper (double divisionsPerOctave)
- virtual std::string toBlueIStatement (double tempering=12.0) const

  Returns a Csound score statement suitable for use by athenaCL: insno, time, duration, dbsp, pch, pan.
- virtual std::string toCsoundIStatement (double tempering=12.0) const
- virtual std::string toCsoundIStatementHeld (int tag, double tempering=12.0) const
- virtual std::string toCsoundIStatementRelease (int tag, double tempering=12.0) const
- virtual std::string toString () const
- virtual ~Event ()

Static Public Member Functions

- static bool & correct_negative_durations ()

  Gets, or sets by reference, a global flag that determines whether Events have all negative durations adjusted to positive durations with a new onset time.

Data Fields

- std::function< void(csound::Score &, csound::Event &) process)

  Process the data in this; called on all Events in a Score as the final state of processing in ScoreModel.
- std::map< std::string, std::string > properties

Static Public Attributes

- static const char * labels [ ]
- static int SORT_ORDER [ ]

6.28.1 Member Enumeration Documentation

6.28.1.1 anonymous enum

```
anonymous enum
```

Enumerator

INDEFINITE	

6.28.1.2 Dimensions

```
enum csound::Event::Dimensions
```

Enumerator

TIME	
DURATION	
STATUS	
INSTRUMENT	
KEY	
VELOCITY	
PHASE	
PAN	
DEPTH	
HEIGHT	
PITCHES	
HOMOGENEITY	
ELEMENT_COUNT	

6.28.2 Constructor & Destructor Documentation

6.28.2.1 Event() [1/6]

```
csound::Event::Event ( )
```

References initialize().

6.28.2.2 Event() [2/6]

```
csound::Event::Event (
            const Event & a )
```

6.28.2.3 Event() [3/6]

```
csound::Event::Event (
            std::string text )
```

References csound::fundamentalDomainByPredicate(), and initialize().

6.28.2.4 Event() [4/6]

```
csound::Event::Event (
            const Eigen::VectorXd & a )
```

6.28.2.5 Event() [5/6]

```
csound::Event::Event (
            double time,
            double duration,
            double status,
            double instrument,
            double key,
            double velocity,
            double phase,
            double pan,
            double depth,
            double height,
            double pitches )
```

References csound::fundamentalDomainByPredicate(), initialize(), and set().

6.28.2.6 Event() [6/6]

```
csound::Event::Event (
            const std::vector< double > & v )
```

6.28.2.7 ~Event()

```
csound::Event::~Event ( )  [virtual]
```

6.28.3 Member Function Documentation

6.28.3.1 clearProperties()

```
void csound::Event::clearProperties ( )  [virtual]
```

References properties.

6.28.3.2 conformToPitchClassSet()

```
void csound::Event::conformToPitchClassSet ( )  [virtual]
```

References csound::Conversions::findClosestPitchClass(), csound::fundamentalDomainByPredicate(), KEY, csound::Conversions::midiTo
csound::Conversions::midiToRoundedOctave(), csound::Conversions::octaveToMidi(), csound::Conversions::pitchClassToMidi(),
PITCHES, and csound::Conversions::round().

6.28.3.3 correct_negative_duration()

```
void csound::Event::correct_negative_duration ( )  [virtual]
```

If the duration of this Event is negative, first adds it to the Event's onset time, i.e.

moves the onset to match the beginning of the duration, then makes the duration positive. This only has an effect if correct_negative_duratiuons() returns true.

References correct_negative_durations(), csound::fundamentalDomainByPredicate(), getDuration(), getTime(), setDuration(), and setTime().

6.28.3.4 correct_negative_durations()

```
bool & csound::Event::correct_negative_durations ( )  [static]
```

Gets, or sets by reference, a global flag that determines whether Events have all negative durations adjusted to positive durations with a new onset time.

The default is to correct all negative durations.

References csound::fundamentalDomainByPredicate().

Referenced by correct_negative_duration(), csound::getCorrectNegativeDurations(), and csound::setCorrectNegativeDurations().

6.28.3.5 createNoteOffEvent()

```
void csound::Event::createNoteOffEvent (
            Event & event ) const  [virtual]
```

References csound::fundamentalDomainByPredicate().

6.28.3.6 dump()

```
void csound::Event::dump (
            std::ostream & stream )  [virtual]
```

References csound::fundamentalDomainByPredicate().

6.28.3.7 getAmplitude()

```
double csound::Event::getAmplitude ( ) const  [virtual]
```

References getVelocity(), and csound::Conversions::midiToAmplitude().

6.28.3.8 getChannel()

```
int csound::Event::getChannel ( ) const  [virtual]
```

MIDI channel numbers are 0-based, Csound instrument numbers are 1-based.

Returns the Csound instrument number minus 1.

References csound::fundamentalDomainByPredicate(), INSTRUMENT, and csound::Conversions::round().

Referenced by csound::ImageToScore2::generateLocally(), and getMidiStatus().

6.28.3.9 getDepth()

```
double csound::Event::getDepth ( ) const  [virtual]
```

References DEPTH.

Referenced by toCsoundIStatement(), toCsoundIStatementHeld(), and toCsoundIStatementRelease().

6.28.3.10 getDuration()

```
double csound::Event::getDuration ( ) const  [virtual]
```

References DURATION.

Referenced by correct_negative_duration(), csound::ChordScore::getScale(), toBlueIStatement(), toCsoundIStatement(), toCsoundIStatementHeld(), toCsoundIStatementRelease(), toString(), and csound::Koch::traverse().

6.28.3.11 getFrequency()

```
double csound::Event::getFrequency ( ) const  [virtual]
```

References getKey(), and csound::Conversions::midiToHz().

6.28.3.12 getGain()

```
double csound::Event::getGain ( ) const  [virtual]
```

References getVelocity(), and csound::Conversions::midiToGain().

6.28.3.13 getHeight()

```
double csound::Event::getHeight ( ) const  [virtual]
```

References HEIGHT.

Referenced by toCsoundIStatement(), toCsoundIStatementHeld(), and toCsoundIStatementRelease().

6.28.3.14 getInstrument()

```
double csound::Event::getInstrument ( ) const  [virtual]
```

MIDI channel numbers are 0-based, Csound instrument numbers are 1-based.

Returns the Csound instrument number.

References INSTRUMENT.

Referenced by csound::ImageToScore2::pixel_to_event(), toBlueIStatement(), toCsoundIStatement(), toCsoundIStatementHeld(), toCsoundIStatementRelease(), and toString().

6.28.3.15 getKey()

```
double csound::Event::getKey ( ) const  [virtual]
```

References KEY.

Referenced by getFrequency(), getKey_tempered(), csound::ImageToScore2::pixel_to_event(), temper(), toCsoundIStatementHeld(), toCsoundIStatementRelease(), toString(), and csound::Koch::traverse().

6.28.3.16 getKey_tempered()

```
double csound::Event::getKey_tempered (
            double tonesPerOctave ) const  [virtual]
```

References csound::fundamentalDomainByPredicate(), getKey(), csound::Conversions::midiToOctave(), csound::Conversions::octaveToM
and csound::Conversions::round().

Referenced by csound::Score::getPitches(), toBlueIStatement(), and toCsoundIStatement().

6.28.3.17 getKeyNumber()

```
int csound::Event::getKeyNumber ( ) const  [virtual]
```

References csound::fundamentalDomainByPredicate(), KEY, and csound::Conversions::round().

Referenced by csound::ImageToScore2::generateLocally().

6.28.3.18 getLeftGain()

```
double csound::Event::getLeftGain ( ) const  [virtual]
```

References getPan(), and csound::Conversions::leftPan().

6.28.3.19 getMidiStatus()

```
int csound::Event::getMidiStatus ( ) const  [virtual]
```

References csound::fundamentalDomainByPredicate(), getChannel(), and getStatusNumber().

6.28.3.20 getOffTime()

```
double csound::Event::getOffTime ( ) const  [virtual]
```

References DURATION, INDEFINITE, and TIME.

Referenced by csound::ImageToScore2::generateLocally().

6.28.3.21 getPan()

```
double csound::Event::getPan ( ) const  [virtual]
```

References PAN.

Referenced by getLeftGain(), getRightGain(), toBlueIStatement(), toCsoundIStatement(), toCsoundIStatementHeld(), toCsoundIStatementRelease(), and toString().

6.28.3.22 getPhase()

```
double csound::Event::getPhase ( ) const  [virtual]
```

References PHASE.

Referenced by toCsoundIStatement(), toCsoundIStatementHeld(), and toCsoundIStatementRelease().

6.28.3.23 getPitches()

```
double csound::Event::getPitches ( ) const  [virtual]
```

References PITCHES.

Referenced by toCsoundIStatement(), toCsoundIStatementHeld(), toCsoundIStatementRelease(), and toString().

6.28.3.24 getProperties()

```
std::string csound::Event::getProperties ( ) const  [virtual]
```

Returns any properties of this as a string consisting of "key"="value" pairs in CSV format.

References csound::fundamentalDomainByPredicate(), and properties.

Referenced by toBlueIStatement(), toCsoundIStatement(), toCsoundIStatementHeld(), toCsoundIStatementRelease(), and toString().

6.28.3.25 getProperty()

```
std::string csound::Event::getProperty (
            std::string name )  [virtual]
```

References properties.

6.28.3.26 getRightGain()

```
double csound::Event::getRightGain ( ) const  [virtual]
```

References getPan(), and csound::Conversions::rightPan().

6.28.3.27 getStatus()

```
double csound::Event::getStatus ( ) const  [virtual]
```

References STATUS.

Referenced by toString().

6.28.3.28 getStatusNumber()

```
int csound::Event::getStatusNumber ( ) const  [virtual]
```

References csound::fundamentalDomainByPredicate(), csound::Conversions::round(), and STATUS.

Referenced by getMidiStatus(), isNoteOff(), and isNoteOn().

6.28.3.29 getTime()

```
double csound::Event::getTime ( ) const  [virtual]
```

References TIME.

Referenced by correct_negative_duration(), csound::ImageToScore2::generateLocally(), csound::ChordScore::getScale(), csound::ImageToScore2::pixel_to_event(), csound::ChordLindenmayer::scoreOperation(), csound::ChordScore::setDuration(), csound::Score::setDuration(), csound::Score::setDurationFromZero(), setOffTime(), toBlueIStatement(), toCsoundIStatement(), toCsoundIStatementHeld(), toCsoundIStatementRelease(), toString(), csound::Cell::transform(), csound::VoiceleadingNode::transform(), and csound::Koch::traverse().

6.28.3.30 getVelocity()

```
double csound::Event::getVelocity ( ) const  [virtual]
```

References VELOCITY.

Referenced by getAmplitude(), getGain(), isNoteOff(), isNoteOn(), csound::ImageToScore2::pixel_to_event(), toBlueIStatement(), toCsoundIStatement(), toCsoundIStatementHeld(), toCsoundIStatementRelease(), toString(), and csound::Koch::traverse().

6.28.3.31 getVelocityNumber()

```
int csound::Event::getVelocityNumber ( ) const  [virtual]
```

References csound::fundamentalDomainByPredicate(), csound::Conversions::round(), and VELOCITY.

6.28.3.32 initialize()

```
void csound::Event::initialize ( )  [virtual]
```

References ELEMENT_COUNT, csound::fundamentalDomainByPredicate(), and HOMOGENEITY.

Referenced by Event(), Event(), and Event().

6.28.3.33 isMatchingEvent()

```
bool csound::Event::isMatchingEvent (
            const Event & event ) const  [virtual]
```

References csound::fundamentalDomainByPredicate(), INSTRUMENT, and csound::Conversions::round().

6.28.3.34 isMatchingNoteOff()

```
bool csound::Event::isMatchingNoteOff (
            const Event & event ) const  [virtual]
```

References csound::fundamentalDomainByPredicate(), INSTRUMENT, isNoteOn(), KEY, and csound::Conversions::round().

6.28.3.35 isMidiEvent()

```
bool csound::Event::isMidiEvent ( ) const  [virtual]
```

References csound::MidiFile::CHANNEL_NOTE_OFF, and STATUS.

6.28.3.36 isNote()

```
bool csound::Event::isNote ( ) const  [virtual]
```

References isNoteOff(), and isNoteOn().

6.28.3.37 isNoteOff()

```
bool csound::Event::isNoteOff ( ) const  [virtual]
```

References csound::MidiFile::CHANNEL_NOTE_OFF, csound::MidiFile::CHANNEL_NOTE_ON, getStatusNumber(), getVelocity(), and csound::Conversions::round().

Referenced by isNote().

6.28.3.38 isNoteOn()

```
bool csound::Event::isNoteOn ( ) const  [virtual]
```

References csound::MidiFile::CHANNEL_NOTE_ON, getStatusNumber(), getVelocity(), and csound::Conversions::round().

Referenced by csound::conformToChord_equivalence(), isMatchingNoteOff(), and isNote().

6.28.3.39 operator=() [1/2]

```
Event & csound::Event::operator= (
            const Eigen::VectorXd & a )  [virtual]
```

6.28.3.40 operator=() [2/2]

```
Event & csound::Event::operator= (
            const Event & a )  [virtual]
```

References process, and properties.

6.28.3.41 removeProperty()

```
void csound::Event::removeProperty (
            std::string nameO )  [virtual]
```

References properties.

6.28.3.42 set()

```
void csound::Event::set (
            double time,
            double duration,
            double status,
            double instrument,
            double key,
            double velocity,
            double phase = 0,
            double pan = 0,
            double depth = 0,
            double height = 0,
            double pitches = 4095 )  [virtual]
```

References DEPTH, DURATION, csound::fundamentalDomainByPredicate(), HEIGHT, INSTRUMENT, KEY, PAN, PHASE, PITCHES, STATUS, TIME, and VELOCITY.

Referenced by Event().

6.28.3.43 setAmplitude()

```
void csound::Event::setAmplitude (
            double amplitude )  [virtual]
```

References csound::Conversions::amplitudeToMidi(), csound::fundamentalDomainByPredicate(), and setVelocity().

6.28.3.44 setChannel()

```
void csound::Event::setChannel (
            int channel )  [virtual]
```

MIDI channel numbers are 0-based, Csound instrument numbers are 1-based.

Sets the Csound instrument number to the channel plus 1.

References csound::fundamentalDomainByPredicate(), and setInstrument().

Referenced by setMidi().

6.28.3.45 setDepth()

```
void csound::Event::setDepth (
            double depth )  [virtual]
```

References DEPTH.

6.28.3.46 setDuration()

```
void csound::Event::setDuration (
            double duration )  [virtual]
```

References DURATION.

Referenced by correct_negative_duration(), csound::note(), and setOffTime().

6.28.3.47 setFrequency()

```
void csound::Event::setFrequency (
            double frequency )  [virtual]
```

References csound::Conversions::hzToMidi(), and KEY.

6.28.3.48 setHeight()

```
void csound::Event::setHeight (
            double height )  [virtual]
```

References csound::fundamentalDomainByPredicate(), and HEIGHT.

6.28.3.49 setInstrument()

```
void csound::Event::setInstrument (
            double instrument )  [virtual]
```

MIDI channel numbers are 0-based, Csound instrument numbers are 1-based.

Sets the Csound instrument number.

References INSTRUMENT.

Referenced by csound::note(), and setChannel().

6.28.3.50 setKey()

```
void csound::Event::setKey (
            double key )  [virtual]
```

References KEY.

Referenced by csound::ChordLindenmayer::chordOperation(), csound::note(), csound::Score::setPitchClassSet(), and temper().

6.28.3.51 setMidi()

```
void csound::Event::setMidi (
            double time,
            char status,
            char key,
            char velocity )  [virtual]
```

References DURATION, csound::fundamentalDomainByPredicate(), INDEFINITE, KEY, setChannel(), STATUS, TIME, and VELOCITY.

6.28.3.52 setOffTime()

```
void csound::Event::setOffTime (
            double offTime )  [virtual]
```

References csound::fundamentalDomainByPredicate(), getTime(), and setDuration().

Referenced by csound::ImageToScore2::generateLocally(), and csound::Score::tieOverlappingNotes().

6.28.3.53 setPan()

```
void csound::Event::setPan (
            double pan )  [virtual]
```

References PAN.

Referenced by csound::note().

6.28.3.54 setPhase()

```
void csound::Event::setPhase (
            double phase )  [virtual]
```

References PHASE.

6.28.3.55 setPitches()

```
void csound::Event::setPitches (
            double pitches )  [virtual]
```

References csound::fundamentalDomainByPredicate(), and PITCHES.

6.28.3.56 setProperty()

```
void csound::Event::setProperty (
            std::string name,
            std::string value )  [virtual]
```

References properties.

6.28.3.57 setStatus()

```
void csound::Event::setStatus (
            double status )  [virtual]
```

References STATUS.

Referenced by csound::KMeansMCRM::deterministic_algorithm(), csound::Lindenmayer::generateLocally(), csound::MCRM::generateLoc
csound::parse_line(), csound::KMeansMCRM::random_algorithm(), and csound::CMaskNode::translate_to_silence().

6.28.3.58 setTime()

```
void csound::Event::setTime (
            double time )  [virtual]
```

References TIME.

Referenced by csound::Score::add(), csound::Score::append(), csound::Score::append_note(), correct_negative_duration(), csound::mean_to_note(), csound::note(), csound::HarmonyIFS::point_to_note(), csound::Cell::transform(), and csound::Sequence::traverse().

6.28.3.59 setVelocity()

```
void csound::Event::setVelocity (
            double velocity )  [virtual]
```

References VELOCITY.

Referenced by csound::note(), and setAmplitude().

6.28.3.60 temper()

```
void csound::Event::temper (
            double divisionsPerOctave )  [virtual]
```

References getKey(), csound::Conversions::midiToOctave(), csound::Conversions::octaveToMidi(), setKey(), and csound::Conversions::temper().

6.28.3.61 toBlueIStatement()

```
std::string csound::Event::toBlueIStatement (
            double tempering = 12.0 ) const  [virtual]
```

Returns a Csound score statement suitable for use by athenaCL: insno, time, duration, dbsp, pch, pan.

Csound dimensions for athenaCL are: i_instrument = p1 i_time = p2 i_duration = p3 i_dbspa = p4 i_pch = p5 optional properties = p12, printed as a string ("'name'='value', ['name'='value']")

References csound::fundamentalDomainByPredicate(), getDuration(), getInstrument(), getKey_tempered(), getPan(), getProperties(), getTime(), getVelocity(), and csound::Conversions::midiToOctave().

6.28.3.62 toCsoundIStatement()

```
std::string csound::Event::toCsoundIStatement (
            double tempering = 12.0 ) const  [virtual]
```

Csound dimensions now are: i_instrument = p1 i_time = p2 i_duration = p3 i_midi_key = p4 i_midi_velocity = p5 k↩
_space_front_to_back = p6 ; Ambisonic X k_space_left_to_right = p7 ; Ambisonic Y k_space_bottom_to_top = p8; Ambisonic Z i_phase = p9 i_pitches = p10 i_homogeneity = p11
optional properties = p12, printed as a string ("'name'='value', ['name'='value']")

References csound::fundamentalDomainByPredicate(), getDepth(), getDuration(), getHeight(), getInstrument(), getKey_tempered(), getPan(), getPhase(), getPitches(), getProperties(), getTime(), getVelocity(), and HOMOGENEITY.

Referenced by csound::RemoveDuplicates::transform().

6.28.3.63 toCsoundIStatementHeld()

```
std::string csound::Event::toCsoundIStatementHeld (
            int tag,
            double tempering = 12.0 ) const  [virtual]
```

References csound::fundamentalDomainByPredicate(), getDepth(), getDuration(), getHeight(), getInstrument(), getKey(), getPan(), getPhase(), getPitches(), getProperties(), getTime(), getVelocity(), HOMOGENEITY, csound::Conversions::midiToOcta
csound::Conversions::octaveToMidi(), csound::Conversions::round(), and csound::Conversions::temper().

6.28.3.64 toCsoundIStatementRelease()

```
std::string csound::Event::toCsoundIStatementRelease (
            int tag,
            double tempering = 12.0 ) const  [virtual]
```

References csound::fundamentalDomainByPredicate(), getDepth(), getDuration(), getHeight(), getInstrument(), getKey(), getPan(), getPhase(), getPitches(), getProperties(), getTime(), getVelocity(), HOMOGENEITY, csound::Conversions::midiToOcta
csound::Conversions::octaveToMidi(), csound::Conversions::round(), and csound::Conversions::temper().

6.28.3.65 toString()

```
std::string csound::Event::toString ( ) const  [virtual]
```

References csound::fundamentalDomainByPredicate(), getDuration(), getInstrument(), getKey(), getPan(), getPitches(), getProperties(), getStatus(), getTime(), and getVelocity().

Referenced by csound::Turtle::__str__(), and csound::ChordLindenmayer::arithmetic().

6.28.4 Field Documentation

6.28.4.1 labels

```
const char * csound::Event::labels  [static]
```

Initial value:

```
= {
    "Time",
    "Duration",
    "Status",
    "Instrument",
    "Key",
    "Velocity",
    "Pan",
    "Depth",
    "Height",
    "Phase",
    "PitchClassSet",
    "Homogeneity"
}
```

Referenced by csound::Rescale::initialize().

6.28.4.2 process

```
std::function<void(csound::Score &, csound::Event &) csound::Event::process)
```

Process the data in this; called on all Events in a Score as the final state of processing in ScoreModel.

Typically, "process" is a closure that contains references to any other data required to process this. Example: put a Chord in the process closure, and when it is called, conform the pitch of this Event to the Chord. The Score is sorted before this is called.

Referenced by operator=().

6.28.4.3 properties

```
std::map<std::string,std::string> csound::Event::properties
```

Referenced by clearProperties(), getProperties(), getProperty(), operator=(), removeProperty(), and setProperty().

6.28.4.4 SORT_ORDER

```
int csound::Event::SORT_ORDER  [static]
```

Initial value:

```
= {
    Event::TIME,
    Event::INSTRUMENT,
    Event::KEY,
    Event::DURATION,
    Event::VELOCITY,
    Event::PAN,
    Event::DEPTH,
    Event::HEIGHT,
    Event::PHASE,
    Event::PITCHES,
    Event::STATUS,
    Event::HOMOGENEITY
}
```

Referenced by csound::operator<().

6.29 csound::Exception Class Reference

Base class for C++ exceptions in the Silence system.

```
#include <Exception.hpp>
```

Public Member Functions

- Exception (std::string message_)
- std::string getMessage () const
- virtual ~Exception ()

6.29.1 Detailed Description

Base class for C++ exceptions in the Silence system.

6.29.2 Constructor & Destructor Documentation

6.29.2.1 Exception()

```
csound::Exception::Exception (
            std::string message_ )  [inline]
```

6.29.2.2 ~Exception()

```
virtual csound::Exception::~Exception ( )  [inline], [virtual]
```

6.29.3 Member Function Documentation

6.29.3.1 getMessage()

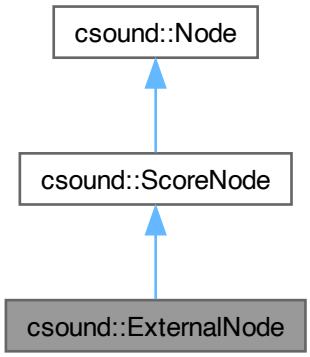
```
std::string csound::Exception::getMessage ( ) const  [inline]
```

6.30 csound::ExternalNode Class Reference

ExternalNode runs a stored script with a specified command line, and imports Csound "i" statements printed by the script to stdout as CsoundAC Event objects in a CsoundAC Score.

```
#include <ExternalNode.hpp>
```

Inheritance diagram for csound::ExternalNode:



Public Member Functions

- virtual void addChild (Node *node)

  Adds an immediate child Node to this.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &collectingScore)

Optionally generate notes into the score.

- virtual void generateLocally ()
- virtual Node * getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual std::string getCommand () const
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual Score & getScore ()
- virtual std::string getScript () const
- virtual void setCommand (std::string command_)
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void setScript (std::string script_)
- virtual void transform (Score &score_from_children)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

Data Fields

- std::vector< Node * > children

  Child Nodes, if any.
- double duration

  If not 0, the score is rescaled to this duration.
- std::string importFilename

Protected Attributes

- std::string command
- Eigen::MatrixXd localCoordinates
- Score score
- std::string script

6.30.1 Detailed Description

ExternalNode runs a stored script with a specified command line, and imports Csound "i" statements printed by the script to stdout as CsoundAC Event objects in a CsoundAC Score.

The format of the "i" statements must be the same as used in CsoundAC's Event::toCsoundIStatement method:

```
p1  Csound instrument number.
p2  Time in seconds from the beginning of the score.
p3  Duration of the note in seconds.
p4  MIDI key number as a real number, may have a fractional part.
p5  MIDI velocity number as a real number.
p6  Spatial location depth (Ambisonic X axis).
p7  Spatial location width (Ambisonic Y axis, stereo pan).
p8  Spatial location height (Ambisonic Z axis).
p9  Audio phase in radians.
p10 Mason number, i.e. a pitch-class set as a sum of powers of 2.
```

Lines of text read from stdout that do not begin with "i " are ignored.

6.30.2 Member Function Documentation

6.30.2.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.30.2.2 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.30.2.3 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.30.2.4 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.30.2.5 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.30.2.6 generate()

```
void csound::ExternalNode::generate (
            Score & score_from_this )  [virtual]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented from csound::ScoreNode.

References csound::fundamentalDomainByPredicate(), generateLocally(), csound::Score::process(), and csound::ScoreNode::score.

6.30.2.7 generateLocally()

```
void csound::ExternalNode::generateLocally ( )  [virtual]
```

References csound::System::debug(), csound::ScoreNode::duration, csound::System::error(), csound::fundamentalDomainByPredicate(), getCommand(), getScript(), csound::System::inform(), csound::parse_line(), csound::ScoreNode::score, csound::Score::setDuration(), and csound::Score::sort().

Referenced by generate(), and main().

6.30.2.8 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.30.2.9 getCommand()

```
std::string csound::ExternalNode::getCommand ( ) const  [virtual]
```

References command.

Referenced by generateLocally().

6.30.2.10 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.30.2.11 getScore()

```
Score & csound::ScoreNode::getScore ( )  [virtual], [inherited]
```

References csound::ScoreNode::score.

Referenced by main().

6.30.2.12 getScript()

```
std::string csound::ExternalNode::getScript ( ) const  [virtual]
```

References script.

Referenced by generateLocally().

6.30.2.13 setCommand()

```
void csound::ExternalNode::setCommand (
            std::string command_ )  [virtual]
```

References command, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.30.2.14 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.30.2.15 setScript()

```
void csound::ExternalNode::setScript (
            std::string script_ )  [virtual]
```

References csound::fundamentalDomainByPredicate(), and script.

Referenced by main().

6.30.2.16 transform()

```
void csound::Node::transform (
            Score & score_from_children )  [virtual], [inherited]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented in csound::Cell, csound::CellRepeat, csound::CellAdd, csound::CellMultiply, csound::CellReflect, csound::CellSelect, csound::CellRemove, csound::CellChord, csound::CellRandom, csound::CellShuffle, csound::CounterpointNode, csound::RemoveDuplicates, csound::Transformer, csound::Random, csound::Rescale, csound::VoiceleadingNode, csound::LispTransformer, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.30.2.17 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord and csound::Node::transform().

6.30.3 Field Documentation

6.30.3.1 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.30.3.2 command

```
std::string csound::ExternalNode::command  [protected]
```

Referenced by getCommand(), and setCommand().

6.30.3.3 duration

```
double csound::ScoreNode::duration  [inherited]
```

If not 0, the score is rescaled to this duration.

Referenced by csound::ScoreNode::generate(), generateLocally(), and csound::Stack::getDuration().

6.30.3.4 importFilename

```
std::string csound::ScoreNode::importFilename  [inherited]
```

Referenced by csound::ScoreNode::generate().

6.30.3.5 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.30.3.6 score

```
Score csound::ScoreNode::score  [protected], [inherited]
```

Referenced by csound::StrangeAttractor::evaluateAttractor(), generate(), csound::ScoreNode::generate(), csound::MCRM::generate(), generateLocally(), csound::ImageToScore2::generateLocally(), csound::Lindenmayer::generateLocally(), csound::Rescale::getRescale(), csound::ScoreNode::getScore(), csound::Lindenmayer::interpret(), csound::MCRM::iterate(), csound::StrangeAttractor::iterate_without_re csound::KMeansMCRM::means_to_notes(), csound::ImageToScore2::pixel_to_event(), csound::StrangeAttractor::render(), csound::Rescale::Rescale(), csound::Rescale::setRescale(), csound::Cell::transform(), csound::Rescale::transform(), csound::CMaskNode::translate_to_silence(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Lindenmayer::updateActual().

6.30.3.7 script

```
std::string csound::ExternalNode::script  [protected]
```

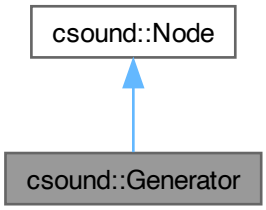
Referenced by getScript(), and setScript().

6.31 csound::Generator Class Reference

Node that uses any callable to implement Node::generate.

```
#include <Node.hpp>
```

Inheritance diagram for csound::Generator:



Public Member Functions

- virtual void addChild (Node *node)

  Adds an immediate child Node to this.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &score)

  Optionally generate notes into the score.
- virtual Node * getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void transform (Score &score_from_children)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

Data Fields

- std::function< void(csound::Score &) callable)
- std::vector< Node * > children

  Child Nodes, if any.

Protected Attributes

- Eigen::MatrixXd localCoordinates

6.31.1 Detailed Description

Node that uses any callable to implement Node::generate.

This is particularly useful as the callable may be a closure that refers to objects outside of the music graph.

6.31.2 Member Function Documentation

6.31.2.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.31.2.2 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.31.2.3 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.31.2.4 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.31.2.5 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.31.2.6 generate()

```
virtual void csound::Generator::generate (
            Score & score_from_this )  [inline], [virtual]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented from csound::Node.

6.31.2.7 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.31.2.8 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.31.2.9 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.31.2.10 transform()

```
void csound::Node::transform (
            Score & score_from_children )  [virtual], [inherited]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented in csound::Cell, csound::CellRepeat, csound::CellAdd, csound::CellMultiply, csound::CellReflect, csound::CellSelect, csound::CellRemove, csound::CellChord, csound::CellRandom, csound::CellShuffle, csound::CounterpointNode, csound::RemoveDuplicates, csound::Transformer, csound::Random, csound::Rescale, csound::VoiceleadingNode, csound::LispTransformer, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.31.2.11 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord and csound::Node::transform().

6.31.3 Field Documentation

6.31.3.1 callable

```
std::function<void(csound::Score &) csound::Generator::callable)
```

6.31.3.2 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.31.3.3 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.32 csound::HarmonyEvent Struct Reference

Associates a Chord with an Event representing a musical note.

```
#include <HarmonyIFS.hpp>
```

Public Member Functions

- const Chord & get_chord () const
- const Event & get_note () const
- void set_chord (const Chord &chord_)
- void set_note (const Event &event)

Data Fields

- Chord chord
- Event note

6.32.1 Detailed Description

Associates a Chord with an Event representing a musical note.

6.32.2 Member Function Documentation

6.32.2.1 get_chord()

```
const Chord & csound::HarmonyEvent::get_chord ( ) const  [inline]
```

References csound::chord().

6.32.2.2 get_note()

```
const Event & csound::HarmonyEvent::get_note ( ) const  [inline]
```

References csound::note().

6.32.2.3 set_chord()

```
void csound::HarmonyEvent::set_chord (
            const Chord & chord_ )  [inline]
```

References csound::chord().

6.32.2.4 set_note()

```
void csound::HarmonyEvent::set_note (
            const Event & event )  [inline]
```

References csound::note().

6.32.3 Field Documentation

6.32.3.1 chord

```
Chord csound::HarmonyEvent::chord
```

6.32.3.2 note

```
Event csound::HarmonyEvent::note
```

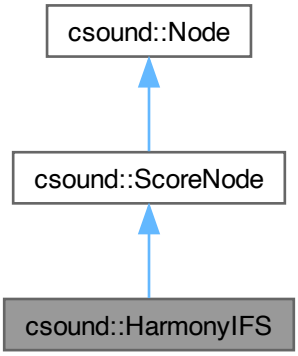
Referenced by csound::HarmonyIFS::point_to_note().

6.33 csound::HarmonyIFS Class Reference

HarmonyIFS is a class for doing algorithmic music composition by means of fractal interpolation functions.

`#include <HarmonyIFS.hpp>`

Inheritance diagram for csound::HarmonyIFS:



Public Member Functions

- virtual HarmonyInterpolationPoint add_interpolation_point (double t, double P, double I, double T, double s_PP, double s_PI, double s_PT, double s_IP, double s_II, double s_IT, double s_TP, double s_TI, double s_TT)

  Adds an interpolation point to the graph of the fractal interpolation function.
- virtual HarmonyInterpolationPoint add_interpolation_point_as_chord (double t_, const Chord &chord, double s_PP_, double s_PI_, double s_PT_, double s_IP_, double s_II_, double s_IT_, double s_TP_, double s_TI_, double s_TT_)

  Adds an interpolation point to the graph of the fractal interpolation function.
- virtual Eigen::MatrixXd & add_transformation ()

  Adds a new affine transformation matrix to the Hutchinson operator.
- virtual void addChild (Node *node)

  Adds an immediate child Node to this.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &collectingScore)

Optionally generate notes into the score.

- virtual void generate_score_attractor (int depth)

  Recursively computes the score graph, translates the points to notes, adds them to the score, ties overlapping notes in the score, and rescales the score.
- virtual Node * getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual Score & getScore ()
- HarmonyIFS ()
- virtual void initialize (int voices_, double range_, double bass_, double note_duration_, bool tie_overlapping_notes, bool remove_duplicate_notes, double g_=1.)

  Initialize the HarmonyIFS for N voices in a range of MIDI keys for a note duration in seconds.
- virtual void initialize_hutchinson_operator ()

  Interpolation points are sorted by time and the corresponding shear transformations for a Hutchinson operator are computed, according to Polychronis Manousopoulos, Vasileios Drakopoulos, and Theoharis Theoharis, "Curve Fitting by Fractal Interpolation.
- virtual void iterate (int depth, int iteration, int index, const HarmonyPoint point)

  Actually computes the score attractor.
- virtual PITV & pitv ()
- virtual HarmonyEvent point_to_note (const HarmonyPoint &point)

  Translates a point in the attractor of the IFS to a note and associated chord.
- virtual void remove_duplicate_notes ()

  Removes duplicate notes from the generated score.
- virtual void set_rotation (int transformation, int dimension1, int dimension2, double degrees)

  Creates a rotation in one plane in one of the affine affine transformation matrices of the Hutchinson operator.
- virtual void set_scaling (int transformation, int dimension, double value)

  Creates a scaling transformation in one of the affine transformation matrices of the Hutchinson operator.
- virtual void set_shear (int transformation, int dimension, double value)

  Creates a shear transformation parallel to one non-time axis in one of the affine transformation matrices of the Hutchinson operator.
- virtual void set_transformation (int transformation, int row, int column, double value)

  Sets the value of a single matrix element in one of the affine transformation matrices of the Hutchinson operator.
- virtual void set_translation (int transformation, int dimension, double value)

  Creates a translation transformation in one of the affine affine transformation matrices of the Hutchinson operator.
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void tie_overlapping_notes ()
- virtual void transform (Score &score_from_children)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual int transformation_count () const

  Returns the number of affine transformation matrices in the Hutchinson operator of the function system that generates the score.
- virtual void translate_score_attractor_to_score ()

  Processes the score attractor (the raw notes in the score) to quantize and rescale certain dimensions, to remove duplicate notes, and to conform pitches to chords.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.
- virtual ~HarmonyIFS ()

Data Fields

- double bass
- std::vector< Node * > children

  Child Nodes, if any.
- double duration

  If not 0, the score is rescaled to this duration.
- double g
- std::vector< Eigen::MatrixXd > hutchinson_operator
- std::string importFilename
- std::vector< HarmonyInterpolationPoint > interpolation_points
- double note_duration
- PITV pitv_
- double range
- bool remove_duplicates
- std::vector< HarmonyEvent > score_attractor
- bool tie_overlaps
- int voices

Protected Attributes

- Eigen::MatrixXd localCoordinates
- Score score

6.33.1 Detailed Description

HarmonyIFS is a class for doing algorithmic music composition by means of fractal interpolation functions.

Scores are generated as the attractors of iterated function systems (IFS) in a score space that has a harmony subspace, in which time is subdivided such that harmony is a linear progression of time.

Usage:

1. Call `add_interpolation_point` several times or more to define the harmony by setting interpolation points for a harmony as a fractal function of time.
2. Call `initialize_hutchinson_operator` to mathematically translate the interpolation points to affine transformation matrices in a Hutchinson operator.
3. Call `set_transformation` as desired to add additional structure to the IFS that generates the score. Do not set matrix elements that will cause harmony to overlap time other than as specified by the interpolation points. However, pitch, time, instrument, and other dimensions may be transformed as desired.
4. Call `generate_score_attractor` with a desired depth of iteration actually generate the score. The HarmonyIFS object may then be included in a music graph, or used as a standalone score generator.

6.33.2 Constructor & Destructor Documentation

6.33.2.1 HarmonyIFS()

```
csound::HarmonyIFS::HarmonyIFS ( )  [inline]
```

6.33.2.2 ~HarmonyIFS()

```
virtual csound::HarmonyIFS::~HarmonyIFS ( )  [inline], [virtual]
```

6.33.3 Member Function Documentation

6.33.3.1 add_interpolation_point()

```
virtual HarmonyInterpolationPoint csound::HarmonyIFS::add_interpolation_point (
            double t,
            double P,
            double I,
            double T,
            double s_PP,
            double s_PI,
            double s_PT,
            double s_IP,
            double s_II,
            double s_IT,
            double s_TP,
            double s_TI,
            double s_TT )  [inline], [virtual]
```

Adds an interpolation point to the graph of the fractal interpolation function.

References csound::I(), csound::T(), and csound::HarmonyInterpolationPoint::toString().

6.33.3.2 add_interpolation_point_as_chord()

```
virtual HarmonyInterpolationPoint csound::HarmonyIFS::add_interpolation_point_as_chord (
            double t_,
            const Chord & chord,
            double s_PP_,
            double s_PI_,
            double s_PT_,
            double s_IP_,
            double s_II_,
            double s_IT_,
            double s_TP_,
            double s_TI_,
            double s_TT_ )  [inline], [virtual]
```

Adds an interpolation point to the graph of the fractal interpolation function.

References csound::chord(), and csound::Chord::toString().

Referenced by main().

6.33.3.3 add_transformation()

```
virtual Eigen::MatrixXd & csound::HarmonyIFS::add_transformation ( )  [inline], [virtual]
```

Adds a new affine transformation matrix to the Hutchinson operator.

The value of this matrix is initially the identity matrix.

6.33.3.4 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.33.3.5 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.33.3.6 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.33.3.7 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.33.3.8 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.33.3.9 generate()

```
void csound::ScoreNode::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented from csound::Node.

Reimplemented in csound::ExternalNode, and csound::MCRM.

References csound::ScoreNode::duration, csound::fundamentalDomainByPredicate(), csound::Score::getCsoundScoreHeader(), csound::ScoreNode::importFilename, csound::Score::load(), csound::Score::process(), csound::ScoreNode::score, csound::Score::setDuration(), and csound::Score::sort().

Referenced by csound::MCRM::generate().

6.33.3.10 generate_score_attractor()

```
virtual void csound::HarmonyIFS::generate_score_attractor (
            int depth )  [inline], [virtual]
```

Recursively computes the score graph, translates the points to notes, adds them to the score, ties overlapping notes in the score, and rescales the score.

This function should be called **before** rendering a music graph that contains this node.

References csound::HarmonyPoint::set_homogeneity(), csound::HarmonyPoint::set_i(), csound::HarmonyPoint::set_k(), csound::HarmonyPoint::set_v(), and csound::toString().

Referenced by main().

6.33.3.11 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.33.3.12 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.33.3.13 getScore()

```
Score & csound::ScoreNode::getScore ( )  [virtual], [inherited]
```

References csound::ScoreNode::score.

Referenced by main().

6.33.3.14 initialize()

```
virtual void csound::HarmonyIFS::initialize (
            int voices_,
            double range_,
            double bass_,
            double note_duration_,
            bool tie_overlapping_notes,
            bool remove_duplicate_notes,
            double g_ = 1.  )  [inline], [virtual]
```

Initialize the HarmonyIFS for N voices in a range of MIDI keys for a note duration in seconds.

g is the generator of transposition.

Referenced by main().

6.33.3.15 initialize_hutchinson_operator()

```
virtual void csound::HarmonyIFS::initialize_hutchinson_operator ( )  [inline], [virtual]
```

Interpolation points are sorted by time and the corresponding shear transformations for a Hutchinson operator are computed, according to Polychronis Manousopoulos, Vasileios Drakopoulos, and Theoharis Theoharis, "Curve Fitting by Fractal Interpolation.

" In: Transactions on Computational Science 1 (Jan. 2008), pp. 85-103. doi: 10.1007/978-3-540-79299-4_4.

Once this function has been called, the non-shear elements of the transformation matrices may be modified. A warning is issued if the modulus of the scaling submatrix of any transformation is >= 0, indicating it is not contractive.

References csound::ge_tolerance(), csound::HarmonyInterpolationPoint::I, csound::interpolation_point_less(), csound::HarmonyInterpolationPoint::P, csound::HarmonyInterpolationPoint::s_II, csound::HarmonyInterpolationPoint::s_IP, csound::HarmonyInterpolationPoint::s_IT, csound::HarmonyInterpolationPoint::s_PI, csound::HarmonyInterpolationPoint::s_PP, csound::HarmonyInterpolationPoint::s_PT, csound::HarmonyInterpolationPoint::s_TI, csound::HarmonyInterpolationPoint::s_TP, csound::HarmonyInterpolationPoint::s_TT, csound::HarmonyInterpolationPoint::t, and csound::HarmonyInterpolationPoint::T.

Referenced by main().

6.33.3.16 iterate()

```
virtual void csound::HarmonyIFS::iterate (
            int depth,
            int iteration,
            int index,
            const HarmonyPoint point )  [inline], [virtual]
```

Actually computes the score attractor.

References csound::T(), and csound::toString().

6.33.3.17 pitv()

```
virtual PITV & csound::HarmonyIFS::pitv ( )  [inline], [virtual]
```

6.33.3.18 point_to_note()

```
virtual HarmonyEvent csound::HarmonyIFS::point_to_note (
            const HarmonyPoint & point )  [inline], [virtual]
```

Translates a point in the attractor of the IFS to a note and associated chord.

References csound::HarmonyPoint::I(), csound::HarmonyPoint::i(), csound::I(), csound::HarmonyPoint::k(), csound::HarmonyEvent::note, csound::HarmonyPoint::P(), csound::Event::setTime(), csound::HarmonyPoint::t(), csound::HarmonyPoint::T(), csound::T(), and csound::HarmonyPoint::v().

6.33.3.19 remove_duplicate_notes()

```
virtual void csound::HarmonyIFS::remove_duplicate_notes ( )  [inline], [virtual]
```

Removes duplicate notes from the generated score.

6.33.3.20 set_rotation()

```
virtual void csound::HarmonyIFS::set_rotation (
            int transformation,
            int dimension1,
            int dimension2,
            double degrees )  [inline], [virtual]
```

Creates a rotation in one plane in one of the affine affine transformation matrices of the Hutchinson operator.

6.33.3.21 set_scaling()

```
virtual void csound::HarmonyIFS::set_scaling (
            int transformation,
            int dimension,
            double value )  [inline], [virtual]
```

Creates a scaling transformation in one of the affine transformation matrices of the Hutchinson operator.

6.33.3.22 set_shear()

```
virtual void csound::HarmonyIFS::set_shear (
            int transformation,
            int dimension,
            double value )  [inline], [virtual]
```

Creates a shear transformation parallel to one non-time axis in one of the affine transformation matrices of the Hutchinson operator.

6.33.3.23 set_transformation()

```
virtual void csound::HarmonyIFS::set_transformation (
            int transformation,
            int row,
            int column,
            double value )  [inline], [virtual]
```

Sets the value of a single matrix element in one of the affine transformation matrices of the Hutchinson operator.

The matrices are homeogenous transformations with 7 dimensions, in column major order. The transformation is by default the identity matrix.

Referenced by main().

6.33.3.24 set_translation()

```
virtual void csound::HarmonyIFS::set_translation (
            int transformation,
            int dimension,
            double value )  [inline], [virtual]
```

Creates a translation transformation in one of the affine affine transformation matrices of the Hutchinson operator.

6.33.3.25 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.33.3.26 tie_overlapping_notes()

```
virtual void csound::HarmonyIFS::tie_overlapping_notes ( )  [inline], [virtual]
```

6.33.3.27 transform()

```
void csound::Node::transform (
            Score & score_from_children )  [virtual], [inherited]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented in csound::Cell, csound::CellRepeat, csound::CellAdd, csound::CellMultiply, csound::CellReflect, csound::CellSelect, csound::CellRemove, csound::CellChord, csound::CellRandom, csound::CellShuffle, csound::CounterpointNode, csound::RemoveDuplicates, csound::Transformer, csound::Random, csound::Rescale, csound::VoiceleadingNode, csound::LispTransformer, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.33.3.28 transformation_count()

```
virtual int csound::HarmonyIFS::transformation_count ( ) const  [inline], [virtual]
```

Returns the number of affine transformation matrices in the Hutchinson operator of the function system that generates the score.

6.33.3.29 translate_score_attractor_to_score()

```
virtual void csound::HarmonyIFS::translate_score_attractor_to_score ( )  [inline], [virtual]
```

Processes the score attractor (the raw notes in the score) to quantize and rescale certain dimensions, to remove duplicate notes, and to conform pitches to chords.

References csound::conformToChord().

6.33.3.30 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord and csound::Node::transform().

6.33.4 Field Documentation

6.33.4.1 bass

```
double csound::HarmonyIFS::bass
```

6.33.4.2 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.33.4.3 duration

```
double csound::ScoreNode::duration  [inherited]
```

If not 0, the score is rescaled to this duration.

Referenced by csound::ScoreNode::generate(), csound::ExternalNode::generateLocally(), and csound::Stack::getDuration().

6.33.4.4 g

```
double csound::HarmonyIFS::g
```

6.33.4.5 hutchinson_operator

```
std::vector<Eigen::MatrixXd> csound::HarmonyIFS::hutchinson_operator
```

6.33.4.6 importFilename

```
std::string csound::ScoreNode::importFilename  [inherited]
```

Referenced by csound::ScoreNode::generate().

6.33.4.7 interpolation_points

```
std::vector<HarmonyInterpolationPoint> csound::HarmonyIFS::interpolation_points
```

6.33.4.8 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.33.4.9 note_duration

```
double csound::HarmonyIFS::note_duration
```

6.33.4.10 pitv_

```
PITV csound::HarmonyIFS::pitv_
```

6.33.4.11 range

```
double csound::HarmonyIFS::range
```

6.33.4.12 remove_duplicates

```
bool csound::HarmonyIFS::remove_duplicates
```

6.33.4.13 score

```
Score csound::ScoreNode::score  [protected], [inherited]
```

Referenced by csound::StrangeAttractor::evaluateAttractor(), csound::ExternalNode::generate(), csound::ScoreNode::generate(), csound::MCRM::generate(), csound::ExternalNode::generateLocally(), csound::ImageToScore2::generateLocally(), csound::Lindenmayer::generateLocally(), csound::Rescale::getRescale(), csound::ScoreNode::getScore(), csound::Lindenmayer::interpre csound::MCRM::iterate(), csound::StrangeAttractor::iterate_without_rendering(), csound::KMeansMCRM::means_to_notes(), csound::ImageToScore2::pixel_to_event(), csound::StrangeAttractor::render(), csound::Rescale::Rescale(), csound::Rescale::setRescale(csound::Cell::transform(), csound::Rescale::transform(), csound::CMaskNode::translate_to_silence(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Lindenmayer::updateActual().

6.33.4.14 score_attractor

```
std::vector<HarmonyEvent> csound::HarmonyIFS::score_attractor
```

6.33.4.15 tie_overlaps

```
bool csound::HarmonyIFS::tie_overlaps
```

6.33.4.16 voices

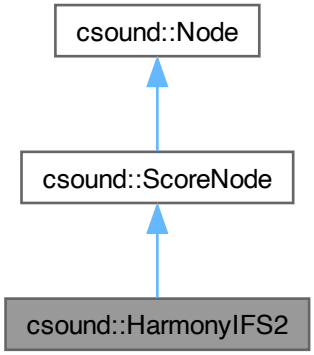
```
int csound::HarmonyIFS::voices
```

6.34 csound::HarmonyIFS2 Class Reference

HarmonyIFS is a class for doing algorithmic music composition by means of fractal interpolation functions.

```
#include <HarmonyIFS2.hpp>
```

Inheritance diagram for csound::HarmonyIFS2:



Public Member Functions

- virtual HarmonyInterpolationPoint2 add_interpolation_point (double t, double P, double I, double T, double V, double s_PP, double s_PI, double s_PT, double s_PV, double s_IP, double s_II, double s_IT, double s_IV, double s_TP, double s_TI, double s_TT, double s_TV, double s_VP, double s_VI, double s_VT, double s_VV)

  Adds an interpolation point to the graph of the fractal interpolation function.
- virtual HarmonyInterpolationPoint2 add_interpolation_point_as_chord (double t_, const Chord &chord, double s_PP_, double s_PI_, double s_PT_, double s_PV_, double s_IP_, double s_II_, double s_IT_, double s_IV_, double s_TP_, double s_TI_, double s_TT_, double s_TV_, double s_VP_, double s_VI_, double s_VT_, double s_VV_)

  Adds an interpolation point to the graph of the fractal interpolation function.
- virtual Eigen::MatrixXd & add_transformation ()

  Adds a new affine transformation matrix to the Hutchinson operator.
- virtual void addChild (Node *node)

  Adds an immediate child Node to this.
- virtual size_t childCount () const

  Returns the number of immediate children of this.

- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &collectingScore)

  Optionally generate notes into the score.
- virtual void generate_score_attractor (int depth)

  Recursively computes the score graph, translates the points to notes, adds them to the score, ties overlapping notes in the score, and rescales the score.
- virtual Node * getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual Score & getScore ()
- HarmonyIFS2 ()
- virtual void initialize (int voices_, double range_, double bass_, double note_duration_, bool tie_overlapping_notes, bool remove_duplicate_notes, double g_=1.)

  Initialize the HarmonyIFS for N voices in a range of MIDI keys for a note duration in seconds.
- virtual void initialize_hutchinson_operator ()

  Interpolation points are sorted by time and the corresponding shear transformations for a Hutchinson operator are computed, according to Polychronis Manousopoulos, Vasileios Drakopoulos, and Theoharis Theoharis, "Curve Fitting by Fractal Interpolation.
- virtual void iterate (int depth, int iteration, int index, const HarmonyPoint2 point)

  Actually computes the score attractor.
- virtual PITV & pitv ()
- virtual void post_process_score ()

  Processes the score attractor (the raw notes in the score) to quantize and rescale certain dimensions, and to remove duplicate notes.
- virtual void remove_duplicate_notes ()

  Removes duplicate notes from the generated score.
- virtual void set_rotation (int transformation, int dimension1, int dimension2, double degrees)

  Creates a rotation in one plane in one of the affine affine transformation matrices of the Hutchinson operator.
- virtual void set_scaling (int transformation, int dimension, double value)

  Creates a scaling transformation in one of the affine transformation matrices of the Hutchinson operator.
- virtual void set_shear (int transformation, int dimension, double value)

  Creates a shear transformation parallel to one non-time axis in one of the affine transformation matrices of the Hutchinson operator.
- virtual void set_transformation (int transformation, int row, int column, double value)

  Sets the value of a single matrix element in one of the affine transformation matrices of the Hutchinson operator.
- virtual void set_translation (int transformation, int dimension, double value)

  Creates a translation transformation in one of the affine affine transformation matrices of the Hutchinson operator.
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void tie_overlapping_notes ()

  Notes in the generated chords have a nominal duration.
- virtual void transform (Score &score_from_children)

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

- virtual int transformation_count () const

  Returns the number of affine transformation matrices in the Hutchinson operator of the function system that generates the score.

- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

- virtual ~HarmonyIFS2 ()

Data Fields

- double bass
- std::vector< Node * > children

  Child Nodes, if any.

- double duration

  If not 0, the score is rescaled to this duration.

- double g
- std::vector< Eigen::MatrixXd > hutchinson_operator
- std::string importFilename
- std::vector< HarmonyInterpolationPoint2 > interpolation_points
- double note_duration
- PITV pitv_
- double range
- bool remove_duplicates
- bool tie_overlaps
- int voices

Protected Attributes

- Eigen::MatrixXd localCoordinates
- Score score

6.34.1 Detailed Description

HarmonyIFS is a class for doing algorithmic music composition by means of fractal interpolation functions.

Scores are generated as the attractors of iterated function systems (IFS) in a score space that has a harmony subspace, in which time is subdivided such that harmony is a linear progression of time.

Usage:

1. Call `add_interpolation_point` several times or more to define the harmony by setting interpolation points for a harmony as a fractal function of time.
2. Call `initialize_hutchinson_operator` to mathematically translate the interpolation points to affine transformation matrices in a Hutchinson operator.
3. Call `set_transformation` as desired to add additional structure to the IFS that generates the score. Do not set matrix elements that will cause harmony to overlap time other than as specified by the interpolation points. However, pitch, time, instrument, and other dimensions may be transformed as desired.
4. Call `generate_score_attractor` with a desired depth of iteration actually generate the score. The HarmonyIFS object may then be included in a music graph, or used as a standalone score generator.

6.34.2 Constructor & Destructor Documentation

6.34.2.1 HarmonyIFS2()

```
csound::HarmonyIFS2::HarmonyIFS2 ( )  [inline]
```

6.34.2.2 ∼HarmonyIFS2()

```
virtual csound::HarmonyIFS2::∼HarmonyIFS2 ( )  [inline], [virtual]
```

6.34.3 Member Function Documentation

6.34.3.1 add_interpolation_point()

```
virtual HarmonyInterpolationPoint2 csound::HarmonyIFS2::add_interpolation_point (
            double t,
            double P,
            double I,
            double T,
            double V,
            double s_PP,
            double s_PI,
            double s_PT,
            double s_PV,
            double s_IP,
            double s_II,
            double s_IT,
            double s_IV,
            double s_TP,
            double s_TI,
            double s_TT,
            double s_TV,
            double s_VP,
            double s_VI,
            double s_VT,
            double s_VV )  [inline], [virtual]
```

Adds an interpolation point to the graph of the fractal interpolation function.

References csound::I(), csound::T(), and csound::HarmonyInterpolationPoint2::toString().

6.34.3.2 add_interpolation_point_as_chord()

```
virtual HarmonyInterpolationPoint2 csound::HarmonyIFS2::add_interpolation_point_as_chord (
            double t_,
            const Chord & chord,
            double s_PP_,
            double s_PI_,
            double s_PT_,
            double s_PV_,
            double s_IP_,
            double s_II_,
            double s_IT_,
            double s_IV_,
            double s_TP_,
            double s_TI_,
            double s_TT_,
            double s_TV_,
            double s_VP_,
            double s_VI_,
            double s_VT_,
            double s_VV_ )  [inline], [virtual]
```

Adds an interpolation point to the graph of the fractal interpolation function.

References csound::chord(), and csound::Chord::toString().

Referenced by main().

6.34.3.3 add_transformation()

```
virtual Eigen::MatrixXd & csound::HarmonyIFS2::add_transformation ( )  [inline], [virtual]
```

Adds a new affine transformation matrix to the Hutchinson operator.

The value of this matrix is initially the identity matrix.

6.34.3.4 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.34.3.5 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.34.3.6 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.34.3.7 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.34.3.8 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.34.3.9 generate()

```
void csound::ScoreNode::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented from csound::Node.

Reimplemented in csound::ExternalNode, and csound::MCRM.

References csound::ScoreNode::duration, csound::fundamentalDomainByPredicate(), csound::Score::getCsoundScoreHeader(), csound::ScoreNode::importFilename, csound::Score::load(), csound::Score::process(), csound::ScoreNode::score, csound::Score::setDuration(), and csound::Score::sort().

Referenced by csound::MCRM::generate().

6.34.3.10 generate_score_attractor()

```
virtual void csound::HarmonyIFS2::generate_score_attractor (
            int depth )  [inline], [virtual]
```

Recursively computes the score graph, translates the points to notes, adds them to the score, ties overlapping notes in the score, and rescales the score.

This function should be called **before** rendering a music graph that contains this node.

References csound::HarmonyPoint2::set_homogeneity(), and csound::toString().

Referenced by main().

6.34.3.11 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.34.3.12 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.34.3.13 getScore()

```
Score & csound::ScoreNode::getScore ( )  [virtual], [inherited]
```

References csound::ScoreNode::score.

Referenced by main().

6.34.3.14 initialize()

```
virtual void csound::HarmonyIFS2::initialize (
            int voices_,
            double range_,
            double bass_,
            double note_duration_,
            bool tie_overlapping_notes,
            bool remove_duplicate_notes,
            double g_ = 1.  )  [inline], [virtual]
```

Initialize the HarmonyIFS for N voices in a range of MIDI keys for a note duration in seconds.

g is the generator of transposition.

Referenced by main().

6.34.3.15 initialize_hutchinson_operator()

```
virtual void csound::HarmonyIFS2::initialize_hutchinson_operator ( )  [inline], [virtual]
```

Interpolation points are sorted by time and the corresponding shear transformations for a Hutchinson operator are computed, according to Polychronis Manousopoulos, Vasileios Drakopoulos, and Theoharis Theoharis, "Curve Fitting by Fractal Interpolation.

" In: Transactions on Computational Science 1 (Jan. 2008), pp. 85-103. doi: 10.1007/978-3-540-79299-4_4.

Once this function has been called, the non-shear elements of the transformation matrices may be modified. A warning is issued if the modulus of the scaling submatrix of any transformation is >= 0, indicating it is not contractive.

References csound::ge_tolerance(), csound::HarmonyInterpolationPoint2::I, csound::interpolation_point_less2(), csound::HarmonyInterpolationPoint2::P, csound::HarmonyInterpolationPoint2::s_II, csound::HarmonyInterpolationPoint2::s_IP, csound::HarmonyInterpolationPoint2::s_IT, csound::HarmonyInterpolationPoint2::s_IV, csound::HarmonyInterpolationPoint2::s_PI, csound::HarmonyInterpolationPoint2::s_PP, csound::HarmonyInterpolationPoint2::s_PT, csound::HarmonyInterpolationPoint2::s_PV, csound::HarmonyInterpolationPoint2::s_TI, csound::HarmonyInterpolationPoint2::s_TP, csound::HarmonyInterpolationPoint2::s_TT, csound::HarmonyInterpolationPoint2::s_TV, csound::HarmonyInterpolationPoint2::s_VI, csound::HarmonyInterpolationPoint2::s_VP, csound::HarmonyInterpolationPoint2::s_VT, csound::HarmonyInterpolationPoint2::s_VV, csound::HarmonyInterpolationPoint2::t, csound::HarmonyInterpolationPoint2::T, and csound::HarmonyInterpolationPoint2::V.

Referenced by main().

6.34.3.16 iterate()

```
virtual void csound::HarmonyIFS2::iterate (
            int depth,
            int iteration,
            int index,
            const HarmonyPoint2 point )  [inline], [virtual]
```

Actually computes the score attractor.

References csound::chord(), csound::HarmonyPoint2::I(), csound::I(), csound::HarmonyPoint2::P(), csound::HarmonyPoint2::t(), csound::HarmonyPoint2::T(), csound::T(), csound::toScore(), csound::Chord::toString(), csound::HarmonyPoint2::toString(), and csound::HarmonyPoint2::V().

6.34.3.17 pitv()

```
virtual PITV & csound::HarmonyIFS2::pitv ( )  [inline], [virtual]
```

6.34.3.18 post_process_score()

```
virtual void csound::HarmonyIFS2::post_process_score ( )  [inline], [virtual]
```

Processes the score attractor (the raw notes in the score) to quantize and rescale certain dimensions, and to remove duplicate notes.

6.34.3.19 remove_duplicate_notes()

```
virtual void csound::HarmonyIFS2::remove_duplicate_notes ( )  [inline], [virtual]
```

Removes duplicate notes from the generated score.

6.34.3.20 set_rotation()

```
virtual void csound::HarmonyIFS2::set_rotation (
            int transformation,
            int dimension1,
            int dimension2,
            double degrees )  [inline], [virtual]
```

Creates a rotation in one plane in one of the affine affine transformation matrices of the Hutchinson operator.

6.34.3.21 set_scaling()

```
virtual void csound::HarmonyIFS2::set_scaling (
            int transformation,
            int dimension,
            double value )  [inline], [virtual]
```

Creates a scaling transformation in one of the affine transformation matrices of the Hutchinson operator.

6.34.3.22 set_shear()

```
virtual void csound::HarmonyIFS2::set_shear (
            int transformation,
            int dimension,
            double value )  [inline], [virtual]
```

Creates a shear transformation parallel to one non-time axis in one of the affine transformation matrices of the Hutchinson operator.

6.34.3.23 set_transformation()

```
virtual void csound::HarmonyIFS2::set_transformation (
            int transformation,
            int row,
            int column,
            double value )  [inline], [virtual]
```

Sets the value of a single matrix element in one of the affine transformation matrices of the Hutchinson operator.

The matrices are homeogenous transformations with 7 dimensions, in column major order. The transformation is by default the identity matrix.

6.34.3.24 set_translation()

```
virtual void csound::HarmonyIFS2::set_translation (
            int transformation,
            int dimension,
            double value )  [inline], [virtual]
```

Creates a translation transformation in one of the affine affine transformation matrices of the Hutchinson operator.

6.34.3.25 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.34.3.26 tie_overlapping_notes()

```
virtual void csound::HarmonyIFS2::tie_overlapping_notes ( )  [inline], [virtual]
```

Notes in the generated chords have a nominal duration.

Notes on the same voice and key that overlap are tied (merged).

6.34.3.27 transform()

```
void csound::Node::transform (
            Score & score_from_children )  [virtual], [inherited]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented in csound::Cell, csound::CellRepeat, csound::CellAdd, csound::CellMultiply, csound::CellReflect, csound::CellSelect, csound::CellRemove, csound::CellChord, csound::CellRandom, csound::CellShuffle, csound::CounterpointNode, csound::RemoveDuplicates, csound::Transformer, csound::Random, csound::Rescale, csound::VoiceleadingNode, csound::LispTransformer, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.34.3.28 transformation_count()

```
virtual int csound::HarmonyIFS2::transformation_count ( ) const  [inline], [virtual]
```

Returns the number of affine transformation matrices in the Hutchinson operator of the function system that generates the score.

6.34.3.29 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord and csound::Node::transform().

6.34.4 Field Documentation

6.34.4.1 bass

```
double csound::HarmonyIFS2::bass
```

6.34.4.2 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.34.4.3 duration

```
double csound::ScoreNode::duration  [inherited]
```

If not 0, the score is rescaled to this duration.

Referenced by csound::ScoreNode::generate(), csound::ExternalNode::generateLocally(), and csound::Stack::getDuration().

6.34.4.4 g

```
double csound::HarmonyIFS2::g
```

6.34.4.5 hutchinson_operator

```
std::vector<Eigen::MatrixXd> csound::HarmonyIFS2::hutchinson_operator
```

6.34.4.6 importFilename

```
std::string csound::ScoreNode::importFilename  [inherited]
```

Referenced by csound::ScoreNode::generate().

6.34.4.7 interpolation_points

```
std::vector<HarmonyInterpolationPoint2> csound::HarmonyIFS2::interpolation_points
```

6.34.4.8 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.34.4.9 note_duration

```
double csound::HarmonyIFS2::note_duration
```

6.34.4.10 pitv_

```
PITV csound::HarmonyIFS2::pitv_
```

6.34.4.11 range

```
double csound::HarmonyIFS2::range
```

6.34.4.12 remove_duplicates

```
bool csound::HarmonyIFS2::remove_duplicates
```

6.34.4.13 score

```
Score csound::ScoreNode::score  [protected], [inherited]
```

Referenced by csound::StrangeAttractor::evaluateAttractor(), csound::ExternalNode::generate(), csound::ScoreNode::generate(), csound::MCRM::generate(), csound::ExternalNode::generateLocally(), csound::ImageToScore2::generateLocally(), csound::Lindenmayer::generateLocally(), csound::Rescale::getRescale(), csound::ScoreNode::getScore(), csound::Lindenmayer::interpre csound::MCRM::iterate(), csound::StrangeAttractor::iterate_without_rendering(), csound::KMeansMCRM::means_to_notes(), csound::ImageToScore2::pixel_to_event(), csound::StrangeAttractor::render(), csound::Rescale::Rescale(), csound::Rescale::setRescale(csound::Cell::transform(), csound::Rescale::transform(), csound::CMaskNode::translate_to_silence(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Lindenmayer::updateActual().

6.34.4.14 tie_overlaps

```
bool csound::HarmonyIFS2::tie_overlaps
```

6.34.4.15 voices

```
int csound::HarmonyIFS2::voices
```

6.35 csound::HarmonyInterpolationPoint Class Reference

Represents an interpolation point with scaling factors for a fractal interpolation function in the **time-harmony subspace** of the score space.

`#include <HarmonyIFS.hpp>`

Public Member Functions

- HarmonyInterpolationPoint ()
- HarmonyInterpolationPoint (const HarmonyInterpolationPoint &other)
- HarmonyInterpolationPoint (double t_, double P_, double I_, double T_, double s_PP_, double s_PI_, double s_PT_, double s_IP_, double s_II_, double s_IT_, double s_TP_, double s_TI_, double s_TT_)
- virtual std::string toString () const
- virtual ~HarmonyInterpolationPoint ()

Data Fields

- double I
- double P
- double s_II
- double s_IP
- double s_IT
- double s_PI
- double s_PP
- double s_PT
- double s_TI
- double s_TP
- double s_TT
- double t
- double T

6.35.1 Detailed Description

Represents an interpolation point with scaling factors for a fractal interpolation function in the **time-harmony subspace** of the score space.

6.35.2 Constructor & Destructor Documentation

6.35.2.1 HarmonyInterpolationPoint() [1/3]

`csound::HarmonyInterpolationPoint::HarmonyInterpolationPoint ( ) [inline]`

6.35.2.2 HarmonyInterpolationPoint() [2/3]

```
csound::HarmonyInterpolationPoint::HarmonyInterpolationPoint (
            const HarmonyInterpolationPoint & other )  [inline]
```

6.35.2.3 HarmonyInterpolationPoint() [3/3]

```
csound::HarmonyInterpolationPoint::HarmonyInterpolationPoint (
            double t_,
            double P_,
            double I_,
            double T_,
            double s_PP_,
            double s_PI_,
            double s_PT_,
            double s_IP_,
            double s_II_,
            double s_IT_,
            double s_TP_,
            double s_TI_,
            double s_TT_ )  [inline]
```

References csound::I(), and csound::T().

6.35.2.4 ~HarmonyInterpolationPoint()

```
virtual csound::HarmonyInterpolationPoint::~HarmonyInterpolationPoint ( )  [inline], [virtual]
```

6.35.3 Member Function Documentation

6.35.3.1 toString()

```
virtual std::string csound::HarmonyInterpolationPoint::toString ( ) const  [inline], [virtual]
```

References csound::I(), and csound::T().

Referenced by csound::HarmonyIFS::add_interpolation_point().

6.35.4 Field Documentation

6.35.4.1 I

```
double csound::HarmonyInterpolationPoint::I
```

Referenced by csound::HarmonyIFS::initialize_hutchinson_operator().

6.35.4.2 P

```
double csound::HarmonyInterpolationPoint::P
```

Referenced by csound::HarmonyIFS::initialize_hutchinson_operator().

6.35.4.3 s_II

```
double csound::HarmonyInterpolationPoint::s_II
```

Referenced by csound::HarmonyIFS::initialize_hutchinson_operator().

6.35.4.4 s_IP

```
double csound::HarmonyInterpolationPoint::s_IP
```

Referenced by csound::HarmonyIFS::initialize_hutchinson_operator().

6.35.4.5 s_IT

```
double csound::HarmonyInterpolationPoint::s_IT
```

Referenced by csound::HarmonyIFS::initialize_hutchinson_operator().

6.35.4.6 s_PI

```
double csound::HarmonyInterpolationPoint::s_PI
```

Referenced by csound::HarmonyIFS::initialize_hutchinson_operator().

6.35.4.7 s_PP

```
double csound::HarmonyInterpolationPoint::s_PP
```

Referenced by csound::HarmonyIFS::initialize_hutchinson_operator().

6.35.4.8 s_PT

```
double csound::HarmonyInterpolationPoint::s_PT
```

Referenced by csound::HarmonyIFS::initialize_hutchinson_operator().

6.35.4.9 s_TI

```
double csound::HarmonyInterpolationPoint::s_TI
```

Referenced by csound::HarmonyIFS::initialize_hutchinson_operator().

6.35.4.10 s_TP

```
double csound::HarmonyInterpolationPoint::s_TP
```

Referenced by csound::HarmonyIFS::initialize_hutchinson_operator().

6.35.4.11 s_TT

```
double csound::HarmonyInterpolationPoint::s_TT
```

Referenced by csound::HarmonyIFS::initialize_hutchinson_operator().

6.35.4.12 t

```
double csound::HarmonyInterpolationPoint::t
```

Referenced by csound::HarmonyIFS::initialize_hutchinson_operator(), and csound::interpolation_point_less().

6.35.4.13 T

```
double csound::HarmonyInterpolationPoint::T
```

Referenced by csound::HarmonyIFS::initialize_hutchinson_operator().

6.36 csound::HarmonyInterpolationPoint2 Class Reference

Represents an interpolation point with scaling factors for a fractal interpolation function in the **time-harmony subspace** of the score space.

```
#include <HarmonyIFS2.hpp>
```

Public Member Functions

- HarmonyInterpolationPoint2 ()
- HarmonyInterpolationPoint2 (const HarmonyInterpolationPoint2 &other)
- HarmonyInterpolationPoint2 (double t_, double P_, double I_, double T_, double V_, double s_PP_, double s_PI_, double s_PT_, double s_PV_, double s_IP_, double s_II_, double s_IT_, double s_IV_, double s_TP_, double s_TI_, double s_TT_, double s_TV_, double s_VP_, double s_VI_, double s_VT_, double s_VV_)
- virtual std::string toString () const
- virtual ~HarmonyInterpolationPoint2 ()

Data Fields

- double I
- double P
- double s_II
- double s_IP
- double s_IT
- double s_IV
- double s_PI
- double s_PP
- double s_PT
- double s_PV
- double s_TI
- double s_TP
- double s_TT
- double s_TV
- double s_VI
- double s_VP
- double s_VT
- double s_VV
- double t
- double T
- double V

6.36.1 Detailed Description

Represents an interpolation point with scaling factors for a fractal interpolation function in the **time-harmony subspace** of the score space.

6.36.2 Constructor & Destructor Documentation

6.36.2.1 HarmonyInterpolationPoint2() [1/3]

```
csound::HarmonyInterpolationPoint2::HarmonyInterpolationPoint2 ( )  [inline]
```

6.36.2.2 HarmonyInterpolationPoint2() [2/3]

```
csound::HarmonyInterpolationPoint2::HarmonyInterpolationPoint2 (
            const HarmonyInterpolationPoint2 & other )  [inline]
```

6.36.2.3 HarmonyInterpolationPoint2() [3/3]

```
csound::HarmonyInterpolationPoint2::HarmonyInterpolationPoint2 (
            double t_,
            double P_,
            double I_,
            double T_,
            double V_,
            double s_PP_,
            double s_PI_,
            double s_PT_,
            double s_PV_,
            double s_IP_,
            double s_II_,
            double s_IT_,
            double s_IV_,
            double s_TP_,
            double s_TI_,
            double s_TT_,
            double s_TV_,
            double s_VP_,
            double s_VI_,
            double s_VT_,
            double s_VV_ )  [inline]
```

References csound::I(), and csound::T().

6.36.2.4 ~HarmonyInterpolationPoint2()

```
virtual csound::HarmonyInterpolationPoint2::~HarmonyInterpolationPoint2 ( )  [inline], [virtual]
```

6.36.3 Member Function Documentation

6.36.3.1 toString()

```
virtual std::string csound::HarmonyInterpolationPoint2::toString ( ) const  [inline], [virtual]
```

References csound::I(), and csound::T().

Referenced by csound::HarmonyIFS2::add_interpolation_point().

6.36.4 Field Documentation

6.36.4.1 I

```
double csound::HarmonyInterpolationPoint2::I
```

Referenced by csound::HarmonyIFS2::initialize_hutchinson_operator().

6.36.4.2 P

```
double csound::HarmonyInterpolationPoint2::P
```

Referenced by csound::HarmonyIFS2::initialize_hutchinson_operator().

6.36.4.3 s_II

```
double csound::HarmonyInterpolationPoint2::s_II
```

Referenced by csound::HarmonyIFS2::initialize_hutchinson_operator().

6.36.4.4 s_IP

```
double csound::HarmonyInterpolationPoint2::s_IP
```

Referenced by csound::HarmonyIFS2::initialize_hutchinson_operator().

6.36.4.5 s_IT

```
double csound::HarmonyInterpolationPoint2::s_IT
```

Referenced by csound::HarmonyIFS2::initialize_hutchinson_operator().

6.36.4.6 s_IV

```
double csound::HarmonyInterpolationPoint2::s_IV
```

Referenced by csound::HarmonyIFS2::initialize_hutchinson_operator().

6.36.4.7 s_PI

```
double csound::HarmonyInterpolationPoint2::s_PI
```

Referenced by csound::HarmonyIFS2::initialize_hutchinson_operator().

6.36.4.8 s_PP

```
double csound::HarmonyInterpolationPoint2::s_PP
```

Referenced by csound::HarmonyIFS2::initialize_hutchinson_operator().

6.36.4.9 s_PT

```
double csound::HarmonyInterpolationPoint2::s_PT
```

Referenced by csound::HarmonyIFS2::initialize_hutchinson_operator().

6.36.4.10 s_PV

```
double csound::HarmonyInterpolationPoint2::s_PV
```

Referenced by csound::HarmonyIFS2::initialize_hutchinson_operator().

6.36.4.11 s_TI

```
double csound::HarmonyInterpolationPoint2::s_TI
```

Referenced by csound::HarmonyIFS2::initialize_hutchinson_operator().

6.36.4.12 s_TP

```
double csound::HarmonyInterpolationPoint2::s_TP
```

Referenced by csound::HarmonyIFS2::initialize_hutchinson_operator().

6.36.4.13 s_TT

```
double csound::HarmonyInterpolationPoint2::s_TT
```

Referenced by csound::HarmonyIFS2::initialize_hutchinson_operator().

6.36.4.14 s_TV

```
double csound::HarmonyInterpolationPoint2::s_TV
```

Referenced by csound::HarmonyIFS2::initialize_hutchinson_operator().

6.36.4.15 s_VI

```
double csound::HarmonyInterpolationPoint2::s_VI
```

Referenced by csound::HarmonyIFS2::initialize_hutchinson_operator().

6.36.4.16 s_VP

```
double csound::HarmonyInterpolationPoint2::s_VP
```

Referenced by csound::HarmonyIFS2::initialize_hutchinson_operator().

6.36.4.17 s_VT

```
double csound::HarmonyInterpolationPoint2::s_VT
```

Referenced by csound::HarmonyIFS2::initialize_hutchinson_operator().

6.36.4.18 s_VV

```
double csound::HarmonyInterpolationPoint2::s_VV
```

Referenced by csound::HarmonyIFS2::initialize_hutchinson_operator().

6.36.4.19 t

```
double csound::HarmonyInterpolationPoint2::t
```

Referenced by csound::HarmonyIFS2::initialize_hutchinson_operator(), and csound::interpolation_point_less2().

6.36.4.20 T

```
double csound::HarmonyInterpolationPoint2::T
```

Referenced by csound::HarmonyIFS2::initialize_hutchinson_operator().

6.36.4.21 V

```
double csound::HarmonyInterpolationPoint2::V
```

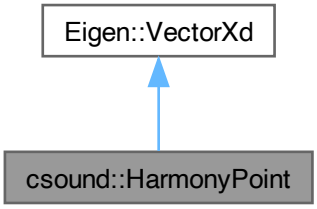
Referenced by csound::HarmonyIFS2::initialize_hutchinson_operator().

6.37 csound::HarmonyPoint Class Reference

Represents a point on a time line in a score space that has a time- harmony subspace.

`#include <HarmonyIFS.hpp>`

Inheritance diagram for csound::HarmonyPoint:



Public Types

- enum Dimensions {
  HP_TIME = 0 , HP_PRIME_FORM , HP_INVERSION , HP_TRANSPOSITION ,
  HP_MIDI_KEY , HP_MIDI_VELOCITY , HP_INSTRUMENT , HP_HOMOGENEITY ,
  HP_ELEMENT_COUNT }

Public Member Functions

- HarmonyPoint ()
- HarmonyPoint (const HarmonyPoint &other)
- virtual double I () const
- virtual double i () const
- virtual void initialize ()
- virtual double k () const
- HarmonyPoint & operator= (const Eigen::VectorXd &other)
- HarmonyPoint & operator= (const HarmonyPoint &other)
- virtual double P () const
- virtual void set_homogeneity (double value)
- virtual void set_I (double value)
- virtual void set_i (double value)
- virtual void set_k (double value)
- virtual void set_P (double value)
- virtual void set_t (double value)
- virtual void set_T (double value)
- virtual void set_v (double value)
- virtual double t () const
- virtual double T () const
- virtual std::string toString () const
- virtual double v () const
- virtual ~HarmonyPoint ()

6.37.1 Detailed Description

Represents a point on a time line in a score space that has a time- harmony subspace.

The point consists of a homogeneous column vector with dimensions:

```
t    Time.
P    Set class (or prime form).
I    Inversion (or reflection in the origin of pitch space).
T    Transposition (or translation in pitch space).
k    MIDI key number (the actual pitch, may be a fraction).
v    MIDI velocity (or loudness).
i    Instrument number (1-based).
1    Homogeneity.
```

At rendering time, the point will be translated to that pitch which most closely matches a pitch-class in that chord defined by P, I, and T.

6.37.2 Member Enumeration Documentation

6.37.2.1 Dimensions

```
enum csound::HarmonyPoint::Dimensions
```

Enumerator

HP_TIME	
HP_PRIME_FORM	
HP_INVERSION	
HP_TRANSPOSITION	
HP_MIDI_KEY	
HP_MIDI_VELOCITY	
HP_INSTRUMENT	
HP_HOMOGENEITY	
HP_ELEMENT_COUNT	

6.37.3 Constructor & Destructor Documentation

6.37.3.1 HarmonyPoint() [1/2]

```
csound::HarmonyPoint::HarmonyPoint ( )  [inline]
```

6.37.3.2 HarmonyPoint() [2/2]

```
csound::HarmonyPoint::HarmonyPoint (
            const HarmonyPoint & other )  [inline]
```

6.37.3.3 ~HarmonyPoint()

```
virtual csound::HarmonyPoint::~HarmonyPoint ( )  [inline], [virtual]
```

6.37.4 Member Function Documentation

6.37.4.1 I()

```
virtual double csound::HarmonyPoint::I ( ) const  [inline], [virtual]
```

Referenced by csound::HarmonyIFS::point_to_note().

6.37.4.2 i()

```
virtual double csound::HarmonyPoint::i ( ) const  [inline], [virtual]
```

Referenced by csound::HarmonyIFS::point_to_note().

6.37.4.3 initialize()

```
virtual void csound::HarmonyPoint::initialize ( )  [inline], [virtual]
```

6.37.4.4 k()

```
virtual double csound::HarmonyPoint::k ( ) const  [inline], [virtual]
```

Referenced by csound::HarmonyIFS::point_to_note().

6.37.4.5 operator=() [1/2]

```
HarmonyPoint & csound::HarmonyPoint::operator= (
            const Eigen::VectorXd & other )  [inline]
```

6.37.4.6 operator=() [2/2]

```
HarmonyPoint & csound::HarmonyPoint::operator= (
            const HarmonyPoint & other )  [inline]
```

6.37.4.7 P()

```
virtual double csound::HarmonyPoint::P ( ) const  [inline], [virtual]
```

Referenced by csound::HarmonyIFS::point_to_note().

6.37.4.8 set_homogeneity()

```
virtual void csound::HarmonyPoint::set_homogeneity (
            double value )  [inline], [virtual]
```

Referenced by csound::HarmonyIFS::generate_score_attractor().

6.37.4.9 set_I()

```
virtual void csound::HarmonyPoint::set_I (
            double value )  [inline], [virtual]
```

6.37.4.10 set_i()

```
virtual void csound::HarmonyPoint::set_i (
            double value )  [inline], [virtual]
```

Referenced by csound::HarmonyIFS::generate_score_attractor().

6.37.4.11 set_k()

```
virtual void csound::HarmonyPoint::set_k (
            double value )  [inline], [virtual]
```

Referenced by csound::HarmonyIFS::generate_score_attractor().

6.37.4.12 set_P()

```
virtual void csound::HarmonyPoint::set_P (
            double value )  [inline], [virtual]
```

6.37.4.13 set_t()

```
virtual void csound::HarmonyPoint::set_t (
            double value )  [inline], [virtual]
```

6.37.4.14 set_T()

```
virtual void csound::HarmonyPoint::set_T (
            double value )  [inline], [virtual]
```

6.37.4.15 set_v()

```
virtual void csound::HarmonyPoint::set_v (
            double value )  [inline], [virtual]
```

Referenced by csound::HarmonyIFS::generate_score_attractor().

6.37.4.16 t()

```
virtual double csound::HarmonyPoint::t ( ) const  [inline], [virtual]
```

Referenced by csound::HarmonyIFS::point_to_note().

6.37.4.17 T()

```
virtual double csound::HarmonyPoint::T ( ) const  [inline], [virtual]
```

Referenced by csound::HarmonyIFS::point_to_note().

6.37.4.18 toString()

```
virtual std::string csound::HarmonyPoint::toString ( ) const  [inline], [virtual]
```

References csound::I(), and csound::T().

6.37.4.19 v()

```
virtual double csound::HarmonyPoint::v ( ) const  [inline], [virtual]
```

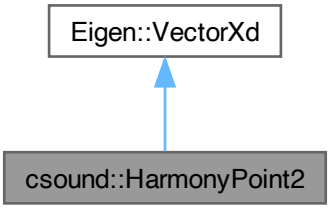
Referenced by csound::HarmonyIFS::point_to_note().

6.38 csound::HarmonyPoint2 Class Reference

Represents a point on a time line in a score space that has a time- harmony subspace.

`#include <HarmonyIFS2.hpp>`

Inheritance diagram for csound::HarmonyPoint2:



Public Types

- enum Dimensions {
  HP_TIME = 0 , HP_PRIME_FORM , HP_INVERSION , HP_TRANSPOSITION ,
  HP_VOICING , HP_HOMOGENEITY , HP_ELEMENT_COUNT }

Public Member Functions

- virtual double get_homogeneity () const
- virtual double get_I () const
- virtual double get_P () const
- virtual double get_t () const
- virtual double get_T () const
- virtual double get_V () const
- HarmonyPoint2 ()
- HarmonyPoint2 (const HarmonyPoint2 &other)
- virtual double I () const
- virtual void initialize ()
- HarmonyPoint2 & operator= (const Eigen::VectorXd &other)
- HarmonyPoint2 & operator= (const HarmonyPoint2 &other)
- virtual double P () const
- virtual void set_homogeneity (double value)
- virtual void set_I (double value)
- virtual void set_P (double value)
- virtual void set_t (double value)
- virtual void set_T (double value)
- virtual void set_V (double value)
- virtual double t () const
- virtual double T () const
- virtual std::string toString () const
- virtual double V () const
- virtual ~HarmonyPoint2 ()

6.38.1 Detailed Description

Represents a point on a time line in a score space that has a time- harmony subspace.

The point consists of a homogeneous column vector with dimensions:

```
t    Time.
P    Set class (or prime form).
I    Inversion (or reflection in the origin of pitch space).
T    Transposition (or translation in pitch space modulo the octave).
V    Permutation of octavewise revoicings within range R.
1    Homogeneity.
```

At rendering time, the point will be translated to that pitch which most closely matches a pitch-class in that chord defined by P, I, and T.

6.38.2 Member Enumeration Documentation

6.38.2.1 Dimensions

```
enum csound::HarmonyPoint2::Dimensions
```

Enumerator

HP_TIME	
HP_PRIME_FORM	
HP_INVERSION	
HP_TRANSPOSITION	
HP_VOICING	
HP_HOMOGENEITY	
HP_ELEMENT_COUNT	

6.38.3 Constructor & Destructor Documentation

6.38.3.1 HarmonyPoint2() [1/2]

```
csound::HarmonyPoint2::HarmonyPoint2 ( )  [inline]
```

6.38.3.2 HarmonyPoint2() [2/2]

```
csound::HarmonyPoint2::HarmonyPoint2 (
            const HarmonyPoint2 & other )  [inline]
```

6.38.3.3 ~HarmonyPoint2()

```
virtual csound::HarmonyPoint2::~HarmonyPoint2 ( )  [inline], [virtual]
```

6.38.4 Member Function Documentation

6.38.4.1 get_homogeneity()

```
virtual double csound::HarmonyPoint2::get_homogeneity ( ) const  [inline], [virtual]
```

6.38.4.2 get_I()

```
virtual double csound::HarmonyPoint2::get_I ( ) const  [inline], [virtual]
```

6.38.4.3 get_P()

```
virtual double csound::HarmonyPoint2::get_P ( ) const  [inline], [virtual]
```

6.38.4.4 get_t()

```
virtual double csound::HarmonyPoint2::get_t ( ) const  [inline], [virtual]
```

6.38.4.5 get_T()

```
virtual double csound::HarmonyPoint2::get_T ( ) const  [inline], [virtual]
```

6.38.4.6 get_V()

```
virtual double csound::HarmonyPoint2::get_V ( ) const  [inline], [virtual]
```

6.38.4.7 I()

```
virtual double csound::HarmonyPoint2::I ( ) const  [inline], [virtual]
```

Referenced by csound::HarmonyIFS2::iterate().

6.38.4.8 initialize()

```
virtual void csound::HarmonyPoint2::initialize ( )  [inline], [virtual]
```

6.38.4.9 operator=() [1/2]

```
HarmonyPoint2 & csound::HarmonyPoint2::operator= (
            const Eigen::VectorXd & other )  [inline]
```

6.38.4.10 operator=() [2/2]

```
HarmonyPoint2 & csound::HarmonyPoint2::operator= (
            const HarmonyPoint2 & other )  [inline]
```

6.38.4.11 P()

```
virtual double csound::HarmonyPoint2::P ( ) const  [inline], [virtual]
```

Referenced by csound::HarmonyIFS2::iterate().

6.38.4.12 set_homogeneity()

```
virtual void csound::HarmonyPoint2::set_homogeneity (
            double value )  [inline], [virtual]
```

Referenced by csound::HarmonyIFS2::generate_score_attractor().

6.38.4.13 set_I()

```
virtual void csound::HarmonyPoint2::set_I (
            double value )  [inline], [virtual]
```

6.38.4.14 set_P()

```
virtual void csound::HarmonyPoint2::set_P (
            double value )  [inline], [virtual]
```

6.38.4.15 set_t()

```
virtual void csound::HarmonyPoint2::set_t (
            double value )  [inline], [virtual]
```

6.38.4.16 set_T()

```
virtual void csound::HarmonyPoint2::set_T (
            double value )  [inline], [virtual]
```

6.38.4.17 set_V()

```
virtual void csound::HarmonyPoint2::set_V (
            double value )  [inline], [virtual]
```

6.38.4.18 t()

```
virtual double csound::HarmonyPoint2::t ( ) const  [inline], [virtual]
```

Referenced by csound::HarmonyIFS2::iterate().

6.38.4.19 T()

```
virtual double csound::HarmonyPoint2::T ( ) const  [inline], [virtual]
```

Referenced by csound::HarmonyIFS2::iterate().

6.38.4.20 toString()

```
virtual std::string csound::HarmonyPoint2::toString ( ) const  [inline], [virtual]
```

References csound::I(), and csound::T().

Referenced by csound::HarmonyIFS2::iterate().

6.38.4.21 V()

```
virtual double csound::HarmonyPoint2::V ( ) const  [inline], [virtual]
```

Referenced by csound::HarmonyIFS2::iterate().

6.39 csound::HyperplaneEquation Struct Reference

```
#include <ChordSpaceBase.hpp>
```

Data Fields

- double constant_term
- Matrix unit_normal_vector

6.39.1 Field Documentation

6.39.1.1 constant_term

```
double csound::HyperplaneEquation::constant_term
```

Referenced by equals(), Hyperplane_Equation_for_Test_Points(), csound::hyperplane_equation_from_random_inversion_flat(), csound::hyperplane_equation_from_singular_value_decomposition(), csound::Chord::initialize_sectors(), and csound::reflect_in_inversior

6.39.1.2 unit_normal_vector

```
Matrix csound::HyperplaneEquation::unit_normal_vector
```

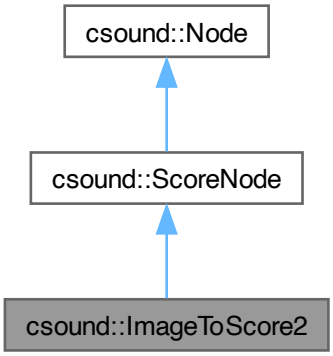
Referenced by equals(), Hyperplane_Equation_for_Test_Points(), csound::hyperplane_equation_from_random_inversion_flat(), csound::hyperplane_equation_from_singular_value_decomposition(), csound::Chord::initialize_sectors(), and csound::reflect_in_inversior

6.40 csound::ImageToScore2 Class Reference

Translates images files to scores.

```
#include <ImageToScore.hpp>
```

Inheritance diagram for csound::ImageToScore2:



Public Member Functions

- virtual void addChild (Node *node)

  Adds an immediate child Node to this.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual void condense (int row_count_)

  Translate the image to the specified number of rows before translating it to notes.
- virtual void contrast (double gain_, double bias_)

  Change the contrast of the image by the specified factors before translating it to notes.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual void dilate (int kernel_shape_, int kernel_size_, int iterations=1)

  Increase the thickness of features in the image before translating it to notes.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void erode (int kernel_shape, int kernel_size_, int iterations=1)

  Decrease the thickness of features in the image before translating it to notes.
- virtual void gaussianBlur (double sigma_x_, double sigma_y_=0, int kernel_size_=9, int kernel_shape_=cv::↩ MORPH_RECT)

  Blur the image using a Gaussian kernel before translating the image to notes.
- virtual void generate (Score &collectingScore)

  Optionally generate notes into the score.
- virtual void generateLocally ()
- virtual Node * getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual std::string getImageFilename () const
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual size_t getMaximumVoiceCount () const
- virtual Score & getScore ()
- ImageToScore2 ()
- virtual void processImage ()

  Perform any image processing, then translate the resulting image to notes.
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void setImageFilename (std::string imageFilename)
- virtual void setMaximumVoiceCount (size_t maximumVoiceCount)
- virtual void sharpen (int kernel_size_, double sigma_x_, double sigma_y_, double alpha_, double beta_, double gamma_)

  Sharpen the image before translating to notes.
- virtual void threshhold (double value_threshhold_)

  Set all values less than the threshhold to zero before translating the image to notes.
- virtual void transform (Score &score_from_children)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.
- void write_processed_file (std::string operation, const cv::Mat &processed_image) const
- virtual ~ImageToScore2 ()

Data Fields

- double alpha = 0
- double beta = 0
- double bias = 0
- std::vector< Node * > children

  Child Nodes, if any.
- bool do_blur = false
- bool do_condense = false
- bool do_contrast = false
- bool do_dilate = false
- bool do_erode = false
- bool do_sharpen = false
- bool do_threshhold = false
- double duration

  If not 0, the score is rescaled to this duration.
- double gain = 0
- double gamma = 0
- std::string importFilename
- int iterations = 1
- int kernel_shape = cv::MORPH_RECT
- int kernel_size = 9
- int row_count = -1
- double sigma_x
- double sigma_y
- double value_threshhold = 0

Protected Member Functions

- virtual void pixel_to_event (int column, int row, const cv::Vec3f &hsv, Event &event) const

Protected Attributes

- std::string image_filename
- Eigen::MatrixXd localCoordinates
- size_t maximum_voice_count = 7
- cv::Mat original_image
- cv::Mat processed_image
- Score score

6.40.1 Detailed Description

Translates images files to scores.

The OpenCV library is used to do an improved mapping from images to scores. Various image processing algorithms may be applied to the original image before the resulting image is translated to notes. Any number of such operations may be specified, but the order of processing is fixed.

6.40.2 Constructor & Destructor Documentation

6.40.2.1 ImageToScore2()

```
csound::ImageToScore2::ImageToScore2 ( )
```

6.40.2.2 ~ImageToScore2()

```
csound::ImageToScore2::~ImageToScore2 ( )  [virtual]
```

6.40.3 Member Function Documentation

6.40.3.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.40.3.2 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.40.3.3 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.40.3.4 condense()

```
void csound::ImageToScore2::condense (
            int row_count_ )  [virtual]
```

Translate the image to the specified number of rows before translating it to notes.

If this operation is performed, it is always the last operation before translating the resulting image to notes.

References do_condense, csound::fundamentalDomainByPredicate(), and row_count.

6.40.3.5 contrast()

```
void csound::ImageToScore2::contrast (
            double gain_,
            double bias_ )  [virtual]
```

Change the contrast of the image by the specified factors before translating it to notes.

References bias, do_contrast, csound::fundamentalDomainByPredicate(), and gain.

6.40.3.6 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.40.3.7 dilate()

```
void csound::ImageToScore2::dilate (
            int kernel_shape_,
            int kernel_size_,
            int iterations = 1 )  [virtual]
```

Increase the thickness of features in the image before translating it to notes.

References do_dilate, csound::fundamentalDomainByPredicate(), iterations, kernel_shape, and kernel_size.

6.40.3.8 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.40.3.9 erode()

```
void csound::ImageToScore2::erode (
            int kernel_shape,
            int kernel_size_,
            int iterations = 1 )  [virtual]
```

Decrease the thickness of features in the image before translating it to notes.

References do_erode, csound::fundamentalDomainByPredicate(), iterations, kernel_shape, and kernel_size.

6.40.3.10 gaussianBlur()

```
void csound::ImageToScore2::gaussianBlur (
            double sigma_x_,
            double sigma_y_ = 0,
            int kernel_size_ = 9,
            int kernel_shape_ = cv::MORPH_RECT )  [virtual]
```

Blur the image using a Gaussian kernel before translating the image to notes.

Kernel size should be odd.

References do_blur, csound::fundamentalDomainByPredicate(), kernel_shape, kernel_size, sigma_x, and sigma_y.

6.40.3.11 generate()

```
void csound::ScoreNode::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented from csound::Node.

Reimplemented in csound::ExternalNode, and csound::MCRM.

References csound::ScoreNode::duration, csound::fundamentalDomainByPredicate(), csound::Score::getCsoundScoreHeader(), csound::ScoreNode::importFilename, csound::Score::load(), csound::Score::process(), csound::ScoreNode::score, csound::Score::setDuration(), and csound::Score::sort().

Referenced by csound::MCRM::generate().

6.40.3.12 generateLocally()

```
void csound::ImageToScore2::generateLocally ( )  [virtual]
```

References csound::Score::append(), csound::System::debug(), csound::System::DEBUGGING_LEVEL, csound::fundamentalDomainBy
csound::Event::getChannel(), csound::Event::getKeyNumber(), getMaximumVoiceCount(), csound::System::getMessageLevel(), csound::Event::getOffTime(), csound::Event::getTime(), csound::System::inform(), pixel_to_event(), processed_image, processImage(), csound::ScoreNode::score, csound::Event::setOffTime(), csound::Score::sort(), and value_threshhold.

6.40.3.13 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.40.3.14 getImageFilename()

```
std::string csound::ImageToScore2::getImageFilename ( ) const  [virtual]
```

References image_filename.

6.40.3.15 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.40.3.16 getMaximumVoiceCount()

```
size_t csound::ImageToScore2::getMaximumVoiceCount ( ) const  [virtual]
```

References maximum_voice_count.

Referenced by generateLocally().

6.40.3.17 getScore()

```
Score & csound::ScoreNode::getScore ( )  [virtual], [inherited]
```

References csound::ScoreNode::score.

Referenced by main().

6.40.3.18 pixel_to_event()

```
void csound::ImageToScore2::pixel_to_event (
            int column,
            int row,
            const cv::Vec3f & hsv,
            Event & event ) const  [protected], [virtual]
```

References csound::fundamentalDomainByPredicate(), csound::Event::getInstrument(), csound::Event::getKey(), csound::Event::getTime(), csound::Event::getVelocity(), processed_image, csound::Score::scaleTargetMinima, csound::Score::scaleTargetRanges, and csound::ScoreNode::score.

Referenced by generateLocally().

6.40.3.19 processImage()

```
void csound::ImageToScore2::processImage ( )  [virtual]
```

Perform any image processing, then translate the resulting image to notes.

References alpha, beta, bias, do_blur, do_condense, do_contrast, do_dilate, do_erode, do_sharpen, do_threshhold, csound::System::error(), csound::fundamentalDomainByPredicate(), gain, gamma, image_filename, csound::System::inform(), iterations, kernel_shape, kernel_size, original_image, processed_image, row_count, sigma_x, sigma_y, value_threshhold, and write_processed_file().

Referenced by generateLocally().

6.40.3.20 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.40.3.21 setImageFilename()

```
void csound::ImageToScore2::setImageFilename (
            std::string imageFilename )  [virtual]
```

References csound::fundamentalDomainByPredicate(), and image_filename.

6.40.3.22 setMaximumVoiceCount()

```
void csound::ImageToScore2::setMaximumVoiceCount (
            size_t maximumVoiceCount )  [virtual]
```

References csound::fundamentalDomainByPredicate(), and maximum_voice_count.

6.40.3.23 sharpen()

```
void csound::ImageToScore2::sharpen (
            int kernel_size_,
            double sigma_x_,
            double sigma_y_,
            double alpha_,
            double beta_,
            double gamma_ )  [virtual]
```

Sharpen the image before translating to notes.

First the image is blurred, and then the blurred image is subtracted from the original image.

References alpha, beta, do_sharpen, csound::fundamentalDomainByPredicate(), gamma, sigma_x, and sigma_y.

6.40.3.24 threshhold()

```
void csound::ImageToScore2::threshhold (
            double value_threshhold_ )  [virtual]
```

Set all values less than the threshhold to zero before translating the image to notes.

References do_threshhold, csound::fundamentalDomainByPredicate(), and value_threshhold.

6.40.3.25 transform()

```
void csound::Node::transform (
            Score & score_from_children )  [virtual], [inherited]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented in csound::Cell, csound::CellRepeat, csound::CellAdd, csound::CellMultiply, csound::CellReflect, csound::CellSelect, csound::CellRemove, csound::CellChord, csound::CellRandom, csound::CellShuffle, csound::CounterpointNode, csound::RemoveDuplicates, csound::Transformer, csound::Random, csound::Rescale, csound::VoiceleadingNode, csound::LispTransformer, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.40.3.26 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord and csound::Node::transform().

6.40.3.27 write_processed_file()

```
void csound::ImageToScore2::write_processed_file (
            std::string operation,
            const cv::Mat & processed_image ) const
```

References csound::fundamentalDomainByPredicate(), image_filename, and processed_image.

Referenced by processImage().

6.40.4 Field Documentation

6.40.4.1 alpha

```
double csound::ImageToScore2::alpha = 0
```

Referenced by processImage(), and sharpen().

6.40.4.2 beta

```
double csound::ImageToScore2::beta = 0
```

Referenced by processImage(), and sharpen().

6.40.4.3 bias

```
double csound::ImageToScore2::bias = 0
```

Referenced by contrast(), and processImage().

6.40.4.4 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.40.4.5 do_blur

```
bool csound::ImageToScore2::do_blur = false
```

Referenced by gaussianBlur(), and processImage().

6.40.4.6 do_condense

```
bool csound::ImageToScore2::do_condense = false
```

Referenced by condense(), and processImage().

6.40.4.7 do_contrast

```
bool csound::ImageToScore2::do_contrast = false
```

Referenced by contrast(), and processImage().

6.40.4.8 do_dilate

```
bool csound::ImageToScore2::do_dilate = false
```

Referenced by dilate(), and processImage().

6.40.4.9 do_erode

```
bool csound::ImageToScore2::do_erode = false
```

Referenced by erode(), and processImage().

6.40.4.10 do_sharpen

```
bool csound::ImageToScore2::do_sharpen = false
```

Referenced by processImage(), and sharpen().

6.40.4.11 do_threshhold

```
bool csound::ImageToScore2::do_threshhold = false
```

Referenced by processImage(), and threshhold().

6.40.4.12 duration

```
double csound::ScoreNode::duration  [inherited]
```

If not 0, the score is rescaled to this duration.

Referenced by csound::ScoreNode::generate(), csound::ExternalNode::generateLocally(), and csound::Stack::getDuration().

6.40.4.13 gain

```
double csound::ImageToScore2::gain = 0
```

Referenced by contrast(), and processImage().

6.40.4.14 gamma

```
double csound::ImageToScore2::gamma = 0
```

Referenced by processImage(), and sharpen().

6.40.4.15 image_filename

```
std::string csound::ImageToScore2::image_filename  [protected]
```

Referenced by getImageFilename(), processImage(), setImageFilename(), and write_processed_file().

6.40.4.16 importFilename

```
std::string csound::ScoreNode::importFilename  [inherited]
```

Referenced by csound::ScoreNode::generate().

6.40.4.17 iterations

```
int csound::ImageToScore2::iterations = 1
```

Referenced by dilate(), erode(), and processImage().

6.40.4.18 kernel_shape

```
int csound::ImageToScore2::kernel_shape = cv::MORPH_RECT
```

Referenced by dilate(), erode(), gaussianBlur(), and processImage().

6.40.4.19 kernel_size

```
int csound::ImageToScore2::kernel_size = 9
```

Referenced by dilate(), erode(), gaussianBlur(), and processImage().

6.40.4.20 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.40.4.21 maximum_voice_count

```
size_t csound::ImageToScore2::maximum_voice_count = 7  [protected]
```

Referenced by getMaximumVoiceCount(), and setMaximumVoiceCount().

6.40.4.22 original_image

```
cv::Mat csound::ImageToScore2::original_image  [protected]
```

Referenced by processImage().

6.40.4.23 processed_image

```
cv::Mat csound::ImageToScore2::processed_image  [protected]
```

Referenced by generateLocally(), pixel_to_event(), processImage(), and write_processed_file().

6.40.4.24 row_count

```
int csound::ImageToScore2::row_count = -1
```

Referenced by condense(), and processImage().

6.40.4.25 score

```
Score csound::ScoreNode::score  [protected], [inherited]
```

Referenced by csound::StrangeAttractor::evaluateAttractor(), csound::ExternalNode::generate(), csound::ScoreNode::generate(), csound::MCRM::generate(), csound::ExternalNode::generateLocally(), generateLocally(), csound::Lindenmayer::generateLocally(), csound::Rescale::getRescale(), csound::ScoreNode::getScore(), csound::Lindenmayer::interpret(), csound::MCRM::iterate(), csound::StrangeAttractor::iterate_without_rendering(), csound::KMeansMCRM::means_to_notes(), pixel_to_event(), csound::StrangeAttractor::render(), csound::Rescale::Rescale(), csound::Rescale::setRescale(), csound::Cell::transform(), csound::Rescale::transform(), csound::CMaskNode::translate_to_silence(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Lindenmayer::updateActual().

6.40.4.26 sigma_x

```
double csound::ImageToScore2::sigma_x
```

Referenced by gaussianBlur(), processImage(), and sharpen().

6.40.4.27 sigma_y

```
double csound::ImageToScore2::sigma_y
```

Referenced by gaussianBlur(), processImage(), and sharpen().

6.40.4.28 value_threshhold

```
double csound::ImageToScore2::value_threshhold = 0
```

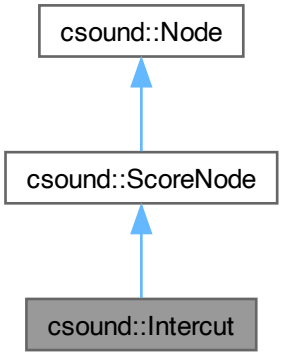
Referenced by generateLocally(), processImage(), and threshhold().

6.41 csound::Intercut Class Reference

The notes produced by each child node are intercut to produce the notes produced by this; e.g.

```
#include <Cell.hpp>
```

Inheritance diagram for csound::Intercut:



Public Member Functions

- virtual void addChild (Node *node)

  Adds an immediate child Node to this.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &collectingScore)

  Optionally generate notes into the score.
- virtual Node * getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual Score & getScore ()
- Intercut ()
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void transform (Score &score_from_children)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &globalCoordinates, Score &collectingScore)

  The notes produced by each child node are intercut to produce the notes produced by this; e.g.
- virtual ~Intercut ()

Data Fields

- std::vector< Node * > children

  Child Nodes, if any.
- double duration

  If not 0, the score is rescaled to this duration.
- std::string importFilename

Protected Attributes

- Eigen::MatrixXd localCoordinates
- Score score

6.41.1 Detailed Description

The notes produced by each child node are intercut to produce the notes produced by this; e.g.

if there are 3 child nodes, then the notes produced by this are node 0 note 0, node 1 note 0, node 2 note 0; node 0 note 1, node 1 note 1, node node 2 note 1; node 0 note 2, node 1 note 2, node 2 note 2, and so on. If the child nodes do not each produce the same number of notes, then production stops with the last note of the longest child.

6.41.2 Constructor & Destructor Documentation

6.41.2.1 Intercut()

```
csound::Intercut::Intercut ( )
```

6.41.2.2 ~Intercut()

```
csound::Intercut::~Intercut ( )  [virtual]
```

6.41.3 Member Function Documentation

6.41.3.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.41.3.2 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.41.3.3 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.41.3.4 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.41.3.5 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.41.3.6 generate()

```
void csound::ScoreNode::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented from csound::Node.

Reimplemented in csound::ExternalNode, and csound::MCRM.

References csound::ScoreNode::duration, csound::fundamentalDomainByPredicate(), csound::Score::getCsoundScoreHeader(), csound::ScoreNode::importFilename, csound::Score::load(), csound::Score::process(), csound::ScoreNode::score, csound::Score::setDuration(), and csound::Score::sort().

Referenced by csound::MCRM::generate().

6.41.3.7 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.41.3.8 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.41.3.9 getScore()

```
Score & csound::ScoreNode::getScore ( )  [virtual], [inherited]
```

References csound::ScoreNode::score.

Referenced by main().

6.41.3.10 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.41.3.11 transform()

```
void csound::Node::transform (
            Score & score_from_children )  [virtual], [inherited]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented in csound::Cell, csound::CellRepeat, csound::CellAdd, csound::CellMultiply, csound::CellReflect, csound::CellSelect, csound::CellRemove, csound::CellChord, csound::CellRandom, csound::CellShuffle, csound::CounterpointNode, csound::RemoveDuplicates, csound::Transformer, csound::Random, csound::Rescale, csound::VoiceleadingNode, csound::LispTransformer, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.41.3.12 traverse()

```
void csound::Intercut::traverse (
            const Eigen::MatrixXd & globalCoordinates,
            Score & collectingScore )  [virtual]
```

The notes produced by each child node are intercut to produce the notes produced by this; e.g.

if there are 3 child nodes, then the notes produced by this are node 0 note 0, node 1 note 0, node 2 note 0; node 0 note 1, node 1 note 1, node node 2 note 1; node 0 note 2, node 1 note 2, node 2 note 2, and so on. If the child nodes do not each produce the same number of notes, then the behavior is controlled by the repeatEach flag. Chords are treated as single notes.

Reimplemented from csound::Node.

References csound::Score::append(), csound::Node::children, csound::eq_tolerance(), csound::Node::getLocalCoordinates(), csound::System::message(), and csound::ScoreNode::score.

6.41.4 Field Documentation

6.41.4.1 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.41.4.2 duration

```
double csound::ScoreNode::duration  [inherited]
```

If not 0, the score is rescaled to this duration.

Referenced by csound::ScoreNode::generate(), csound::ExternalNode::generateLocally(), and csound::Stack::getDuration().

6.41.4.3 importFilename

```
std::string csound::ScoreNode::importFilename  [inherited]
```

Referenced by csound::ScoreNode::generate().

6.41.4.4 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.41.4.5 score

```
Score csound::ScoreNode::score  [protected], [inherited]
```

Referenced by csound::StrangeAttractor::evaluateAttractor(), csound::ExternalNode::generate(), csound::ScoreNode::generate(), csound::MCRM::generate(), csound::ExternalNode::generateLocally(), csound::ImageToScore2::generateLocally(), csound::Lindenmayer::generateLocally(), csound::Rescale::getRescale(), csound::ScoreNode::getScore(), csound::Lindenmayer::interpre csound::MCRM::iterate(), csound::StrangeAttractor::iterate_without_rendering(), csound::KMeansMCRM::means_to_notes(), csound::ImageToScore2::pixel_to_event(), csound::StrangeAttractor::render(), csound::Rescale::Rescale(), csound::Rescale::setRescale(csound::Cell::transform(), csound::Rescale::transform(), csound::CMaskNode::translate_to_silence(), traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Lindenmayer::updateActual().

6.42 csound::is_cl_object< T > Struct Template Reference

`#include <Lisp.hpp>`

Static Public Attributes

- static constexpr bool p = false

6.42.1 Field Documentation

6.42.1.1 p

```
template<typename T >
constexpr bool csound::is_cl_object< T >::p = false  [static], [constexpr]
```

6.43 csound::is_cl_object< cl_object > Struct Reference

`#include <Lisp.hpp>`

Static Public Attributes

- static constexpr bool p = true

6.43.1 Field Documentation

6.43.1.1 p

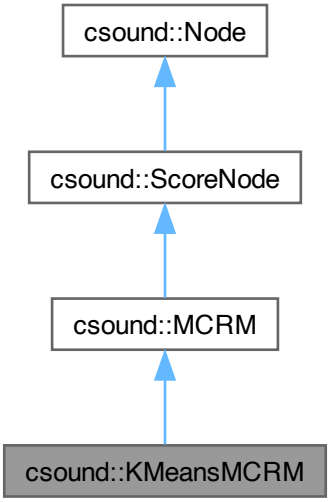
```
constexpr bool csound::is_cl_object< cl_object >::p = true  [static], [constexpr]
```

6.44 csound::KMeansMCRM Class Reference

Uses k-means clustering to translate the accumulated samples that approximate the measure on the iterated function system implemented by the multiple copy reducing machine algorithm into a specified number of notes.

`#include <MCRM.hpp>`

Inheritance diagram for csound::KMeansMCRM:



Public Types

- enum { MEASURE_DIMENSIONS =6 }
- enum ALGORITHM_TYPE { RANDOM = 1 , DETERMINISTIC }

  The type of algorithm used.

Public Member Functions

- virtual void addChild (Node *node)

  Adds an immediate child Node to this.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.

- virtual void deterministic_algorithm ()
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &score)

  Optionally generate notes into the score.
- virtual void generateLocally ()
- virtual Node * getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual Score & getScore ()
- virtual void iterate (int depth, size_t p, const Event &event, double weight)
- KMeansMCRM ()
- virtual void means_to_notes ()
- virtual void random_algorithm ()
- void resize (size_t transformations)
- void setDepth (int depth)
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- void setTransformationElement (size_t index, size_t row, size_t column, double value)
- void setWeight (size_t precursor, size_t successor, double weight)
- virtual void transform (Score &score_from_children)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.
- virtual ~KMeansMCRM ()

Data Fields

- ALGORITHM_TYPE algorithm_type
- std::vector< Node * > children

  Child Nodes, if any.
- double duration

  If not 0, the score is rescaled to this duration.
- std::string importFilename
- size_t means_count

  The same as k: the number of centroids or means to be computed from the samples, and later translated to notes.
- size_t sample_count

  The number of times a sample is to be generated, for the random algorithm.
- std::vector< std::array< double, MEASURE_DIMENSIONS > > samples

  The accumulated samples that approximate the measure of the IFS.

Protected Attributes

- int depth
- Eigen::MatrixXd localCoordinates
- Score score
- std::vector< Eigen::MatrixXd > transformations
- Eigen::MatrixXd weights

6.44.1 Detailed Description

Uses k-means clustering to translate the accumulated samples that approximate the measure on the iterated function system implemented by the multiple copy reducing machine algorithm into a specified number of notes.

6.44.2 Member Enumeration Documentation

6.44.2.1 anonymous enum

```
anonymous enum
```

Enumerator

MEASURE_DIMENSIONS	

6.44.2.2 ALGORITHM_TYPE

```
enum csound::KMeansMCRM::ALGORITHM_TYPE
```

The type of algorithm used.

Enumerator

RANDOM	
DETERMINISTIC	

6.44.3 Constructor & Destructor Documentation

6.44.3.1 KMeansMCRM()

```
csound::KMeansMCRM::KMeansMCRM ( )
```

6.44.3.2 ~KMeansMCRM()

```
csound::KMeansMCRM::~KMeansMCRM ( )  [virtual]
```

6.44.4 Member Function Documentation

6.44.4.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.44.4.2 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.44.4.3 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.44.4.4 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.44.4.5 deterministic_algorithm()

```
void csound::KMeansMCRM::deterministic_algorithm ( )  [virtual]
```

References csound::MCRM::depth, csound::fundamentalDomainByPredicate(), csound::System::inform(), iterate(), csound::Event::setStatus(), csound::System::startTiming(), and csound::System::stopTiming().

Referenced by generateLocally().

6.44.4.6 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.44.4.7 generate()

```
void csound::MCRM::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented from csound::ScoreNode.

References csound::ScoreNode::generate(), csound::MCRM::generateLocally(), and csound::ScoreNode::score.

6.44.4.8 generateLocally()

```
void csound::KMeansMCRM::generateLocally ( )  [virtual]
```

Reimplemented from csound::MCRM.

References algorithm_type, DETERMINISTIC, deterministic_algorithm(), csound::fundamentalDomainByPredicate(), csound::System::inform(), means_to_notes(), RANDOM, random_algorithm(), samples, csound::System::startTiming(), and csound::System::stopTiming().

6.44.4.9 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.44.4.10 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.44.4.11 getScore()

```
Score & csound::ScoreNode::getScore ( )  [virtual], [inherited]
```

References csound::ScoreNode::score.

Referenced by main().

6.44.4.12 iterate()

```
void csound::KMeansMCRM::iterate (
            int depth,
            size_t p,
            const Event & event,
            double weight )  [virtual]
```

Reimplemented from csound::MCRM.

References csound::fundamentalDomainByPredicate(), csound::System::inform(), iterate(), samples, csound::MCRM::transformations, and csound::MCRM::weights.

Referenced by deterministic_algorithm(), and iterate().

6.44.4.13 means_to_notes()

```
void csound::KMeansMCRM::means_to_notes ( )  [virtual]
```

References csound::Score::append(), csound::fundamentalDomainByPredicate(), csound::System::inform(), csound::mean_to_note(), means_count, samples, csound::ScoreNode::score, csound::System::startTiming(), and csound::System::stopTiming().

Referenced by generateLocally().

6.44.4.14 random_algorithm()

```
void csound::KMeansMCRM::random_algorithm ( )  [virtual]
```

References csound::fundamentalDomainByPredicate(), csound::System::inform(), sample_count, samples, csound::Event::setStatus(), csound::System::startTiming(), csound::System::stopTiming(), csound::MCRM::transformations, and csound::MCRM::weights.

Referenced by generateLocally().

6.44.4.15 resize()

```
void csound::MCRM::resize (
            size_t transformations )  [inherited]
```

References csound::Node::createTransform(), csound::fundamentalDomainByPredicate(), csound::MCRM::transformations, and csound::MCRM::weights.

6.44.4.16 setDepth()

```
void csound::MCRM::setDepth (
            int depth )  [inherited]
```

References csound::MCRM::depth.

6.44.4.17 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.44.4.18 setTransformationElement()

```
void csound::MCRM::setTransformationElement (
            size_t index,
            size_t row,
            size_t column,
            double value )  [inherited]
```

References csound::MCRM::transformations.

6.44.4.19 setWeight()

```
void csound::MCRM::setWeight (
            size_t precursor,
            size_t successor,
            double weight )  [inherited]
```

References csound::fundamentalDomainByPredicate(), and csound::MCRM::weights.

6.44.4.20 transform()

```
void csound::Node::transform (
            Score & score_from_children )  [virtual], [inherited]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented in csound::Cell, csound::CellRepeat, csound::CellAdd, csound::CellMultiply, csound::CellReflect, csound::CellSelect, csound::CellRemove, csound::CellChord, csound::CellRandom, csound::CellShuffle, csound::CounterpointNode, csound::RemoveDuplicates, csound::Transformer, csound::Random, csound::Rescale, csound::VoiceleadingNode, csound::LispTransformer, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.44.4.21 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord and csound::Node::transform().

6.44.5 Field Documentation

6.44.5.1 algorithm_type

```
ALGORITHM_TYPE csound::KMeansMCRM::algorithm_type
```

Referenced by generateLocally().

6.44.5.2 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.44.5.3 depth

```
int csound::MCRM::depth  [protected], [inherited]
```

Referenced by deterministic_algorithm(), csound::MCRM::generateLocally(), and csound::MCRM::setDepth().

6.44.5.4 duration

```
double csound::ScoreNode::duration  [inherited]
```

If not 0, the score is rescaled to this duration.

Referenced by csound::ScoreNode::generate(), csound::ExternalNode::generateLocally(), and csound::Stack::getDuration().

6.44.5.5 importFilename

```
std::string csound::ScoreNode::importFilename  [inherited]
```

Referenced by csound::ScoreNode::generate().

6.44.5.6 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.44.5.7 means_count

```
size_t csound::KMeansMCRM::means_count
```

The same as k: the number of centroids or means to be computed from the samples, and later translated to notes.

Referenced by means_to_notes().

6.44.5.8 sample_count

```
size_t csound::KMeansMCRM::sample_count
```

The number of times a sample is to be generated, for the random algorithm.

Referenced by random_algorithm().

6.44.5.9 samples

```
std::vector< std::array<double, MEASURE_DIMENSIONS> > csound::KMeansMCRM::samples
```

The accumulated samples that approximate the measure of the IFS.

Referenced by generateLocally(), iterate(), means_to_notes(), and random_algorithm().

6.44.5.10 score

```
Score csound::ScoreNode::score  [protected], [inherited]
```

Referenced by csound::StrangeAttractor::evaluateAttractor(), csound::ExternalNode::generate(), csound::ScoreNode::generate(), csound::MCRM::generate(), csound::ExternalNode::generateLocally(), csound::ImageToScore2::generateLocally(), csound::Lindenmayer::generateLocally(), csound::Rescale::getRescale(), csound::ScoreNode::getScore(), csound::Lindenmayer::interpre csound::MCRM::iterate(), csound::StrangeAttractor::iterate_without_rendering(), means_to_notes(), csound::ImageToScore2::pixel_to_ev csound::StrangeAttractor::render(), csound::Rescale::Rescale(), csound::Rescale::setRescale(), csound::Cell::transform(), csound::Rescale::transform(), csound::CMaskNode::translate_to_silence(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Lindenmayer::updateActual().

6.44.5.11 transformations

```
std::vector< Eigen::MatrixXd > csound::MCRM::transformations  [protected], [inherited]
```

Referenced by csound::MCRM::iterate(), iterate(), random_algorithm(), csound::MCRM::resize(), and csound::MCRM::setTransformationE

6.44.5.12 weights

```
Eigen::MatrixXd csound::MCRM::weights  [protected], [inherited]
```

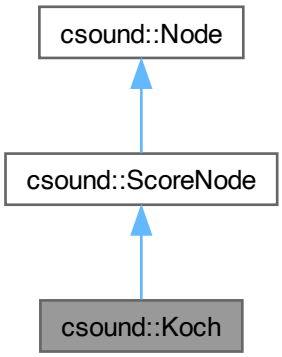
Referenced by csound::MCRM::iterate(), iterate(), random_algorithm(), csound::MCRM::resize(), and csound::MCRM::setWeight().

6.45 csound::Koch Class Reference

All notes produced by child[N - 1] are rescaled and stacked on top of each note produced by child[N - 2], and so on.

```
#include <Cell.hpp>
```

Inheritance diagram for csound::Koch:



Public Member Functions

- virtual void addChild (Node ∗node)

  Adds an immediate child Node to this.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &collectingScore)

  Optionally generate notes into the score.
- virtual Node ∗ getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual Score & getScore ()
- Koch ()
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void setPitchOffsetForLayer (int layer, double pitch)
- virtual void transform (Score &score_from_children)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &globalCoordinates, Score &score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.
- virtual ∼Koch ()

Data Fields

- std::vector< Node ∗ > children

  Child Nodes, if any.
- double duration

  If not 0, the score is rescaled to this duration.
- std::string importFilename
- std::map< int, double > pitchOffsetsForLayers

Protected Attributes

- Eigen::MatrixXd localCoordinates
- Score score

6.45.1 Detailed Description

All notes produced by child[N - 1] are rescaled and stacked on top of each note produced by child[N - 2], and so on.

6.45.2 Constructor & Destructor Documentation

6.45.2.1 Koch()

```
csound::Koch::Koch ( )
```

6.45.2.2 ~Koch()

```
csound::Koch::~Koch ( )  [virtual]
```

6.45.3 Member Function Documentation

6.45.3.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.45.3.2 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.45.3.3 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.45.3.4 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.45.3.5 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.45.3.6 generate()

```
void csound::ScoreNode::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented from csound::Node.

Reimplemented in csound::ExternalNode, and csound::MCRM.

References csound::ScoreNode::duration, csound::fundamentalDomainByPredicate(), csound::Score::getCsoundScoreHeader(), csound::ScoreNode::importFilename, csound::Score::load(), csound::Score::process(), csound::ScoreNode::score, csound::Score::setDuration(), and csound::Score::sort().

Referenced by csound::MCRM::generate().

6.45.3.7 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.45.3.8 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), traverse(), and csound::Sequence::traverse().

6.45.3.9 getScore()

```
Score & csound::ScoreNode::getScore ( )  [virtual], [inherited]
```

References csound::ScoreNode::score.

Referenced by main().

6.45.3.10 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.45.3.11 setPitchOffsetForLayer()

```
void csound::Koch::setPitchOffsetForLayer (
            int layer,
            double pitch )  [virtual]
```

References pitchOffsetsForLayers.

6.45.3.12 transform()

```
void csound::Node::transform (
            Score & score_from_children )  [virtual], [inherited]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented in csound::Cell, csound::CellRepeat, csound::CellAdd, csound::CellMultiply, csound::CellReflect, csound::CellSelect, csound::CellRemove, csound::CellChord, csound::CellRandom, csound::CellShuffle, csound::CounterpointNode, csound::RemoveDuplicates, csound::Transformer, csound::Random, csound::Rescale, csound::VoiceleadingNode, csound::LispTransformer, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.45.3.13 traverse()

```
void csound::Koch::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented from csound::Node.

References csound::Score::append(), csound::Node::children, csound::Event::DURATION, csound::Event::ELEMENT_COUNT, csound::Score::findScale(), csound::Score::getDuration(), csound::Event::getDuration(), csound::Event::getKey(), csound::Node::getLocalCoordinates(), csound::Event::getTime(), csound::Event::getVelocity(), csound::Event::HOMOGENEITY, csound::Event::KEY, csound::System::message(), pitchOffsetsForLayers, csound::Score::scaleActualMinima, csound::ScoreNode::score, csound::Score::sort(), csound::Event::TIME, and csound::Event::VELOCITY.

6.45.4 Field Documentation

6.45.4.1 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), traverse(), and csound::Sequence::traverse().

6.45.4.2 duration

```
double csound::ScoreNode::duration  [inherited]
```

If not 0, the score is rescaled to this duration.

Referenced by csound::ScoreNode::generate(), csound::ExternalNode::generateLocally(), and csound::Stack::getDuration().

6.45.4.3 importFilename

```
std::string csound::ScoreNode::importFilename  [inherited]
```

Referenced by csound::ScoreNode::generate().

6.45.4.4 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.45.4.5 pitchOffsetsForLayers

```
std::map<int, double> csound::Koch::pitchOffsetsForLayers
```

Referenced by setPitchOffsetForLayer(), and traverse().

6.45.4.6 score

```
Score csound::ScoreNode::score  [protected], [inherited]
```

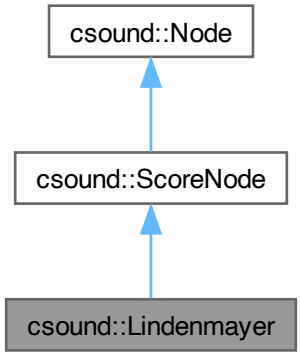
Referenced by csound::StrangeAttractor::evaluateAttractor(), csound::ExternalNode::generate(), csound::ScoreNode::generate(), csound::MCRM::generate(), csound::ExternalNode::generateLocally(), csound::ImageToScore2::generateLocally(), csound::Lindenmayer::generateLocally(), csound::Rescale::getRescale(), csound::ScoreNode::getScore(), csound::Lindenmayer::interpre csound::MCRM::iterate(), csound::StrangeAttractor::iterate_without_rendering(), csound::KMeansMCRM::means_to_notes(), csound::ImageToScore2::pixel_to_event(), csound::StrangeAttractor::render(), csound::Rescale::Rescale(), csound::Rescale::setRescale(csound::Cell::transform(), csound::Rescale::transform(), csound::CMaskNode::translate_to_silence(), csound::Intercut::traverse(), csound::Stack::traverse(), traverse(), and csound::Lindenmayer::updateActual().

6.46 csound::Lindenmayer Class Reference

This class implements a Lindenmayer system in music space for a turtle that writes either notes into a score, or Jones-Parks grains into a memory soundfile.

```
#include <Lindenmayer.hpp>
```

Inheritance diagram for csound::Lindenmayer:



Public Member Functions

- virtual void addChild (Node *node)

  Adds an immediate child Node to this.
- virtual void addRule (std::string command, std::string replacement)
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

Recursively clears all child Nodes of this.

- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &collectingScore)

  Optionally generate notes into the score.
- virtual void generateLocally ()
- virtual double getAngle () const
- virtual std::string getAxiom () const
- virtual Node * getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual int getIterationCount () const
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual std::string getReplacement (std::string command)
- virtual Score & getScore ()
- Lindenmayer ()
- virtual void setAngle (double angle)
- virtual void setAxiom (std::string axiom)
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void setIterationCount (int count)
- virtual void transform (Score &score_from_children)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.
- virtual ~Lindenmayer ()

Data Fields

- std::vector< Node * > children

  Child Nodes, if any.
- double duration

  If not 0, the score is rescaled to this duration.
- std::string importFilename

Protected Member Functions

- virtual Eigen::MatrixXd createRotation (int dimension1, int dimension2, double angle) const
- virtual int getDimension (char dimension) const
- virtual void initialize ()
- virtual void interpret (std::string command, bool render)
- virtual void rewrite ()
- virtual void updateActual (Event &event)

Protected Attributes

- double angle
- std::string axiom
- clock_t beganAt
- clock_t elapsed
- clock_t endedAt
- int iterationCount
- Eigen::MatrixXd localCoordinates
- std::map< std::string, std::string > rules
- Score score
- Event turtle
- Event turtleOrientation
- std::stack< Event > turtleOrientationStack
- std::stack< Event > turtleStack
- Event turtleStep
- std::stack< Event > turtleStepStack

6.46.1 Detailed Description

This class implements a Lindenmayer system in music space for a turtle that writes either notes into a score, or Jones-Parks grains into a memory soundfile.

The Z dimension of note space is used for chirp rate. The actions of the turtle are rescaled to fit the specified bounding hypercube. The turtle commands are represented by letters (all n default to 1):

- G = Write the current state of the turtle into the soundfile as a grain.
- Mn = Translate the turtle by adding to its state its step times its orientation times n.
- Rabn = Rotate the turtle from dimension a to dimension b by angle 2 pi / (angleCount $*$ n)
- Uan = Vary the turtle state on dimension a by a normalized (-1 through +1) uniformly distributed random variable times n.
- Gan = Vary the turtle state on dimension a by a normalized (-1 through +1) Gaussian random variable times n.
- T=an = Assign to dimension a of the turtle state the value n.
- T$*$an = Multiply dimension a of the turtle state by n.
- T/an = Divide dimension a of the turtle state by n.
- T+an = Add to dimension a of the turtle state the value n.
- T-an = Subtract from dimension a of the turtle state the value n.
- S=an = Assign to dimension a of the turtle step the value n.
- S$*$an = Multiply dimension a of the turtle step by n.
- S/an = Divide dimension a of the turtle step by n.
- S+an = Add to dimension a of the turtle step the value n.
- S-an = Subtract from dimension a of the turtle step the value n.

- [= Push the current state of the turtle state onto a stack.
-] = Pop the current state of the turtle from the stack.

The abbreviations for the dimensions are:

1. i = instrument
2. t = time
3. d = duration
4. k = MIDI key number
5. v = MIDI velocity number
6. p = phase
7. x = pan
8. y = height
9. z = depth
10. s = pitch-class set as Mason number

6.46.2 Constructor & Destructor Documentation

6.46.2.1 Lindenmayer()

```
csound::Lindenmayer::Lindenmayer ( )
```

6.46.2.2 ~Lindenmayer()

```
csound::Lindenmayer::~Lindenmayer ( )  [virtual]
```

6.46.3 Member Function Documentation

6.46.3.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.46.3.2 addRule()

```
void csound::Lindenmayer::addRule (
            std::string command,
            std::string replacement )  [virtual]
```

References csound::fundamentalDomainByPredicate(), and rules.

6.46.3.3 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.46.3.4 clear()

```
void csound::Lindenmayer::clear ( )  [virtual]
```

Recursively clears all child Nodes of this.

Reimplemented from csound::Node.

References initialize(), rules, turtleOrientationStack, turtleStack, and turtleStepStack.

6.46.3.5 createRotation()

```
Eigen::MatrixXd csound::Lindenmayer::createRotation (
            int dimension1,
            int dimension2,
            double angle ) const  [protected], [virtual]
```

References angle, csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by interpret().

6.46.3.6 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.46.3.7 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.46.3.8 generate()

```
void csound::ScoreNode::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented from csound::Node.

Reimplemented in csound::ExternalNode, and csound::MCRM.

References csound::ScoreNode::duration, csound::fundamentalDomainByPredicate(), csound::Score::getCsoundScoreHeader(), csound::ScoreNode::importFilename, csound::Score::load(), csound::Score::process(), csound::ScoreNode::score, csound::Score::setDuration(), and csound::Score::sort().

Referenced by csound::MCRM::generate().

6.46.3.9 generateLocally()

```
void csound::Lindenmayer::generateLocally ( )  [virtual]
```

References axiom, csound::MidiFile::CHANNEL_NOTE_ON, csound::fundamentalDomainByPredicate(), getReplacement(), initialize(), interpret(), iterationCount, csound::Score::scaleActualMinima, csound::Score::scaleActualRanges, csound::ScoreNode::score, csound::Event::setStatus(), and turtle.

6.46.3.10 getAngle()

```
double csound::Lindenmayer::getAngle ( ) const  [virtual]
```

References angle.

6.46.3.11 getAxiom()

```
std::string csound::Lindenmayer::getAxiom ( ) const  [virtual]
```

References axiom.

6.46.3.12 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.46.3.13 getDimension()

```
int csound::Lindenmayer::getDimension (
            char dimension ) const  [protected], [virtual]
```

References csound::Event::DEPTH, csound::Event::DURATION, csound::Event::HEIGHT, csound::Event::INSTRUMENT, csound::Event::KEY, csound::Event::PAN, csound::Event::PHASE, csound::Event::PITCHES, csound::Event::TIME, and csound::Event::VELOCITY.

Referenced by interpret().

6.46.3.14 getIterationCount()

```
int csound::Lindenmayer::getIterationCount ( ) const  [virtual]
```

References iterationCount.

6.46.3.15 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.46.3.16 getReplacement()

```
std::string csound::Lindenmayer::getReplacement (
            std::string command )  [virtual]
```

References csound::fundamentalDomainByPredicate(), and rules.

Referenced by generateLocally().

6.46.3.17 getScore()

```
Score & csound::ScoreNode::getScore ( )  [virtual], [inherited]
```

References csound::ScoreNode::score.

Referenced by main().

6.46.3.18 initialize()

```
void csound::Lindenmayer::initialize ( )  [protected], [virtual]
```

References csound::Event::HOMOGENEITY, csound::Event::TIME, turtle, turtleOrientation, and turtleStep.

Referenced by clear(), and generateLocally().

6.46.3.19 interpret()

```
void csound::Lindenmayer::interpret (
            std::string command,
            bool render )  [protected], [virtual]
```

References angle, createRotation(), csound::System::error(), csound::fundamentalDomainByPredicate(), getDimension(), csound::Event::HOMOGENEITY, csound::Conversions::modulus(), csound::Event::PITCHES, csound::ScoreNode::score, csound::Conversions::stringToDouble(), csound::Conversions::trim(), turtle, turtleOrientation, turtleOrientationStack, turtleStack, turtleStep, turtleStepStack, and updateActual().

Referenced by generateLocally().

6.46.3.20 rewrite()

```
void csound::Lindenmayer::rewrite ( )  [protected], [virtual]
```

References axiom, csound::System::debug(), csound::fundamentalDomainByPredicate(), iterationCount, and rules.

6.46.3.21 setAngle()

```
void csound::Lindenmayer::setAngle (
            double angle )  [virtual]
```

References angle.

6.46.3.22 setAxiom()

```
void csound::Lindenmayer::setAxiom (
            std::string axiom )  [virtual]
```

References axiom.

6.46.3.23 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.46.3.24 setIterationCount()

```
void csound::Lindenmayer::setIterationCount (
            int count )  [virtual]
```

References iterationCount.

6.46.3.25 transform()

```
void csound::Node::transform (
            Score & score_from_children )  [virtual], [inherited]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented in csound::Cell, csound::CellRepeat, csound::CellAdd, csound::CellMultiply, csound::CellReflect, csound::CellSelect, csound::CellRemove, csound::CellChord, csound::CellRandom, csound::CellShuffle, csound::CounterpointNode, csound::RemoveDuplicates, csound::Transformer, csound::Random, csound::Rescale, csound::VoiceleadingNode, csound::LispTransformer, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.46.3.26 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord and csound::Node::transform().

6.46.3.27 updateActual()

```
void csound::Lindenmayer::updateActual (
            Event & event )  [protected], [virtual]
```

References csound::fundamentalDomainByPredicate(), csound::Score::scaleActualMinima, csound::Score::scaleActualRanges, and csound::ScoreNode::score.

Referenced by interpret().

6.46.4 Field Documentation

6.46.4.1 angle

```
double csound::Lindenmayer::angle  [protected]
```

Referenced by createRotation(), getAngle(), interpret(), and setAngle().

6.46.4.2 axiom

```
std::string csound::Lindenmayer::axiom  [protected]
```

Referenced by generateLocally(), getAxiom(), rewrite(), and setAxiom().

6.46.4.3 beganAt

```
clock_t csound::Lindenmayer::beganAt  [protected]
```

6.46.4.4 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.46.4.5 duration

```
double csound::ScoreNode::duration  [inherited]
```

If not 0, the score is rescaled to this duration.

Referenced by csound::ScoreNode::generate(), csound::ExternalNode::generateLocally(), and csound::Stack::getDuration().

6.46.4.6 elapsed

```
clock_t csound::Lindenmayer::elapsed  [protected]
```

6.46.4.7 endedAt

```
clock_t csound::Lindenmayer::endedAt  [protected]
```

6.46.4.8 importFilename

```
std::string csound::ScoreNode::importFilename  [inherited]
```

Referenced by csound::ScoreNode::generate().

6.46.4.9 iterationCount

```
int csound::Lindenmayer::iterationCount  [protected]
```

Referenced by generateLocally(), getIterationCount(), rewrite(), and setIterationCount().

6.46.4.10 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.46.4.11 rules

```
std::map<std::string, std::string> csound::Lindenmayer::rules  [protected]
```

Referenced by addRule(), clear(), getReplacement(), and rewrite().

6.46.4.12 score

```
Score csound::ScoreNode::score  [protected], [inherited]
```

Referenced by csound::StrangeAttractor::evaluateAttractor(), csound::ExternalNode::generate(), csound::ScoreNode::generate(), csound::MCRM::generate(), csound::ExternalNode::generateLocally(), csound::ImageToScore2::generateLocally(), generateLocally(), csound::Rescale::getRescale(), csound::ScoreNode::getScore(), interpret(), csound::MCRM::iterate(), csound::StrangeAttractor::iterate_without_rendering(), csound::KMeansMCRM::means_to_notes(), csound::ImageToScore2::pixel_to_eve csound::StrangeAttractor::render(), csound::Rescale::Rescale(), csound::Rescale::setRescale(), csound::Cell::transform(), csound::Rescale::transform(), csound::CMaskNode::translate_to_silence(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and updateActual().

6.46.4.13 turtle

```
Event csound::Lindenmayer::turtle  [protected]
```

Referenced by generateLocally(), initialize(), and interpret().

6.46.4.14 turtleOrientation

```
Event csound::Lindenmayer::turtleOrientation  [protected]
```

Referenced by initialize(), and interpret().

6.46.4.15 turtleOrientationStack

```
std::stack<Event> csound::Lindenmayer::turtleOrientationStack  [protected]
```

Referenced by clear(), and interpret().

6.46.4.16 turtleStack

```
std::stack<Event> csound::Lindenmayer::turtleStack  [protected]
```

Referenced by clear(), and interpret().

6.46.4.17 turtleStep

```
Event csound::Lindenmayer::turtleStep  [protected]
```

Referenced by initialize(), and interpret().

6.46.4.18 turtleStepStack

```
std::stack<Event> csound::Lindenmayer::turtleStepStack  [protected]
```

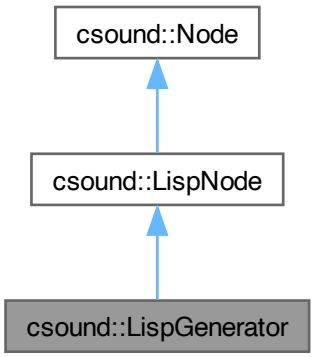
Referenced by clear(), and interpret().

6.47 csound::LispGenerator Class Reference

Node that uses Lisp code to generate Events.

```
#include <Lisp.hpp>
```

Inheritance diagram for csound::LispGenerator:



Public Member Functions

- virtual void addChild (Node *node)

  Adds an immediate child Node to this.
- virtual void appendTopLevelForm (const std::string code)

  Sets the Lisp code that will generate or transform a Silence score.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &score_from_this)

  To generate a score_from_this, the Lisp code should end by returning a Common Music seq object, which will be translated to score_from_this.
- virtual Node * getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual double getNumberFromForm (const std::string &form)
- virtual std::string getStringFromForm (const std::string &form)
- virtual std::vector< std::string > & getTopLevelForms ()
- LispGenerator ()
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void transform (Score &score_from_children)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.
- virtual ~LispGenerator ()

Data Fields

- std::vector< Node * > children

  Child Nodes, if any.

Protected Attributes

- Eigen::MatrixXd localCoordinates
- std::vector< std::string > top_level_forms

6.47.1 Detailed Description

Node that uses Lisp code to generate Events.

6.47.2 Constructor & Destructor Documentation

6.47.2.1 LispGenerator()

```
csound::LispGenerator::LispGenerator ( )
```

6.47.2.2 ~LispGenerator()

```
csound::LispGenerator::~LispGenerator ( )  [virtual]
```

6.47.3 Member Function Documentation

6.47.3.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.47.3.2 appendTopLevelForm()

```
void csound::LispNode::appendTopLevelForm (
            const std::string code )  [virtual], [inherited]
```

Sets the Lisp code that will generate or transform a Silence score.

Please note, each top-level form must be appended in sequence, e.g. `require` and `in-package` should be added before a `progn` containing score generating forms.

References csound::fundamentalDomainByPredicate(), and csound::LispNode::top_level_forms.

Referenced by main().

6.47.3.3 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.47.3.4 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.47.3.5 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.47.3.6 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.47.3.7 generate()

```
void csound::LispGenerator::generate (
            Score & score_from_this )  [virtual]
```

To generate a score_from_this, the Lisp code should end by returning a Common Music seq object, which will be translated to score_from_this.

Reimplemented from csound::Node.

References csound::System::debug(), csound::evaluate_form(), csound::fundamentalDomainByPredicate(), csound::System::inform(), csound::seqToScore(), and csound::LispNode::top_level_forms.

6.47.3.8 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.47.3.9 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.47.3.10 getNumberFromForm()

```
double csound::LispNode::getNumberFromForm (
            const std::string & form )  [virtual], [inherited]
```

References csound::evaluate_form(), and csound::fundamentalDomainByPredicate().

6.47.3.11 getStringFromForm()

```
std::string csound::LispNode::getStringFromForm (
            const std::string & form )  [virtual], [inherited]
```

References csound::evaluate_form(), csound::fundamentalDomainByPredicate(), and csound::to_std_string().

6.47.3.12 getTopLevelForms()

```
std::vector< std::string > & csound::LispNode::getTopLevelForms ( )  [virtual], [inherited]
```

References csound::LispNode::top_level_forms.

6.47.3.13 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.47.3.14 transform()

```
void csound::Node::transform (
            Score & score_from_children )  [virtual], [inherited]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented in csound::Cell, csound::CellRepeat, csound::CellAdd, csound::CellMultiply, csound::CellReflect, csound::CellSelect, csound::CellRemove, csound::CellChord, csound::CellRandom, csound::CellShuffle, csound::CounterpointNode, csound::RemoveDuplicates, csound::Transformer, csound::Random, csound::Rescale, csound::VoiceleadingNode, csound::LispTransformer, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.47.3.15 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord and csound::Node::transform().

6.47.4 Field Documentation

6.47.4.1 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.47.4.2 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.47.4.3 top_level_forms

```
std::vector<std::string> csound::LispNode::top_level_forms  [protected], [inherited]
```

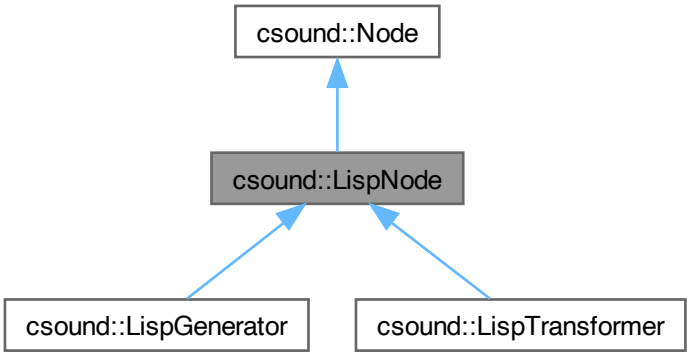
Referenced by csound::LispNode::appendTopLevelForm(), generate(), csound::LispNode::getTopLevelForms(), and csound::LispTransformer::transform().

6.48 csound::LispNode Class Reference

Base class for Nodes that can use embedded Lisp code to generate or transform Events.

`#include <Lisp.hpp>`

Inheritance diagram for csound::LispNode:



Public Member Functions

- virtual void addChild (Node *node)
  Adds an immediate child Node to this.
- virtual void appendTopLevelForm (const std::string code)
  Sets the Lisp code that will generate or transform a Silence score.
- virtual size_t childCount () const
  Returns the number of immediate children of this.
- virtual void clear ()
  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()
  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)
  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &score_from_this)
  Optionally generate notes into the score.
- virtual Node * getChild (size_t index)
  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const
  Returns the local transformation of coordinate system.
- virtual double getNumberFromForm (const std::string &form)
- virtual std::string getStringFromForm (const std::string &form)

- virtual std::vector< std::string > & getTopLevelForms ()
- LispNode ()
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void transform (Score &score_from_children)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.
- virtual ~LispNode ()

Data Fields

- std::vector< Node * > children

  Child Nodes, if any.

Protected Attributes

- Eigen::MatrixXd localCoordinates
- std::vector< std::string > top_level_forms

6.48.1 Detailed Description

Base class for Nodes that can use embedded Lisp code to generate or transform Events.

In order to use the Common Music or nudruz packages to do this, first execute the following sequence of calls:

```
csound::initialize_ecl(argc, (char **)argv);
csound::evaluate_form("(require :asdf)");
csound::evaluate_form("(require :nudruz)");
csound::evaluate_form("(in-package :cm)");
```

6.48.2 Constructor & Destructor Documentation

6.48.2.1 LispNode()

```
csound::LispNode::LispNode ( )
```

6.48.2.2 ~LispNode()

```
csound::LispNode::~LispNode ( )  [virtual]
```

6.48.3 Member Function Documentation

6.48.3.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.48.3.2 appendTopLevelForm()

```
void csound::LispNode::appendTopLevelForm (
            const std::string code )  [virtual]
```

Sets the Lisp code that will generate or transform a Silence score.

Please note, each top-level form must be appended in sequence, e.g. `require` and `in-package` should be added before a `progn` containing score generating forms.

References csound::fundamentalDomainByPredicate(), and top_level_forms.

Referenced by main().

6.48.3.3 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.48.3.4 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.48.3.5 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.48.3.6 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.48.3.7 generate()

```
void csound::Node::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented in csound::ExternalNode, csound::ScoreNode, csound::ChordLindenmayer, csound::MCRM, csound::Generator, csound::Random, csound::LispGenerator, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.48.3.8 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.48.3.9 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.48.3.10 getNumberFromForm()

```
double csound::LispNode::getNumberFromForm (
            const std::string & form )  [virtual]
```

References csound::evaluate_form(), and csound::fundamentalDomainByPredicate().

6.48.3.11 getStringFromForm()

```
std::string csound::LispNode::getStringFromForm (
            const std::string & form )  [virtual]
```

References csound::evaluate_form(), csound::fundamentalDomainByPredicate(), and csound::to_std_string().

6.48.3.12 getTopLevelForms()

```
std::vector< std::string > & csound::LispNode::getTopLevelForms ( )  [virtual]
```

References top_level_forms.

6.48.3.13 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.48.3.14 transform()

```
void csound::Node::transform (
            Score & score_from_children )  [virtual], [inherited]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented in csound::Cell, csound::CellRepeat, csound::CellAdd, csound::CellMultiply, csound::CellReflect, csound::CellSelect, csound::CellRemove, csound::CellChord, csound::CellRandom, csound::CellShuffle, csound::CounterpointNode, csound::RemoveDuplicates, csound::Transformer, csound::Random, csound::Rescale, csound::VoiceleadingNode, csound::LispTransformer, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.48.3.15 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord and csound::Node::transform().

6.48.4 Field Documentation

6.48.4.1 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.48.4.2 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.48.4.3 top_level_forms

```
std::vector<std::string> csound::LispNode::top_level_forms  [protected]
```

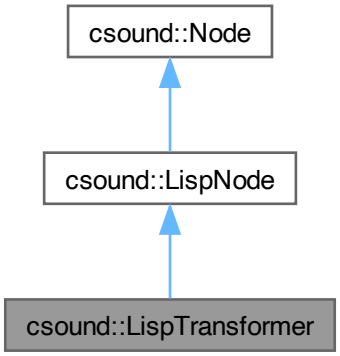
Referenced by appendTopLevelForm(), csound::LispGenerator::generate(), getTopLevelForms(), and csound::LispTransformer::transform(

6.49 csound::LispTransformer Class Reference

Node that uses Lisp code to transform Events produced by child Nodes.

```
#include <Lisp.hpp>
```

Inheritance diagram for csound::LispTransformer:



Public Member Functions

- virtual void addChild (Node *node)

  Adds an immediate child Node to this.
- virtual void appendTopLevelForm (const std::string code)

  Sets the Lisp code that will generate or transform a Silence score.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &score_from_this)

  Optionally generate notes into the score.
- virtual Node * getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual double getNumberFromForm (const std::string &form)
- virtual std::string getStringFromForm (const std::string &form)
- virtual std::vector< std::string > & getTopLevelForms ()
- LispTransformer ()
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void transform (Score &score_from_children)

  To transform score_from_children, the Lisp code should operate on a Common Music seq instance named "score-from-children", which will be translated from score_from_children before transformation, and then translated back to score_↩ from_children after transformation.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.
- virtual ~LispTransformer ()

Data Fields

- std::vector< Node * > children

  Child Nodes, if any.

Protected Attributes

- Eigen::MatrixXd localCoordinates
- std::vector< std::string > top_level_forms

6.49.1 Detailed Description

Node that uses Lisp code to transform Events produced by child Nodes.

6.49.2 Constructor & Destructor Documentation

6.49.2.1 LispTransformer()

```
csound::LispTransformer::LispTransformer ( )
```

6.49.2.2 ~LispTransformer()

```
csound::LispTransformer::~LispTransformer ( )  [virtual]
```

6.49.3 Member Function Documentation

6.49.3.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.49.3.2 appendTopLevelForm()

```
void csound::LispNode::appendTopLevelForm (
            const std::string code )  [virtual], [inherited]
```

Sets the Lisp code that will generate or transform a Silence score.

Please note, each top-level form must be appended in sequence, e.g. `require` and `in-package` should be added before a `progn` containing score generating forms.

References csound::fundamentalDomainByPredicate(), and csound::LispNode::top_level_forms.

Referenced by main().

6.49.3.3 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.49.3.4 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.49.3.5 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.49.3.6 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.49.3.7 generate()

```
void csound::Node::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented in csound::ExternalNode, csound::ScoreNode, csound::ChordLindenmayer, csound::MCRM, csound::Generator, csound::Random, csound::LispGenerator, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.49.3.8 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.49.3.9 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.49.3.10 getNumberFromForm()

```
double csound::LispNode::getNumberFromForm (
            const std::string & form )  [virtual], [inherited]
```

References csound::evaluate_form(), and csound::fundamentalDomainByPredicate().

6.49.3.11 getStringFromForm()

```
std::string csound::LispNode::getStringFromForm (
            const std::string & form )  [virtual], [inherited]
```

References csound::evaluate_form(), csound::fundamentalDomainByPredicate(), and csound::to_std_string().

6.49.3.12 getTopLevelForms()

```
std::vector< std::string > & csound::LispNode::getTopLevelForms ( )  [virtual], [inherited]
```

References csound::LispNode::top_level_forms.

6.49.3.13 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.49.3.14 transform()

```
void csound::LispTransformer::transform (
            Score & score_from_children )  [virtual]
```

To transform score_from_children, the Lisp code should operate on a Common Music seq instance named "score-from-children", which will be translated from score_from_children before transformation, and then translated back to score_from_children after transformation.

Please note, before doing any of this, the asdf, nudruz, and cm packages will be loaded.

Reimplemented from csound::Node.

References csound::evaluate_form(), csound::fundamentalDomainByPredicate(), csound::System::inform(), csound::scoreToSeq(), csound::seqToScore(), and csound::LispNode::top_level_forms.

6.49.3.15 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord and csound::Node::transform().

6.49.4 Field Documentation

6.49.4.1 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.49.4.2 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.49.4.3 top_level_forms

```
std::vector<std::string> csound::LispNode::top_level_forms  [protected], [inherited]
```

Referenced by csound::LispNode::appendTopLevelForm(), csound::LispGenerator::generate(), csound::LispNode::getTopLevelForms(), and transform().

6.50 csound::Logger Class Reference

`#include <System.hpp>`

Public Member Functions

- Logger ()
- virtual void write (const char *text)
- virtual ~Logger ()

6.50.1 Constructor & Destructor Documentation

6.50.1.1 Logger()

```
SILENCE_PUBLIC csound::Logger::Logger ( )
```

6.50.1.2 ~Logger()

```
SILENCE_PUBLIC csound::Logger::~Logger ( )  [virtual]
```

6.50.2 Member Function Documentation

6.50.2.1 write()

```
SILENCE_PUBLIC void csound::Logger::write (
            const char * text )  [virtual]
```

References csound::fundamentalDomainByPredicate().

6.51 csound::MatrixCell Struct Reference

Public Member Functions

- MatrixCell ()

Data Fields

- std::vector< double > a
- std::vector< double > b
- double d
- size_t i
- size_t j
- std::vector< double > s
- std::vector< double > v

6.51.1 Constructor & Destructor Documentation

6.51.1.1 MatrixCell()

```
csound::MatrixCell::MatrixCell ( )  [inline]
```

6.51.2 Field Documentation

6.51.2.1 a

```
std::vector<double> csound::MatrixCell::a
```

6.51.2.2 b

```
std::vector<double> csound::MatrixCell::b
```

6.51.2.3 d

```
double csound::MatrixCell::d
```

Referenced by csound::minimumCell().

6.51.2.4 i

```
size_t csound::MatrixCell::i
```

6.51.2.5 j

```
size_t csound::MatrixCell::j
```

6.51.2.6 s

```
std::vector<double> csound::MatrixCell::s
```

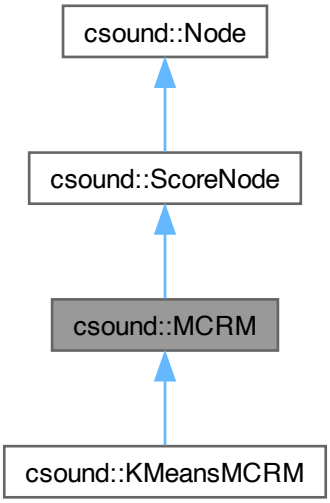
6.51.2.7 v

```
std::vector<double> csound::MatrixCell::v
```

6.52 csound::MCRM Class Reference

`#include <MCRM.hpp>`

Inheritance diagram for csound::MCRM:



Public Member Functions

- virtual void addChild (Node *node)

  Adds an immediate child Node to this.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &score)

  Optionally generate notes into the score.
- virtual void generateLocally ()
- virtual Node * getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.

- virtual Score & getScore ()
- MCRM ()
- void resize (size_t transformations)
- void setDepth (int depth)
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- void setTransformationElement (size_t index, size_t row, size_t column, double value)
- void setWeight (size_t precursor, size_t successor, double weight)
- virtual void transform (Score &score_from_children)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.
- virtual ~MCRM ()

Data Fields

- std::vector< Node * > children

  Child Nodes, if any.
- double duration

  If not 0, the score is rescaled to this duration.
- std::string importFilename

Protected Member Functions

- virtual void iterate (int depth, size_t p, const Event &event, double weight)

Protected Attributes

- int depth
- Eigen::MatrixXd localCoordinates
- Score score
- std::vector< Eigen::MatrixXd > transformations
- Eigen::MatrixXd weights

6.52.1 Constructor & Destructor Documentation

6.52.1.1 MCRM()

```
csound::MCRM::MCRM ( )
```

6.52.1.2 ~MCRM()

```
csound::MCRM::~MCRM ( )  [virtual]
```

6.52.2 Member Function Documentation

6.52.2.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.52.2.2 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.52.2.3 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.52.2.4 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and resize().

6.52.2.5 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.52.2.6 generate()

```
void csound::MCRM::generate (
            Score & score_from_this )  [virtual]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented from csound::ScoreNode.

References csound::ScoreNode::generate(), generateLocally(), and csound::ScoreNode::score.

6.52.2.7 generateLocally()

```
void csound::MCRM::generateLocally ( )  [virtual]
```

Reimplemented in csound::KMeansMCRM.

References depth, csound::fundamentalDomainByPredicate(), iterate(), and csound::Event::setStatus().

Referenced by generate().

6.52.2.8 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.52.2.9 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.52.2.10 getScore()

```
Score & csound::ScoreNode::getScore ( )  [virtual], [inherited]
```

References csound::ScoreNode::score.

Referenced by main().

6.52.2.11 iterate()

```
void csound::MCRM::iterate (
            int depth,
            size_t p,
            const Event & event,
            double weight )  [protected], [virtual]
```

Reimplemented in csound::KMeansMCRM.

References csound::fundamentalDomainByPredicate(), iterate(), csound::ScoreNode::score, transformations, and weights.

Referenced by generateLocally(), and iterate().

6.52.2.12 resize()

```
void csound::MCRM::resize (
            size_t transformations )
```

References csound::Node::createTransform(), csound::fundamentalDomainByPredicate(), transformations, and weights.

6.52.2.13 setDepth()

```
void csound::MCRM::setDepth (
            int depth )
```

References depth.

6.52.2.14 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.52.2.15 setTransformationElement()

```
void csound::MCRM::setTransformationElement (
            size_t index,
            size_t row,
            size_t column,
            double value )
```

References transformations.

6.52.2.16 setWeight()

```
void csound::MCRM::setWeight (
            size_t precursor,
            size_t successor,
            double weight )
```

References csound::fundamentalDomainByPredicate(), and weights.

6.52.2.17 transform()

```
void csound::Node::transform (
            Score & score_from_children )  [virtual], [inherited]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented in csound::Cell, csound::CellRepeat, csound::CellAdd, csound::CellMultiply, csound::CellReflect, csound::CellSelect, csound::CellRemove, csound::CellChord, csound::CellRandom, csound::CellShuffle, csound::CounterpointNode, csound::RemoveDuplicates, csound::Transformer, csound::Random, csound::Rescale, csound::VoiceleadingNode, csound::LispTransformer, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.52.2.18 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord and csound::Node::transform().

6.52.3 Field Documentation

6.52.3.1 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.52.3.2 depth

```
int csound::MCRM::depth  [protected]
```

Referenced by csound::KMeansMCRM::deterministic_algorithm(), generateLocally(), and setDepth().

6.52.3.3 duration

```
double csound::ScoreNode::duration  [inherited]
```

If not 0, the score is rescaled to this duration.

Referenced by csound::ScoreNode::generate(), csound::ExternalNode::generateLocally(), and csound::Stack::getDuration().

6.52.3.4 importFilename

```
std::string csound::ScoreNode::importFilename  [inherited]
```

Referenced by csound::ScoreNode::generate().

6.52.3.5 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.52.3.6 score

```
Score csound::ScoreNode::score  [protected], [inherited]
```

Referenced by csound::StrangeAttractor::evaluateAttractor(), csound::ExternalNode::generate(), csound::ScoreNode::generate(), generate(), csound::ExternalNode::generateLocally(), csound::ImageToScore2::generateLocally(), csound::Lindenmayer::generateLocally csound::Rescale::getRescale(), csound::ScoreNode::getScore(), csound::Lindenmayer::interpret(), iterate(), csound::StrangeAttractor::iter csound::KMeansMCRM::means_to_notes(), csound::ImageToScore2::pixel_to_event(), csound::StrangeAttractor::render(), csound::Rescale::Rescale(), csound::Rescale::setRescale(), csound::Cell::transform(), csound::Rescale::transform(), csound::CMaskNode::translate_to_silence(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Lindenmayer::updateActual().

6.52.3.7 transformations

```
std::vector< Eigen::MatrixXd > csound::MCRM::transformations  [protected]
```

Referenced by iterate(), csound::KMeansMCRM::iterate(), csound::KMeansMCRM::random_algorithm(), resize(), and setTransformationElement().

6.52.3.8 weights

```
Eigen::MatrixXd csound::MCRM::weights  [protected]
```

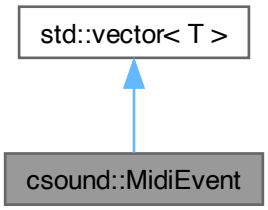
Referenced by iterate(), csound::KMeansMCRM::iterate(), csound::KMeansMCRM::random_algorithm(), resize(), and setWeight().

6.53 csound::MidiEvent Class Reference

This class is used to store ALL Midi messages.

`#include <Midifile.hpp>`

Inheritance diagram for csound::MidiEvent:



Public Member Functions

- virtual int getChannelNybble () const
- virtual int getKey () const
- virtual unsigned char getMetaData (int i) const
- virtual size_t getMetaSize () const
- virtual int getMetaType () const
- virtual int getStatus () const
- virtual int getStatusNybble () const
- virtual int getVelocity () const
- virtual bool isChannelVoiceMessage () const
- virtual bool isNoteOff () const
- virtual bool isNoteOn () const
- virtual bool matchesNoteOffEvent (const MidiEvent &offEvent) const
- MidiEvent ()
- MidiEvent (const MidiEvent &a)
- MidiEvent & operator= (const MidiEvent &a)
- virtual unsigned char readByte (std::istream &stream)
- virtual void readIn (std::istream &stream, MidiFile &midiFile)
- virtual std::string toString () const
- virtual void writeOut (std::ostream &stream, const MidiFile &midiFile, int lastTick) const
- virtual ~MidiEvent ()

Data Fields

- T elements

  STL member.
- int ticks
- double time

Friends

- bool operator< (const MidiEvent &a, const MidiEvent &b)

6.53.1 Detailed Description

This class is used to store ALL Midi messages.

6.53.2 Constructor & Destructor Documentation

6.53.2.1 MidiEvent() [1/2]

```
csound::MidiEvent::MidiEvent ( )
```

6.53.2.2 MidiEvent() [2/2]

```
csound::MidiEvent::MidiEvent (
            const MidiEvent & a )
```

6.53.2.3 ~MidiEvent()

```
csound::MidiEvent::~MidiEvent ( )  [virtual]
```

6.53.3 Member Function Documentation

6.53.3.1 getChannelNybble()

```
int csound::MidiEvent::getChannelNybble ( ) const  [virtual]
```

Referenced by matchesNoteOffEvent().

6.53.3.2 getKey()

```
int csound::MidiEvent::getKey ( ) const  [virtual]
```

Referenced by matchesNoteOffEvent().

6.53.3.3 getMetaData()

```
unsigned char csound::MidiEvent::getMetaData (
            int i ) const  [virtual]
```

Referenced by readIn().

6.53.3.4 getMetaSize()

```
size_t csound::MidiEvent::getMetaSize ( ) const  [virtual]
```

References csound::fundamentalDomainByPredicate().

Referenced by writeOut().

6.53.3.5 getMetaType()

```
int csound::MidiEvent::getMetaType ( ) const  [virtual]
```

References getStatus(), and csound::MidiFile::META_EVENT.

Referenced by readIn(), and writeOut().

6.53.3.6 getStatus()

```
int csound::MidiEvent::getStatus ( ) const  [virtual]
```

Referenced by getMetaType(), readIn(), toString(), and writeOut().

6.53.3.7 getStatusNybble()

```
int csound::MidiEvent::getStatusNybble ( ) const  [virtual]
```

Referenced by isChannelVoiceMessage(), isNoteOff(), isNoteOn(), readIn(), and writeOut().

6.53.3.8 getVelocity()

```
int csound::MidiEvent::getVelocity ( ) const  [virtual]
```

Referenced by isNoteOff(), and isNoteOn().

6.53.3.9 isChannelVoiceMessage()

```
bool csound::MidiEvent::isChannelVoiceMessage ( ) const  [virtual]
```

References csound::MidiFile::CHANNEL_NOTE_OFF, csound::MidiFile::CHANNEL_PITCH_BEND, and getStatusNybble().

6.53.3.10 isNoteOff()

```
bool csound::MidiEvent::isNoteOff ( ) const  [virtual]
```

References csound::MidiFile::CHANNEL_NOTE_OFF, csound::MidiFile::CHANNEL_NOTE_ON, getStatusNybble(), and getVelocity().

6.53.3.11 isNoteOn()

```
bool csound::MidiEvent::isNoteOn ( ) const  [virtual]
```

References csound::MidiFile::CHANNEL_NOTE_ON, getStatusNybble(), and getVelocity().

Referenced by matchesNoteOffEvent().

6.53.3.12 matchesNoteOffEvent()

```
bool csound::MidiEvent::matchesNoteOffEvent (
            const MidiEvent & offEvent ) const  [virtual]
```

References csound::fundamentalDomainByPredicate(), getChannelNybble(), getKey(), isNoteOn(), and time.

6.53.3.13 operator=()

```
MidiEvent & csound::MidiEvent::operator= (
            const MidiEvent & a )
```

References ticks, and time.

6.53.3.14 readByte()

```
unsigned char csound::MidiEvent::readByte (
            std::istream & stream )  [virtual]
```

References csound::fundamentalDomainByPredicate().

Referenced by readIn().

6.53.3.15 readIn()

```
void csound::MidiEvent::readIn (
            std::istream & stream,
            MidiFile & midiFile )  [virtual]
```

References csound::MidiFile::CHANNEL_AFTER_TOUCH, csound::MidiFile::CHANNEL_CONTROL_CHANGE, csound::MidiFile::CHANNEL_KEY_PRESSURE, csound::MidiFile::CHANNEL_NOTE_OFF, csound::MidiFile::CHANNEL_NOTE_ON, csound::MidiFile::CHANNEL_PITCH_BEND, csound::MidiFile::CHANNEL_PROGRAM_CHANGE, csound::MidiFile::currentTime, csound::System::debug(), csound::System::error(), csound::fundamentalDomainByPredicate(), getMetaData(), getMetaType(), getStatus(), getStatusNybble(), csound::System::inform(), csound::MidiFile::META_END_OF_TRACK, csound::MidiFile::META_EVENT, csound::MidiFile::META_SEQUENCER_SPECIFIC, csound::MidiFile::META_SET_TEMPO, csound::MidiFile::META_TIME_SIGNATURE, readByte(), csound::MidiFile::readVariableLength(), csound::MidiFile::SYSTEM_EXCLUSIV ticks, time, and toString().

Referenced by csound::MidiTrack::readIn().

6.53.3.16 toString()

```
std::string csound::MidiEvent::toString ( ) const  [virtual]
```

References csound::fundamentalDomainByPredicate(), getStatus(), ticks, and time.

Referenced by readIn().

6.53.3.17 writeOut()

```
void csound::MidiEvent::writeOut (
            std::ostream & stream,
            const MidiFile & midiFile,
            int lastTick ) const  [virtual]
```

References csound::fundamentalDomainByPredicate(), getMetaSize(), getMetaType(), getStatus(), getStatusNybble(), ticks, and csound::MidiFile::writeVariableLength().

Referenced by csound::MidiTrack::writeOut().

6.53.4 Friends And Related Symbol Documentation

6.53.4.1 operator<

```
bool operator< (
            const MidiEvent & a,
            const MidiEvent & b )  [friend]
```

6.53.5 Field Documentation

6.53.5.1 elements

```
T std::vector< T >::elements  [inherited]
```

STL member.

6.53.5.2 ticks

```
int csound::MidiEvent::ticks
```

Referenced by csound::MidiFile::dump(), csound::MidiEventComparator::operator()(), operator=(), readIn(), toString(), and writeOut().

6.53.5.3 time

```
double csound::MidiEvent::time
```

Referenced by matchesNoteOffEvent(), operator=(), readIn(), and toString().

6.54 csound::MidiEventComparator Struct Reference

```
#include <Midifile.hpp>
```

Public Member Functions

- bool operator() (const MidiEvent &a, const MidiEvent &b)

6.54.1 Member Function Documentation

6.54.1.1 operator()()

```
bool csound::MidiEventComparator::operator() (
            const MidiEvent & a,
            const MidiEvent & b )  [inline]
```

References csound::MidiEvent::ticks.

6.55 csound::MidiFile Class Reference

Reads and writes format 0 and format 1 standard MIDI files.

`#include <Midifile.hpp>`

Public Types

- enum MetaEventTypes {
  META_SEQUENCE_NUMBER = 0x00 , META_TEXT_EVENT = 0x01 , META_COPYRIGHT_NOTICE = 0x02 ,
  META_SEQUENCE_NAME = 0x03 ,
  META_INSTRUMENT_NAME = 0x04 , META_LYRIC = 0x05 , META_MARKER = 0x06 , META_CUE_POINT = 0x07 ,
  META_CHANNEL_PREFIX = 0x20 , META_END_OF_TRACK = 0x2f , META_SET_TEMPO = 0x51 ,
  META_SMPTE_OFFSET = 0x54 ,
  META_TIME_SIGNATURE = 0x58 , META_KEY_SIGNATURE = 0x59 , META_SEQUENCER_SPECIFIC = 0x74
  }
- enum MidiControllers {
  CONTROLLER_MOD_WHEEL = 1 , CONTROLLER_BREATH = 2 , CONTROLLER_FOOT = 4 , CONTROLLER_BALANCE = 8 ,
  CONTROLLER_PAN = 10 , CONTROLLER_EXPRESSION = 11 , CONTROLLER_DAMPER_PEDAL = 0x40 ,
  CONTROLLER_PORTAMENTO = 0x41 ,
  CONTROLLER_SOSTENUTO = 0x42 , CONTROLLER_SOFT_PEDAL = 0x43 , CONTROLLER_GENERAL_4 = 0x44 , CONTROLLER_HOLD_2 = 0x45 ,
  CONTROLLER_7GENERAL_5 = 0x50 , CONTROLLER_GENERAL_6 = 0x51 , CONTROLLER_GENERAL_7 = 0x52 , CONTROLLER_GENERAL_8 = 0x53 ,
  CONTROLLER_TREMOLO_DEPTH = 0x5c , CONTROLLER_CHORUS_DEPTH = 0x5d , CONTROLLER_DETUNE = 0x5e , CONTROLLER_PHASER_DEPTH = 0x5f ,
  CONTROLLER_DATA_INC = 0x60 , CONTROLLER_DATA_DEC = 0x61 , CONTROLLER_NON_REG_LSB = 0x62 , CONTROLLER_NON_REG_MSB = 0x63 ,
  CONTROLLER_REG_LSB = 0x64 , CONTROLLER_REG_MSG = 0x65 , CONTROLLER_CONTINUOUS_AFTERTOUCH = 128 }
- enum MidiEventTypes {
  CHANNEL_NOTE_OFF = 0x80 , CHANNEL_NOTE_ON = 0x90 , CHANNEL_KEY_PRESSURE = 0xa0 ,
  CHANNEL_CONTROL_CHANGE = 0xb0 ,
  CHANNEL_PROGRAM_CHANGE = 0xc0 , CHANNEL_AFTER_TOUCH = 0xd0 , CHANNEL_PITCH_BEND = 0xe0 , SYSTEM_EXCLUSIVE = 0xf0 ,
  SYSTEM_MIDI_TIME_CODE = 0xf1 , SYSTEM_SONG_POSITION_POINTER = 0xf2 , SYSTEM_SONG_SELECT = 0xf3 , SYSTEM_TUNE_REQUEST = 0xf6 ,
  SYSTEM_END_OF_EXCLUSIVE = 0xf7 , SYSTEM_TIMING_CLOCK = 0xf8 , SYSTEM_START = 0xfa ,
  SYSTEM_CONTINUE = 0xfb ,
  SYSTEM_STOP = 0xfc , SYSTEM_ACTIVE_SENSING = 0xfe , META_EVENT = 0xff }

Public Member Functions

- virtual void clear ()
- void computeTimes ()
- virtual void dump (std::ostream &stream)
- virtual void load (std::string filename)
- MidiFile ()
- virtual void read (std::istream &stream)
- virtual void save (std::string filename)
- virtual void write (std::ostream &stream)
- virtual ~MidiFile ()

Static Public Member Functions

- static int chunkName (int a, int b, int c, int d)
- static int readInt (std::istream &stream)
- static short readShort (std::istream &stream)
- static int readVariableLength (std::istream &stream)
- static int toInt (int c1, int c2, int c3, int c4)
- static short toShort (int c1, int c2)
- static void writeInt (std::ostream &stream, int value)
- static void writeShort (std::ostream &stream, short value)
- static void writeVariableLength (std::ostream &stream, int value)

Data Fields

- double currentSecondsPerTick
- int currentTick
- double currentTime
- unsigned char lastStatus
- double microsecondsPerQuarterNote
- MidiHeader midiHeader
- std::vector< MidiTrack > midiTracks
- TempoMap tempoMap

6.55.1 Detailed Description

Reads and writes format 0 and format 1 standard MIDI files.

6.55.2 Member Enumeration Documentation

6.55.2.1 MetaEventTypes

`enum csound::MidiFile::MetaEventTypes`

Enumerator

META_SEQUENCE_NUMBER	
META_TEXT_EVENT	
META_COPYRIGHT_NOTICE	
META_SEQUENCE_NAME	
META_INSTRUMENT_NAME	
META_LYRIC	
META_MARKER	
META_CUE_POINT	
META_CHANNEL_PREFIX	
META_END_OF_TRACK	
META_SET_TEMPO	
META_SMPTE_OFFSET	
META_TIME_SIGNATURE	
META_KEY_SIGNATURE	
META_SEQUENCER_SPECIFIC	

6.55.2.2 MidiControllers

enum csound::MidiFile::MidiControllers

Enumerator

CONTROLLER_MOD_WHEEL	
CONTROLLER_BREATH	
CONTROLLER_FOOT	
CONTROLLER_BALANCE	
CONTROLLER_PAN	
CONTROLLER_EXPRESSION	
CONTROLLER_DAMPER_PEDAL	
CONTROLLER_PORTAMENTO	
CONTROLLER_SOSTENUTO	
CONTROLLER_SOFT_PEDAL	
CONTROLLER_GENERAL_4	
CONTROLLER_HOLD_2	
CONTROLLER_7GENERAL_5	
CONTROLLER_GENERAL_6	
CONTROLLER_GENERAL_7	
CONTROLLER_GENERAL_8	
CONTROLLER_TREMOLO_DEPTH	
CONTROLLER_CHORUS_DEPTH	
CONTROLLER_DETUNE	
CONTROLLER_PHASER_DEPTH	
CONTROLLER_DATA_INC	
CONTROLLER_DATA_DEC	
CONTROLLER_NON_REG_LSB	
CONTROLLER_NON_REG_MSB	
CONTROLLER_REG_LSB	
CONTROLLER_REG_MSG	
CONTROLLER_CONTINUOUS_AFTERTOUCH	

6.55.2.3 MidiEventTypes

enum csound::MidiFile::MidiEventTypes

Enumerator

CHANNEL_NOTE_OFF	
CHANNEL_NOTE_ON	
CHANNEL_KEY_PRESSURE	
CHANNEL_CONTROL_CHANGE	
CHANNEL_PROGRAM_CHANGE	
CHANNEL_AFTER_TOUCH	

Enumerator

CHANNEL_PITCH_BEND	
SYSTEM_EXCLUSIVE	
SYSTEM_MIDI_TIME_CODE	
SYSTEM_SONG_POSITION_POINTER	
SYSTEM_SONG_SELECT	
SYSTEM_TUNE_REQUEST	
SYSTEM_END_OF_EXCLUSIVE	
SYSTEM_TIMING_CLOCK	
SYSTEM_START	
SYSTEM_CONTINUE	
SYSTEM_STOP	
SYSTEM_ACTIVE_SENSING	
META_EVENT	

6.55.3 Constructor & Destructor Documentation

6.55.3.1 MidiFile()

```
csound::MidiFile::MidiFile ( )
```

References clear().

6.55.3.2 ~MidiFile()

```
csound::MidiFile::~MidiFile ( )  [virtual]
```

6.55.4 Member Function Documentation

6.55.4.1 chunkName()

```
int csound::MidiFile::chunkName (
            int a,
            int b,
            int c,
            int d )  [static]
```

Referenced by csound::Chunk::Chunk().

6.55.4.2 clear()

```
void csound::MidiFile::clear ( )  [virtual]
```

References csound::MidiHeader::clear(), computeTimes(), currentSecondsPerTick, currentTick, currentTime, lastStatus, microsecondsPerQuarterNote, midiHeader, midiTracks, and tempoMap.

Referenced by MidiFile(), and read().

6.55.4.3 computeTimes()

```
void csound::MidiFile::computeTimes ( )
```

References currentSecondsPerTick, currentTick, csound::fundamentalDomainByPredicate(), microsecondsPerQuarterNote, midiHeader, tempoMap, and csound::MidiHeader::timeFormat.

Referenced by clear(), and read().

6.55.4.4 dump()

```
void csound::MidiFile::dump (
            std::ostream & stream )  [virtual]
```

References csound::fundamentalDomainByPredicate(), csound::Chunk::id, midiHeader, midiTracks, csound::MidiEvent::ticks, csound::MidiHeader::timeFormat, csound::MidiHeader::trackCount, and csound::MidiHeader::type.

6.55.4.5 load()

```
void csound::MidiFile::load (
            std::string filename )  [virtual]
```

References csound::fundamentalDomainByPredicate(), and read().

6.55.4.6 read()

```
void csound::MidiFile::read (
            std::istream & stream )  [virtual]
```

References clear(), computeTimes(), currentTick, currentTime, csound::fundamentalDomainByPredicate(), midiHeader, midiTracks, csound::MidiHeader::read(), csound::MidiTrack::readIn(), and csound::MidiHeader::trackCount.

Referenced by load().

6.55.4.7 readInt()

```
int csound::MidiFile::readInt (
            std::istream & stream )  [static]
```

References csound::fundamentalDomainByPredicate(), and toInt().

Referenced by csound::Chunk::read().

6.55.4.8 readShort()

```
short csound::MidiFile::readShort (
            std::istream & stream )  [static]
```

References csound::fundamentalDomainByPredicate(), and toShort().

Referenced by csound::MidiHeader::read().

6.55.4.9 readVariableLength()

```
int csound::MidiFile::readVariableLength (
            std::istream & stream )  [static]
```

References csound::fundamentalDomainByPredicate().

Referenced by csound::MidiEvent::readIn().

6.55.4.10 save()

```
void csound::MidiFile::save (
            std::string filename )  [virtual]
```

References csound::fundamentalDomainByPredicate(), and write().

6.55.4.11 toInt()

```
int csound::MidiFile::toInt (
            int c1,
            int c2,
            int c3,
            int c4 )  [static]
```

References csound::fundamentalDomainByPredicate().

Referenced by readInt().

6.55.4.12 toShort()

```
short csound::MidiFile::toShort (
            int c1,
            int c2 )  [static]
```

References csound::fundamentalDomainByPredicate().

Referenced by readShort().

6.55.4.13 write()

```
void csound::MidiFile::write (
            std::ostream & stream )  [virtual]
```

References csound::fundamentalDomainByPredicate(), midiHeader, midiTracks, csound::MidiHeader::trackCount, and csound::MidiHeader::write().

Referenced by save().

6.55.4.14 writeInt()

```
void csound::MidiFile::writeInt (
            std::ostream & stream,
            int value )  [static]
```

References csound::fundamentalDomainByPredicate().

Referenced by csound::Chunk::markChunkEnd(), and csound::Chunk::write().

6.55.4.15 writeShort()

```
void csound::MidiFile::writeShort (
            std::ostream & stream,
            short value )  [static]
```

References csound::fundamentalDomainByPredicate().

Referenced by csound::MidiHeader::write().

6.55.4.16 writeVariableLength()

```
void csound::MidiFile::writeVariableLength (
            std::ostream & stream,
            int value )  [static]
```

References csound::fundamentalDomainByPredicate().

Referenced by csound::MidiEvent::writeOut().

6.55.5 Field Documentation

6.55.5.1 currentSecondsPerTick

```
double csound::MidiFile::currentSecondsPerTick
```

Referenced by clear(), and computeTimes().

6.55.5.2 currentTick

```
int csound::MidiFile::currentTick
```

Referenced by clear(), computeTimes(), and read().

6.55.5.3 currentTime

```
double csound::MidiFile::currentTime
```

Referenced by clear(), read(), and csound::MidiEvent::readIn().

6.55.5.4 lastStatus

```
unsigned char csound::MidiFile::lastStatus
```

Referenced by clear().

6.55.5.5 microsecondsPerQuarterNote

```
double csound::MidiFile::microsecondsPerQuarterNote
```

Referenced by clear(), and computeTimes().

6.55.5.6 midiHeader

```
MidiHeader csound::MidiFile::midiHeader
```

Referenced by clear(), computeTimes(), dump(), read(), and write().

6.55.5.7 midiTracks

```
std::vector<MidiTrack> csound::MidiFile::midiTracks
```

Referenced by clear(), dump(), read(), and write().

6.55.5.8 tempoMap

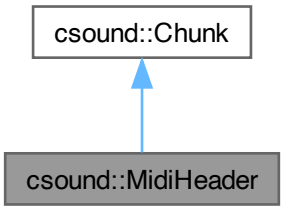
```
TempoMap csound::MidiFile::tempoMap
```

Referenced by clear(), and computeTimes().

6.56 csound::MidiHeader Class Reference

```
#include <Midifile.hpp>
```

Inheritance diagram for csound::MidiHeader:



Public Member Functions

- virtual void clear ()
- virtual void markChunkEnd (std::ostream &stream)
- virtual void markChunkSize (std::ostream &stream)
- virtual void markChunkStart (std::ostream &stream)
- MidiHeader ()
- MidiHeader (const MidiHeader &a)
- MidiHeader & operator= (const MidiHeader &a)
- virtual void read (std::istream &stream)
- virtual void write (std::ostream &stream)
- virtual ~MidiHeader ()

Data Fields

- int chunkEnd
- int chunkSize
- int chunkSizePosition
- int chunkStart
- int id
- short timeFormat
- short trackCount
- short type

6.56.1 Constructor & Destructor Documentation

6.56.1.1 MidiHeader() [1/2]

```
csound::MidiHeader::MidiHeader ( )
```

References clear().

6.56.1.2 MidiHeader() [2/2]

```
csound::MidiHeader::MidiHeader (
            const MidiHeader & a )
```

6.56.1.3 ~MidiHeader()

```
csound::MidiHeader::~MidiHeader ( )  [virtual]
```

References clear().

6.56.2 Member Function Documentation

6.56.2.1 clear()

```
void csound::MidiHeader::clear ( )  [virtual]
```

References timeFormat, trackCount, and type.

Referenced by csound::MidiFile::clear(), MidiHeader(), and ~MidiHeader().

6.56.2.2 markChunkEnd()

```
void csound::Chunk::markChunkEnd (
            std::ostream & stream )  [virtual], [inherited]
```

References csound::Chunk::chunkEnd, csound::Chunk::chunkSize, csound::Chunk::chunkSizePosition, csound::Chunk::chunkStart, csound::fundamentalDomainByPredicate(), and csound::MidiFile::writeInt().

Referenced by write(), and csound::MidiTrack::writeOut().

6.56.2.3 markChunkSize()

```
void csound::Chunk::markChunkSize (
            std::ostream & stream )  [virtual], [inherited]
```

References csound::Chunk::chunkSizePosition, and csound::fundamentalDomainByPredicate().

Referenced by csound::Chunk::write().

6.56.2.4 markChunkStart()

```
void csound::Chunk::markChunkStart (
            std::ostream & stream )  [virtual], [inherited]
```

References csound::Chunk::chunkStart, and csound::fundamentalDomainByPredicate().

Referenced by csound::Chunk::write().

6.56.2.5 operator=()

```
MidiHeader & csound::MidiHeader::operator= (
            const MidiHeader & a )
```

References csound::Chunk::chunkEnd, csound::Chunk::chunkSize, csound::Chunk::chunkSizePosition, csound::Chunk::chunkStart, csound::Chunk::id, timeFormat, trackCount, and type.

6.56.2.6 read()

```
void csound::MidiHeader::read (
            std::istream & stream )  [virtual]
```

Reimplemented from csound::Chunk.

References csound::fundamentalDomainByPredicate(), csound::Chunk::read(), csound::MidiFile::readShort(), timeFormat, trackCount, and type.

Referenced by csound::MidiFile::read().

6.56.2.7 write()

```
void csound::MidiHeader::write (
            std::ostream & stream )  [virtual]
```

Reimplemented from csound::Chunk.

References csound::fundamentalDomainByPredicate(), csound::Chunk::markChunkEnd(), timeFormat, trackCount, type, csound::Chunk::write(), and csound::MidiFile::writeShort().

Referenced by csound::MidiFile::write().

6.56.3 Field Documentation

6.56.3.1 chunkEnd

```
int csound::Chunk::chunkEnd  [inherited]
```

Referenced by csound::Chunk::markChunkEnd(), csound::Chunk::operator=(), operator=(), and csound::MidiTrack::operator=().

6.56.3.2 chunkSize

```
int csound::Chunk::chunkSize  [inherited]
```

Referenced by csound::Chunk::markChunkEnd(), csound::Chunk::operator=(), operator=(), csound::MidiTrack::operator=(), csound::Chunk::read(), and csound::Chunk::write().

6.56.3.3 chunkSizePosition

```
int csound::Chunk::chunkSizePosition  [inherited]
```

Referenced by csound::Chunk::markChunkEnd(), csound::Chunk::markChunkSize(), csound::Chunk::operator=(), operator=(), and csound::MidiTrack::operator=().

6.56.3.4 chunkStart

```
int csound::Chunk::chunkStart  [inherited]
```

Referenced by csound::Chunk::markChunkEnd(), csound::Chunk::markChunkStart(), csound::Chunk::operator=(), operator=(), and csound::MidiTrack::operator=().

6.56.3.5 id

```
int csound::Chunk::id  [inherited]
```

Referenced by csound::MidiFile::dump(), csound::Chunk::operator=(), operator=(), and csound::MidiTrack::operator=().

6.56.3.6 timeFormat

```
short csound::MidiHeader::timeFormat
```

Referenced by clear(), csound::MidiFile::computeTimes(), csound::MidiFile::dump(), operator=(), read(), and write().

6.56.3.7 trackCount

```
short csound::MidiHeader::trackCount
```

Referenced by clear(), csound::MidiFile::dump(), operator=(), read(), csound::MidiFile::read(), write(), and csound::MidiFile::write().

6.56.3.8 type

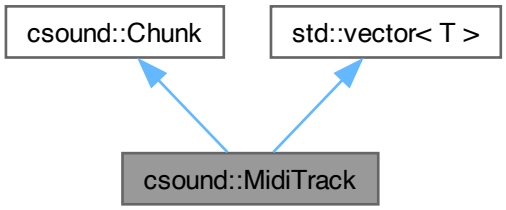
```
short csound::MidiHeader::type
```

Referenced by clear(), csound::MidiFile::dump(), operator=(), read(), and write().

6.57 csound::MidiTrack Class Reference

`#include <Midifile.hpp>`

Inheritance diagram for csound::MidiTrack:



Public Member Functions

- virtual void markChunkEnd (std::ostream &stream)
- virtual void markChunkSize (std::ostream &stream)
- virtual void markChunkStart (std::ostream &stream)
- MidiTrack ()
- MidiTrack & operator= (const MidiTrack &a)
- virtual void read (std::istream &stream)
- virtual void readIn (std::istream &stream, MidiFile &midiFile)
- virtual void write (std::ostream &stream)
- virtual void writeOut (std::ostream &stream, MidiFile &midiFile)
- virtual ~MidiTrack ()

Data Fields

- int chunkEnd
- int chunkSize
- int chunkSizePosition
- int chunkStart
- T elements

  STL member.
- int id

6.57.1 Constructor & Destructor Documentation

6.57.1.1 MidiTrack()

`csound::MidiTrack::MidiTrack ( )`

6.57.1.2 ~MidiTrack()

```
csound::MidiTrack::~MidiTrack ( )  [virtual]
```

6.57.2 Member Function Documentation

6.57.2.1 markChunkEnd()

```
void csound::Chunk::markChunkEnd (
            std::ostream & stream )  [virtual], [inherited]
```

References csound::Chunk::chunkEnd, csound::Chunk::chunkSize, csound::Chunk::chunkSizePosition, csound::Chunk::chunkStart, csound::fundamentalDomainByPredicate(), and csound::MidiFile::writeInt().

Referenced by csound::MidiHeader::write(), and writeOut().

6.57.2.2 markChunkSize()

```
void csound::Chunk::markChunkSize (
            std::ostream & stream )  [virtual], [inherited]
```

References csound::Chunk::chunkSizePosition, and csound::fundamentalDomainByPredicate().

Referenced by csound::Chunk::write().

6.57.2.3 markChunkStart()

```
void csound::Chunk::markChunkStart (
            std::ostream & stream )  [virtual], [inherited]
```

References csound::Chunk::chunkStart, and csound::fundamentalDomainByPredicate().

Referenced by csound::Chunk::write().

6.57.2.4 operator=()

```
MidiTrack & csound::MidiTrack::operator= (
            const MidiTrack & a )
```

References csound::Chunk::chunkEnd, csound::Chunk::chunkSize, csound::Chunk::chunkSizePosition, csound::Chunk::chunkStart, csound::fundamentalDomainByPredicate(), and csound::Chunk::id.

6.57.2.5 read()

```
void csound::Chunk::read (
            std::istream & stream )  [virtual], [inherited]
```

Reimplemented in csound::MidiHeader.

References csound::Chunk::chunkSize, csound::fundamentalDomainByPredicate(), csound::System::inform(), csound::MidiFile::readInt(), and csound::System::warn().

Referenced by csound::MidiHeader::read(), and readIn().

6.57.2.6 readIn()

```
void csound::MidiTrack::readIn (
            std::istream & stream,
            MidiFile & midiFile )  [virtual]
```

References csound::fundamentalDomainByPredicate(), csound::MidiFile::META_END_OF_TRACK, csound::Chunk::read(), and csound::MidiEvent::readIn().

Referenced by csound::MidiFile::read().

6.57.2.7 write()

```
void csound::Chunk::write (
            std::ostream & stream )  [virtual], [inherited]
```

Reimplemented in csound::MidiHeader.

References csound::Chunk::chunkSize, csound::fundamentalDomainByPredicate(), csound::Chunk::markChunkSize(), csound::Chunk::markChunkStart(), and csound::MidiFile::writeInt().

Referenced by csound::MidiHeader::write(), and writeOut().

6.57.2.8 writeOut()

```
void csound::MidiTrack::writeOut (
            std::ostream & stream,
            MidiFile & midiFile )  [virtual]
```

References csound::fundamentalDomainByPredicate(), csound::Chunk::markChunkEnd(), csound::Chunk::write(), and csound::MidiEvent::writeOut().

6.57.3 Field Documentation

6.57.3.1 chunkEnd

```
int csound::Chunk::chunkEnd  [inherited]
```

Referenced by csound::Chunk::markChunkEnd(), csound::Chunk::operator=(), csound::MidiHeader::operator=(), and operator=().

6.57.3.2 chunkSize

```
int csound::Chunk::chunkSize  [inherited]
```

Referenced by csound::Chunk::markChunkEnd(), csound::Chunk::operator=(), csound::MidiHeader::operator=(), operator=(), csound::Chunk::read(), and csound::Chunk::write().

6.57.3.3 chunkSizePosition

```
int csound::Chunk::chunkSizePosition  [inherited]
```

Referenced by csound::Chunk::markChunkEnd(), csound::Chunk::markChunkSize(), csound::Chunk::operator=(), csound::MidiHeader::operator=(), and operator=().

6.57.3.4 chunkStart

```
int csound::Chunk::chunkStart  [inherited]
```

Referenced by csound::Chunk::markChunkEnd(), csound::Chunk::markChunkStart(), csound::Chunk::operator=(), csound::MidiHeader::operator=(), and operator=().

6.57.3.5 elements

```
T std::vector< T >::elements  [inherited]
```

STL member.

6.57.3.6 id

```
int csound::Chunk::id  [inherited]
```

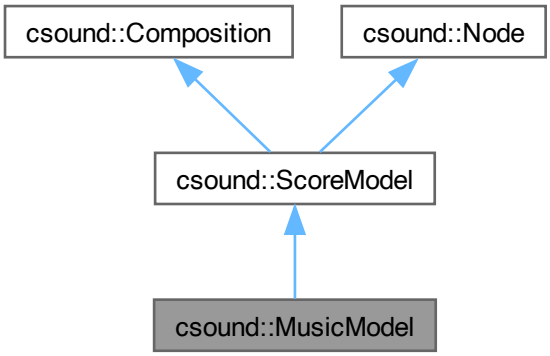
Referenced by csound::MidiFile::dump(), csound::Chunk::operator=(), csound::MidiHeader::operator=(), and operator=().

6.58 csound::MusicModel Class Reference

A ScoreModel that uses Csound to render generated scores, via the CppSound class.

`#include <MusicModel.hpp>`

Inheritance diagram for csound::MusicModel:



Public Member Functions

- virtual void addChild (Node *node)

  Adds an immediate child Node to this.
- virtual void arrange (int oldInstrumentNumber, int newInstrumentNumber)

  Re-assign instrument number for export to Csound score (convenience wrapper for Score::arrange()).
- virtual void arrange (int oldInstrumentNumber, int newInstrumentNumber, double gain)

  Re-assign instrument number and adjust gain for export to Csound score (convenience wrapper for Score::arrange()).
- virtual void arrange (int oldInstrumentNumber, int newInstrumentNumber, double gain, double pan)

  Re-assign instrument number, adjust gain, and change pan for export to Csound score (convenience wrapper for Score::arrange()).
- virtual void arrange (int silenceInstrumentNumber, std::string csoundInstrumentName)

  Re-assign instrument by name for export to Csound score.
- virtual void arrange (int silenceInstrumentNumber, std::string csoundInstrumentName, double gain)

  Re-assign instrument by name and adjust gains for export to Csound score.
- virtual void arrange (int silenceInstrumentNumber, std::string csoundInstrumentName, double gain, double pan)

  Re-assign instrument by name, adjust gain, and change pan for export to Csound score.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

  Clear all contents of this.
- virtual void clearOutputSoundfileName ()

- virtual int cppsoundCleanup ()
- virtual int cppsoundCompile (int argc, const char **argv)
- virtual int cppsoundCompileCsdText (const std::string &csd_text)
- virtual std::string cppsoundGetCommand () const
- virtual void cppsoundInputMessage (const std::string &message) const
- virtual int cppsoundLoad (std::string filename)
- virtual int cppsoundPerform ()
- virtual int cppsoundPerformKsmps ()
- virtual void cppsoundReset ()
- virtual void cppsoundSetCommand (const std::string &command)
- virtual void cppsoundSetFilename (const std::string &filename)
- virtual int cppsoundStart ()
- virtual void cppsoundStop ()
- virtual void createCsoundScore (std::string addToScore="", double extendSeconds=0.)

  Translate the generated score to a Csound score and export it for performance.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual void csoundArgv (int argc, const char **argv)

  Does not use the csound::Composition options; passes argc and argv directly to Csound.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual int generate ()

  Generates a score based on a music graph defined by the child nodes of this.
- virtual void generate (Score &score_from_this)

  Optionally generate notes into the score.
- virtual void generateAllNames ()

  Generates all filenames and other text based on required stem, output_directory, filename extension, and metadata.
- virtual std::string getAlbum () const
- virtual std::string getArtist () const
- virtual std::string getAuthor () const
- virtual std::string getBasename () const

  Returns the complete basename of the file, i.e., the output directory plus the stem.
- virtual std::string getCdSoundfileFilepath () const

  Returns a soundfile name for a CD audio track based on the filename of this, by appending ".cd.wav" to the filename.
- virtual Node * getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual bool getConformPitches () const

  Returns whether or not the pitches in generated scores will be conformed to the nearest equally tempered pitch.
- virtual std::string getCopyright () const
- virtual std::string getCsoundCommand () const

  Return Csound command line (convenience wrapper for CppSound::getCommand()).
- virtual std::string getCsoundOrchestra () const

  Return the Csound orchestra (convenience wrapper for CppSound::getOrchestra()).
- virtual std::string getCsoundScoreHeader () const

  Return the Csound score header that is prepended to generated scores.
- virtual double getDuration () const

  Returns the duration to which all times and durations of all events will be rescaled.
- virtual double getExtendSeconds () const

- virtual std::string getFileFilepath () const

  Returns the complete basename of the file, i.e., the output directory plus the filename.
- virtual std::string getFilename () const

  Returns the stem of this, which is used as a base for derived filenames (soundfile, MIDI file, etc.).
- virtual std::string getFomusfileFilepath () const

  Returns a MusicXML filename based on the filename of this, by appending ".fms" to the filename.
- virtual std::string getLicense () const
- virtual std::string getLilypondfileFilepath () const

  Returns a MusicXML filename based on the filename of this, by appending ".ly" to the filename.
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual std::string getMidifileFilepath () const

  Returns a MIDI filename based on the filename of this, by appending ".mid" to the filename.
- virtual std::string getMp3SoundfileFilepath () const

  Returns a soundfile name for an MP3 file based on the filename of this, by appending ".mp3" to the filename.
- virtual std::string getMusicXmlfileFilepath () const

  Returns a MusicXML filename based on the filename of this, by appending ".xml" to the filename.
- virtual std::string getNormalizedSoundfileFilepath () const

  Returns a soundfile name based on the filename of this, by appending ".norm.wav" to the filename.
- virtual std::string getOutputDirectory () const

  Returns the directory in which to place the output files of this.
- virtual std::string getOutputSoundfileFilepath () const

  Returns a soundfile name based on the filename of this, by appending ".wav" to the filename, which is the default, or a non-default ouput name which need not be a file but must be set using setOutputSoundfileName().
- virtual std::string getPerformanceRightsOrganization () const
- virtual Score & getScore ()

  Return the self-contained Score.
- virtual intptr_t getThis ()

  Returns the address of this as a long integer.
- virtual Node * getThisNode ()

  Returns the address of this as a Node pointer.
- virtual bool getTieOverlappingNotes () const

  Returns whether or not overlapping notes in generated scores are replaced by one note.
- virtual std::string getTimestamp () const

  Returns the time the score was generated.
- virtual std::string getTitle () const
- virtual double getTonesPerOctave () const

  Returns the number of equally tempered intervals per octave (the default is 12, 0 means non-equally tempered).
- virtual std::string getYear () const
- virtual void initialize ()
- MusicModel ()
- virtual int normalizeOutputSoundfile (double levelDb=-3.0)

  Assuming the score has been rendered, uses sox to translate the output soundfile to a normalized soundfile.
- virtual int perform ()

  Uses Csound to perform the current score.
- virtual int performAll ()

  Convenience function that calls performMaster(), and translateMaster().
- virtual int performMaster ()

Convenience function that calls saveMidi(), saveMusicXML(), and perform().

- virtual int processArgs (const std::vector< std::string > &args)

  Pass the invoking program's command-line arguments to processArgs() and it will perform the following commands:
- virtual int processArgv (int argc, const char **argv)

  Pass the invoking program's command-line arguments to processArgs() and it will perform with possibly back-end-dependent options.
- virtual void removeArrangement ()

  Remove instrument number, gain, and pan assignments (convenience wrapper for Score::removeArrangement()).
- virtual int render ()

  Convenience function that erases the existing score, appends optional text to it, invokes generate(), invokes createCsoundScore(), and invokes perform().
- virtual int renderAll ()

  Convenience function that calls clear(), generate(), performAll().
- virtual void setAlbum (std::string value)
- virtual void setArtist (std::string value)
- virtual void setAuthor (std::string value)
- virtual void setConformPitches (bool conformPitches)

  Sets whether or not the pitches in generated scores will be conformed to the nearest equally tempered pitch.
- virtual void setCopyright (std::string value)
- virtual void setCsoundCommand (std::string command)

  Set Csound command line (convenience wrapper for CppSound::setCommand()).
- virtual void setCsoundOrchestra (std::string orchestra)

  Set the Csound orchestra (convenience wrapper for CppSound::setOrchestra()).
- virtual void setCsoundScoreHeader (std::string header)

  Set a Csound score fragment to be prepended to the generated score (createCsoundScore is called with it).
- virtual void setDuration (double seconds)

  At the end of processing, if the defined duration is not zero, the times and durations of all events are rescaled to the defined duration.
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void setExtendSeconds (double extendSeconds_)
- virtual void setFilename (std::string filename)

  Sets the filename of this – basically, the title of the composition.
- virtual void setLicense (std::string value)
- virtual void setOutputDirectory (std::string directory)

  Sets the directory in which to place the output files of this.
- virtual void setOutputSoundfileName (std::string name)

  Sets a non-default output name (could be an audio device not a file).
- virtual void setPerformanceRightsOrganization (std::string value)
- virtual void setScore (Score &score)

  Sets the score in this to the indicated score.
- virtual void setTieOverlappingNotes (bool tieOverlappingNotes)

  Sets whether or not overlapping notes in generated scores are replaced by one note.
- virtual void setTitle (std::string value)
- virtual void setTonesPerOctave (double tonesPerOctave)

  Sets the number of equally tempered intervals per octave (the default is 12, 0 means non-equally tempered).
- virtual void setYear (std::string value)
- virtual void stop ()

- virtual int tagFile (std::string filename) const
- virtual void transform (Score &score_from_children)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual int translateMaster ()

  Convenience function that calls rescaleOutputSoundfile(), translateToCdAudio(), and translateToMp3().
- virtual int translateToCdAudio (double levelDb=-3.0)

  Assuming the score has been rendered, uses sox to translate the output soundfile to normalized CD-audio format.
- virtual int translateToMp3 (double bitrate=256.01, double levelDb=-3.0)

  Assuming the score has been rendered, uses sox and LAME to translate the output soundfile to normalized MP3 format.
- virtual int translateToMp4 ()

  Assuming the score has been rendered, uses sox and ffmpeg to translate the output soundfile to a normalized mp4 video suitable for uploading to YouTube.
- virtual int translateToNotation (const std::vector< std::string > partNames=std::vector< std::string >(), std::string header="")

  Saves the generated score in Fomus format and uses Fomus and Lilypond to translate that to a PDF of music notation.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.
- virtual void write (const char *text)

  Write as if to stderr.
- virtual ~MusicModel ()

Static Public Member Functions

- static std::string generateFilename ()

  Generates a versioned filename.
- static std::string makeTimestamp ()

  Returns the current locale time as a string.

Data Fields

- std::vector< Node * > children

  Child Nodes, if any.
- int threadCount

Protected Attributes

- std::string album

  Optional metadata.
- std::string artist

  Required metadata.
- std::string author

  Required metadata.
- std::string base_filepath

  Generated.

- Score baseScore
- std::string bext_description

  Generated.
- std::string bext_orig_ref

  Generated.
- std::string bext_originator

  Generated.
- std::string cd_quality_filepath

  Generated.
- bool conformPitches
- std::string copyright

  Required metadata.
- CppSound * cppSound

  Pointer to a Csound object that is used to render scores.
- CppSound cppSound_

  Self-contained Csound object.
- std::string csoundScoreHeader

  Prepended to generated score.
- double duration
- double extendSeconds = -1
- std::string flac_filepath

  Generated.
- std::string label

  Generated.
- std::string license

  Required metadata.
- Eigen::MatrixXd localCoordinates
- std::string master_filepath

  Generated.
- std::string midi_filepath

  Generated.
- std::string mp3_filepath

  Generated.
- std::string mp4_filepath

  Generated.
- std::string normalized_master_filepath

  Generated.
- std::string notes

  Optional metadata, defaults to "Electroacoustic Music.".
- std::string output_directory

  Required.
- std::string output_filename
- std::string performance_rights_organization

  Optional metadata.
- Score & score
- std::string spectrogram_filepath

  Generated.

- std::string stem

  Required.
- bool tieOverlappingNotes
- std::string timestamp

  Generated.
- double tonesPerOctave
- std::string track

  Optional metadata.
- std::string year

  Required metadata.

6.58.1 Detailed Description

A ScoreModel that uses Csound to render generated scores, via the CppSound class.

6.58.2 Constructor & Destructor Documentation

6.58.2.1 MusicModel()

```
csound::MusicModel::MusicModel ( )
```

6.58.2.2 ~MusicModel()

```
csound::MusicModel::~MusicModel ( )  [virtual]
```

6.58.3 Member Function Documentation

6.58.3.1 addChild()

```
virtual void csound::ScoreModel::addChild (
            Node * node )  [inline], [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented from csound::Node.

Referenced by main().

6.58.3.2 arrange() [1/6]

```
void csound::MusicModel::arrange (
            int oldInstrumentNumber,
            int newInstrumentNumber )  [virtual]
```

Re-assign instrument number for export to Csound score (convenience wrapper for Score::arrange()).

References csound::Score::arrange(), csound::fundamentalDomainByPredicate(), and csound::Composition::score.

Referenced by arrange(), arrange(), and arrange().

6.58.3.3 arrange() [2/6]

```
void csound::MusicModel::arrange (
            int oldInstrumentNumber,
            int newInstrumentNumber,
            double gain )  [virtual]
```

Re-assign instrument number and adjust gain for export to Csound score (convenience wrapper for Score::arrange()).

References csound::Score::arrange(), csound::fundamentalDomainByPredicate(), and csound::Composition::score.

6.58.3.4 arrange() [3/6]

```
void csound::MusicModel::arrange (
            int oldInstrumentNumber,
            int newInstrumentNumber,
            double gain,
            double pan )  [virtual]
```

Re-assign instrument number, adjust gain, and change pan for export to Csound score (convenience wrapper for Score::arrange()).

References csound::Score::arrange(), csound::fundamentalDomainByPredicate(), and csound::Composition::score.

6.58.3.5 arrange() [4/6]

```
void csound::MusicModel::arrange (
            int silenceInstrumentNumber,
            std::string csoundInstrumentName )  [virtual]
```

Re-assign instrument by name for export to Csound score.

References arrange(), cppSound, csound::fundamentalDomainByPredicate(), and CsoundFile::getInstrumentNumber().

6.58.3.6 arrange() [5/6]

```
void csound::MusicModel::arrange (
            int silenceInstrumentNumber,
            std::string csoundInstrumentName,
            double gain )  [virtual]
```

Re-assign instrument by name and adjust gains for export to Csound score.

References arrange(), cppSound, csound::fundamentalDomainByPredicate(), and CsoundFile::getInstrumentNumber().

6.58.3.7 arrange() [6/6]

```
void csound::MusicModel::arrange (
            int silenceInstrumentNumber,
            std::string csoundInstrumentName,
            double gain,
            double pan )  [virtual]
```

Re-assign instrument by name, adjust gain, and change pan for export to Csound score.

References arrange(), cppSound, csound::fundamentalDomainByPredicate(), and CsoundFile::getInstrumentNumber().

6.58.3.8 childCount()

```
virtual size_t csound::ScoreModel::childCount ( ) const  [inline], [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented from csound::Node.

6.58.3.9 clear()

```
void csound::MusicModel::clear ( )  [virtual]
```

Clear all contents of this.

Reimplemented from csound::ScoreModel.

References csound::Composition::clear(), csound::Node::clear(), cppSound, and CsoundFile::removeScore().

6.58.3.10 clearOutputSoundfileName()

```
void csound::Composition::clearOutputSoundfileName ( )  [virtual], [inherited]
```

References csound::Composition::output_filename.

6.58.3.11 cppsoundCleanup()

```
virtual int csound::MusicModel::cppsoundCleanup ( )  [inline], [virtual]
```

6.58.3.12 cppsoundCompile()

```
virtual int csound::MusicModel::cppsoundCompile (
            int argc,
            const char ** argv )  [inline], [virtual]
```

6.58.3.13 cppsoundCompileCsdText()

```
virtual int csound::MusicModel::cppsoundCompileCsdText (
            const std::string & csd_text )  [inline], [virtual]
```

References csd_text.

6.58.3.14 cppsoundGetCommand()

```
virtual std::string csound::MusicModel::cppsoundGetCommand ( ) const  [inline], [virtual]
```

6.58.3.15 cppsoundInputMessage()

```
virtual void csound::MusicModel::cppsoundInputMessage (
            const std::string & message ) const  [inline], [virtual]
```

6.58.3.16 cppsoundLoad()

```
virtual int csound::MusicModel::cppsoundLoad (
            std::string filename )  [inline], [virtual]
```

6.58.3.17 cppsoundPerform()

```
virtual int csound::MusicModel::cppsoundPerform ( )  [inline], [virtual]
```

6.58.3.18 cppsoundPerformKsmps()

```
virtual int csound::MusicModel::cppsoundPerformKsmps ( )  [inline], [virtual]
```

6.58.3.19 cppsoundReset()

```
virtual void csound::MusicModel::cppsoundReset ( )  [inline], [virtual]
```

6.58.3.20 cppsoundSetCommand()

```
virtual void csound::MusicModel::cppsoundSetCommand (
            const std::string & command )  [inline], [virtual]
```

6.58.3.21 cppsoundSetFilename()

```
virtual void csound::MusicModel::cppsoundSetFilename (
            const std::string & filename )  [inline], [virtual]
```

6.58.3.22 cppsoundStart()

```
virtual int csound::MusicModel::cppsoundStart ( )  [inline], [virtual]
```

6.58.3.23 cppsoundStop()

```
virtual void csound::MusicModel::cppsoundStop ( )  [inline], [virtual]
```

6.58.3.24 createCsoundScore()

```
void csound::MusicModel::createCsoundScore (
            std::string addToScore = "",
            double extendSeconds = 0.  )  [virtual]
```

Translate the generated score to a Csound score and export it for performance.

The time given by extendSeconds is used for a concluding e statement.

References CsoundFile::addScoreLine(), csound::Composition::conformPitches, cppSound, extendSeconds, csound::fundamentalDomainByPredicate(), csound::Score::getCsoundScore(), csound::System::inform(), CsoundFile::removeScore(), csound::Composition::score, and csound::Composition::tonesPerOctave.

Referenced by perform().

6.58.3.25 createTransform()

```
virtual Eigen::MatrixXd csound::ScoreModel::createTransform ( )  [inline], [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented from csound::Node.

6.58.3.26 csoundArgv()

```
void csound::MusicModel::csoundArgv (
            int argc,
            const char ** argv )  [virtual]
```

Does not use the csound::Composition options; passes argc and argv directly to Csound.

References csound::fundamentalDomainByPredicate(), generate(), csound::Composition::generateAllNames(), csound::Composition::getMidifileFilepath(), csound::Composition::getScore(), csound::System::inform(), render(), csound::Score::save(), and setCsoundCommand().

6.58.3.27 element()

```
virtual double & csound::ScoreModel::element (
            size_t row,
            size_t column )  [inline], [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented from csound::Node.

6.58.3.28 generate() [1/2]

```
int csound::MusicModel::generate ( )  [virtual]
```

Generates a score based on a music graph defined by the child nodes of this.

Reimplemented from csound::ScoreModel.

References csound::Node::children, cppSound, csound::Composition::duration, csound::fundamentalDomainByPredicate(), csound::ScoreModel::getLocalCoordinates(), csound::Composition::getMidifileFilepath(), csound::System::message(), csound::Score::process(), CsoundFile::removeScore(), csound::Score::save(), csound::Composition::score, csound::Score::setDuration(), and csound::ScoreModel::traverse().

Referenced by csoundArgv(), processArgs(), and render().

6.58.3.29 generate() [2/2]

```
virtual void csound::ScoreModel::generate (
            Score & score_from_this )  [inline], [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented from csound::Node.

6.58.3.30 generateAllNames()

```
void csound::Composition::generateAllNames ( )  [virtual], [inherited]
```

Generates all filenames and other text based on required stem, output_directory, filename extension, and metadata.

References csound::Composition::album, csound::Composition::artist, csound::Composition::author, csound::Composition::base_filepath, csound::Composition::bext_description, csound::Composition::bext_orig_ref, csound::Composition::bext_originator, csound::Composition::cd_quality_filepath, csound::Composition::copyright, csound::Composition::flac_filepath, csound::fundamentalDomainByPredicate(), csound::Composition::getLicense(), csound::Composition::getOutputDirectory(), csound::Composition::getTitle(), csound::System::inform(), csound::Composition::label, csound::Composition::makeTimestamp(), csound::Composition::master_filepath, csound::Composition::midi_filepath, csound::Composition::mp3_filepath, csound::Composition::mp4_filepath, csound::Composition::normalized_master_filepath, csound::Composition::notes, csound::Composition::output_directory, csound::Composition::performance_rights_organization, csound::Composition::spectrogram_filep csound::Composition::stem, csound::Composition::timestamp, csound::Composition::track, and csound::Composition::year.

Referenced by csoundArgv(), csound::Composition::processArgs(), and processArgs().

6.58.3.31 generateFilename()

```
std::string csound::Composition::generateFilename ( )  [static], [inherited]
```

Generates a versioned filename.

References csound::fundamentalDomainByPredicate(), and csound::Composition::makeTimestamp().

6.58.3.32 getAlbum()

```
std::string csound::Composition::getAlbum ( ) const  [virtual], [inherited]
```

References csound::Composition::album.

Referenced by csound::Composition::tagFile(), and csound::Composition::translateToMp3().

6.58.3.33 getArtist()

```
std::string csound::Composition::getArtist ( ) const  [virtual], [inherited]
```

References csound::Composition::artist.

Referenced by csound::Composition::tagFile(), and csound::Composition::translateToNotation().

6.58.3.34 getAuthor()

```
std::string csound::Composition::getAuthor ( ) const  [virtual], [inherited]
```

References csound::Composition::author.

Referenced by csound::Composition::tagFile(), and csound::Composition::translateToMp3().

6.58.3.35 getBasename()

```
std::string csound::Composition::getBasename ( ) const  [virtual], [inherited]
```

Returns the complete basename of the file, i.e., the output directory plus the stem.

References csound::Composition::base_filepath.

Referenced by csound::Composition::getFomusfileFilepath(), and csound::Composition::getLilypondfileFilepath().

6.58.3.36 getCdSoundfileFilepath()

```
std::string csound::Composition::getCdSoundfileFilepath ( ) const  [virtual], [inherited]
```

Returns a soundfile name for a CD audio track based on the filename of this, by appending ".cd.wav" to the filename.

References csound::Composition::cd_quality_filepath.

Referenced by csound::Composition::translateToCdAudio(), and csound::Composition::translateToMp3().

6.58.3.37 getChild()

```
virtual Node * csound::ScoreModel::getChild (
            size_t index )  [inline], [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented from csound::Node.

6.58.3.38 getConformPitches()

```
bool csound::Composition::getConformPitches ( ) const  [virtual], [inherited]
```

Returns whether or not the pitches in generated scores will be conformed to the nearest equally tempered pitch.

References csound::Composition::conformPitches.

Referenced by csound::ScoreModel::generate().

6.58.3.39 getCopyright()

```
std::string csound::Composition::getCopyright ( ) const  [virtual], [inherited]
```

References csound::Composition::copyright.

Referenced by csound::Composition::tagFile(), csound::Composition::translateToMp3(), and csound::Composition::translateToMp4().

6.58.3.40 getCsoundCommand()

```
std::string csound::MusicModel::getCsoundCommand ( ) const  [virtual]
```

Return Csound command line (convenience wrapper for CppSound::getCommand()).

References cppSound, csound::fundamentalDomainByPredicate(), CsoundFile::getCommand(), csound::Composition::getOutputSoundfil
and threadCount.

Referenced by perform().

6.58.3.41 getCsoundOrchestra()

```
std::string csound::MusicModel::getCsoundOrchestra ( ) const  [virtual]
```

Return the Csound orchestra (convenience wrapper for CppSound::getOrchestra()).

References cppSound, and CsoundFile::getOrchestra().

6.58.3.42 getCsoundScoreHeader()

```
std::string csound::MusicModel::getCsoundScoreHeader ( ) const  [virtual]
```

Return the Csound score header that is prepended to generated scores.

References csoundScoreHeader.

6.58.3.43 getDuration()

```
double csound::Composition::getDuration ( ) const  [virtual], [inherited]
```

Returns the duration to which all times and durations of all events will be rescaled.

If the duration is 0, no rescaling is performed.

References csound::Composition::duration.

6.58.3.44 getExtendSeconds()

```
virtual double csound::MusicModel::getExtendSeconds ( ) const  [inline], [virtual]
```

Referenced by perform().

6.58.3.45 getFileFilepath()

```
std::string csound::Composition::getFileFilepath ( ) const  [virtual], [inherited]
```

Returns the complete basename of the file, i.e., the output directory plus the filename.

References csound::Composition::base_filepath.

6.58.3.46 getFilename()

```
std::string csound::Composition::getFilename ( ) const  [virtual], [inherited]
```

Returns the stem of this, which is used as a base for derived filenames (soundfile, MIDI file, etc.).

References csound::Composition::stem.

6.58.3.47 getFomusfileFilepath()

```
std::string csound::Composition::getFomusfileFilepath ( ) const  [virtual], [inherited]
```

Returns a MusicXML filename based on the filename of this, by appending ".fms" to the filename.

References csound::Composition::getBasename().

Referenced by csound::Composition::translateToNotation().

6.58.3.48 getLicense()

```
std::string csound::Composition::getLicense ( ) const  [virtual], [inherited]
```

References csound::Composition::license.

Referenced by csound::Composition::generateAllNames(), and csound::Composition::tagFile().

6.58.3.49 getLilypondfileFilepath()

```
std::string csound::Composition::getLilypondfileFilepath ( ) const  [virtual], [inherited]
```

Returns a MusicXML filename based on the filename of this, by appending ".ly" to the filename.

References csound::Composition::getBasename().

6.58.3.50 getLocalCoordinates()

```
virtual Eigen::MatrixXd csound::ScoreModel::getLocalCoordinates ( ) const  [inline], [virtual],
[inherited]
```

Returns the local transformation of coordinate system.

Reimplemented from csound::Node.

Referenced by generate(), and csound::ScoreModel::generate().

6.58.3.51 getMidifileFilepath()

```
std::string csound::Composition::getMidifileFilepath ( ) const  [virtual], [inherited]
```

Returns a MIDI filename based on the filename of this, by appending ".mid" to the filename.

References csound::Composition::midi_filepath.

Referenced by csoundArgv(), generate(), and processArgs().

6.58.3.52 getMp3SoundfileFilepath()

```
std::string csound::Composition::getMp3SoundfileFilepath ( ) const  [virtual], [inherited]
```

Returns a soundfile name for an MP3 file based on the filename of this, by appending ".mp3" to the filename.

References csound::Composition::mp3_filepath.

Referenced by csound::Composition::translateToMp3().

6.58.3.53 getMusicXmlfileFilepath()

```
std::string csound::Composition::getMusicXmlfileFilepath ( ) const  [virtual], [inherited]
```

Returns a MusicXML filename based on the filename of this, by appending ".xml" to the filename.

References csound::Composition::base_filepath.

6.58.3.54 getNormalizedSoundfileFilepath()

```
std::string csound::Composition::getNormalizedSoundfileFilepath ( ) const  [virtual], [inherited]
```

Returns a soundfile name based on the filename of this, by appending ".norm.wav" to the filename.

References csound::Composition::normalized_master_filepath.

Referenced by csound::Composition::normalizeOutputSoundfile(), and processArgs().

6.58.3.55 getOutputDirectory()

```
std::string csound::Composition::getOutputDirectory ( ) const  [virtual], [inherited]
```

Returns the directory in which to place the output files of this.

References csound::fundamentalDomainByPredicate(), and csound::Composition::output_directory.

Referenced by csound::Composition::generateAllNames().

6.58.3.56 getOutputSoundfileFilepath()

```
std::string csound::Composition::getOutputSoundfileFilepath ( ) const  [virtual], [inherited]
```

Returns a soundfile name based on the filename of this, by appending ".wav" to the filename, which is the default, or a non-default ouput name which need not be a file but must be set using setOutputSoundfileName().

References csound::Composition::master_filepath, and csound::Composition::output_filename.

Referenced by getCsoundCommand(), csound::Composition::normalizeOutputSoundfile(), perform(), processArgs(), csound::Composition::translateMaster(), and csound::Composition::translateToCdAudio().

6.58.3.57 getPerformanceRightsOrganization()

```
std::string csound::Composition::getPerformanceRightsOrganization ( ) const  [virtual], [inherited]
```

References csound::Composition::performance_rights_organization.

6.58.3.58 getScore()

```
Score & csound::Composition::getScore ( )  [virtual], [inherited]
```

Return the self-contained Score.

References csound::Composition::score.

Referenced by csoundArgv(), and processArgs().

6.58.3.59 getThis()

```
intptr_t csound::MusicModel::getThis ( )  [virtual]
```

Returns the address of this as a long integer.

Reimplemented from csound::ScoreModel.

References csound::fundamentalDomainByPredicate().

6.58.3.60 getThisNode()

```
Node * csound::MusicModel::getThisNode ( )  [virtual]
```

Returns the address of this as a Node pointer.

Reimplemented from csound::ScoreModel.

6.58.3.61 getTieOverlappingNotes()

```
bool csound::Composition::getTieOverlappingNotes ( ) const  [virtual], [inherited]
```

Returns whether or not overlapping notes in generated scores are replaced by one note.

References csound::Composition::tieOverlappingNotes.

Referenced by csound::ScoreModel::generate().

6.58.3.62 getTimestamp()

```
std::string csound::Composition::getTimestamp ( ) const  [virtual], [inherited]
```

Returns the time the score was generated.

References csound::Composition::timestamp.

Referenced by csound::Composition::tagFile().

6.58.3.63 getTitle()

```
std::string csound::Composition::getTitle ( ) const  [virtual], [inherited]
```

References csound::Composition::stem.

Referenced by csound::Composition::generateAllNames(), csound::Composition::tagFile(), csound::Composition::translateToMp3(), csound::Composition::translateToMp4(), and csound::Composition::translateToNotation().

6.58.3.64 getTonesPerOctave()

```
double csound::Composition::getTonesPerOctave ( ) const  [virtual], [inherited]
```

Returns the number of equally tempered intervals per octave (the default is 12, 0 means non-equally tempered).

References csound::Composition::tonesPerOctave.

Referenced by csound::ScoreModel::generate().

6.58.3.65 getYear()

```
std::string csound::Composition::getYear ( ) const  [virtual], [inherited]
```

References csound::Composition::year.

6.58.3.66 initialize()

```
void csound::MusicModel::initialize ( )  [virtual]
```

Reimplemented from csound::ScoreModel.

6.58.3.67 makeTimestamp()

```
std::string csound::Composition::makeTimestamp ( )  [static], [inherited]
```

Returns the current locale time as a string.

References csound::fundamentalDomainByPredicate().

Referenced by csound::Composition::generateAllNames(), and csound::Composition::generateFilename().

6.58.3.68 normalizeOutputSoundfile()

```
int csound::Composition::normalizeOutputSoundfile (
            double levelDb = -3.0 )  [virtual], [inherited]
```

Assuming the score has been rendered, uses sox to translate the output soundfile to a normalized soundfile.

References csound::fundamentalDomainByPredicate(), csound::Composition::getNormalizedSoundfileFilepath(), csound::Composition::getOutputSoundfileFilepath(), csound::System::inform(), and csound::Composition::tagFile().

Referenced by csound::Composition::translateMaster().

6.58.3.69 perform()

```
int csound::MusicModel::perform ( )  [virtual]
```

Uses Csound to perform the current score.

If a Csound command has been set in this, that is used; otherwise, if an output soundfile has been specified in this, a command line is generated and used; otherwise, a default command line is used.

In all cases, a CSD file is generated in memory and rendered.

Reimplemented from csound::Composition.

References cppSound, createCsoundScore(), csoundScoreHeader, csound::fundamentalDomainByPredicate(), CsoundFile::getCommand(), CsoundFile::getCSD(), CppSound::getCsound(), getCsoundCommand(), getExtendSeconds(), csound::Composition::getOutputSoundfileFilepath(), csound::System::message(), CsoundFile::setCommand(), csound::System::startTiming(), and csound::System::stopTiming().

Referenced by render().

6.58.3.70 performAll()

```
int csound::Composition::performAll ( )  [virtual], [inherited]
```

Convenience function that calls performMaster(), and translateMaster().

References csound::fundamentalDomainByPredicate(), csound::System::inform(), csound::System::message(), csound::Composition::performMaster(), csound::System::startTiming(), csound::System::stopTiming(), and csound::Composition::translate

Referenced by csound::Composition::renderAll().

6.58.3.71 performMaster()

```
int csound::Composition::performMaster ( )  [virtual], [inherited]
```

Convenience function that calls saveMidi(), saveMusicXML(), and perform().

References csound::fundamentalDomainByPredicate(), csound::System::inform(), and csound::Composition::perform().

Referenced by csound::Composition::performAll().

6.58.3.72 processArgs()

```
int csound::MusicModel::processArgs (
            const std::vector< std::string > & args )  [virtual]
```

Pass the invoking program's command-line arguments to processArgs() and it will perform the following commands:

–csound Render generated score using set Csound orchestra. –midi Render generated score as MIDI file and play it (default). –pianoteq Play generated MIDI sequence file with Pianoteq. –pianoteq-wav Render score to soundfile using Pianoteq, post-process it, and play it. –playmidi Play generated MIDI filev post-process it, and play it. –playwav Play rendered or normalized output soundfile. –post Post-process Csound output soundfile: normalize, CD, MP3, tag, and play it.

If none of these are given, all command-line arguments are passed directly to Csound.

Reimplemented from csound::Composition.

References csound::fundamentalDomainByPredicate(), generate(), csound::Composition::generateAllNames(), csound::Composition::getMidifileFilepath(), csound::Composition::getNormalizedSoundfileFilepath(), csound::Composition::getOutputSou csound::Composition::getScore(), csound::System::inform(), render(), csound::Score::save(), setCsoundCommand(), csound::Composition::setOutputDirectory(), csound::Composition::translateMaster(), and csound::Composition::translateToNotation().

6.58.3.73 processArgv()

```
int csound::Composition::processArgv (
            int argc,
            const char ** argv )  [virtual], [inherited]
```

Pass the invoking program's command-line arguments to processArgs() and it will perform with possibly back-end-dependent options.

Default implementation calls the std::string overload.

References csound::fundamentalDomainByPredicate(), and csound::Composition::processArgs().

Referenced by main().

6.58.3.74 removeArrangement()

```
void csound::MusicModel::removeArrangement ( )  [virtual]
```

Remove instrument number, gain, and pan assignments (convenience wrapper for Score::removeArrangement()).

References csound::Score::removeArrangement(), and csound::Composition::score.

6.58.3.75 render()

```
int csound::MusicModel::render ( )  [virtual]
```

Convenience function that erases the existing score, appends optional text to it, invokes generate(), invokes createCsoundScore(), and invokes perform().

Reimplemented from csound::Composition.

References csound::fundamentalDomainByPredicate(), generate(), and perform().

Referenced by csoundArgv(), and processArgs().

6.58.3.76 renderAll()

```
int csound::Composition::renderAll ( )  [virtual], [inherited]
```

Convenience function that calls clear(), generate(), performAll().

References csound::Composition::clear(), csound::fundamentalDomainByPredicate(), csound::Composition::generate(), and csound::Composition::performAll().

Referenced by csound::Composition::processArgs().

6.58.3.77 setAlbum()

```
void csound::Composition::setAlbum (
            std::string value )  [virtual], [inherited]
```

References csound::Composition::album.

Referenced by main().

6.58.3.78 setArtist()

```
void csound::Composition::setArtist (
            std::string value )  [virtual], [inherited]
```

References csound::Composition::artist.

6.58.3.79 setAuthor()

```
void csound::Composition::setAuthor (
            std::string value )  [virtual], [inherited]
```

References csound::Composition::author.

Referenced by main().

6.58.3.80 setConformPitches()

```
void csound::Composition::setConformPitches (
            bool conformPitches )  [virtual], [inherited]
```

Sets whether or not the pitches in generated scores will be conformed to the nearest equally tempered pitch.

References csound::Composition::conformPitches.

6.58.3.81 setCopyright()

```
void csound::Composition::setCopyright (
            std::string value )  [virtual], [inherited]
```

References csound::Composition::copyright.

6.58.3.82 setCsoundCommand()

```
void csound::MusicModel::setCsoundCommand (
            std::string command )  [virtual]
```

Set Csound command line (convenience wrapper for CppSound::setCommand()).

References cppSound, and CsoundFile::setCommand().

Referenced by csoundArgv(), and processArgs().

6.58.3.83 setCsoundOrchestra()

```
void csound::MusicModel::setCsoundOrchestra (
            std::string orchestra )  [virtual]
```

Set the Csound orchestra (convenience wrapper for CppSound::setOrchestra()).

References cppSound, and CsoundFile::setOrchestra().

Referenced by main().

6.58.3.84 setCsoundScoreHeader()

```
void csound::MusicModel::setCsoundScoreHeader (
            std::string header )  [virtual]
```

Set a Csound score fragment to be prepended to the generated score (createCsoundScore is called with it).

References csoundScoreHeader, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.58.3.85 setDuration()

```
void csound::Composition::setDuration (
            double seconds )  [virtual], [inherited]
```

At the end of processing, if the defined duration is not zero, the times and durations of all events are rescaled to the defined duration.

References csound::Composition::duration, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.58.3.86 setElement()

```
virtual void csound::ScoreModel::setElement (
            size_t row,
            size_t column,
            double value )  [inline], [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented from csound::Node.

6.58.3.87 setExtendSeconds()

```
virtual void csound::MusicModel::setExtendSeconds (
            double extendSeconds_ )  [inline], [virtual]
```

6.58.3.88 setFilename()

```
void csound::Composition::setFilename (
            std::string filename )  [virtual], [inherited]
```

Sets the filename of this – basically, the title of the composition.

References csound::Composition::stem.

6.58.3.89 setLicense()

```
void csound::Composition::setLicense (
            std::string value )  [virtual], [inherited]
```

References csound::Composition::license.

6.58.3.90 setOutputDirectory()

```
void csound::Composition::setOutputDirectory (
            std::string directory )  [virtual], [inherited]
```

Sets the directory in which to place the output files of this.

The directory name must end with a directory separator.

References csound::fundamentalDomainByPredicate(), and csound::Composition::output_directory.

Referenced by processArgs().

6.58.3.91 setOutputSoundfileName()

```
void csound::Composition::setOutputSoundfileName (
            std::string name )  [virtual], [inherited]
```

Sets a non-default output name (could be an audio device not a file).

References csound::Composition::output_filename.

6.58.3.92 setPerformanceRightsOrganization()

```
void csound::Composition::setPerformanceRightsOrganization (
            std::string value )  [virtual], [inherited]
```

References csound::Composition::performance_rights_organization.

Referenced by main().

6.58.3.93 setScore()

```
void csound::Composition::setScore (
            Score & score )  [virtual], [inherited]
```

Sets the score in this to the indicated score.

References csound::fundamentalDomainByPredicate(), and csound::Composition::score.

6.58.3.94 setTieOverlappingNotes()

```
void csound::Composition::setTieOverlappingNotes (
            bool tieOverlappingNotes )  [virtual], [inherited]
```

Sets whether or not overlapping notes in generated scores are replaced by one note.

References csound::fundamentalDomainByPredicate(), and csound::Composition::tieOverlappingNotes.

Referenced by main().

6.58.3.95 setTitle()

```
void csound::Composition::setTitle (
            std::string value )  [virtual], [inherited]
```

References csound::Composition::stem.

Referenced by main().

6.58.3.96 setTonesPerOctave()

```
void csound::Composition::setTonesPerOctave (
            double tonesPerOctave )  [virtual], [inherited]
```

Sets the number of equally tempered intervals per octave (the default is 12, 0 means non-equally tempered).

References csound::Composition::tonesPerOctave.

6.58.3.97 setYear()

```
void csound::Composition::setYear (
            std::string value )  [virtual], [inherited]
```

References csound::Composition::year.

Referenced by main().

6.58.3.98 stop()

```
void csound::MusicModel::stop ( )  [virtual]
```

References cppSound, csound::System::inform(), and CppSound::stop().

6.58.3.99 tagFile()

```
int csound::Composition::tagFile (
            std::string filename ) const  [virtual], [inherited]
```

References csound::fundamentalDomainByPredicate(), csound::Composition::getAlbum(), csound::Composition::getArtist(), csound::Composition::getAuthor(), csound::Composition::getCopyright(), csound::Composition::getLicense(), csound::Composition::getTi csound::Composition::getTitle(), and csound::System::inform().

Referenced by csound::Composition::normalizeOutputSoundfile(), csound::Composition::translateMaster(), and csound::Composition::translateToCdAudio().

6.58.3.100 transform()

```
virtual void csound::ScoreModel::transform (
            Score & score_from_children )  [inline], [virtual], [inherited]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented from csound::Node.

6.58.3.101 translateMaster()

```
int csound::Composition::translateMaster ( )  [virtual], [inherited]
```

Convenience function that calls rescaleOutputSoundfile(), translateToCdAudio(), and translateToMp3().

References csound::fundamentalDomainByPredicate(), csound::Composition::getOutputSoundfileFilepath(), csound::System::inform(), csound::System::message(), csound::Composition::normalizeOutputSoundfile(), csound::System::startTiming(), csound::System::stopTiming(), csound::Composition::tagFile(), csound::Composition::translateToCdAudio(), csound::Composition::transla and csound::Composition::translateToMp4().

Referenced by csound::Composition::performAll(), and processArgs().

6.58.3.102 translateToCdAudio()

```
int csound::Composition::translateToCdAudio (
            double levelDb = -3.0 )  [virtual], [inherited]
```

Assuming the score has been rendered, uses sox to translate the output soundfile to normalized CD-audio format.

References csound::fundamentalDomainByPredicate(), csound::Composition::getCdSoundfileFilepath(), csound::Composition::getOutput csound::System::inform(), and csound::Composition::tagFile().

Referenced by csound::Composition::translateMaster().

6.58.3.103 translateToMp3()

```
int csound::Composition::translateToMp3 (
            double bitrate = 256.01,
            double levelDb = -3.0 )  [virtual], [inherited]
```

Assuming the score has been rendered, uses sox and LAME to translate the output soundfile to normalized MP3 format.

References csound::Composition::author, csound::fundamentalDomainByPredicate(), csound::Composition::getAlbum(), csound::Composition::getAuthor(), csound::Composition::getCdSoundfileFilepath(), csound::Composition::getCopyright(), csound::Composition::getMp3SoundfileFilepath(), csound::Composition::getTitle(), csound::System::inform(), and csound::Composition::year.

Referenced by csound::Composition::translateMaster().

6.58.3.104 translateToMp4()

```
int csound::Composition::translateToMp4 ( )  [virtual], [inherited]
```

Assuming the score has been rendered, uses sox and ffmpeg to translate the output soundfile to a normalized mp4 video suitable for uploading to YouTube.

References csound::Composition::album, csound::Composition::artist, csound::Composition::author, csound::Composition::cd_quality_file csound::Composition::copyright, csound::fundamentalDomainByPredicate(), csound::Composition::getCopyright(), csound::Composition::getTitle(), csound::System::inform(), csound::Composition::master_filepath, csound::Composition::mp4_filepath, csound::Composition::notes, csound::Composition::performance_rights_organization, csound::Composition::spectrogram_filepath, csound::Composition::stem, csound::Composition::track, and csound::Composition::year.

Referenced by csound::Composition::translateMaster().

6.58.3.105 translateToNotation()

```
int csound::Composition::translateToNotation (
            const std::vector< std::string > partNames = std::vector<std::string>(),
            std::string header = "" )  [virtual], [inherited]
```

Saves the generated score in Fomus format and uses Fomus and Lilypond to translate that to a PDF of music notation.

A meter of 4/4 and a tempo of MM 120 is assumed. A vector of part names may be supplied.

References csound::Composition::duration, csound::fundamentalDomainByPredicate(), csound::Composition::getArtist(), csound::Score::getDuration(), csound::Composition::getFomusfileFilepath(), csound::Composition::getTitle(), csound::iterator(), csound::Conversions::round(), csound::Composition::score, and csound::Score::sort().

Referenced by processArgs().

6.58.3.106 traverse()

```
virtual void csound::ScoreModel::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [inline], [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented from csound::Node.

Referenced by generate(), and csound::ScoreModel::generate().

6.58.3.107 write()

```
void csound::Composition::write (
            const char * text )  [virtual], [inherited]
```

Write as if to stderr.

References csound::fundamentalDomainByPredicate(), and csound::System::message().

6.58.4 Field Documentation

6.58.4.1 album

```
std::string csound::Composition::album  [protected], [inherited]
```

Optional metadata.

Referenced by csound::Composition::generateAllNames(), csound::Composition::getAlbum(), csound::Composition::setAlbum(), and csound::Composition::translateToMp4().

6.58.4.2 artist

```
std::string csound::Composition::artist  [protected], [inherited]
```

Required metadata.

Allows for performer, etc. to differ from author. Defaults to author.

Referenced by csound::Composition::generateAllNames(), csound::Composition::getArtist(), csound::Composition::setArtist(), and csound::Composition::translateToMp4().

6.58.4.3 author

```
std::string csound::Composition::author  [protected], [inherited]
```

Required metadata.

Referenced by csound::Composition::generateAllNames(), csound::Composition::getAuthor(), csound::Composition::setAuthor(), csound::Composition::translateToMp3(), and csound::Composition::translateToMp4().

6.58.4.4 base_filepath

```
std::string csound::Composition::base_filepath  [protected], [inherited]
```

Generated.

The dirname and stem of the output files.

Referenced by csound::Composition::generateAllNames(), csound::Composition::getBasename(), csound::Composition::getFileFilepath(), and csound::Composition::getMusicXmlfileFilepath().

6.58.4.5 baseScore

```
Score csound::Composition::baseScore  [protected], [inherited]
```

6.58.4.6 bext_description

```
std::string csound::Composition::bext_description  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames().

6.58.4.7 bext_orig_ref

```
std::string csound::Composition::bext_orig_ref  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames().

6.58.4.8 bext_originator

```
std::string csound::Composition::bext_originator  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames().

6.58.4.9 cd_quality_filepath

```
std::string csound::Composition::cd_quality_filepath  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames(), csound::Composition::getCdSoundfileFilepath(), and csound::Composition::translateToMp4().

6.58.4.10 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.58.4.11 conformPitches

```
bool csound::Composition::conformPitches  [protected], [inherited]
```

Referenced by createCsoundScore(), csound::Composition::getConformPitches(), and csound::Composition::setConformPitches().

6.58.4.12 copyright

```
std::string csound::Composition::copyright  [protected], [inherited]
```

Required metadata.

Referenced by csound::Composition::generateAllNames(), csound::Composition::getCopyright(), csound::Composition::setCopyright(), and csound::Composition::translateToMp4().

6.58.4.13 cppSound

```
CppSound* csound::MusicModel::cppSound  [protected]
```

Pointer to a Csound object that is used to render scores.

Defaults to the internal Csound object, but can be re-set to an external Csound object.

Referenced by arrange(), arrange(), arrange(), clear(), createCsoundScore(), generate(), getCsoundCommand(), getCsoundOrchestra(), perform(), setCsoundCommand(), setCsoundOrchestra(), and stop().

6.58.4.14 cppSound_

```
CppSound csound::MusicModel::cppSound_  [protected]
```

Self-contained Csound object.

6.58.4.15 csoundScoreHeader

```
std::string csound::MusicModel::csoundScoreHeader  [protected]
```

Prepended to generated score.

Referenced by getCsoundScoreHeader(), perform(), and setCsoundScoreHeader().

6.58.4.16 duration

```
double csound::Composition::duration  [protected], [inherited]
```

Referenced by generate(), csound::ScoreModel::generate(), csound::Composition::getDuration(), csound::Composition::setDuration(), and csound::Composition::translateToNotation().

6.58.4.17 extendSeconds

```
double csound::MusicModel::extendSeconds = -1  [protected]
```

Referenced by createCsoundScore().

6.58.4.18 flac_filepath

```
std::string csound::Composition::flac_filepath  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames().

6.58.4.19 label

```
std::string csound::Composition::label  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames().

6.58.4.20 license

```
std::string csound::Composition::license  [protected], [inherited]
```

Required metadata.

Defaults to Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International.

Referenced by csound::Composition::getLicense(), and csound::Composition::setLicense().

6.58.4.21 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.58.4.22 master_filepath

```
std::string csound::Composition::master_filepath  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames(), csound::Composition::getOutputSoundfileFilepath(), and csound::Composition::translateToMp4().

6.58.4.23 midi_filepath

```
std::string csound::Composition::midi_filepath  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames(), and csound::Composition::getMidifileFilepath().

6.58.4.24 mp3_filepath

```
std::string csound::Composition::mp3_filepath  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames(), and csound::Composition::getMp3SoundfileFilepath().

6.58.4.25 mp4_filepath

```
std::string csound::Composition::mp4_filepath  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames(), and csound::Composition::translateToMp4().

6.58.4.26 normalized_master_filepath

```
std::string csound::Composition::normalized_master_filepath  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames(), and csound::Composition::getNormalizedSoundfileFilepath().

6.58.4.27 notes

```
std::string csound::Composition::notes  [protected], [inherited]
```

Optional metadata, defaults to "Electroacoustic Music.".

Referenced by csound::Composition::generateAllNames(), and csound::Composition::translateToMp4().

6.58.4.28 output_directory

```
std::string csound::Composition::output_directory  [protected], [inherited]
```

Required.

The target directory of the output files. Defaults to the current working directory.

Referenced by csound::Composition::generateAllNames(), csound::Composition::getOutputDirectory(), and csound::Composition::setOutp

6.58.4.29 output_filename

```
std::string csound::Composition::output_filename  [protected], [inherited]
```

Referenced by csound::Composition::clearOutputSoundfileName(), csound::Composition::getOutputSoundfileFilepath(), and csound::Composition::setOutputSoundfileName().

6.58.4.30 performance_rights_organization

```
std::string csound::Composition::performance_rights_organization  [protected], [inherited]
```

Optional metadata.

Referenced by csound::Composition::generateAllNames(), csound::Composition::getPerformanceRightsOrganization(), csound::Composition::setPerformanceRightsOrganization(), and csound::Composition::translateToMp4().

6.58.4.31 score

```
Score& csound::Composition::score  [protected], [inherited]
```

Referenced by arrange(), arrange(), arrange(), csound::Composition::clear(), createCsoundScore(), generate(), csound::ScoreModel::generate(), csound::Composition::getScore(), removeArrangement(), csound::Composition::setScore(), and csound::Composition::translateToNotation().

6.58.4.32 spectrogram_filepath

```
std::string csound::Composition::spectrogram_filepath  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames(), and csound::Composition::translateToMp4().

6.58.4.33 stem

```
std::string csound::Composition::stem  [protected], [inherited]
```

Required.

The stem must be a valid filename and also represents the title. All other names, text, and commands are generated from directory, stem, filename extensions, and required metadata.

Referenced by csound::Composition::generateAllNames(), csound::Composition::getFilename(), csound::Composition::getTitle(), csound::Composition::setFilename(), csound::Composition::setTitle(), and csound::Composition::translateToMp4().

6.58.4.34 threadCount

```
int csound::MusicModel::threadCount
```

Referenced by getCsoundCommand().

6.58.4.35 tieOverlappingNotes

```
bool csound::Composition::tieOverlappingNotes  [protected], [inherited]
```

Referenced by csound::Composition::getTieOverlappingNotes(), and csound::Composition::setTieOverlappingNotes().

6.58.4.36 timestamp

```
std::string csound::Composition::timestamp  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames(), and csound::Composition::getTimestamp().

6.58.4.37 tonesPerOctave

```
double csound::Composition::tonesPerOctave  [protected], [inherited]
```

Referenced by createCsoundScore(), csound::Composition::getTonesPerOctave(), and csound::Composition::setTonesPerOctave().

6.58.4.38 track

```
std::string csound::Composition::track  [protected], [inherited]
```

Optional metadata.

Referenced by csound::Composition::generateAllNames(), and csound::Composition::translateToMp4().

6.58.4.39 year

```
std::string csound::Composition::year  [protected], [inherited]
```

Required metadata.

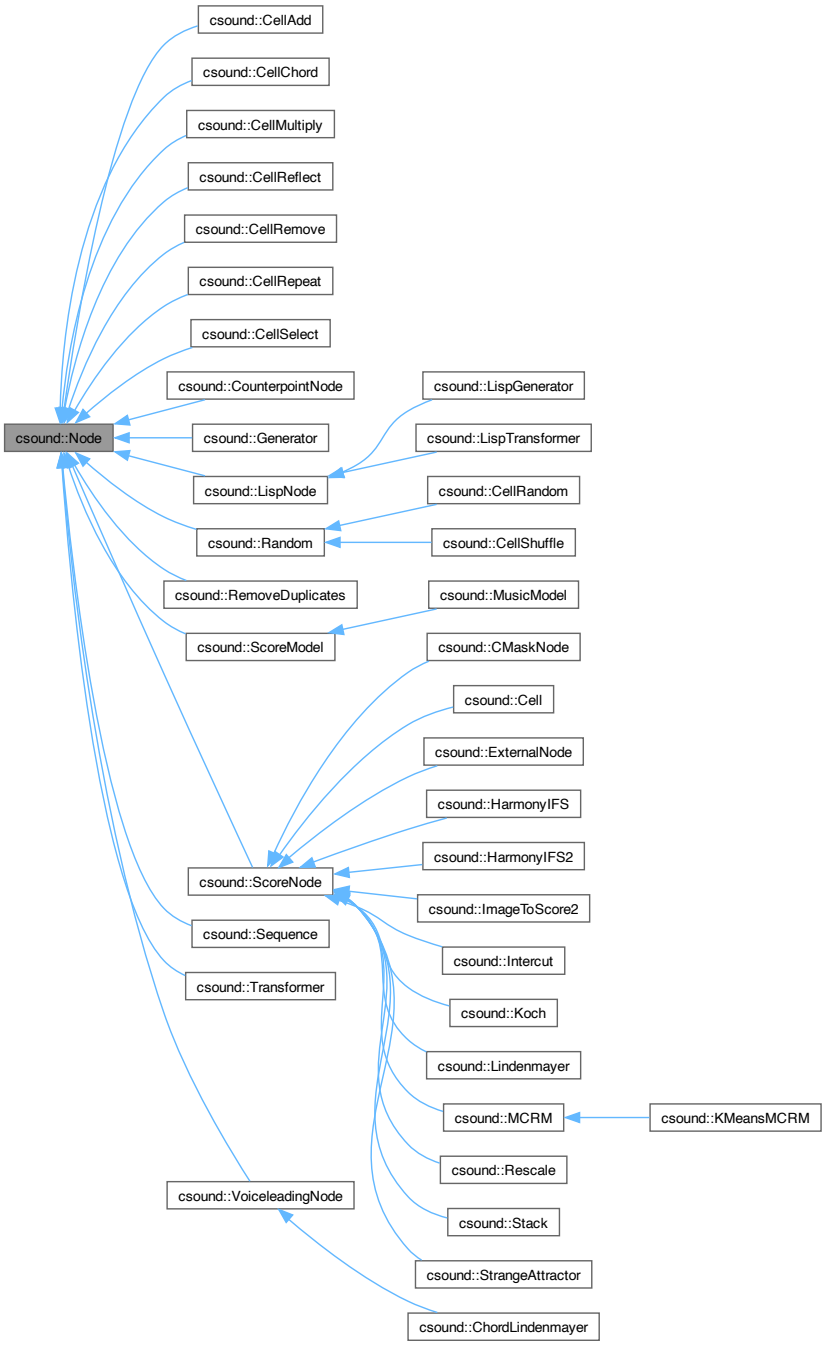
Referenced by csound::Composition::generateAllNames(), csound::Composition::getYear(), csound::Composition::setYear(), csound::Composition::translateToMp3(), and csound::Composition::translateToMp4().

6.59 csound::Node Class Reference

Base class for all music graph nodes in the Silence system.

`#include <Node.hpp>`

Inheritance diagram for csound::Node:



Public Member Functions

- virtual void addChild (Node ∗node)

  Adds an immediate child Node to this.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &score_from_this)

  Optionally generate notes into the score.
- virtual Node ∗ getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- Node ()
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void transform (Score &score_from_children)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.
- virtual ∼Node ()

Data Fields

- std::vector< Node ∗ > children

  Child Nodes, if any.

Protected Attributes

- Eigen::MatrixXd localCoordinates

6.59.1 Detailed Description

Base class for all music graph nodes in the Silence system.

A music graph consists of a root Node, with any number of child Nodes, which in turn may have any number of child Nodes. To render the music graph, the root Node::traverse function is called, which performs a depth-first recursive traversal of the music graph.

Nodes can transform silence::Events produced by child nodes. Nodes can generate silence::Events.

Nodes can also perform score generation and processing outside of a music graph.

Nodes follow the pattern of initalization with controlling parameters, and then invocation either within a music graph, or by calling Node::generate or Node::transform outside of a music graph.

6.59.2 Constructor & Destructor Documentation

6.59.2.1 Node()

```
csound::Node::Node ( )
```

References createTransform(), csound::Event::ELEMENT_COUNT, and localCoordinates.

6.59.2.2 ~Node()

```
csound::Node::~Node ( )  [virtual]
```

6.59.3 Member Function Documentation

6.59.3.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.59.3.2 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References children.

6.59.3.3 clear()

```
void csound::Node::clear ( )  [virtual]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References children, clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), clear(), and csound::ScoreModel::clear().

6.59.3.4 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by Node(), and csound::MCRM::resize().

6.59.3.5 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References localCoordinates.

6.59.3.6 generate()

```
void csound::Node::generate (
            Score & score_from_this )  [virtual]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented in csound::ExternalNode, csound::ScoreNode, csound::ChordLindenmayer, csound::MCRM, csound::Generator, csound::Random, csound::LispGenerator, and csound::ScoreModel.

Referenced by traverse().

6.59.3.7 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References children.

6.59.3.8 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.59.3.9 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References localCoordinates.

6.59.3.10 transform()

```
void csound::Node::transform (
            Score & score_from_children )  [virtual]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented in csound::Cell, csound::CellRepeat, csound::CellAdd, csound::CellMultiply, csound::CellReflect, csound::CellSelect, csound::CellRemove, csound::CellChord, csound::CellRandom, csound::CellShuffle, csound::CounterpointNode, csound::RemoveDuplicates, csound::Transformer, csound::Random, csound::Rescale, csound::VoiceleadingNode, csound::LispTransformer, and csound::ScoreModel.

Referenced by traverse().

6.59.3.11 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References children, csound::fundamentalDomainByPredicate(), generate(), getLocalCoordinates(), and transform().

6.59.4 Field Documentation

6.59.4.1 children

```
std::vector<Node *> csound::Node::children
```

Child Nodes, if any.

Referenced by addChild(), childCount(), clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), getChild(), traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.59.4.2 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected]
```

Referenced by element(), getLocalCoordinates(), Node(), and setElement().

6.60 csound::PITV Class Reference

This class implements a cyclic additive group for all chords under cardinality, permutational, and range equivalence.

```
#include <ChordSpaceBase.hpp>
```

Public Member Functions

- Eigen::VectorXi fromChord (const Chord &chord, bool printme=false) const

  Returns the indices of prime form, inversion, transposition, and voicing for a chord, as the first 4 elements, respectively, of a homogeneous vector.
- virtual int getCountI () const
- virtual int getCountP () const
- virtual int getCountT () const
- virtual int getCountV () const
- virtual int getG () const
- virtual int getN () const
- virtual int getRange () const
- virtual void initialize (int N_, double range_, double g_=1., bool printme=false)
- virtual void list (bool listheader=true, bool listps=false, bool listvoicings=false) const
- virtual void preinitialize (int N_, double range_, double g_=1.0)
- std::vector< Chord > toChord (int P, int I, int T, int V, bool printme=false) const

  Returns the chord for the indices of prime form, inversion, transposition, and voicing.
- std::vector< Chord > toChord_vector (const Eigen::VectorXi &pitv, bool printme=false) const
- virtual ~PITV ()

Data Fields

- int countI
- int countP
- int countT
- int countV
- double g

  The generator of transposition.
- std::map< Chord, int > indexesForPs
- int N
- std::set< Chord > normal_forms
- std::map< int, Chord > PsForIndexes
- double range

  The 0-based range of the chord space.

6.60.1 Detailed Description

This class implements a cyclic additive group for all chords under cardinality, permutational, and range equivalence.

It is formed by the direct product of prime form equivalence or P, inversional equivalence or I, transpositional equivalence or T, and equivalence under octavewise revoicing within range R or V. The group is thus PITV = P x I x T x V. Therefore, operations on the P, I, T, or V subgroups may be used to independently and orthogonally transform the respective symmetry of any chord. Some of these operations will reflect in RP.

NOTE: Prime form rather than OPTI is used because prime form abstracts from voicings (i.e. from the sectors of the OPT cyclical region).

6.60.2 Constructor & Destructor Documentation

6.60.2.1 ~PITV()

```
SILENCE_PUBLIC csound::PITV::~PITV ( )  [inline], [virtual]
```

6.60.3 Member Function Documentation

6.60.3.1 fromChord()

```
Eigen::VectorXi csound::PITV::fromChord (
            const Chord & chord,
            bool printme = false ) const  [inline]
```

Returns the indices of prime form, inversion, transposition, and voicing for a chord, as the first 4 elements, respectively, of a homogeneous vector.

NOTE: Where there are singularities in the quotient spaces for chords, there may be several chords that belong to the same equivalence class. In such cases, any of several chords at a singular point of the fundamental domain will return the same P.

References csound::chord(), csound::Chord::eOP(), csound::Chord::eppcs(), csound::I(), csound::indexForOctavewiseRevoicing(), csound::Chord::inverse_prime_form(), csound::Chord::normal_form(), csound::octavewiseRevoicing(), csound::Chord::prime_form(), csound::print_chord(), and csound::T().

Referenced by test_pitv(), and test_pitv().

6.60.3.2 getCountI()

```
SILENCE_PUBLIC int csound::PITV::getCountI ( ) const  [inline], [virtual]
```

6.60.3.3 getCountP()

```
SILENCE_PUBLIC int csound::PITV::getCountP ( ) const  [inline], [virtual]
```

6.60.3.4 getCountT()

```
SILENCE_PUBLIC int csound::PITV::getCountT ( ) const  [inline], [virtual]
```

6.60.3.5 getCountV()

```
SILENCE_PUBLIC int csound::PITV::getCountV ( ) const  [inline], [virtual]
```

6.60.3.6 getG()

```
SILENCE_PUBLIC int csound::PITV::getG ( ) const  [inline], [virtual]
```

6.60.3.7 getN()

```
SILENCE_PUBLIC int csound::PITV::getN ( ) const  [inline], [virtual]
```

6.60.3.8 getRange()

```
SILENCE_PUBLIC int csound::PITV::getRange ( ) const  [inline], [virtual]
```

6.60.3.9 initialize()

```
SILENCE_PUBLIC void csound::PITV::initialize (
            int N_,
            double range_,
            double g_ = 1.,
            bool printme = false )  [inline], [virtual]
```

References csound::Chord::clamp(), csound::Chord::inverse_prime_form(), csound::iterator(), csound::next(), csound::Chord::normal_form(), csound::Chord::prime_form(), and csound::print_chord().

Referenced by main(), and test_pitv().

6.60.3.10 list()

```
SILENCE_PUBLIC void csound::PITV::list (
            bool listheader = true,
            bool listps = false,
            bool listvoicings = false ) const  [inline], [virtual]
```

References csound::chord(), csound::Chord::eOP(), csound::Chord::eOPTT(), csound::Chord::eOPTTI(), csound::Chord::inverse_prime_f
csound::Chord::opt_domain_sectors(), csound::Chord::prime_form(), csound::print_chord(), and csound::Chord::toString().

Referenced by main(), and test_pitv().

6.60.3.11 preinitialize()

```
SILENCE_PUBLIC void csound::PITV::preinitialize (
            int N_,
            double range_,
            double g_ = 1.0 )  [inline], [virtual]
```

References csound::chord(), csound::OCTAVE(), csound::octavewiseRevoicings(), and csound::Chord::resize().

6.60.3.12 toChord()

```
std::vector< Chord > csound::PITV::toChord (
            int P,
            int I,
            int T,
            int V,
            bool printme = false ) const  [inline]
```

Returns the chord for the indices of prime form, inversion, transposition, and voicing.

The chord is not in RP; rather, each voice of the chord's OP may have zero or more octaves added to it.

References csound::I(), csound::octavewiseRevoicing(), csound::print_chord(), csound::T(), and csound::toString().

Referenced by test_pitv(), and test_pitv().

6.60.3.13 toChord_vector()

```
SILENCE_PUBLIC std::vector< Chord > csound::PITV::toChord_vector (
            const Eigen::VectorXi & pitv,
            bool printme = false ) const  [inline]
```

Referenced by test_pitv().

6.60.4 Field Documentation

6.60.4.1 countI

```
int csound::PITV::countI
```

Referenced by test_pitv().

6.60.4.2 countP

```
int csound::PITV::countP
```

Referenced by test_pitv().

6.60.4.3 countT

```
int csound::PITV::countT
```

Referenced by test_pitv().

6.60.4.4 countV

```
int csound::PITV::countV
```

Referenced by test_pitv().

6.60.4.5 g

```
double csound::PITV::g
```

The generator of transposition.

6.60.4.6 indexesForPs

```
std::map<Chord, int> csound::PITV::indexesForPs
```

6.60.4.7 N

```
int csound::PITV::N
```

6.60.4.8 normal_forms

```
std::set<Chord> csound::PITV::normal_forms
```

6.60.4.9 PsForIndexes

```
std::map<int, Chord> csound::PITV::PsForIndexes
```

6.60.4.10 range

```
double csound::PITV::range
```

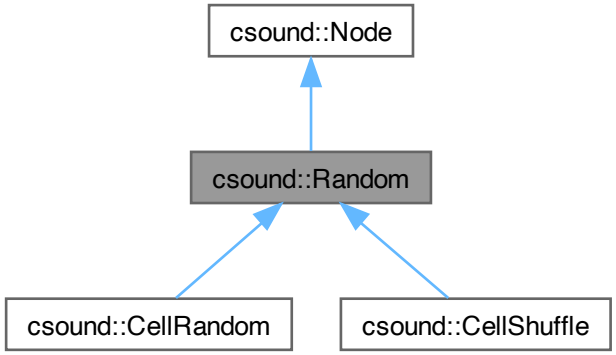
The 0-based range of the chord space.

6.61 csound::Random Class Reference

A random value will be sampled from the specified distribution, translated and scaled as specified, and set in the specified row and column of the local coordinates.

`#include <Random.hpp>`

Inheritance diagram for csound::Random:



Public Member Functions

- virtual void addChild (Node *node)
  Adds an immediate child Node to this.
- virtual size_t childCount () const
  Returns the number of immediate children of this.
- virtual void clear ()
  Recursively clears all child Nodes of this.
- virtual void createDistribution (std::string distribution)
- virtual Eigen::MatrixXd createTransform ()
  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)
  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &score)
  Optionally generate notes into the score.
- virtual Node * getChild (size_t index)
  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const
  Returns the local transformation of coordinate system.
- virtual Eigen::MatrixXd getRandomCoordinates ()
- Random ()

- virtual double sample ()
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void transform (Score &score)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.
- virtual ~Random ()

Static Public Member Functions

- static void seed (int s)

Data Fields

- double a
- double b
- double c
- std::vector< Node * > children

  Child Nodes, if any.
- int column
- std::string distribution
- int eventCount
- bool incrementTime
- double Lambda
- double maximum
- double mean
- double minimum
- double q
- int row
- double sigma

Static Public Attributes

- static std::mt19937 mersenneTwister

Protected Attributes

- std::bernoulli_distribution bernoulli_distribution_generator
- std::exponential_distribution exponential_distribution_generator
- void * generator_
- std::geometric_distribution geometric_distribution_generator
- Eigen::MatrixXd localCoordinates
- std::lognormal_distribution lognormal_distribution_generator
- std::normal_distribution normal_distribution_generator
- std::uniform_int_distribution< std::int64_t > uniform_int_generator
- std::uniform_real_distribution uniform_real_generator
- std::uniform_int_distribution< std::int32_t > uniform_smallint_generator

6.61.1 Detailed Description

A random value will be sampled from the specified distribution, translated and scaled as specified, and set in the specified row and column of the local coordinates.

The resulting matrix will be used in place of the local coordinates when traversing the music graph. If eventCount is greater than zero, a new event will be created for each of eventCount samples, which will be transformed by the newly sampled local coordinates.

6.61.2 Constructor & Destructor Documentation

6.61.2.1 Random()

```
csound::Random::Random ( )
```

References distribution.

6.61.2.2 ~Random()

```
csound::Random::~Random ( )  [virtual]
```

6.61.3 Member Function Documentation

6.61.3.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.61.3.2 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.61.3.3 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.61.3.4 createDistribution()

```
void csound::Random::createDistribution (
            std::string distribution )  [virtual]
```

References bernoulli_distribution_generator, distribution, exponential_distribution_generator, generator_, geometric_distribution_generato Lambda, lognormal_distribution_generator, maximum, mean, minimum, normal_distribution_generator, q, sigma, uniform_int_generator, uniform_real_generator, and uniform_smallint_generator.

Referenced by generate(), csound::StrangeAttractor::StrangeAttractor(), csound::CellRandom::transform(), and transform().

6.61.3.5 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.61.3.6 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.61.3.7 generate()

```
void csound::Random::generate (
            Score & score_from_this )  [virtual]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented from csound::Node.

References createDistribution(), distribution, eventCount, csound::fundamentalDomainByPredicate(), getRandomCoordinates(), and incrementTime.

6.61.3.8 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.61.3.9 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.61.3.10 getRandomCoordinates()

```
Eigen::MatrixXd csound::Random::getRandomCoordinates ( )  [virtual]
```

References csound::fundamentalDomainByPredicate(), csound::Node::getLocalCoordinates(), csound::Event::HOMOGENEITY, and sample().

Referenced by generate(), and transform().

6.61.3.11 sample()

```
double csound::Random::sample ( )  [virtual]
```

References bernoulli_distribution_generator, exponential_distribution_generator, generator_, geometric_distribution_generator, lognormal_distribution_generator, mersenneTwister, normal_distribution_generator, uniform_int_generator, uniform_real_generator, and uniform_smallint_generator.

Referenced by csound::StrangeAttractor::calculateFractalDimension(), csound::StrangeAttractor::codeRandomize(), getRandomCoordinates(), csound::StrangeAttractor::render(), csound::StrangeAttractor::shuffleRandomNumbers(), and csound::CellRandom::transform().

6.61.3.12 seed()

```
void csound::Random::seed (
            int s )  [static]
```

References mersenneTwister.

6.61.3.13 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.61.3.14 transform()

```
void csound::Random::transform (
            Score & score_from_children )  [virtual]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented from csound::Node.

Reimplemented in csound::CellRandom, and csound::CellShuffle.

References createDistribution(), distribution, eventCount, csound::fundamentalDomainByPredicate(), and getRandomCoordinates().

6.61.3.15 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord and csound::Node::transform().

6.61.4 Field Documentation

6.61.4.1 a

```
double csound::Random::a
```

6.61.4.2 b

```
double csound::Random::b
```

6.61.4.3 bernoulli_distribution_generator

```
std::bernoulli_distribution csound::Random::bernoulli_distribution_generator  [protected]
```

Referenced by createDistribution(), and sample().

6.61.4.4 c

```
double csound::Random::c
```

6.61.4.5 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.61.4.6 column

```
int csound::Random::column
```

6.61.4.7 distribution

```
std::string csound::Random::distribution
```

Referenced by createDistribution(), generate(), Random(), and transform().

6.61.4.8 eventCount

```
int csound::Random::eventCount
```

Referenced by generate(), and transform().

6.61.4.9 exponential_distribution_generator

```
std::exponential_distribution csound::Random::exponential_distribution_generator  [protected]
```

Referenced by createDistribution(), and sample().

6.61.4.10 generator_

```
void* csound::Random::generator_  [protected]
```

Referenced by createDistribution(), and sample().

6.61.4.11 geometric_distribution_generator

```
std::geometric_distribution csound::Random::geometric_distribution_generator  [protected]
```

Referenced by createDistribution(), and sample().

6.61.4.12 incrementTime

```
bool csound::Random::incrementTime
```

Referenced by generate().

6.61.4.13 Lambda

```
double csound::Random::Lambda
```

Referenced by createDistribution().

6.61.4.14 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.61.4.15 lognormal_distribution_generator

```
std::lognormal_distribution csound::Random::lognormal_distribution_generator  [protected]
```

Referenced by createDistribution(), and sample().

6.61.4.16 maximum

```
double csound::Random::maximum
```

Referenced by createDistribution().

6.61.4.17 mean

```
double csound::Random::mean
```

Referenced by createDistribution().

6.61.4.18 mersenneTwister

```
std::mt19937 csound::Random::mersenneTwister  [static]
```

Referenced by sample(), seed(), and csound::CellShuffle::transform().

6.61.4.19 minimum

```
double csound::Random::minimum
```

Referenced by createDistribution().

6.61.4.20 normal_distribution_generator

```
std::normal_distribution csound::Random::normal_distribution_generator  [protected]
```

Referenced by createDistribution(), and sample().

6.61.4.21 q

```
double csound::Random::q
```

Referenced by createDistribution().

6.61.4.22 row

```
int csound::Random::row
```

6.61.4.23 sigma

```
double csound::Random::sigma
```

Referenced by createDistribution().

6.61.4.24 uniform_int_generator

```
std::uniform_int_distribution<std::int64_t> csound::Random::uniform_int_generator  [protected]
```

Referenced by createDistribution(), and sample().

6.61.4.25 uniform_real_generator

```
std::uniform_real_distribution csound::Random::uniform_real_generator  [protected]
```

Referenced by createDistribution(), and sample().

6.61.4.26 uniform_smallint_generator

```
std::uniform_int_distribution<std::int32_t> csound::Random::uniform_smallint_generator  [protected]
```

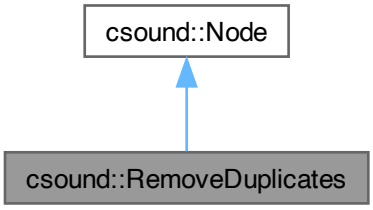
Referenced by createDistribution(), and sample().

6.62 csound::RemoveDuplicates Class Reference

Removes all duplicate events produced by the child nodes of this.

`#include <Node.hpp>`

Inheritance diagram for csound::RemoveDuplicates:



Public Member Functions

- virtual void addChild (Node *node)

  Adds an immediate child Node to this.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &score_from_this)

  Optionally generate notes into the score.
- virtual Node * getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void transform (Score &score)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

Data Fields

- std::vector< Node * > children
  Child Nodes, if any.

Protected Attributes

- Eigen::MatrixXd localCoordinates

6.62.1 Detailed Description

Removes all duplicate events produced by the child nodes of this.

6.62.2 Member Function Documentation

6.62.2.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.62.2.2 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.62.2.3 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.62.2.4 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.62.2.5 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.62.2.6 generate()

```
void csound::Node::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented in csound::ExternalNode, csound::ScoreNode, csound::ChordLindenmayer, csound::MCRM, csound::Generator, csound::Random, csound::LispGenerator, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.62.2.7 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.62.2.8 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.62.2.9 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.62.2.10 transform()

```
void csound::RemoveDuplicates::transform (
            Score & score_from_children )  [virtual]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented from csound::Node.

References csound::fundamentalDomainByPredicate(), and csound::Event::toCsoundIStatement().

6.62.2.11 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord
and csound::Node::transform().

6.62.3 Field Documentation

6.62.3.1 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.62.3.2 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

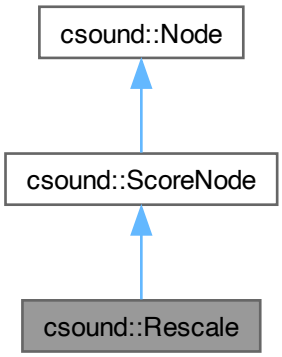
Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.63 csound::Rescale Class Reference

Rescales all child events to fit a bounding hypercube in music space.

```
#include <Rescale.hpp>
```

Inheritance diagram for csound::Rescale:



Public Member Functions

- virtual void addChild (Node *node)

  Adds an immediate child Node to this.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &collectingScore)

  Optionally generate notes into the score.
- virtual Node * getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual void getRescale (int dimension, bool &rescaleMinimum, bool &rescaleRange, double &targetMinimum, double &targetRange)
- virtual Score & getScore ()
- virtual void initialize ()
- Rescale ()
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void setRescale (int dimension, bool rescaleMinimum, bool rescaleRange, double targetMinimum, double targetRange)
- virtual void transform (Score &score)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.
- virtual ~Rescale ()

Data Fields

- std::vector< Node * > children

  Child Nodes, if any.
- double duration

  If not 0, the score is rescaled to this duration.
- std::string importFilename

Protected Attributes

- Eigen::MatrixXd localCoordinates
- Score score

6.63.1 Detailed Description

Rescales all child events to fit a bounding hypercube in music space.

No, some, or all dimensions may be rescaled to fit the minimum alone, the range alone, or both the minimum and the range.

6.63.2 Constructor & Destructor Documentation

6.63.2.1 Rescale()

```
csound::Rescale::Rescale ( )
```

References csound::Event::ELEMENT_COUNT, initialize(), csound::Score::rescaleMinima, csound::Score::rescaleRanges, and csound::ScoreNode::score.

6.63.2.2 ~Rescale()

```
csound::Rescale::~Rescale ( )  [virtual]
```

6.63.3 Member Function Documentation

6.63.3.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.63.3.2 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.63.3.3 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.63.3.4 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.63.3.5 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.63.3.6 generate()

```
void csound::ScoreNode::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented from csound::Node.

Reimplemented in csound::ExternalNode, and csound::MCRM.

References csound::ScoreNode::duration, csound::fundamentalDomainByPredicate(), csound::Score::getCsoundScoreHeader(), csound::ScoreNode::importFilename, csound::Score::load(), csound::Score::process(), csound::ScoreNode::score, csound::Score::setDuration(), and csound::Score::sort().

Referenced by csound::MCRM::generate().

6.63.3.7 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.63.3.8 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.63.3.9 getRescale()

```
void csound::Rescale::getRescale (
            int dimension,
            bool & rescaleMinimum,
            bool & rescaleRange,
            double & targetMinimum,
            double & targetRange )  [virtual]
```

References csound::fundamentalDomainByPredicate(), csound::Score::rescaleMinima, csound::Score::rescaleRanges, csound::Score::scaleTargetMinima, csound::Score::scaleTargetRanges, and csound::ScoreNode::score.

6.63.3.10 getScore()

```
Score & csound::ScoreNode::getScore ( )  [virtual], [inherited]
```

References csound::ScoreNode::score.

Referenced by main().

6.63.3.11 initialize()

```
void csound::Rescale::initialize ( )  [virtual]
```

References csound::Event::ELEMENT_COUNT, and csound::Event::labels.

Referenced by Rescale().

6.63.3.12 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.63.3.13 setRescale()

```
void csound::Rescale::setRescale (
            int dimension,
            bool rescaleMinimum,
            bool rescaleRange,
            double targetMinimum,
            double targetRange )  [virtual]
```

References csound::fundamentalDomainByPredicate(), csound::Score::rescaleMinima, csound::Score::rescaleRanges, csound::Score::scaleTargetMinima, csound::Score::scaleTargetRanges, and csound::ScoreNode::score.

Referenced by main().

6.63.3.14 transform()

```
void csound::Rescale::transform (
            Score & score_from_children )  [virtual]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented from csound::Node.

References csound::Event::ELEMENT_COUNT, csound::fundamentalDomainByPredicate(), csound::Score::rescaleMinima, csound::Score::rescaleRanges, csound::Score::scaleTargetMinima, csound::Score::scaleTargetRanges, and csound::ScoreNode::score.

6.63.3.15 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord and csound::Node::transform().

6.63.4 Field Documentation

6.63.4.1 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.63.4.2 duration

```
double csound::ScoreNode::duration  [inherited]
```

If not 0, the score is rescaled to this duration.

Referenced by csound::ScoreNode::generate(), csound::ExternalNode::generateLocally(), and csound::Stack::getDuration().

6.63.4.3 importFilename

```
std::string csound::ScoreNode::importFilename  [inherited]
```

Referenced by csound::ScoreNode::generate().

6.63.4.4 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.63.4.5 score

```
Score csound::ScoreNode::score  [protected], [inherited]
```

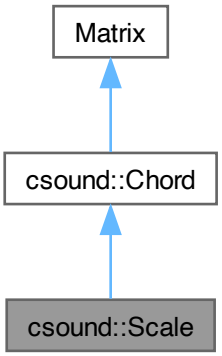
Referenced by csound::StrangeAttractor::evaluateAttractor(), csound::ExternalNode::generate(), csound::ScoreNode::generate(), csound::MCRM::generate(), csound::ExternalNode::generateLocally(), csound::ImageToScore2::generateLocally(), csound::Lindenmayer::generateLocally(), getRescale(), csound::ScoreNode::getScore(), csound::Lindenmayer::interpret(), csound::MCRM::iterate(), csound::StrangeAttractor::iterate_without_rendering(), csound::KMeansMCRM::means_to_notes(), csound::ImageToScore2::pixel_to_event(), csound::StrangeAttractor::render(), Rescale(), setRescale(), csound::Cell::transform(), transform(), csound::CMaskNode::translate_to_silence(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Lindenmayer::updateActual().

6.64 csound::Scale Class Reference

Scale as a class; must be created with the name of the scale.

```
#include <ChordSpaceBase.hpp>
```

Inheritance diagram for csound::Scale:



Public Types

- enum {
  PITCH = 0 , DURATION = 1 , LOUDNESS = 2 , INSTRUMENT = 3 ,
  PAN = 4 , COUNT = 5 }

Public Member Functions

- virtual Chord a (int arpeggiation, double &resultPitch, int &resultVoice) const

  Returns the ith arpeggiation, current voice, and corresponding revoicing of the chord.
- virtual Chord ceiling (double g=1.) const

  Returns a new chord whose pitches are the ceilings of this chord's pitches, with respect to the generator of transposition g, which defaults to 1 semitone.
- virtual Chord center () const

  Returns the maximally even chord in the chord's space, e.g.
- virtual Chord chord (int scale_degree, int voices, int interval=3) const

  Returns the chord for the indicated scale degree, number of voices in the chord, and interval in scale degrees of the chord (defaults to thirds, or 3; the actual number of scale steps between chord pitches is interval - 1).
- virtual void clamp (double g=1.)

  Rounds the pitches in this chord to the nearest integer multiple of g, the generator of transposition.
- virtual Chord clone () const
- virtual bool contains (double pitch_) const

  Returns whether or not the chord contains the pitch.
- virtual size_t count (double pitch) const

  Returns the number of voices in this chord, the same as the number of dimensions in this chord space.
- virtual Chord cycle (int stride=1) const

  Returns a copy of the chord cyclically permuted by a stride, by default 1.
- virtual int degree (const Chord &chord_, int interval=3) const

  Returns the scale degree of the Chord in this Scale; if the Chord does not belong to this Scale, -1 is returned.
- virtual double distanceToOrigin () const

  Returns the Euclidean distance of this chord from its space's origin.
- virtual double distanceToUnisonDiagonal () const

  Returns the Euclidean distance from this chord to the unison diagonal of its chord space.
- virtual Chord eI (int opt_sector=0) const

  Returns the equivalent of the chord within a fundamental domain of inversional equivalence.
- virtual Chord eO () const

  Returns the equivalent of the chord within the representative fundamental domain of octave equivalence.
- virtual Chord eOP () const

  Returns the equivalent of the chord within the representative fundamental domain of octave and permutational equivalence.
- virtual Chord eOPI (int opt_sector=0) const

  Returns the equivalent of the chord within a fundamental domain of octave, permutational, and inversional equivalence.
- virtual Chord eOPT (int opt_sector=0) const

  Returns the equivalent of the chord within a fundamental domain of octave, permutational, and transpositional equivalence.
- virtual Chord eOPTI (int opt_sector=0) const

  Returns the equivalent of the chord within a fundamental domain of range, permutational, transpositional, and inversional equivalence.
- virtual Chord eOPTT (double g=1., int opt_sector=0) const

Returns the equivalent of the chord within a fundamental domain of octave, permutational, and transpositional equivalence but in the equal temperament generated by g.

- virtual Chord eOPTTI (double g=1., int opt_sector=0) const

  Returns the equivalent of the chord within a fundamental domain of range, permutational, transpositional, and inversional equivalence but in the equal temperament generated by g.

- virtual Chord eOT () const

  Returns the equivalent of the chord within the representative fundamental domain of octave and transpositional equivalence.

- virtual Chord eOTT (double g=1.) const

  Returns the equivalent of the chord within a fundamental domain of octave and transpositional equivalence but in the equal temperament generated by g.

- virtual Chord eP () const

  Returns the equivalent of the chord within the representative fundamental domain of permutational equivalence.

- virtual Chord epcs () const

  Returns the equivalent of the chord under pitch-class equivalence, i.e.

- virtual Chord eppcs () const

  Returns the equivalent of the chord under pitch-class equivalence, i.e.

- virtual bool equals (const Chord &other) const

  Returns whether the voices of this chord equal the voices of the other.

- virtual Chord eR (double range) const

  Returns the equivalent of the chord within the representative fundamental domain of a range equivalence.

- virtual Chord eRP (double range) const

  Returns the equivalent of the chord within the representative fundamental domain of range and permutational equivalence.

- virtual Chord eRPI (double range, int opt_sector=0) const

  Returns the equivalent of the chord within a fundamental domain of range, permutational, and inversional equivalence.

- virtual Chord eRPT (double range, int opt_sector=0) const

  Returns the equivalent of the chord within a fundamental domain of range, permutational, and transpositional equivalence.

- virtual Chord eRPTI (double range, int opt_sector=0) const

  Returns the equivalent of the chord within the representative fundamental domain of range, permutational, transpositional, and inversional equivalence.

- virtual std::vector< Chord > eRPTs (double range=OCTAVE()) const

  Returns all equivalents of the chord within all fundamental domains of range, permutational, and transpositional equivalence.

- virtual Chord eRPTT (double range, double g=1., int opt_sector=0) const

  Returns the equivalent of the chord within a fundamental domain of range, permutational, and transpositional equivalence, in the equal temperament generated by g; the same as chord type.

- virtual Chord eRPTTI (double range, double g=1., int opt_sector=0) const

  Returns the equivalent of the chord within a fundamental domain of range, permutational, transpositional, and inversional equivalence.

- virtual std::vector< Chord > eRPTTs (double range, double g=1.) const

  Returns all equivalents of the chord within all fundamental domains of range, permutational, and transpositional equivalence in the equal temperament generated by g; equivalent to all inversions of the chord in the musician's sense.

- virtual Chord eT () const

  Returns the equivalent of the chord within a fundamental domain of range, permutational, transpositional, and inversional equivalence in the equal temperament generated by g; the same as set class.

- virtual Chord et () const

  Returns the equivalent of the chord within the fundamental domain of transposition to 0.

- virtual Chord eTT (double g=1.) const

Returns the equivalent of the chord within the representative fundamental domain of transpositonal equivalence and the equal temperament generated by g, i.e., returns the chord transposed such that its layer is 0 or, under transposition, the positive layer closest to 0.

- virtual Chord floor () const

  Returns a new chord whose pitches are the floors of this chord's pitches.
- virtual void fromString (std::string text)

  Rebuilds the chord's pitches (only) from a line of text.
- virtual double getDuration (int voice=0) const
- virtual double getInstrument (int voice=0) const
- virtual double getLoudness (int voice=0) const
- virtual double getPan (int voice=0) const
- virtual double getPitch (int voice) const
- virtual double & getPitchReference (int voice)
- virtual std::string getTypeName () const

  Returns the type name, e.g.
- virtual bool greater (const Chord &other) const

  Returns whether the voices of this chord are greater than the voices of the other.
- virtual bool greater_equals (const Chord &other) const

  Returns whether the voices of this chord are greater than or equal to the voices of the other.
- virtual HyperplaneEquation hyperplane_equation (int opt_sector) const

  Returns the hyperplane equation for the inversion flat that evenly divides the fundamental domain in the indicated sector of the OPT cyclical region.
- virtual Chord I (double center=0.0) const

  Inverts the chord by another chord that is on the unison diagonal, by default the origin.
- virtual bool Iform (const Chord &Y, double g=1.) const

  Returns whether the chord is an inversional form of Y with interval size g.
- virtual std::string information () const

  Print much information about the chord including whether it is within important equivalence classes, or what its equivalents would be.
- virtual std::string information_debug (int opt_sector) const

  Print much information about the chord including whether it is within important equivalence classes, or what its equivalents would be.
- virtual std::string information_sector (int opt_sector) const

  Print much information about the chord including whether it is within important equivalence classes, or what its equivalents would be.
- virtual void initialize_sectors ()

  Initializes the fundamental domains (sectors) of the cyclical regions of OPT equivalence and OPTI equivalence, as well as the hyperplane equations that define the inversion flat in each OPT sector.
- virtual Chord inverse_prime_form () const

  Returns this chord as the inverse standard "prime form.".
- virtual bool is_compact (double range=12.) const

  Returns whether this chord has a compact voicing.
- virtual bool is_minor () const

  Returns whether this chord is "minor" in the sense of having the smallest "wraparound interval" of all its voicings.
- virtual bool is_opt_sector (int opt_sector=0) const

  Returns whether or not this chord lies within the indicated sector of the cyclical region of OPT fundamental domains.
- virtual bool is_opti_sector (int opti_sector=0) const

  Returns whether or not this chord lies within the indicated sector of the cyclical region of OPTI fundamental domains.
- virtual bool isel (int opt_sector=0) const

- virtual bool iseI_chord (Chord *inverse, int opt_sector=0) const

  Returns whether the chord is within a fundamental domain of inversional equivalence.
- virtual bool iseO () const

  Returns whether the chord is within the representative fundamental domain of octave equivalence.
- virtual bool iseOP () const

  Returns whether the chord is within the representative fundamental domain of octave and permutational equivalence.
- virtual bool iseOPI (int opt_sector=0) const

  Returns whether the chord is within a fundamental domain of octave, permutational, and inversional equivalence.
- virtual bool iseOPT (int opt_sector=0) const

  Returns whether the chord is within a fundamental domainof octave, permutational, and transpositional equivalence.
- virtual bool iseOPTI (int opt_sector=0) const

  Returns whether the chord is within a fundamental domain of octave, permutational, transpositional, and inversional equivalence.
- virtual bool iseOPTT (double g=1., int opt_sector=0) const

  Returns whether the chord is within a fundamental domain of octave, permutational, and transpositional equivalence in the equal temperament generated by g.
- virtual bool iseOPTTI (double g=1., int opt_sector=0) const

  Returns whether the chord is within a fundamental domain of octave, permutational, transpositional, and inversional equivalence in the equal temperament generated by g.
- virtual bool iseOT () const

  Returns whether the chord is within the representative fundamental domain of octave and transpositional equivalence.
- virtual bool iseOTT (double g=1.) const

  Returns whether the chord is within the representative fundamental domain of octave and translational equivalence in the equal temperament generated by g.
- virtual bool iseP () const

  Returns whether the chord is within the representative fundamental domain of permutational equivalence.
- virtual bool isepcs () const

  Returns whether the chord is within the fundamental domain of pitch-class equivalence, i.e.
- virtual bool iseR (double range_) const

  Returns whether the chord is within the representative fundamental domain of the indicated range equivalence.
- virtual bool iseRP (double range) const

  Returns whether the chord is within the representative fundamental domain of range and permutational equivalence.
- virtual bool iseRPI (double range, int opt_sector=0) const

  Returns whether the chord is within a fundamental domain of range, permutational, and inversional equivalence.
- virtual bool iseRPT (double range, int opt_sector=0) const

  Returns whether the chord is within a fundamental domain of range, permutational, and transpositional equivalence.
- virtual bool iseRPTI (double range, int opt_sector=0) const

  Returns whether the chord is within a fundamental domain of range, permutational, transpositional, and inversional equivalence.
- virtual bool iseRPTT (double range, double g=1., int opt_sector=0) const

  Returns whether the chord is within a fundamental domain of range, permutational, and transpositional equivalence in the equal temperament generated by g.
- virtual bool iseRPTTI (double range, double g=1., int opt_sector=0) const

  Returns whether the chord is within a fundamental domain of range, permutational, transpositional, and inversional equivalence in the ' equal temperament generated by g.
- virtual bool iseRT (double range) const

  Returns whether the chord is within the representative fundamental domain of range and transpositional equivalence.
- virtual bool iseRTT (double range, double g=1.) const

Returns whether the chord is within a fundamental domain of range and transpositional equivalence in the equal temperament generated by g.

- virtual bool iseT () const

  Returns whether the chord is within the representative fundamental domain of transpositional equivalence.

- virtual bool iset () const

  Returns whether the chord is within the fundamental domain of transposition to 0.

- virtual bool iseTT (double g=1.) const

  Returns whether the chord is within the representative fundamental domain of transpositional equivalence in the equal temperament generated by g.

- virtual Chord K () const

  Returns the chord inverted by the sum of its first two voices.

- virtual Chord K_range (double range) const
- virtual double layer () const

  Returns the sum of the pitches in the chord.

- virtual bool lesser (const Chord &other) const

  Returns whether the voices of this chord are less than the voices of the other.

- virtual bool lesser_equals (const Chord &other) const

  Returns whether the voices of this chord are less than or equal to the voices of the other.

- virtual std::vector< double > max () const

  Returns the highest pitch in the chord, and also the voice index of that pitch.

- virtual double maximumInterval () const

  Returns the maximum interval within the chord.

- virtual std::vector< double > min () const

  Returns the lowest pitch in the chord, and also the voice index of that pitch.

- virtual double minimumInterval () const

  Returns the minimum interval within the chord.

- virtual std::vector< Scale > modulations (const Chord &chord) const

  Returns a list of common modulations, that is, other major or harmonic minor Scales to which the Chord belongs; optionally the Chord can first be resized (e.g.

- virtual void modulations_for_scale_types (std::vector< Scale > &result, const Chord &current_chord, int voices_, const std::vector< std::string > &type_names) const

  For any Chord belonging to this Scale, returns in the argument a list of other Scales to which that Chord also belongs.

- virtual std::vector< Scale > modulations_for_voices (const Chord &chord, int voices) const
- virtual Chord move (int voice, double interval) const

  Move 1 voice of the chord.

- virtual std::string name () const

  Returns the name of this Scale.

- virtual Chord normal_form () const

  Returns this chord as its standard "normal form.".

- virtual Chord normal_order () const

  Returns this chord in standard "normal order." For a very clear explanation, see: https://www.mta.ca/pc-set/pc-set_new/pages/page04/page04.html and http://openmusictheory.com/normalOrder.html/.

- virtual Chord nrD () const

  Performs the dominant transformation (which is not a neo-Reimannian transformation).

- virtual Chord nrH () const

  Performs the neo-Riemannian hexatonic pole transformation.

- virtual Chord nrL () const

Performs the neo-Riemannian Lettonwechsel transformation.

- virtual Chord nrN () const

  Performs the neo-Riemannian Nebenverwandt transformation.

- virtual Chord nrP () const

  Performs the neo-Riemannian parallel transformation.

- virtual Chord nrR () const

  Performs the neo-Riemannian parallel transformation.

- virtual Chord nrS () const

  Performs the neo-Riemannian Slide transformation.

- virtual operator std::vector< double > () const
- virtual Scale & operator= (const Scale &other)
- virtual std::vector< Chord > opt_domain (int sector) const

  Returns the vertices of the OPT fundamental domain for the indicated sector of the cyclical region.

- virtual std::vector< int > opt_domain_sectors () const

  Returns the zero-based index(s) of the sector(s) within the cyclical region of OPT fundamental domains to which the chord belongs.

- virtual std::vector< Chord > opti_domain (int sector) const

  Returns the vertices of the OPTI fundamental domain for the indicated sector of the cyclical region.

- virtual std::vector< int > opti_domain_sectors () const

  Returns the zero-based index(s) of the sector(s) within the cyclical region of OPTI fundamental domains to which the chord belongs.

- virtual Chord origin () const

  Returns the origin of the chord's space.

- virtual std::vector< Chord > permutations () const

  Returns the permutations of the pitches in a chord.

- virtual Chord prime_form () const

  Returns this chord as its standard "prime form.".

- virtual Chord Q (double x, const Chord &m, double g=1.) const

  Returns the contextual transposition of the chord by x with respect to m with minimum interval size g.

- virtual Chord reflect (int opt_sector) const

  Reflects the chord in the inversion flat of the indicated OPT domain sector.

- virtual std::vector< Scale > relative_tonicizations (const Chord &current_chord, int secondary_function=5, int voices=-1) const

  Returns a list of common relative tonicizations for the Chord, that is, the other major or harmonic minor Scales for which that Chord could be mutated to have the secondary function.

- virtual void relative_tonicizations_for_scale_types (std::vector< Scale > &result, const Chord &current_chord, int secondary_function, int voices, const std::vector< std::string > &type_names) const

  Returns the relative tonicizations of the Chord, that is, the scales for which that Chord could be mutated to have the secondary function, if that is possible.

- virtual void resize (size_t voiceN)
- Scale ()

  Default constructor, an empty Scale.

- Scale (std::string name)

  Creates a Scale by name, e.g.

- Scale (std::string name, const Chord &scale_pitches)

  Creates a Scale with a new name as a set of pitches.

- Scale (std::string name, const std::vector< double > &scale_pitches)

  Creates a Scale with a new name as a set of pitches.

- virtual std::vector< Chord > secondary (const Chord &current_chord, int secondary_function=5, int voices_=-1) const

  Returns the current Chord mutated, if possible, to one or more function(s) with respect to another Chord in its Scale.
- virtual bool self_inverse (int opt_sector=0) const

  Returns whether or not this chord is invariant under reflection in the inversion flat of the indicated OPT sector.
- virtual double semitones_for_degree (int scale_degree) const

  Returns the number of semitones (may be whole or fractional) from the tonic (as 0) of this Scale to the indicated scale degree, which is wrapped around by octave equivalence.
- virtual void setDuration (double value, int voice=-1)
- virtual void setInstrument (double value, int voice=-1)
- virtual void setLoudness (double value, int voice=-1)
- virtual void setPan (double value, int voice=-1)
- virtual void setPitch (int voice, double value)
- virtual Chord T (double interval) const

  Transposes the chord by the indicated interval (may be a fraction).
- virtual Chord T_voiceleading (const Chord &voiceleading)

  Transposes the chord by the indicated voiceleading (passed as a Chord of directed intervals).
- virtual bool test (const char *caption="") const

  Tests the internal consistency of the predicates ("iseX") and transformations ("eX") of this chord, and prints a report.
- virtual bool Tform (const Chord &Y, double g=1.) const

  Returns whether the chord is a transpositional form of Y with interval size g.
- virtual double tonic () const

  Returns the pitch-class that is the tonic or root of this Scale.
- virtual std::vector< Scale > tonicizations (const Chord &current_chord, int voices=-1) const

  Returns all major or minor Scales for which the current Chord is the tonic (scale degree 1).
- virtual std::string toString () const

  Returns a string representation of the chord's pitches (only).
- virtual Scale transpose (double semitones) const

  Returns a copy of this Scale transposed by the indicated number of semitones.
- virtual Chord transpose_degrees (const Chord &chord, int scale_degrees, int interval=3) const

  Returns a Chord transposed by the indicated number of scale degrees; the chord as passed must belong to this Scale, and the interval must be the same as that used to generate the Chord; (defaults to thirds, or 3; the actual number of scale steps between chord pitches is interval - 1).
- virtual Scale transpose_to_degree (int degrees) const

  Returns a copy of this Scale transposed to the indicated scale degree.
- virtual Chord v (int direction=1) const

  Returns a copy of the chord 'inverted' in the musician's sense, i.e.
- virtual Chord voiceleading (const Chord &destination) const

  Returns the transpositions (as a Chord of directed intervals) that takes this chord to the destination chord.
- virtual size_t voices () const

  Returns the number of voices in this chord; that is, the number of dimensions in the chord space for this chord.
- virtual std::vector< Chord > voicings () const

  Returns all the 'inversions' (in the musician's sense) or octavewise revoicings of the chord.
- virtual ~Scale ()

Static Public Member Functions

- static std::map< int, std::vector< Chord > > & cyclical_regions_for_dimensionalities ()

  For each chord space of dimensions 3 <= n <= 12, there is one cyclical region of n fundamental domains of OPT equivalence.
- static std::map< int, std::vector< HyperplaneEquation > > & hyperplane_equations_for_opt_sectors ()

  For each chord space of dimensions 3 <= n <= 12, there are n fundamental domains (sectors) of OPT equivalence.
- static std::map< int, std::vector< std::vector< Chord > > > & opt_sectors_for_dimensionalities ()

  For each chord space of dimensions 3 <= n <= 12, there are n fundamental domains (sectors) of OPT equivalence.
- static std::map< int, std::vector< std::vector< Chord > > > & opt_simplexes_for_dimensionalities ()

  Returns a collection of vertices for the OPT fundamental domains; each has an added vertex to make a simplex for chord location.
- static std::map< int, std::vector< std::vector< Chord > > > & opti_sectors_for_dimensionalities ()

  For each chord space of dimensions 3 <= n <= 12, there are n fundamental domains (sectors) of OPTI equivalence.
- static std::map< int, std::vector< std::vector< Chord > > > & opti_simplexes_for_dimensionalities ()

  Returns a collection of vertices for the OPTI fundamental domains that have an added vertex to make a simplex for chord location.
- static double rownd (double x, int places=12)

  Rounds the value of x to the specified number of decimal places.

Protected Attributes

- std::string type_name

6.64.1 Detailed Description

Scale as a class; must be created with the name of the scale.

Inherits from Chord. Note that inherited Chord member functions such as T and I return Chords, not Scales.

6.64.2 Member Enumeration Documentation

6.64.2.1 anonymous enum

```
anonymous enum  [inherited]
```

Enumerator

PITCH	
DURATION	
LOUDNESS	
INSTRUMENT	
PAN	
COUNT	

6.64.3 Constructor & Destructor Documentation

6.64.3.1 Scale() [1/4]

```
SILENCE_PUBLIC csound::Scale::Scale ( )  [inline]
```

Default constructor, an empty Scale.

6.64.3.2 Scale() [2/4]

```
SILENCE_PUBLIC csound::Scale::Scale (
            std::string name )  [inline]
```

Creates a Scale by name, e.g.

'C major'. If the named Scale does not already exist, an empty Scale without a name is created.

References csound::scale(), and csound::Chord::voices().

6.64.3.3 Scale() [3/4]

```
SILENCE_PUBLIC csound::Scale::Scale (
            std::string name,
            const Chord & scale_pitches )  [inline]
```

Creates a Scale with a new name as a set of pitches.

These must start in octave 0 and be in ascending order, but otherwise may have any value in semitones or fractions of semitones; this permits the construction of new scales with any temperament and with any interval content. If a Scale with the proposed name already exists, that Scale is returned. New Scales are also stored as new named Scales.

References csound::add_scale(), csound::Chord::getPitch(), and csound::Chord::voices().

6.64.3.4 Scale() [4/4]

```
SILENCE_PUBLIC csound::Scale::Scale (
            std::string name,
            const std::vector< double > & scale_pitches )  [inline]
```

Creates a Scale with a new name as a set of pitches.

These must start in octave 0 and be in ascending order, but otherwise may have any value in semitones or fractions of semitones; this permits the construction of new scales with any temperament and with any interval content. If a Scale with the proposed name already exists, that Scale is replaced. New Scales are also stored as new named Scales.

References csound::add_scale().

6.64.3.5 ~Scale()

```
SILENCE_PUBLIC csound::Scale::~Scale ( )  [inline], [virtual]
```

6.64.4 Member Function Documentation

6.64.4.1 a()

```
Chord csound::Chord::a (
            int arpeggiation,
            double & resultPitch,
            int & resultVoice ) const  [inline], [virtual], [inherited]
```

Returns the ith arpeggiation, current voice, and corresponding revoicing of the chord.

Positive arpeggiations start with the lowest voice of the chord and revoice up; negative arpeggiations start with the highest voice of the chord and revoice down.

References csound::Chord::getPitch(), and csound::Chord::voices().

6.64.4.2 ceiling()

```
Chord csound::Chord::ceiling (
            double g = 1.  )  const  [inline], [virtual], [inherited]
```

Returns a new chord whose pitches are the ceilings of this chord's pitches, with respect to the generator of transposition g, which defaults to 1 semitone.

References CHORD_SPACE_DEBUG, csound::print_chord(), and csound::Chord::setPitch().

Referenced by csound::equate< EQUIVALENCE_RELATION_Tg >(), main(), and csound::predicate< EQUIVALENCE_RELATION_Tg >

6.64.4.3 center()

```
Chord csound::Chord::center ( ) const  [inline], [virtual], [inherited]
```

Returns the maximally even chord in the chord's space, e.g.

the augmented triad for 3 dimensions.

References csound::OCTAVE(), and csound::Chord::setPitch().

Referenced by csound::hyperplane_equation_from_random_inversion_flat(), csound::Chord::initialize_sectors(), main(), csound::reflect_by_householder(), csound::reflect_in_central_diagonal(), and csound::reflect_in_central_point().

6.64.4.4 chord()

```
SILENCE_PUBLIC Chord csound::Scale::chord (
            int scale_degree,
            int voices,
            int interval = 3 ) const  [inline], [virtual]
```

Returns the chord for the indicated scale degree, number of voices in the chord, and interval in scale degrees of the chord (defaults to thirds, or 3; the actual number of scale steps between chord pitches is interval - 1).

References csound::chord().

Referenced by csound::ChordLindenmayer::scaleDegreeOperation(), and csound::ChordLindenmayer::scaleOperation().

6.64.4.5 clamp()

```
void csound::Chord::clamp (
            double g = 1.  )  [inline], [virtual], [inherited]
```

Rounds the pitches in this chord to the nearest integer multiple of g, the generator of transposition.

This is valid only if g goes evenly into 12 (the octave), i.e. in 12/g tone equal temperament.

References csound::OCTAVE().

Referenced by csound::PITV::initialize().

6.64.4.6 clone()

```
virtual Chord csound::Chord::clone ( ) const  [inline], [virtual], [inherited]
```

6.64.4.7 contains()

```
bool csound::Chord::contains (
            double pitch_ ) const  [inline], [virtual], [inherited]
```

Returns whether or not the chord contains the pitch.

References csound::eq_tolerance().

Referenced by main().

6.64.4.8 count()

```
size_t csound::Chord::count (
            double pitch ) const  [inline], [virtual], [inherited]
```

Returns the number of voices in this chord, the same as the number of dimensions in this chord space.

References csound::eq_tolerance().

Referenced by main(), csound::parallelFifth(), and csound::voiceleadingSimpler().

6.64.4.9 cycle()

```
Chord csound::Chord::cycle (
            int stride = 1 ) const  [inline], [virtual], [inherited]
```

Returns a copy of the chord cyclically permuted by a stride, by default 1.

The direction of rotation is by default the same as musicians' first inversion, second inversion, and so on; but negative sign will reverse the direction of rotation.

- 1 is pop the front and push it on the back, shifting the middle down. 0 1 2 3 4 => 1 2 3 4 0
- 1 is pop the back and push it on the front, shifting the middle up. 0 1 2 3 4 => 4 0 1 2 3

Referenced by main(), csound::Chord::permutations(), and csound::Chord::v().

6.64.4.10 cyclical_regions_for_dimensionalities()

```
std::map< int, std::vector< Chord > > & csound::Chord::cyclical_regions_for_dimensionalities ( )
[inline], [static], [inherited]
```

For each chord space of dimensions 3 <= n <= 12, there is one cyclical region of n fundamental domains of OPT equivalence.

The vertices of the cyclical region consist of the n octavewise revoicings of the origin. This function returns a global collection of these cyclical regions.

6.64.4.11 degree()

```
SILENCE_PUBLIC int csound::Scale::degree (
            const Chord & chord_,
            int interval = 3 ) const  [inline], [virtual]
```

Returns the scale degree of the Chord in this Scale; if the Chord does not belong to this Scale, -1 is returned.

References csound::chord(), csound::Chord::eOP(), and csound::Chord::voices().

Referenced by csound::ChordLindenmayer::scaleOperation().

6.64.4.12 distanceToOrigin()

```
double csound::Chord::distanceToOrigin ( ) const  [inline], [virtual], [inherited]
```

Returns the Euclidean distance of this chord from its space's origin.

References csound::euclidean().

Referenced by main().

6.64.4.13 distanceToUnisonDiagonal()

```
double csound::Chord::distanceToUnisonDiagonal ( ) const  [inline], [virtual], [inherited]
```

Returns the Euclidean distance from this chord to the unison diagonal of its chord space.

References csound::euclidean(), and csound::Chord::setPitch().

Referenced by main().

6.64.4.14 eI()

```
Chord csound::Chord::eI (
            int opt_sector = 0 ) const  [inline], [virtual], [inherited]
```

Returns the equivalent of the chord within a fundamental domain of inversional equivalence.

References csound::equate< EQUIVALENCE_RELATION_I >(), and csound::OCTAVE().

Referenced by csound::equate< EQUIVALENCE_RELATION_RPI >(), and main().

6.64.4.15 eO()

```
Chord csound::Chord::eO ( ) const  [inline], [virtual], [inherited]
```

Returns the equivalent of the chord within the representative fundamental domain of octave equivalence.

References csound::OCTAVE().

Referenced by main().

6.64.4.16 eOP()

```
Chord csound::Chord::eOP ( ) const  [inline], [virtual], [inherited]
```

Returns the equivalent of the chord within the representative fundamental domain of octave and permutational equivalence.

References csound::OCTAVE().

Referenced by csound::addVoice(), csound::ChordLindenmayer::chordOperation(), degree(), csound::fill(), csound::PITV::fromChord(), is_k_dual(), csound::PITV::list(), csound::ChordLindenmayer::modalityOperation(), csound::octavewiseRevoicings(), csound::compare_by_op::operator()(), csound::removeVoice(), test_pitv(), csound::transpose_degrees(), and csound::voiceleadingClosestRange().

6.64.4.17 eOPI()

```
Chord csound::Chord::eOPI (
            int opt_sector = 0 ) const  [inline], [virtual], [inherited]
```

Returns the equivalent of the chord within a fundamental domain of octave, permutational, and inversional equivalence.

References csound::OCTAVE().

6.64.4.18 eOPT()

```
Chord csound::Chord::eOPT (
            int opt_sector = 0 ) const  [inline], [virtual], [inherited]
```

Returns the equivalent of the chord within a fundamental domain of octave, permutational, and transpositional equivalence.

References csound::OCTAVE().

6.64.4.19 eOPTI()

```
Chord csound::Chord::eOPTI (
            int opt_sector = 0 ) const  [inline], [virtual], [inherited]
```

Returns the equivalent of the chord within a fundamental domain of range, permutational, transpositional, and inversional equivalence.

References csound::OCTAVE().

6.64.4.20 eOPTT()

```
Chord csound::Chord::eOPTT (
            double g = 1.,
            int opt_sector = 0 ) const  [inline], [virtual], [inherited]
```

Returns the equivalent of the chord within a fundamental domain of octave, permutational, and transpositional equivalence but in the equal temperament generated by g.

References csound::OCTAVE().

Referenced by is_k_dual(), csound::PITV::list(), and setDifference().

6.64.4.21 eOPTTI()

```
Chord csound::Chord::eOPTTI (
            double g = 1.,
            int opt_sector = 0 ) const  [inline], [virtual], [inherited]
```

Returns the equivalent of the chord within a fundamental domain of range, permutational, transpositional, and inversional equivalence but in the equal temperament generated by g.

References csound::OCTAVE().

Referenced by csound::PITV::list(), and test_pitv().

6.64.4.22 eOT()

```
Chord csound::Chord::eOT ( ) const  [inline], [virtual], [inherited]
```

Returns the equivalent of the chord within the representative fundamental domain of octave and transpositional equivalence.

6.64.4.23 eOTT()

```
Chord csound::Chord::eOTT (
            double g = 1.  )  const  [inline], [virtual], [inherited]
```

Returns the equivalent of the chord within a fundamental domain of octave and transpositional equivalence but in the equal temperament generated by g.

6.64.4.24 eP()

```
Chord csound::Chord::eP ( ) const  [inline], [virtual], [inherited]
```

Returns the equivalent of the chord within the representative fundamental domain of permutational equivalence.

References csound::equate< EQUIVALENCE_RELATION_P >(), and csound::OCTAVE().

Referenced by csound::Chord::eppcs(), csound::hyperplane_equation_from_random_inversion_flat(), csound::Chord::Iform(), main(), and csound::Chord::Tform().

6.64.4.25 epcs()

```
Chord csound::Chord::epcs ( ) const  [inline], [virtual], [inherited]
```

Returns the equivalent of the chord under pitch-class equivalence, i.e.

the pitch-class set of the chord.

References csound::chord(), csound::epc(), and csound::Chord::setPitch().

Referenced by csound::conformToChord_equivalence(), csound::Chord::Iform(), main(), and csound::Chord::Tform().

6.64.4.26 eppcs()

```
Chord csound::Chord::eppcs ( ) const  [inline], [virtual], [inherited]
```

Returns the equivalent of the chord under pitch-class equivalence, i.e.

the pitch-class set of the chord, sorted by pitch-class.

References csound::chord(), csound::Chord::eP(), csound::epc(), and csound::Chord::setPitch().

Referenced by csound::PITV::fromChord().

6.64.4.27 equals()

```
bool csound::Chord::equals (
            const Chord & other ) const  [inline], [virtual], [inherited]
```

Returns whether the voices of this chord equal the voices of the other.

Referenced by test_pitv().

6.64.4.28 eR()

```
Chord csound::Chord::eR (
            double range ) const  [inline], [virtual], [inherited]
```

Returns the equivalent of the chord within the representative fundamental domain of a range equivalence.

References csound::equate< EQUIVALENCE_RELATION_R >().

6.64.4.29 eRP()

```
Chord csound::Chord::eRP (
            double range ) const  [inline], [virtual], [inherited]
```

Returns the equivalent of the chord within the representative fundamental domain of range and permutational equivalence.

References csound::equate< EQUIVALENCE_RELATION_RP >().

Referenced by csound::equate< EQUIVALENCE_RELATION_RPI >(), and csound::Chord::K_range().

6.64.4.30 eRPI()

```
Chord csound::Chord::eRPI (
            double range,
            int opt_sector = 0 ) const  [inline], [virtual], [inherited]
```

Returns the equivalent of the chord within a fundamental domain of range, permutational, and inversional equivalence.

References csound::equate< EQUIVALENCE_RELATION_RPI >().

6.64.4.31 eRPT()

```
Chord csound::Chord::eRPT (
            double range,
            int opt_sector = 0 ) const  [inline], [virtual], [inherited]
```

Returns the equivalent of the chord within a fundamental domain of range, permutational, and transpositional equivalence.

References csound::equate< EQUIVALENCE_RELATION_RPT >().

6.64.4.32 eRPTI()

```
Chord csound::Chord::eRPTI (
            double range,
            int opt_sector = 0 ) const  [inline], [virtual], [inherited]
```

Returns the equivalent of the chord within the representative fundamental domain of range, permutational, transpositional, and inversional equivalence.

References csound::equate< EQUIVALENCE_RELATION_RPTI >().

6.64.4.33 eRPTs()

```
std::vector< Chord > csound::Chord::eRPTs (
            double range = OCTAVE() ) const  [inline], [virtual], [inherited]
```

Returns all equivalents of the chord within all fundamental domains of range, permutational, and transpositional equivalence.

Referenced by csound::equate< EQUIVALENCE_RELATION_RPT >().

6.64.4.34 eRPTT()

```
Chord csound::Chord::eRPTT (
            double range,
            double g = 1.,
            int opt_sector = 0 ) const  [inline], [virtual], [inherited]
```

Returns the equivalent of the chord within a fundamental domain of range, permutational, and transpositional equivalence, in the equal temperament generated by g; the same as chord type.

References csound::equate< EQUIVALENCE_RELATION_RPTg >().

6.64.4.35 eRPTTI()

```
Chord csound::Chord::eRPTTI (
            double range,
            double g = 1.,
            int opt_sector = 0 ) const  [inline], [virtual], [inherited]
```

Returns the equivalent of the chord within a fundamental domain of range, permutational, transpositional, and inversional equivalence.

References csound::equate< EQUIVALENCE_RELATION_RPTgI >().

6.64.4.36 eRPTTs()

```
std::vector< Chord > csound::Chord::eRPTTs (
            double range,
            double g = 1.  )  const  [inline], [virtual], [inherited]
```

Returns all equivalents of the chord within all fundamental domains of range, permutational, and transpositional equivalence in the equal temperament generated by g; equivalent to all inversions of the chord in the musician's sense.

Referenced by csound::equate< EQUIVALENCE_RELATION_RPTg >().

6.64.4.37 eT()

```
Chord csound::Chord::eT ( ) const  [inline], [virtual], [inherited]
```

Returns the equivalent of the chord within a fundamental domain of range, permutational, transpositional, and inversional equivalence in the equal temperament generated by g; the same as set class.

References csound::equate< EQUIVALENCE_RELATION_T >(), and csound::OCTAVE().

Referenced by csound::equate< EQUIVALENCE_RELATION_Tg >(), csound::hyperplane_equation_from_random_inversion_flat(), csound::Chord::initialize_sectors(), main(), csound::predicate< EQUIVALENCE_RELATION_Tg >(), and csound::reflect_by_householder

6.64.4.38 et()

```
Chord csound::Chord::et ( ) const  [inline], [virtual], [inherited]
```

Returns the equivalent of the chord within the fundamental domain of transposition to 0.

References csound::T().

Referenced by main().

6.64.4.39 eTT()

```
Chord csound::Chord::eTT (
            double g = 1.  )  const  [inline], [virtual], [inherited]
```

Returns the equivalent of the chord within the representative fundamental domain of transpositonal equivalence and the equal temperament generated by g, i.e., returns the chord transposed such that its layer is 0 or, under transposition, the positive layer closest to 0.

NOTE: Does NOT return the result under any other equivalence class.

References csound::equate< EQUIVALENCE_RELATION_Tg >(), and csound::OCTAVE().

Referenced by main().

6.64.4.40 floor()

```
Chord csound::Chord::floor ( ) const  [inline], [virtual], [inherited]
```

Returns a new chord whose pitches are the floors of this chord's pitches.

References csound::Chord::setPitch().

Referenced by main().

6.64.4.41 fromString()

```
void csound::Chord::fromString (
            std::string text )  [inline], [virtual], [inherited]
```

Rebuilds the chord's pitches (only) from a line of text.

6.64.4.42 getDuration()

```
double csound::Chord::getDuration (
            int voice = 0 ) const  [inline], [virtual], [inherited]
```

Referenced by csound::note(), and csound::toScore().

6.64.4.43 getInstrument()

```
double csound::Chord::getInstrument (
            int voice = 0 ) const  [inline], [virtual], [inherited]
```

Referenced by csound::note(), and csound::toScore().

6.64.4.44 getLoudness()

```
double csound::Chord::getLoudness (
            int voice = 0 ) const  [inline], [virtual], [inherited]
```

Referenced by csound::note(), and csound::toScore().

6.64.4.45 getPan()

```
double csound::Chord::getPan (
            int voice = 0 ) const  [inline], [virtual], [inherited]
```

Referenced by csound::note(), and csound::toScore().

6.64.4.46 getPitch()

```
double csound::Chord::getPitch (
            int voice ) const  [inline], [virtual], [inherited]
```

Referenced by csound::Chord::a(), csound::addVoice(), csound::ChordLindenmayer::arithmetic(), csound::chord(), csound::closestPitch(), csound::equate< EQUIVALENCE_RELATION_P >(), csound::equate< EQUIVALENCE_RELATION_r >(), csound::euclidean(), csound::hyperplane_equation_from_random_inversion_flat(), csound::Chord::K(), csound::midpoint(), csound::next(), csound::note(), csound::operator<(), csound::operator==(), csound::operator>(), csound::predicate< EQUIVALENCE_RE csound::predicate< EQUIVALENCE_RELATION_r >(), csound::reflect_in_central_diagonal(), csound::reflect_in_central_point(), csound::reflect_in_unison_diagonal(), csound::scale(), Scale(), csound::Chord::T_voiceleading(), test_pitv(), csound::toScore(), transpose(), csound::Chord::v(), csound::voiceleading(), csound::Chord::voiceleading(), csound::voiceleadingClosestR and csound::voiceleadingSmoothness().

6.64.4.47 getPitchReference()

```
double & csound::Chord::getPitchReference (
            int voice )  [inline], [virtual], [inherited]
```

6.64.4.48 getTypeName()

```
SILENCE_PUBLIC std::string csound::Scale::getTypeName ( ) const  [inline], [virtual]
```

Returns the type name, e.g.

"major" or "whole tone," of this. This name will probably be invalid if the interval structure of this has been changed, e.g. by inversion.

Referenced by operator=().

6.64.4.49 greater()

```
bool csound::Chord::greater (
            const Chord & other ) const  [inline], [virtual], [inherited]
```

Returns whether the voices of this chord are greater than the voices of the other.

6.64.4.50 greater_equals()

```
bool csound::Chord::greater_equals (
            const Chord & other ) const  [inline], [virtual], [inherited]
```

Returns whether the voices of this chord are greater than or equal to the voices of the other.

6.64.4.51 hyperplane_equation()

```
HyperplaneEquation csound::Chord::hyperplane_equation (
            int opt_sector ) const  [inline], [virtual], [inherited]
```

Returns the hyperplane equation for the inversion flat that evenly divides the fundamental domain in the indicated sector of the OPT cyclical region.

Referenced by csound::reflect_by_householder(), and csound::reflect_in_inversion_flat().

6.64.4.52 hyperplane_equations_for_opt_sectors()

```
std::map< int, std::vector< HyperplaneEquation > > & csound::Chord::hyperplane_equations_for_↩
opt_sectors ( )  [inline], [static], [inherited]
```

For each chord space of dimensions 3 <= n <= 12, there are n fundamental domains (sectors) of OPT equivalence.

For each OPT fundamental domain, there is a inversion flat that evenly divides the OPT fundamental domain into 2 OPTI fundamental domains. This function returns a global collection of the hyperplane equations that define these inversion flats.

6.64.4.53 I()

```
Chord csound::Chord::I (
            double center = 0.0 ) const  [inline], [virtual], [inherited]
```

Inverts the chord by another chord that is on the unison diagonal, by default the origin.

NOTE: Does NOT return an equivalent under any requivalence relation.

References csound::I(), and csound::Chord::setPitch().

Referenced by csound::ChordLindenmayer::chordOperation(), csound::Chord::Iform(), main(), csound::ChordLindenmayer::modalityOpera and test_pitv().

6.64.4.54 Iform()

```
bool csound::Chord::Iform (
            const Chord & Y,
            double g = 1.  )  const  [inline], [virtual], [inherited]
```

Returns whether the chord is an inversional form of Y with interval size g.

Only works in equal temperament.

References csound::Chord::eP(), csound::Chord::epcs(), csound::Chord::I(), and csound::OCTAVE().

6.64.4.55 information()

```
std::string csound::Chord::information ( ) const  [inline], [virtual], [inherited]
```

Print much information about the chord including whether it is within important equivalence classes, or what its equivalents would be.

Referenced by main(), test_pitv(), test_pitv(), transpose(), and csound::transpose_degrees().

6.64.4.56 information_debug()

```
std::string csound::Chord::information_debug (
            int opt_sector ) const  [inline], [virtual], [inherited]
```

Print much information about the chord including whether it is within important equivalence classes, or what its equivalents would be.

The printout first enables then restores debugging diagnostics.

Referenced by main().

6.64.4.57 information_sector()

```
std::string csound::Chord::information_sector (
            int opt_sector ) const  [inline], [virtual], [inherited]
```

Print much information about the chord including whether it is within important equivalence classes, or what its equivalents would be.

References csound::print_chord(), csound::print_opti_sectors(), csound::reflect_in_inversion_flat(), and csound::toString().

6.64.4.58 initialize_sectors()

```
void csound::Chord::initialize_sectors ( )  [inline], [virtual], [inherited]
```

Initializes the fundamental domains (sectors) of the cyclical regions of OPT equivalence and OPTI equivalence, as well as the hyperplane equations that define the inversion flat in each OPT sector.

The cyclical region C of OPT for n voices is the (n-1)-simplicial region of R^n / T with n vertices at A_i = [0^(n - i), 12^n)]_T, for 0 <= i < n. These are the n octavewise revoicings of the origin.

(1) To obtain the fundamental regions of OPT in C, for dimensions 0 <= d < n, replace C[(d+n-1)n] with the center of C c to give OPT_d.

(2) To obtain the fundamental regions for OPTI in C for dimensions 0 <= d < n, replace OPT_d[(d+n-2)n] with the midpoint of OPT_d[(d+n)n] => OPT_d[(d+n-2)n] to give OPTI_d_0, and replace OPT_d[(d+n)n] with the midpoint of OPT_d[(d+n)n] => OPT_d[(d+n-2)n] to give OPTI_d_1.

(3) A vector that is normal to the inversion flat in OPT_d is then OPT_d[(d+n)n] => OPT_d[d+n-2)n]. Normalizing this vector gives the unit normal vector u for the inversion flat. Then the hyperplane equation for the inversion flat is u and its constant term is u dot c.

NOTE:

In this code, sector vertices are NOT permuted.

The reason for starting with C[n-1] is to include the origin in the 0th fundamental domain, because we regard OPT sector 0 as the *representative* fundamental domain of OPT.

This code is based on the construction of Noam Elkies described in the *Generalized Chord Spaces* draft by Callender, Quinn, and Tymoczko.

References csound::Chord::center(), CHORD_SPACE_DEBUG, csound::HyperplaneEquation::constant_term, csound::Chord::eT(), csound::midpoint(), csound::Chord::setPitch(), csound::Chord::T(), csound::Chord::toString(), csound::toString(), and csound::HyperplaneEquation::unit_normal_vector.

6.64.4.59 inverse_prime_form()

```
Chord csound::Chord::inverse_prime_form ( ) const  [inline], [virtual], [inherited]
```

Returns this chord as the inverse standard "prime form.".

NOTE: The code here does NOT remove duplicate pitch-classes.

References csound::I(), and csound::inverse_prime_forms_for_chords().

Referenced by csound::PITV::fromChord(), csound::PITV::initialize(), and csound::PITV::list().

6.64.4.60 is_compact()

```
bool csound::Chord::is_compact (
            double range = 12.  )  const  [inline], [virtual], [inherited]
```

Returns whether this chord has a compact voicing.

This identifies whether the chord belongs to the representative fundamental domain of the OPT equivalence class. In Tymoczko's 1-based notation: x[1] + 12 - x[N] <= x[i + 1] - x[i], 1 <= i < N - 1 In 0-based notation: x[0] + 12 - x[N-1] <= x[i + 1] - x[i], 0 <= i < N - 2

References csound::le_tolerance().

6.64.4.61 is_minor()

```
bool csound::Chord::is_minor ( ) const  [inline], [virtual], [inherited]
```

Returns whether this chord is "minor" in the sense of having the smallest "wraparound interval" of all its voicings.

References csound::gt_tolerance(), and csound::lt_tolerance().

6.64.4.62 is_opt_sector()

```
bool csound::Chord::is_opt_sector (
            int opt_sector = 0 ) const  [inline], [virtual], [inherited]
```

Returns whether or not this chord lies within the indicated sector of the cyclical region of OPT fundamental domains.

Referenced by csound::predicate< EQUIVALENCE_RELATION_RPT >(), csound::predicate< EQUIVALENCE_RELATION_RPTg >(), csound::predicate< EQUIVALENCE_RELATION_RPTgI >(), and csound::predicate< EQUIVALENCE_RELATION_RPTI >().

6.64.4.63 is_opti_sector()

```
bool csound::Chord::is_opti_sector (
            int opti_sector = 0 ) const  [inline], [virtual], [inherited]
```

Returns whether or not this chord lies within the indicated sector of the cyclical region of OPTI fundamental domains.

Referenced by csound::predicate< EQUIVALENCE_RELATION_I >().

6.64.4.64 isel()

```
bool csound::Chord::iseI (
            int opt_sector = 0 ) const  [inline], [virtual], [inherited]
```

Referenced by main().

6.64.4.65 isel_chord()

```
bool csound::Chord::iseI_chord (
            Chord * inverse,
            int opt_sector = 0 ) const  [inline], [virtual], [inherited]
```

Returns whether the chord is within a fundamental domain of inversional equivalence.

References csound::OCTAVE().

6.64.4.66 iseO()

```
bool csound::Chord::iseO ( ) const  [inline], [virtual], [inherited]
```

Returns whether the chord is within the representative fundamental domain of octave equivalence.

References csound::OCTAVE().

Referenced by main().

6.64.4.67 iseOP()

```
bool csound::Chord::iseOP ( ) const  [inline], [virtual], [inherited]
```

Returns whether the chord is within the representative fundamental domain of octave and permutational equivalence.

References csound::OCTAVE().

6.64.4.68 iseOPI()

```
bool csound::Chord::iseOPI (
            int opt_sector = 0 ) const  [inline], [virtual], [inherited]
```

Returns whether the chord is within a fundamental domain of octave, permutational, and inversional equivalence.

References csound::OCTAVE().

6.64.4.69 iseOPT()

```
bool csound::Chord::iseOPT (
            int opt_sector = 0 ) const  [inline], [virtual], [inherited]
```

Returns whether the chord is within a fundamental domainof octave, permutational, and transpositional equivalence.

References csound::OCTAVE().

6.64.4.70 iseOPTI()

```
bool csound::Chord::iseOPTI (
            int opt_sector = 0 ) const  [inline], [virtual], [inherited]
```

Returns whether the chord is within a fundamental domain of octave, permutational, transpositional, and inversional equivalence.

References csound::OCTAVE().

6.64.4.71 iseOPTT()

```
bool csound::Chord::iseOPTT (
            double g = 1.,
            int opt_sector = 0 ) const  [inline], [virtual], [inherited]
```

Returns whether the chord is within a fundamental domain of octave, permutational, and transpositional equivalence in the equal temperament generated by g.

References csound::OCTAVE().

6.64.4.72 iseOPTTI()

```
bool csound::Chord::iseOPTTI (
            double g = 1.,
            int opt_sector = 0 ) const  [inline], [virtual], [inherited]
```

Returns whether the chord is within a fundamental domain of octave, permutational, transpositional, and inversional equivalence in the equal temperament generated by g.

References csound::OCTAVE().

6.64.4.73 iseOT()

```
virtual bool csound::Chord::iseOT ( ) const  [inline], [virtual], [inherited]
```

Returns whether the chord is within the representative fundamental domain of octave and transpositional equivalence.

6.64.4.74 iseOTT()

```
virtual bool csound::Chord::iseOTT (
            double g = 1.  )  const  [inline], [virtual], [inherited]
```

Returns whether the chord is within the representative fundamental domain of octave and translational equivalence in the equal temperament generated by g.

6.64.4.75 iseP()

```
bool csound::Chord::iseP ( ) const  [inline], [virtual], [inherited]
```

Returns whether the chord is within the representative fundamental domain of permutational equivalence.

References csound::OCTAVE().

Referenced by main().

6.64.4.76 isepcs()

```
bool csound::Chord::isepcs ( ) const  [inline], [virtual], [inherited]
```

Returns whether the chord is within the fundamental domain of pitch-class equivalence, i.e.

is a pitch-class set.

References csound::epc(), and csound::eq_tolerance().

Referenced by main().

6.64.4.77 iseR()

```
bool csound::Chord::iseR (
            double range_ ) const  [inline], [virtual], [inherited]
```

Returns whether the chord is within the representative fundamental domain of the indicated range equivalence.

Referenced by csound::equate< EQUIVALENCE_RELATION_R >().

6.64.4.78 iseRP()

```
bool csound::Chord::iseRP (
            double range ) const  [inline], [virtual], [inherited]
```

Returns whether the chord is within the representative fundamental domain of range and permutational equivalence.

6.64.4.79 iseRPI()

```
bool csound::Chord::iseRPI (
            double range,
            int opt_sector = 0 ) const  [inline], [virtual], [inherited]
```

Returns whether the chord is within a fundamental domain of range, permutational, and inversional equivalence.

6.64.4.80 iseRPT()

```
bool csound::Chord::iseRPT (
            double range,
            int opt_sector = 0 ) const  [inline], [virtual], [inherited]
```

Returns whether the chord is within a fundamental domain of range, permutational, and transpositional equivalence.

6.64.4.81 iseRPTI()

```
bool csound::Chord::iseRPTI (
            double range,
            int opt_sector = 0 ) const  [inline], [virtual], [inherited]
```

Returns whether the chord is within a fundamental domain of range, permutational, transpositional, and inversional equivalence.

6.64.4.82 iseRPTT()

```
bool csound::Chord::iseRPTT (
            double range,
            double g = 1.,
            int opt_sector = 0 ) const  [inline], [virtual], [inherited]
```

Returns whether the chord is within a fundamental domain of range, permutational, and transpositional equivalence in the equal temperament generated by g.

6.64.4.83 iseRPTTI()

```
bool csound::Chord::iseRPTTI (
            double range,
            double g = 1.,
            int opt_sector = 0 ) const  [inline], [virtual], [inherited]
```

Returns whether the chord is within a fundamental domain of range, permutational, transpositional, and inversional equivalence in the ' equal temperament generated by g.

6.64.4.84 iseRT()

```
virtual bool csound::Chord::iseRT (
            double range ) const  [inline], [virtual], [inherited]
```

Returns whether the chord is within the representative fundamental domain of range and transpositional equivalence.

6.64.4.85 iseRTT()

```
virtual bool csound::Chord::iseRTT (
            double range,
            double g = 1.  )  const  [inline], [virtual], [inherited]
```

Returns whether the chord is within a fundamental domain of range and transpositional equivalence in the equal temperament generated by g.

6.64.4.86 iseT()

```
bool csound::Chord::iseT ( ) const  [inline], [virtual], [inherited]
```

Returns whether the chord is within the representative fundamental domain of transpositional equivalence.

References csound::OCTAVE().

Referenced by main().

6.64.4.87 iset()

```
bool csound::Chord::iset ( ) const  [inline], [virtual], [inherited]
```

Returns whether the chord is within the fundamental domain of transposition to 0.

Referenced by main().

6.64.4.88 iseTT()

```
bool csound::Chord::iseTT (
            double g = 1.  )  const  [inline], [virtual], [inherited]
```

Returns whether the chord is within the representative fundamental domain of transpositional equivalence in the equal temperament generated by g.

References csound::OCTAVE().

Referenced by main().

6.64.4.89 K()

```
Chord csound::Chord::K ( ) const  [inline], [virtual], [inherited]
```

Returns the chord inverted by the sum of its first two voices.

References csound::chord(), csound::epc(), csound::Chord::getPitch(), csound::Chord::setPitch(), and csound::Chord::voices().

Referenced by csound::ChordLindenmayer::chordOperation(), is_k_dual(), and csound::ChordLindenmayer::modalityOperation().

6.64.4.90 K_range()

```
Chord csound::Chord::K_range (
            double range ) const  [inline], [virtual], [inherited]
```

References csound::chord(), and csound::Chord::eRP().

6.64.4.91 layer()

```
double csound::Chord::layer ( ) const  [inline], [virtual], [inherited]
```

Returns the sum of the pitches in the chord.

Referenced by csound::equate< EQUIVALENCE_RELATION_R >(), csound::equate< EQUIVALENCE_RELATION_T >(), csound::predicate< EQUIVALENCE_RELATION_R >(), csound::predicate< EQUIVALENCE_RELATION_T >(), csound::predicate< EQUIVALENCE_RELATION_Tg >(), csound::reflect_in_central_diagonal(), and csound::reflect_in_unison_diagonal(

6.64.4.92 lesser()

```
bool csound::Chord::lesser (
            const Chord & other ) const  [inline], [virtual], [inherited]
```

Returns whether the voices of this chord are less than the voices of the other.

6.64.4.93 lesser_equals()

```
bool csound::Chord::lesser_equals (
            const Chord & other ) const  [inline], [virtual], [inherited]
```

Returns whether the voices of this chord are less than or equal to the voices of the other.

6.64.4.94 max()

```
std::vector< double > csound::Chord::max ( ) const  [inline], [virtual], [inherited]
```

Returns the highest pitch in the chord, and also the voice index of that pitch.

References csound::gt_tolerance().

Referenced by csound::equate< EQUIVALENCE_RELATION_R >(), and csound::predicate< EQUIVALENCE_RELATION_R >().

6.64.4.95 maximumInterval()

```
double csound::Chord::maximumInterval ( ) const  [inline], [virtual], [inherited]
```

Returns the maximum interval within the chord.

References csound::gt_tolerance().

Referenced by main().

6.64.4.96 min()

```
std::vector< double > csound::Chord::min ( ) const  [inline], [virtual], [inherited]
```

Returns the lowest pitch in the chord, and also the voice index of that pitch.

References csound::lt_tolerance().

Referenced by main(), csound::next(), and csound::predicate< EQUIVALENCE_RELATION_R >().

6.64.4.97 minimumInterval()

```
double csound::Chord::minimumInterval ( ) const  [inline], [virtual], [inherited]
```

Returns the minimum interval within the chord.

References csound::lt_tolerance().

Referenced by main().

6.64.4.98 modulations()

```
SILENCE_PUBLIC std::vector< Scale > csound::Scale::modulations (
            const Chord & chord ) const  [inline], [virtual]
```

Returns a list of common modulations, that is, other major or harmonic minor Scales to which the Chord belongs; optionally the Chord can first be resized (e.g.

from a 9th chord to a triad) in order to find more or fewer possible modulations.

References csound::chord(), and csound::Chord::voices().

6.64.4.99 modulations_for_scale_types()

```
SILENCE_PUBLIC void csound::Scale::modulations_for_scale_types (
            std::vector< Scale > & result,
            const Chord & current_chord,
            int voices_,
            const std::vector< std::string > & type_names ) const  [inline], [virtual]
```

For any Chord belonging to this Scale, returns in the argument a list of other Scales to which that Chord also belongs.

Switching to one of these Scales will perform some sort of modulation. The list of scale type names restricts the types of Scale that will be returned.

References csound::chord(), csound::scale(), csound::unique_scales(), and csound::Chord::voices().

6.64.4.100 modulations_for_voices()

```
SILENCE_PUBLIC std::vector< Scale > csound::Scale::modulations_for_voices (
            const Chord & chord,
            int voices ) const  [inline], [virtual]
```

References csound::chord().

Referenced by csound::ChordLindenmayer::scaleOperation().

6.64.4.101 move()

```
Chord csound::Chord::move (
            int voice,
            double interval ) const  [inline], [virtual], [inherited]
```

Move 1 voice of the chord.

NOTE: Does NOT return an equivalent under any requivalence relation.

References csound::chord(), csound::Chord::setPitch(), and csound::T().

6.64.4.102 name()

```
SILENCE_PUBLIC std::string csound::Scale::name ( ) const  [inline], [virtual]
```

Returns the name of this Scale.

Reimplemented from csound::Chord.

References csound::nameForPitchClass().

Referenced by transpose().

6.64.4.103 normal_form()

```
Chord csound::Chord::normal_form ( ) const  [inline], [virtual], [inherited]
```

Returns this chord as its standard "normal form.".

NOTE: The code here does NOT remove duplicate pitch-classes.

References csound::normal_forms_for_chords().

Referenced by csound::PITV::fromChord(), csound::PITV::initialize(), csound::compare_by_normal_form::operator()(), and setDifference().

6.64.4.104 normal_order()

```
Chord csound::Chord::normal_order ( ) const  [inline], [virtual], [inherited]
```

Returns this chord in standard "normal order." For a very clear explanation, see: https://www.↩mta.ca/pc-set/pc-set_new/pages/page04/page04.html and http://openmusictheory.↩com/normalOrder.html/.

NOTE: The code here does NOT remove duplicate pitch-classes. "Normal order" is the most compact ordering to the left of pitch-classes in a chord, measured by pitch-class interval.

References csound::lt_tolerance(), and csound::OCTAVE().

Referenced by csound::compare_by_normal_order::operator()().

6.64.4.105 nrD()

```
Chord csound::Chord::nrD ( ) const  [inline], [virtual], [inherited]
```

Performs the dominant transformation (which is not a neo-Reimannian transformation).

The result is returned in OP.

References csound::T().

6.64.4.106 nrH()

```
Chord csound::Chord::nrH ( ) const  [inline], [virtual], [inherited]
```

Performs the neo-Riemannian hexatonic pole transformation.

The result is returned in OP.

References csound::Chord::nrL(), and csound::Chord::nrP().

6.64.4.107 nrL()

```
Chord csound::Chord::nrL ( ) const  [inline], [virtual], [inherited]
```

Performs the neo-Riemannian Lettonwechsel transformation.

The result is returned in OP.

Referenced by csound::Chord::nrH(), and csound::Chord::nrN().

6.64.4.108 nrN()

```
Chord csound::Chord::nrN ( ) const  [inline], [virtual], [inherited]
```

Performs the neo-Riemannian Nebenverwandt transformation.

The result is returned in NP.

References csound::Chord::nrL(), and csound::Chord::nrP().

6.64.4.109 nrP()

```
Chord csound::Chord::nrP ( ) const  [inline], [virtual], [inherited]
```

Performs the neo-Riemannian parallel transformation.

The result is returned in OP.

Referenced by csound::Chord::nrH(), csound::Chord::nrN(), and csound::Chord::nrS().

6.64.4.110 nrR()

```
Chord csound::Chord::nrR ( ) const  [inline], [virtual], [inherited]
```

Performs the neo-Riemannian parallel transformation.

Referenced by csound::Chord::nrS().

6.64.4.111 nrS()

```
Chord csound::Chord::nrS ( ) const  [inline], [virtual], [inherited]
```

Performs the neo-Riemannian Slide transformation.

The result is returned in OP.

References csound::Chord::nrP(), and csound::Chord::nrR().

6.64.4.112 operator std::vector< double >()

```
csound::Chord::operator std::vector< double > ( ) const  [inline], [virtual], [inherited]
```

6.64.4.113 operator=()

```
SILENCE_PUBLIC Scale & csound::Scale::operator= (
            const Scale & other )  [inline], [virtual]
```

References getTypeName().

6.64.4.114 opt_domain()

```
std::vector< Chord > csound::Chord::opt_domain (
            int sector ) const  [inline], [virtual], [inherited]
```

Returns the vertices of the OPT fundamental domain for the indicated sector of the cyclical region.

6.64.4.115 opt_domain_sectors()

```
std::vector< int > csound::Chord::opt_domain_sectors ( ) const  [inline], [virtual], [inherited]
```

Returns the zero-based index(s) of the sector(s) within the cyclical region of OPT fundamental domains to which the chord belongs.

A chord on a vertex, edge, or facet shared by more than one sector belongs to each of them; the center of the cyclical region belongs to all of the sectors. Sectors are generated by rotation of a fundamental domain around the central axis (equivalently, by the octavewise revoicing of chords) and correspond to "chord inversion" in the musician's sense.

Referenced by csound::PITV::list(), and csound::reflect_by_householder().

6.64.4.116 opt_sectors_for_dimensionalities()

```
std::map< int, std::vector< std::vector< Chord > > > & csound::Chord::opt_sectors_for_dimensionalities
( )  [inline], [static], [inherited]
```

For each chord space of dimensions 3 <= n <= 12, there are n fundamental domains (sectors) of OPT equivalence.

This function returns a global collection of these sectors.

6.64.4.117 opt_simplexes_for_dimensionalities()

```
std::map< int, std::vector< std::vector< Chord > > > & csound::Chord::opt_simplexes_for_dimensionalities
( )  [inline], [static], [inherited]
```

Returns a collection of vertices for the OPT fundamental domains; each has an added vertex to make a simplex for chord location.

6.64.4.118 opti_domain()

```
std::vector< Chord > csound::Chord::opti_domain (
            int sector ) const  [inline], [virtual], [inherited]
```

Returns the vertices of the OPTI fundamental domain for the indicated sector of the cyclical region.

6.64.4.119 opti_domain_sectors()

```
std::vector< int > csound::Chord::opti_domain_sectors ( ) const  [inline], [virtual], [inherited]
```

Returns the zero-based index(s) of the sector(s) within the cyclical region of OPTI fundamental domains to which the chord belongs.

A chord on a vertex, edge, or facet shared by more than one sector belongs to each them; the center of the cyclical region belongs to all of the sectors. Sectors are generated by rotation of a fundamental domain (equivalently, by the octavewise revoicing of chords) and correspond to "chord inversion" in the musician's ordinary sense. SCOPED_DEBUGGING debug;

References CHORD_SPACE_DEBUG, csound::distance_to_points(), csound::lt_tolerance(), and csound::toString().

Referenced by csound::print_chord(), and csound::print_opti_sectors().

6.64.4.120 opti_sectors_for_dimensionalities()

```
std::map< int, std::vector< std::vector< Chord > > > & csound::Chord::opti_sectors_for_dimensionalities
( )  [inline], [static], [inherited]
```

For each chord space of dimensions 3 <= n <= 12, there are n fundamental domains (sectors) of OPTI equivalence.

This function returns a global collection of these sectors.

6.64.4.121 opti_simplexes_for_dimensionalities()

```
std::map< int, std::vector< std::vector< Chord > > > & csound::Chord::opti_simplexes_for_↩
dimensionalities ( )  [inline], [static], [inherited]
```

Returns a collection of vertices for the OPTI fundamental domains that have an added vertex to make a simplex for chord location.

6.64.4.122 origin()

```
Chord csound::Chord::origin ( ) const  [inline], [virtual], [inherited]
```

Returns the origin of the chord's space.

References csound::Chord::resize().

Referenced by csound::reflect_in_unison_diagonal().

6.64.4.123 permutations()

```
std::vector< Chord > csound::Chord::permutations ( ) const  [inline], [virtual], [inherited]
```

Returns the permutations of the pitches in a chord.

The permutations are always returned in the same order.

References csound::Chord::cycle().

Referenced by main().

6.64.4.124 prime_form()

```
Chord csound::Chord::prime_form ( ) const  [inline], [virtual], [inherited]
```

Returns this chord as its standard "prime form.".

NOTE: The code here does NOT remove duplicate pitch-classes.

References csound::I(), and csound::prime_forms_for_chords().

Referenced by csound::PITV::fromChord(), csound::PITV::initialize(), and csound::PITV::list().

6.64.4.125 Q()

```
Chord csound::Chord::Q (
            double x,
            const Chord & m,
            double g = 1.  )  const  [inline], [virtual], [inherited]
```

Returns the contextual transposition of the chord by x with respect to m with minimum interval size g.

NOTE: Does NOT return an equivalent under any requivalence relation.

References csound::T().

Referenced by csound::ChordLindenmayer::chordOperation().

6.64.4.126 reflect()

```
Chord csound::Chord::reflect (
            int opt_sector ) const  [inline], [virtual], [inherited]
```

Reflects the chord in the inversion flat of the indicated OPT domain sector.

References csound::reflect_in_inversion_flat().

Referenced by main().

6.64.4.127 relative_tonicizations()

```
SILENCE_PUBLIC std::vector< Scale > csound::Scale::relative_tonicizations (
            const Chord & current_chord,
            int secondary_function = 5,
            int voices = -1 ) const  [inline], [virtual]
```

Returns a list of common relative tonicizations for the Chord, that is, the other major or harmonic minor Scales for which that Chord could be mutated to have the secondary function.

If that is not possible, an empty result is returned.

6.64.4.128 relative_tonicizations_for_scale_types()

```
SILENCE_PUBLIC void csound::Scale::relative_tonicizations_for_scale_types (
            std::vector< Scale > & result,
            const Chord & current_chord,
            int secondary_function,
            int voices,
            const std::vector< std::string > & type_names ) const  [inline], [virtual]
```

Returns the *relative* tonicizations of the Chord, that is, the scales for which that Chord could be mutated to have the secondary function, if that is possible.

The list of scale types is used to restrict the types of Scales that are returned.

References csound::chord(), CHORD_SPACE_DEBUG, csound::Chord::name(), csound::Chord::toString(), and csound::Chord::voices().

6.64.4.129 resize()

```
void csound::Chord::resize (
            size_t voiceN )  [inline], [virtual], [inherited]
```

Referenced by csound::addVoice(), csound::chord(), csound::chordForName(), csound::fill(), csound::gather(), csound::iterator(), main(), csound::Chord::origin(), csound::PITV::preinitialize(), csound::removeVoice(), csound::scaleForName(), transpose(), and csound::transpose_degrees().

6.64.4.130 rownd()

```
double csound::Chord::rownd (
            double x,
            int places = 12 )  [inline], [static], [inherited]
```

Rounds the value of x to the specified number of decimal places.

6.64.4.131 secondary()

```
SILENCE_PUBLIC std::vector< Chord > csound::Scale::secondary (
            const Chord & current_chord,
            int secondary_function = 5,
            int voices_ = -1 ) const  [inline], [virtual]
```

Returns the current Chord mutated, if possible, to one or more function(s) with respect to another Chord in its Scale.

Not "secondary function of this chord," but "this chord as secondary function of another (tonicized) chord." If that is not possible, an empty Chord is returned. The number of voices defaults to that of the current Chord. Can be used to generate secondary dominants (function = 5), secondary supertonics (function = 2), secondary subtonics (function = 6), and so on. It is then up to the user to perform an appropriate progression by number of scale degrees in the original Scale.

References csound::Chord::voices().

6.64.4.132 self_inverse()

```
bool csound::Chord::self_inverse (
            int opt_sector = 0 ) const  [inline], [virtual], [inherited]
```

Returns whether or not this chord is invariant under reflection in the inversion flat of the indicated OPT sector.

Such are the shared vertices, edges, and facets of those fundamental domains that involve inversional equivalence.

References csound::reflect_in_inversion_flat().

Referenced by csound::predicate< EQUIVALENCE_RELATION_I >().

6.64.4.133 semitones_for_degree()

```
SILENCE_PUBLIC double csound::Scale::semitones_for_degree (
            int scale_degree ) const  [inline], [virtual]
```

Returns the number of semitones (may be whole or fractional) from the tonic (as 0) of this Scale to the indicated scale degree, which is wrapped around by octave equivalence.

6.64.4.134 setDuration()

```
void csound::Chord::setDuration (
            double value,
            int voice = -1 )  [inline], [virtual], [inherited]
```

6.64.4.135 setInstrument()

```
void csound::Chord::setInstrument (
            double value,
            int voice = -1 )  [inline], [virtual], [inherited]
```

6.64.4.136 setLoudness()

```
void csound::Chord::setLoudness (
            double value,
            int voice = -1 )  [inline], [virtual], [inherited]
```

6.64.4.137 setPan()

```
void csound::Chord::setPan (
            double value,
            int voice = -1 )  [inline], [virtual], [inherited]
```

6.64.4.138 setPitch()

```
void csound::Chord::setPitch (
            int voice,
            double value )  [inline], [virtual], [inherited]
```

Referenced by csound::addVoice(), csound::ChordLindenmayer::arithmetic(), csound::Chord::ceiling(), csound::Chord::center(), csound::chord(), csound::Chord::distanceToUnisonDiagonal(), csound::Chord::epcs(), csound::Chord::eppcs(), csound::equate< EQUIVALENCE_RELATION_r >(), csound::equate< EQUIVALENCE_RELATION_R >(), csound::fill(), csound::Chord::floor(), csound::gather(), csound::hyperplane_equation_from_random_inversion_flat(), csound::Chord::I(), csound::Chord::initialize_sectors(), csound::iterator(), csound::Chord::K(), main(), csound::midpoint(), csound::Chord::move(), csound::next(), csound::reflect_by_householder(), csound::reflect_in_inversion_flat(), csound::Chord::T(), csound::Chord::T_voiceleading(test_pitv(), transpose(), csound::Chord::v(), csound::voiceleading(), csound::Chord::voiceleading(), and csound::voiceleadingClosestRang

6.64.4.139 T()

```
Chord csound::Chord::T (
            double interval ) const  [inline], [virtual], [inherited]
```

Transposes the chord by the indicated interval (may be a fraction).

NOTE: Does NOT return an equivalent under any requivalence relation.

References csound::Chord::setPitch(), and csound::T().

Referenced by csound::ChordLindenmayer::chordOperation(), csound::equate< EQUIVALENCE_RELATION_T >(), csound::equate< EQUIVALENCE_RELATION_Tg >(), csound::fill(), csound::Chord::initialize_sectors(), main(), csound::ChordLindenmayer::modalityOperation(), csound::predicate< EQUIVALENCE_RELATION_Tg >(), csound::reflect_in_central_di csound::reflect_in_unison_diagonal(), csound::Chord::Tform(), and transpose().

6.64.4.140 T_voiceleading()

```
Chord csound::Chord::T_voiceleading (
            const Chord & voiceleading )  [inline], [virtual], [inherited]
```

Transposes the chord by the indicated voiceleading (passed as a Chord of directed intervals).

NOTE: Does NOT return an equivalent under any equivalence relation.

References csound::Chord::getPitch(), csound::Chord::setPitch(), and csound::voiceleading().

6.64.4.141 test()

```
bool csound::Chord::test (
            const char * caption = "" ) const  [inline], [virtual], [inherited]
```

Tests the internal consistency of the predicates ("iseX") and transformations ("eX") of this chord, and prints a report.

References csound::toString().

6.64.4.142 Tform()

```
bool csound::Chord::Tform (
            const Chord & Y,
            double g = 1.  )  const  [inline], [virtual], [inherited]
```

Returns whether the chord is a transpositional form of Y with interval size g.

Only works in equal temperament.

References csound::Chord::eP(), csound::Chord::epcs(), csound::OCTAVE(), and csound::Chord::T().

6.64.4.143 tonic()

```
SILENCE_PUBLIC double csound::Scale::tonic ( ) const  [inline], [virtual]
```

Returns the pitch-class that is the tonic or root of this Scale.

6.64.4.144 tonicizations()

```
SILENCE_PUBLIC std::vector< Scale > csound::Scale::tonicizations (
            const Chord & current_chord,
            int voices = -1 ) const  [inline], [virtual]
```

Returns all major or minor Scales for which the current Chord is the tonic (scale degree 1).

The number of voices defaults to that of the current Chord, but may be larger or smaller.

NOTE: Here, tonicizations are modulations in which the Chord has degree 1, i.e. is the tonic chord.

References csound::chord(), CHORD_SPACE_DEBUG, csound::Chord::name(), csound::Chord::toString(), and csound::Chord::voices().

6.64.4.145 toString()

```
std::string csound::Chord::toString ( ) const  [inline], [virtual], [inherited]
```

Returns a string representation of the chord's pitches (only).

Quadratic complexity, but short enough not to matter.

Referenced by csound::HarmonyIFS::add_interpolation_point_as_chord(), csound::HarmonyIFS2::add_interpolation_point_as_chord(), csound::ChordLindenmayer::chordOperation(), csound::equate< EQUIVALENCE_RELATION_R >(), csound::fill(), csound::fundamentalDomainByPredicate(), csound::fundamentalDomainByTransformation(), csound::indexForOctavewiseRevoicing(), csound::Chord::initialize_sectors(), is_k_dual(), csound::HarmonyIFS2::iterate(), csound::PITV::list(), main(), csound::octavewiseRevoicing(), csound::octavewiseRevoicings(), csound::predicate< EQUIVALENCE_RELATION_I >(), csound::predicate< EQUIVALENCE_RELATION_T >(), csound::print_chord(), csound::reflect_by_householder(), relative_tonicizations_for_scale_types(), csound::scale(), csound::ChordLindenmayer::scaleDegreeOperation(), csound::ChordLindenmayer::scaleOperation(), csound::ChordLindenmayer::scoreOperation(), setDifference(), tonicizations(), transpose(), and csound::transpose_degrees().

6.64.4.146 transpose()

```
SILENCE_PUBLIC Scale csound::Scale::transpose (
            double semitones ) const  [inline], [virtual]
```

Returns a copy of this Scale transposed by the indicated number of *semitones*.

References CHORD_SPACE_DEBUG, csound::ge_tolerance(), csound::Chord::getPitch(), csound::Chord::information(), csound::lt_tolerance(), name(), csound::nameForPitchClass(), csound::OCTAVE(), csound::Chord::resize(), csound::Chord::setPitch(), csound::Chord::T(), csound::T(), csound::Chord::toString(), and type_name.

6.64.4.147 transpose_degrees()

```
SILENCE_PUBLIC Chord csound::Scale::transpose_degrees (
            const Chord & chord,
            int scale_degrees,
            int interval = 3 ) const  [inline], [virtual]
```

Returns a Chord transposed by the indicated number of *scale degrees*; the chord as passed must belong to this Scale, and the interval must be the same as that used to generate the Chord; (defaults to thirds, or 3; the actual number of scale steps between chord pitches is interval - 1).

References csound::chord(), and csound::transpose_degrees().

6.64.4.148 transpose_to_degree()

```
SILENCE_PUBLIC Scale csound::Scale::transpose_to_degree (
            int degrees ) const  [inline], [virtual]
```

Returns a copy of this Scale transposed to the indicated *scale degree*.

References CHORD_SPACE_DEBUG.

6.64.4.149 v()

```
Chord csound::Chord::v (
            int direction = 1 ) const  [inline], [virtual], [inherited]
```

Returns a copy of the chord 'inverted' in the musician's sense, i.e.

revoiced by cyclically permuting the chord and adding (or subtracting) an octave to the highest (or lowest) voice. The revoicing will move the chord up or down in pitch. A positive direction is the same as a musician's first inversion, second inversion, etc.

References csound::chord(), csound::Chord::cycle(), csound::Chord::getPitch(), csound::OCTAVE(), and csound::Chord::setPitch().

Referenced by csound::scale(), and csound::Chord::voicings().

6.64.4.150 voiceleading()

```
Chord csound::Chord::voiceleading (
            const Chord & destination ) const  [inline], [virtual], [inherited]
```

Returns the transpositions (as a Chord of directed intervals) that takes this chord to the destination chord.

NOTE: Makes no assumption that both chords are in the same equivalence class.

References csound::Chord::getPitch(), and csound::Chord::setPitch().

6.64.4.151 voices()

```
size_t csound::Chord::voices ( ) const  [inline], [virtual], [inherited]
```

Returns the number of voices in this chord; that is, the number of dimensions in the chord space for this chord.

Referenced by csound::Chord::a(), csound::addVoice(), csound::ChordLindenmayer::arithmetic(), csound::chord(), csound::closestPitch(), degree(), csound::equate< EQUIVALENCE_RELATION_P >(), csound::equate< EQUIVALENCE_RELATION_r > csound::equate< EQUIVALENCE_RELATION_T >(), csound::equivalentDegree(), csound::euclidean(), csound::Chord::K(), csound::midpoint(), modulations(), modulations_for_scale_types(), csound::next(), csound::notes(), csound::operator<(), csound::operator==(), csound::operator>(), csound::predicate< EQUIVALENCE_RELATION_P >(), csound::predicate< EQUIVALENCE csound::reflect_by_householder(), csound::reflect_in_central_diagonal(), csound::reflect_in_central_point(), csound::reflect_in_inversion_ csound::reflect_in_unison_diagonal(), relative_tonicizations_for_scale_types(), csound::removeVoice(), Scale(), Scale(), secondary(), tonicizations(), csound::toScore(), csound::transpose_degrees(), csound::voiceleading(), csound::voiceleadingClosestRange(), and csound::voiceleadingSmoothness().

6.64.4.152 voicings()

```
std::vector< Chord > csound::Chord::voicings ( ) const  [inline], [virtual], [inherited]
```

Returns all the 'inversions' (in the musician's sense) or octavewise revoicings of the chord.

The first voice is transposed up by one octave, and all voices are then rotated "left" so the transposed voice becomes the last voice.

References csound::chord(), and csound::Chord::v().

6.64.5 Field Documentation

6.64.5.1 type_name

```
std::string csound::Scale::type_name  [protected]
```

Referenced by transpose().

6.65 csound::SCOPED_DEBUGGING Struct Reference

```
#include <ChordSpaceBase.hpp>
```

Public Member Functions

- SCOPED_DEBUGGING ()
- ~SCOPED_DEBUGGING ()

Data Fields

- int prior_state = false

6.65.1 Constructor & Destructor Documentation

6.65.1.1 SCOPED_DEBUGGING()

```
csound::SCOPED_DEBUGGING::SCOPED_DEBUGGING ( )  [inline]
```

References csound::CHORD_SPACE_DEBUGGING(), and csound::SCOPED_DEBUGGING_FLAG().

6.65.1.2 ~SCOPED_DEBUGGING()

```
csound::SCOPED_DEBUGGING::~SCOPED_DEBUGGING ( )  [inline]
```

References csound::CHORD_SPACE_DEBUGGING(), and csound::SCOPED_DEBUGGING_FLAG().

6.65.2 Field Documentation

6.65.2.1 prior_state

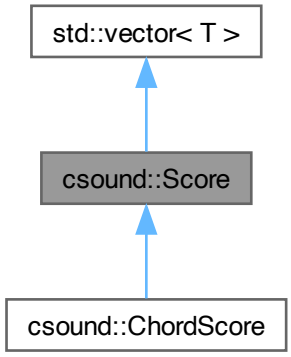
```
int csound::SCOPED_DEBUGGING::prior_state = false
```

6.66 csound::Score Class Reference

Base class for collections of events in music space.

`#include <Score.hpp>`

Inheritance diagram for csound::Score:



Public Member Functions

- virtual void add (double time, double duration, double status, double instrument, double key, double velocity, double phase=0, double pan=0, double depth=0, double height=0, double pitches=4095)
- virtual void append (double time, double duration, double status, double instrument, double key, double velocity, double phase=0, double pan=0, double depth=0, double height=0, double pitches=4095)
- virtual void append (Event event)
- virtual void append_event (Event event)
- virtual void append_note (double time, double duration, double status, double instrument, double key, double velocity, double phase=0, double pan=0, double depth=0, double height=0, double pitches=4095)
- virtual void appendToCsoundScoreHeader (const std::string &text)
- virtual void arrange (int oldInstrumentNumber, int newInstrumentNumber)

  Re-assign instrument number for export to Csound score.
- virtual void arrange (int oldInstrumentNumber, int newInstrumentNumber, double gain)

  Re-assign instrument number and adjust gain for export to Csound score.
- virtual void arrange (int oldInstrumentNumber, int newInstrumentNumber, double gain, double pan)

  Re-assign instrument number, adjust gain, and change pan for export to Csound score.
- virtual void arrange_all (int oldInstrumentNumber, int newInstrumentNumber, double gain, double pan)
- virtual void dump (std::ostream &stream)
- virtual void findScale ()
- virtual std::string getBlueScore (double tonesPerOctave=12.0, bool conformPitches=false)

  Translate the Silence events in this to a Csound score for blue, that is, to a list of i statements with with inso, time, duration, dbsp, pch, pan.

- virtual std::string getCsoundScore (double tonesPerOctave=12.0, bool conformPitches=false)

  Translate the Silence events in this to a Csound score, that is, to a list of i statements.
- virtual std::string getCsoundScoreHeader () const
- virtual double getDuration ()

  Returns the time from the first event to the last event.
- virtual double getDurationFromZero () const

  Returns the time from 0 to the final off time; this assumes that no events start before time 0.
- virtual std::vector< double > getPitches (size_t begin, size_t end, size_t divisionsPerOctave=12) const

  Return a vector containing the MIDI key numbers in the specified segment of the score.
- virtual std::vector< double > getPT (size_t begin, size_t end, double lowest, double range, size_t divisionsPer↩ Octave=12) const

  For the specified segment of the score, return the indexes for the prime chord and its transposition, within the specified range.
- virtual std::vector< double > getPTV (size_t begin, size_t end, double lowest, double range, size_t divisions↩ PerOctave=12) const

  For the specified segment of the score, return the indexes for the prime chord, its transposition, and their voicing within the specified range.
- virtual std::vector< bool > & getRescaleMinima ()
- virtual std::vector< bool > & getRescaleRanges ()
- virtual const Event & getScaleActualMinima () const
- virtual const Event & getScaleActualRanges () const
- virtual Event & getScaleTargetMinima ()
- virtual Event & getScaleTargetRanges ()
- virtual std::vector< double > getVoicing (size_t begin, size_t end, size_t divisionsPerOctave=12) const

  Iterate over each note from the beginning to end of the segment; sort the unique pitches; return those unique pitches which also have unique pitch-class sets, in order from lowest to highest in pitch; this has the effect of returning the "inversion" or "voicing", in the musician's informal sense, of the pitches in that segment of the score.
- virtual int indexAfterTime (double time)

  Return the index of the first event after the specified time, that is return "end" for the time; if the time is not found, return the size of the score.
- virtual int indexAtTime (double time)

  Return the index of the first event at or after the specified time, that is, return "begin" for the time; if the time is not found, return the size of the score.
- virtual double indexToTime (size_t index)

  Return the time of the first event at or after the specified index; if the index is not found, return DBL_MAX.
- void initialize ()
- virtual void load (std::istream &stream)
- virtual void load (std::string filename)

  Loads score data from a MIDI (.mid) file, or a MusicXML (.xml) file.
- virtual void load_filename (std::string filename)
- virtual void process ()

  Calls Event::process on all Events in this.
- virtual void remove (size_t index)
- virtual void removeArrangement ()

  Remove instrument number, gain, and pan assignments.
- virtual void rescale ()
- virtual void rescale (Event &event)
- virtual void rescale (int dimension, bool rescaleMinimum, double minimum, bool rescaleRange=false, double range=0.0)
- virtual void rescale_event (Event &event)

- virtual void save (std::ostream &stream)

  Save as a MIDI file, format 1.
- virtual void save (std::string filename)

  Save as a MIDI file, format 1 (.mid) file, or as a partwise MusicXML (.xml) file, or as a Fomus music notation (.fms) file.
- virtual void save_filename (std::string filename)
- Score ()
- virtual void setCsoundScoreHeader (const std::string &text)
- virtual void setDuration (double targetDuration)

  Multiply existing times and durations by (targetDuration / getDuration()), i.e.
- virtual void setDurationFromZero (double targetDuration)
- virtual void setK (size_t priorBegin, size_t begin, size_t end, double base, double range)

  Find the non-unique pitch-class set of the prior segment; invert the set such that the inversion's first two pitch-classes are exchanged from the origina; conform the pitches of the current segment to that inversion.
- virtual void setKL (size_t priorBegin, size_t begin, size_t end, double base, double range, bool avoidParallels=true)

  Find the non-unique pitch-class set of the prior segment; invert the set such that the inversion's first two pitch-classes are exchanged from the original; conform the pitches of the current segment to that inversion, using the closest voice-leading from the pitches of the prior segment, optionally avoiding parallel fifths.
- virtual void setKV (size_t priorBegin, size_t begin, size_t end, double V, double base, double range)

  Find the non-unique pitch-class set of the prior segment; invert the set such that the inversion's first two pitch-classes are exchanged from the original; conform the pitches of the current segment to that inversion, with voicing V.
- virtual void setPitchClassSet (size_t begin, size_t end, const std::vector< double > &pcs, size_t divisionsPer↩ Octave=12)

  Set the pitches of the specified segment of the score to the specified pitch-class set.
- virtual void setPitches (size_t begin, size_t end, const std::vector< double > &pitches)

  Set the pitches of the specified segment of the score to the specified pitches.
- virtual void setPT (size_t begin, size_t end, double prime, double transposition, double lowest, double range, size_t divisionsPerOctave=12)

  For the specified segment of the score, adjust the pitches to match the specified indexes for the prime chord and its transposition within the specified range.
- virtual void setPTV (size_t begin, size_t end, double prime, double transposition, double voicing, double lowest, double range, size_t divisionsPerOctave=12)

  For the specified segment of the score, adjust the pitches to match the specified indexes for the prime chord, its transposition, and their voicing within the specified range.
- virtual void setQ (size_t priorBegin, size_t begin, size_t end, double Q, const std::vector< double > &context, double base, double range)

  Find the non-unique pitch-class set of the prior segment; transpose the set up by Q if the set is a T-form of the context, or down by Q if the set is an I-form of the context; then conform the pitches of the current segment to that set.
- virtual void setQL (size_t priorBegin, size_t begin, size_t end, double Q, const std::vector< double > &context, double base, double range, bool avoidParallels=true)

  Find the non-unique pitch-class set of the prior segment; transpose the set up by Q if the set is a T-form of the context, or down by Q if the set is an I-form of the context; then conform the pitches of the segment to that set, using the closest voice-leading from the pitches of the prior segment, optionally avoiding parallel fifths.
- virtual void setQV (size_t priorBegin, size_t begin, size_t end, double Q, const std::vector< double > &context, double V, double base, double range)

  Find the non-unique pitch-class set of the prior segment; transpose the set up by Q if the set is a T-form of the context, or down by Q if the set is an I-form of the context; then conform the pitches of the current segment to that set, with the voicing V.
- virtual void setVoicing (size_t begin, size_t end, const std::vector< double > &voicing, double range, size_t divisionsPerOctave=12)

  Move the pitches in the segment as little as possible to make them have the same ordering of pitch-class sets as the voicing, from the bottom to the top of the range.

- virtual void sort ()

  Sort all events in the score by time, instrument number, pitch, duration, loudness, and other dimensions as given by Event::SORT_ORDER.
- virtual void temper (double tonesPerOctave=12.0)

  Confirm pitches in this score to the closest pitch in the indicated system of equal temperament.
- virtual void tieOverlappingNotes (bool considerInstrumentNumber=false)

  If the score contains two notes of the same pitch and loudness greater than 0 that overlap in time, extend the earlier note and discard the later note.
- virtual std::string toJson ()

  Translates most of this Score to JSON:
- virtual std::string toString ()
- virtual void transform (const Eigen::MatrixXd &transformation)

  Multiply each event in this by the transformation.
- virtual void voicelead (size_t beginSource, size_t endSource, size_t beginTarget, size_t endTarget, const std::vector< double > &targetPitches, double lowest, double range, bool avoidParallelFifths, size_t divisionsPerOctave=12)

  Performs voice-leading between the specified segments of the score within the specified range, using the specified target pitches.
- virtual void voicelead (size_t beginSource, size_t endSource, size_t beginTarget, size_t endTarget, double lowest, double range, bool avoidParallelFifths, size_t divisionsPerOctave=12)

  Performs voice-leading between the specified segments of the score within the specified range.
- virtual void voicelead_pitches (size_t beginSource, size_t endSource, size_t beginTarget, size_t endTarget, const std::vector< double > &targetPitches, double lowest, double range, bool avoidParallelFifths, size_t divisionsPerOctave=12)
- virtual void voicelead_segments (size_t beginSource, size_t endSource, size_t beginTarget, size_t endTarget, double lowest, double range, bool avoidParallelFifths, size_t divisionsPerOctave=12)
- virtual ~Score ()

Static Public Member Functions

- static void getScale (std::vector< Event > &score, int dimension, size_t beginAt, size_t endAt, double &minimum, double &range)

  Save as a MIDI file, format 1.
- static void setScale (std::vector< Event > &score, int dimension, bool rescaleMinimum, bool rescaleRange, size_t beginAt, size_t endAt, double targetMinimum, double targetRange)

Data Fields

- std::string csound_score_header

  Arbitrary text that is prepended to the Csound score.
- T elements

  STL member.
- std::map< int, double > gains
- MidiFile midifile
- std::map< int, double > pans
- std::map< int, double > reassignments
- std::vector< bool > rescaleMinima
- std::vector< bool > rescaleRanges
- Event scaleActualMaxima
- Event scaleActualMinima
- Event scaleActualRanges
- Event scaleTargetMinima
- Event scaleTargetRanges

Protected Member Functions

- void createMusicModel ()

6.66.1 Detailed Description

Base class for collections of events in music space.

Can order events by time.

The implementation is a std::vector of Events. The elements of the vector are value objects, not references.

6.66.2 Constructor & Destructor Documentation

6.66.2.1 Score()

```
csound::Score::Score ( )
```

References initialize().

6.66.2.2 ~Score()

```
csound::Score::~Score ( )  [virtual]
```

6.66.3 Member Function Documentation

6.66.3.1 add()

```
void csound::Score::add (
            double time,
            double duration,
            double status,
            double instrument,
            double key,
            double velocity,
            double phase = 0,
            double pan = 0,
            double depth = 0,
            double height = 0,
            double pitches = 4095 )  [virtual]
```

References csound::fundamentalDomainByPredicate(), and csound::Event::setTime().

6.66.3.2 append() [1/2]

```
void csound::Score::append (
            double time,
            double duration,
            double status,
            double instrument,
            double key,
            double velocity,
            double phase = 0,
            double pan = 0,
            double depth = 0,
            double height = 0,
            double pitches = 4095 )  [virtual]
```

References csound::fundamentalDomainByPredicate(), and csound::Event::setTime().

6.66.3.3 append() [2/2]

```
void csound::Score::append (
            Event event )  [virtual]
```

References csound::fundamentalDomainByPredicate().

Referenced by csound::ChordLindenmayer::chordOperation(), csound::ImageToScore2::generateLocally(), load(), csound::KMeansMCRM::means_to_notes(), csound::ChordLindenmayer::noteOperation(), csound::notes(), csound::StrangeAttractor::ren csound::seqToScore(), csound::toScore(), csound::CounterpointNode::transform(), csound::CMaskNode::translate_to_silence(), csound::Intercut::traverse(), csound::Stack::traverse(), and csound::Koch::traverse().

6.66.3.4 append_event()

```
void csound::Score::append_event (
            Event event )  [virtual]
```

References csound::fundamentalDomainByPredicate().

6.66.3.5 append_note()

```
void csound::Score::append_note (
            double time,
            double duration,
            double status,
            double instrument,
            double key,
            double velocity,
            double phase = 0,
            double pan = 0,
            double depth = 0,
            double height = 0,
            double pitches = 4095 )  [virtual]
```

References csound::fundamentalDomainByPredicate(), and csound::Event::setTime().

6.66.3.6 appendToCsoundScoreHeader()

```
void csound::Score::appendToCsoundScoreHeader (
            const std::string & text )  [virtual]
```

References csound_score_header, and csound::fundamentalDomainByPredicate().

Referenced by csound::CMaskNode::translate_to_silence().

6.66.3.7 arrange() [1/3]

```
void csound::Score::arrange (
            int oldInstrumentNumber,
            int newInstrumentNumber )  [virtual]
```

Re-assign instrument number for export to Csound score.

References csound::fundamentalDomainByPredicate(), and reassignments.

Referenced by csound::MusicModel::arrange(), csound::MusicModel::arrange(), and csound::MusicModel::arrange().

6.66.3.8 arrange() [2/3]

```
void csound::Score::arrange (
            int oldInstrumentNumber,
            int newInstrumentNumber,
            double gain )  [virtual]
```

Re-assign instrument number and adjust gain for export to Csound score.

References csound::fundamentalDomainByPredicate(), gains, and reassignments.

6.66.3.9 arrange() [3/3]

```
void csound::Score::arrange (
            int oldInstrumentNumber,
            int newInstrumentNumber,
            double gain,
            double pan )  [virtual]
```

Re-assign instrument number, adjust gain, and change pan for export to Csound score.

References csound::fundamentalDomainByPredicate(), gains, pans, and reassignments.

6.66.3.10 arrange_all()

```
void csound::Score::arrange_all (
            int oldInstrumentNumber,
            int newInstrumentNumber,
            double gain,
            double pan )  [virtual]
```

References csound::fundamentalDomainByPredicate(), gains, pans, and reassignments.

6.66.3.11 createMusicModel()

```
void csound::Score::createMusicModel ( )  [protected]
```

6.66.3.12 dump()

```
void csound::Score::dump (
            std::ostream & stream )  [virtual]
```

References csound::fundamentalDomainByPredicate().

Referenced by toString().

6.66.3.13 findScale()

```
void csound::Score::findScale ( )  [virtual]
```

References csound::Event::ELEMENT_COUNT, csound::fundamentalDomainByPredicate(), getScale(), scaleActualMaxima, scaleActualMinima, scaleActualRanges, and sort().

Referenced by toJson(), csound::VoiceleadingNode::transform(), and csound::Koch::traverse().

6.66.3.14 getBlueScore()

```
std::string csound::Score::getBlueScore (
            double tonesPerOctave = 12.0,
            bool conformPitches = false )  [virtual]
```

Translate the Silence events in this to a Csound score for blue, that is, to a list of i statements with with inso, time, duration, dbsp, pch, pan.

References csound_score_header, csound::fundamentalDomainByPredicate(), gains, pans, reassignments, and sort().

6.66.3.15 getCsoundScore()

```
std::string csound::Score::getCsoundScore (
            double tonesPerOctave = 12.0,
            bool conformPitches = false )  [virtual]
```

Translate the Silence events in this to a Csound score, that is, to a list of i statements.

The Silence events are rounded off to the nearest equally tempered pitch by the specified number of tones per octave; if this argument is zero, the pitch is not tempered. The Silence events are conformed to the nearest pitch-class set in the pitch-class set dimension of the event, if the conform pitches argument is true; otherwise, the pitches are not conformed.

References csound_score_header, csound::fundamentalDomainByPredicate(), gains, pans, reassignments, and sort().

Referenced by csound::MusicModel::createCsoundScore(), and main().

6.66.3.16 getCsoundScoreHeader()

```
std::string csound::Score::getCsoundScoreHeader ( ) const  [virtual]
```

References csound_score_header.

Referenced by csound::ScoreNode::generate().

6.66.3.17 getDuration()

```
double csound::Score::getDuration ( )  [virtual]
```

Returns the time from the first event to the last event.

Reimplemented in csound::ChordScore.

References csound::fundamentalDomainByPredicate(), and sort().

Referenced by csound::VoiceleadingNode::apply(), csound::scoreToSeq(), setDuration(), csound::VoiceleadingNode::transform(), csound::Composition::translateToNotation(), and csound::Koch::traverse().

6.66.3.18 getDurationFromZero()

```
double csound::Score::getDurationFromZero ( ) const  [virtual]
```

Returns the time from 0 to the final off time; this assumes that no events start before time 0.

sort();

References csound::fundamentalDomainByPredicate().

Referenced by setDurationFromZero().

6.66.3.19 getPitches()

```
std::vector< double > csound::Score::getPitches (
            size_t begin,
            size_t end,
            size_t divisionsPerOctave = 12 ) const  [virtual]
```

Return a vector containing the MIDI key numbers in the specified segment of the score.

References csound::chord(), csound::fundamentalDomainByPredicate(), csound::Event::getKey_tempered(), csound::System::inform(), and csound::printChord().

Referenced by getPT(), getPTV(), getVoicing(), setK(), setKL(), setKV(), setPT(), setQ(), setQL(), setQV(), voicelead(), and voicelead().

6.66.3.20 getPT()

```
std::vector< double > csound::Score::getPT (
            size_t begin,
            size_t end,
            double lowest,
            double range,
            size_t divisionsPerOctave = 12 ) const  [virtual]
```

For the specified segment of the score, return the indexes for the prime chord and its transposition, within the specified range.

See: http://ruccas.org/pub/Gogins/music_atoms.pdf

References csound::chord(), csound::fundamentalDomainByPredicate(), getPitches(), csound::Voicelead::pitchClassSetToPandT(), and csound::Voicelead::uniquePcs().

6.66.3.21 getPTV()

```
std::vector< double > csound::Score::getPTV (
            size_t begin,
            size_t end,
            double lowest,
            double range,
            size_t divisionsPerOctave = 12 ) const  [virtual]
```

For the specified segment of the score, return the indexes for the prime chord, its transposition, and their voicing within the specified range.

Each of these indexes forms an additive cyclic group.

See: http://ruccas.org/pub/Gogins/music_atoms.pdf

References csound::chord(), csound::Voicelead::chordToPTV(), csound::fundamentalDomainByPredicate(), and getPitches().

Referenced by csound::VoiceleadingNode::apply().

6.66.3.22 getRescaleMinima()

```
std::vector< bool > & csound::Score::getRescaleMinima ( )  [virtual]
```

References rescaleMinima.

6.66.3.23 getRescaleRanges()

```
std::vector< bool > & csound::Score::getRescaleRanges ( )  [virtual]
```

References rescaleRanges.

6.66.3.24 getScale()

```
void csound::Score::getScale (
            std::vector< Event > & score,
            int dimension,
            size_t beginAt,
            size_t endAt,
            double & minimum,
            double & range )  [static]
```

Save as a MIDI file, format 1.

References csound::fundamentalDomainByPredicate(), csound::max(), csound::min(), and csound::Event::TIME.

Referenced by findScale(), and setScale().

6.66.3.25 getScaleActualMinima()

```
const Event & csound::Score::getScaleActualMinima ( ) const  [virtual]
```

References scaleActualMinima.

6.66.3.26 getScaleActualRanges()

```
const Event & csound::Score::getScaleActualRanges ( ) const  [virtual]
```

References scaleActualRanges.

6.66.3.27 getScaleTargetMinima()

```
Event & csound::Score::getScaleTargetMinima ( )  [virtual]
```

References scaleTargetMinima.

6.66.3.28 getScaleTargetRanges()

```
Event & csound::Score::getScaleTargetRanges ( )  [virtual]
```

References scaleTargetRanges.

6.66.3.29 getVoicing()

```
std::vector< double > csound::Score::getVoicing (
            size_t begin,
            size_t end,
            size_t divisionsPerOctave = 12 ) const  [virtual]
```

Iterate over each note from the beginning to end of the segment; sort the unique pitches; return those unique pitches which also have unique pitch-class sets, in order from lowest to highest in pitch; this has the effect of returning the "inversion" or "voicing", in the musician's informal sense, of the pitches in that segment of the score.

References csound::fundamentalDomainByPredicate(), getPitches(), csound::System::inform(), csound::Voicelead::pc(), csound::printChord(), and csound::Voicelead::uniquePcs().

Referenced by voicelead(), and voicelead().

6.66.3.30 indexAfterTime()

```
int csound::Score::indexAfterTime (
            double time )  [virtual]
```

Return the index of the first event after the specified time, that is return "end" for the time; if the time is not found, return the size of the score.

Iterating from indexAtTime(t1) to indexAfterTime(t2) is guaranteed to iterate over all and only those events included from and including t1 and up to but not including t2.

References csound::fundamentalDomainByPredicate().

Referenced by csound::VoiceleadingNode::transform().

6.66.3.31 indexAtTime()

```
int csound::Score::indexAtTime (
            double time )  [virtual]
```

Return the index of the first event at or after the specified time, that is, return "begin" for the time; if the time is not found, return the size of the score.

Iterating from indexAtTime(t1) to indexAfterTime(t2) is guaranteed to iterate over all and only those events included between t1 and t2.

References csound::fundamentalDomainByPredicate().

Referenced by csound::VoiceleadingNode::transform().

6.66.3.32 indexToTime()

```
double csound::Score::indexToTime (
            size_t index )  [virtual]
```

Return the time of the first event at or after the specified index; if the index is not found, return DBL_MAX.

References csound::fundamentalDomainByPredicate().

6.66.3.33 initialize()

```
void csound::Score::initialize ( )
```

References csound_score_header, csound::Event::DEPTH, csound::Event::DURATION, csound::Event::HEIGHT, csound::Event::HOMOGENEITY, csound::Event::INSTRUMENT, csound::Event::KEY, csound::Event::PAN, csound::Event::PHASE, csound::Event::PITCHES, rescaleMinima, rescaleRanges, scaleTargetMinima, scaleTargetRanges, csound::Event::STATUS, csound::Event::TIME, and csound::Event::VELOCITY.

Referenced by Score().

6.66.3.34 load() [1/2]

```
void csound::Score::load (
            std::istream & stream )  [virtual]
```

References append(), csound::fundamentalDomainByPredicate(), and csound::iterator().

6.66.3.35 load() [2/2]

```
void csound::Score::load (
            std::string filename )  [virtual]
```

Loads score data from a MIDI (.mid) file, or a MusicXML (.xml) file.

Non-sounding data is ignored.

References csound::System::error(), csound::fundamentalDomainByPredicate(), csound::System::inform(), and load().

Referenced by csound::ScoreNode::generate(), load(), and load_filename().

6.66.3.36 load_filename()

```
void csound::Score::load_filename (
            std::string filename )  [virtual]
```

References load().

6.66.3.37 process()

```
void csound::Score::process ( )  [virtual]
```

Calls Event::process on all Events in this.

References csound::fundamentalDomainByPredicate(), csound::System::inform(), and sort().

Referenced by csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::ExternalNode::generate(), and csound::ScoreNode::generate().

6.66.3.38 remove()

```
void csound::Score::remove (
            size_t index )  [virtual]
```

References csound::fundamentalDomainByPredicate().

6.66.3.39 removeArrangement()

```
void csound::Score::removeArrangement ( )  [virtual]
```

Remove instrument number, gain, and pan assignments.

References gains, pans, and reassignments.

Referenced by csound::MusicModel::removeArrangement().

6.66.3.40 rescale() [1/3]

```
void csound::Score::rescale ( )  [virtual]
```

References csound::Event::ELEMENT_COUNT, csound::fundamentalDomainByPredicate(), rescaleMinima, rescaleRanges, scaleTargetMinima, scaleTargetRanges, setScale(), and sort().

Referenced by rescale_event().

6.66.3.41 rescale() [2/3]

```
void csound::Score::rescale (
            Event & event )  [virtual]
```

References csound::Event::HOMOGENEITY, rescaleMinima, rescaleRanges, csound::scale(), scaleActualMinima, scaleActualRanges, scaleTargetMinima, and scaleTargetRanges.

6.66.3.42 rescale() [3/3]

```
void csound::Score::rescale (
            int dimension,
            bool rescaleMinimum,
            double minimum,
            bool rescaleRange = false,
            double range = 0.0 )  [virtual]
```

References csound::fundamentalDomainByPredicate(), and setScale().

6.66.3.43 rescale_event()

```
void csound::Score::rescale_event (
            Event & event )  [virtual]
```

References csound::fundamentalDomainByPredicate(), and rescale().

6.66.3.44 save() [1/2]

```
void csound::Score::save (
            std::ostream & stream )  [virtual]
```

Save as a MIDI file, format 1.

References csound::fundamentalDomainByPredicate(), csound::note(), and sort().

6.66.3.45 save() [2/2]

```
void csound::Score::save (
            std::string filename )  [virtual]
```

Save as a MIDI file, format 1 (.mid) file, or as a partwise MusicXML (.xml) file, or as a Fomus music notation (.fms) file.

Only sounding data is saved.

References csound::System::error(), csound::fundamentalDomainByPredicate(), csound::System::inform(), save(), and sort().

Referenced by csound::MusicModel::csoundArgv(), csound::MusicModel::generate(), csound::MusicModel::processArgs(), save(), and save_filename().

6.66.3.46 save_filename()

```
void csound::Score::save_filename (
            std::string filename )  [virtual]
```

References save().

6.66.3.47 setCsoundScoreHeader()

```
void csound::Score::setCsoundScoreHeader (
            const std::string & text )  [virtual]
```

References csound_score_header, and csound::fundamentalDomainByPredicate().

6.66.3.48 setDuration()

```
void csound::Score::setDuration (
            double targetDuration )  [virtual]
```

Multiply existing times and durations by (targetDuration / getDuration()), i.e.

stretch or shrink musical time.

Reimplemented in csound::ChordScore.

References csound::fundamentalDomainByPredicate(), getDuration(), and csound::Event::getTime().

Referenced by csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::ScoreNode::generate(), csound::ExternalNode::generateLocally(), csound::CellShuffle::transform(), and csound::Stack::traverse().

6.66.3.49 setDurationFromZero()

```
void csound::Score::setDurationFromZero (
            double targetDuration )  [virtual]
```

References csound::fundamentalDomainByPredicate(), getDurationFromZero(), and csound::Event::getTime().

6.66.3.50 setK()

```
void csound::Score::setK (
            size_t priorBegin,
            size_t begin,
            size_t end,
            double base,
            double range )  [virtual]
```

Find the non-unique pitch-class set of the prior segment; invert the set such that the inversion's first two pitch-classes are exchanged from the origina; conform the pitches of the current segment to that inversion.

References csound::fundamentalDomainByPredicate(), getPitches(), csound::Voicelead::K(), csound::printChord(), setPitchClassSet(), and csound::Voicelead::uniquePcs().

Referenced by csound::VoiceleadingNode::apply().

6.66.3.51 setKL()

```
void csound::Score::setKL (
            size_t priorBegin,
            size_t begin,
            size_t end,
            double base,
            double range,
            bool avoidParallels = true )  [virtual]
```

Find the non-unique pitch-class set of the prior segment; invert the set such that the inversion's first two pitch-classes are exchanged from the original; conform the pitches of the current segment to that inversion, using the closest voice-leading from the pitches of the prior segment, optionally avoiding parallel fifths.

References csound::fundamentalDomainByPredicate(), getPitches(), csound::Voicelead::K(), csound::Voicelead::uniquePcs(), and voicelead().

Referenced by csound::VoiceleadingNode::apply().

6.66.3.52 setKV()

```
void csound::Score::setKV (
            size_t priorBegin,
            size_t begin,
            size_t end,
            double V,
            double base,
            double range )  [virtual]
```

Find the non-unique pitch-class set of the prior segment; invert the set such that the inversion's first two pitch-classes are exchanged from the original; conform the pitches of the current segment to that inversion, with voicing V.

References csound::fundamentalDomainByPredicate(), getPitches(), csound::Voicelead::K(), csound::Voicelead::pitchClassSetToPandT() setPTV(), and csound::Voicelead::uniquePcs().

Referenced by csound::VoiceleadingNode::apply().

6.66.3.53 setPitchClassSet()

```
void csound::Score::setPitchClassSet (
            size_t begin,
            size_t end,
            const std::vector< double > & pcs,
            size_t divisionsPerOctave = 12 )  [virtual]
```

Set the pitches of the specified segment of the score to the specified pitch-class set.

Each pitch in the score is moved to the closest pitch-class in the specified set.

References csound::Voicelead::conformToPitchClassSet(), csound::fundamentalDomainByPredicate(), and csound::Event::setKey().

Referenced by csound::VoiceleadingNode::apply(), setK(), setPT(), setQ(), and voicelead().

6.66.3.54 setPitches()

```
void csound::Score::setPitches (
            size_t begin,
            size_t end,
            const std::vector< double > & pitches )  [virtual]
```

Set the pitches of the specified segment of the score to the specified pitches.

Each pitch in the score is moved to the closest pitch in the specified pitches.

References csound::Voicelead::closestPitch(), and csound::fundamentalDomainByPredicate().

Referenced by setPTV(), voicelead(), and voicelead().

6.66.3.55 setPT()

```
void csound::Score::setPT (
            size_t begin,
            size_t end,
            double prime,
            double transposition,
            double lowest,
            double range,
            size_t divisionsPerOctave = 12 )  [virtual]
```

For the specified segment of the score, adjust the pitches to match the specified indexes for the prime chord and its transposition within the specified range.

See: http://ruccas.org/pub/Gogins/music_atoms.pdf

References csound::fundamentalDomainByPredicate(), getPitches(), csound::System::inform(), csound::Voicelead::pAndTtoPitchClassSe
csound::printChord(), setPitchClassSet(), csound::T(), and csound::Voicelead::uniquePcs().

Referenced by csound::VoiceleadingNode::apply().

6.66.3.56 setPTV()

```
void csound::Score::setPTV (
            size_t begin,
            size_t end,
            double prime,
            double transposition,
            double voicing,
            double lowest,
            double range,
            size_t divisionsPerOctave = 12 )  [virtual]
```

For the specified segment of the score, adjust the pitches to match the specified indexes for the prime chord, its transposition, and their voicing within the specified range.

Each of these indexes forms an additive cyclic group.

See: http://ruccas.org/pub/Gogins/music_atoms.pdf

References csound::fundamentalDomainByPredicate(), csound::System::inform(), csound::printChord(), csound::Voicelead::ptvToChord(),
setPitches(), csound::T(), and csound::Voicelead::uniquePcs().

Referenced by csound::VoiceleadingNode::apply(), setKV(), and setQV().

6.66.3.57 setQ()

```
void csound::Score::setQ (
            size_t priorBegin,
            size_t begin,
            size_t end,
            double Q,
            const std::vector< double > & context,
            double base,
            double range )  [virtual]
```

Find the non-unique pitch-class set of the prior segment; transpose the set up by Q if the set is a T-form of the context, or down by Q if the set is an I-form of the context; then conform the pitches of the current segment to that set.

The context will be reduced or doubled as required to match the cardinality of the set.

References csound::fundamentalDomainByPredicate(), getPitches(), csound::System::inform(), csound::matchContextSize(), csound::printChord(), csound::Voicelead::Q(), setPitchClassSet(), and csound::Voicelead::uniquePcs().

Referenced by csound::VoiceleadingNode::apply().

6.66.3.58 setQL()

```
void csound::Score::setQL (
            size_t priorBegin,
            size_t begin,
            size_t end,
            double Q,
            const std::vector< double > & context,
            double base,
            double range,
            bool avoidParallels = true )  [virtual]
```

Find the non-unique pitch-class set of the prior segment; transpose the set up by Q if the set is a T-form of the context, or down by Q if the set is an I-form of the context; then conform the pitches of the segment to that set, using the closest voice-leading from the pitches of the prior segment, optionally avoiding parallel fifths.

The context will be reduced or doubled as required to match the cardinality of the set.

References csound::fundamentalDomainByPredicate(), getPitches(), csound::matchContextSize(), csound::printChord(), csound::Voicelead::Q(), csound::Voicelead::uniquePcs(), and voicelead().

Referenced by csound::VoiceleadingNode::apply().

6.66.3.59 setQV()

```
void csound::Score::setQV (
            size_t priorBegin,
            size_t begin,
            size_t end,
            double Q,
            const std::vector< double > & context,
            double V,
            double base,
            double range )  [virtual]
```

Find the non-unique pitch-class set of the prior segment; transpose the set up by Q if the set is a T-form of the context, or down by Q if the set is an I-form of the context; then conform the pitches of the current segment to that set, with the voicing V.

The context will be reduced or doubled as required to match the cardinality of the set.

References csound::fundamentalDomainByPredicate(), getPitches(), csound::matchContextSize(), csound::Voicelead::pitchClassSetToPa csound::printChord(), csound::Voicelead::Q(), setPTV(), and csound::Voicelead::uniquePcs().

Referenced by csound::VoiceleadingNode::apply().

6.66.3.60 setScale()

```
void csound::Score::setScale (
            std::vector< Event > & score,
            int dimension,
            bool rescaleMinimum,
            bool rescaleRange,
            size_t beginAt,
            size_t endAt,
            double targetMinimum,
            double targetRange )  [static]
```

References csound::System::debug(), csound::fundamentalDomainByPredicate(), getScale(), csound::Event::PITCHES, and csound::scale().

Referenced by rescale(), and rescale().

6.66.3.61 setVoicing()

```
void csound::Score::setVoicing (
            size_t begin,
            size_t end,
            const std::vector< double > & voicing,
            double range,
            size_t divisionsPerOctave = 12 )  [virtual]
```

Move the pitches in the segment as little as possible to make them have the same ordering of pitch-class sets as the voicing, from the bottom to the top of the range.

This has the effect of "inverting" or "re-voicing", in the musician's informal sense, the pitches in that segment of the score.

References csound::Voicelead::conformToPitchClassSet(), csound::fundamentalDomainByPredicate(), csound::Voicelead::pc(), and csound::Voicelead::pcs().

6.66.3.62 sort()

```
void csound::Score::sort ( )  [virtual]
```

Sort all events in the score by time, instrument number, pitch, duration, loudness, and other dimensions as given by Event::SORT_ORDER.

References csound::fundamentalDomainByPredicate().

Referenced by findScale(), csound::ScoreModel::generate(), csound::ScoreNode::generate(), csound::ExternalNode::generateLocally(), csound::ImageToScore2::generateLocally(), getBlueScore(), getCsoundScore(), getDuration(), process(), rescale(), save(), save(), tieOverlappingNotes(), toJson(), csound::Cell::transform(), csound::CellRepeat::transform(), csound::CounterpointNode::tra csound::VoiceleadingNode::transform(), csound::Composition::translateToNotation(), and csound::Koch::traverse().

6.66.3.63 temper()

```
void csound::Score::temper (
            double tonesPerOctave = 12.0 )  [virtual]
```

Confirm pitches in this score to the closest pitch in the indicated system of equal temperament.

References csound::fundamentalDomainByPredicate().

Referenced by csound::ScoreModel::generate().

6.66.3.64 tieOverlappingNotes()

```
void csound::Score::tieOverlappingNotes (
            bool considerInstrumentNumber = false )  [virtual]
```

If the score contains two notes of the same pitch and loudness greater than 0 that overlap in time, extend the earlier note and discard the later note.

References csound::fundamentalDomainByPredicate(), csound::Event::setOffTime(), and sort().

Referenced by csound::ScoreModel::generate(), and csound::ChordLindenmayer::tieOverlappingNotes().

6.66.3.65 toJson()

```
std::string csound::Score::toJson ( )  [virtual]
```

Translates most of this Score to JSON:

1. The vector of Events, sorted and otherwise massaged.
2. The actual minima, maxima, and ranges. The JSON schema is: { events: [[],...], minima: [],' maxima: [], ranges: [] }; This is useful, e.g., for sending a complete score to the JavaScript context of a Web page for display using WebGL or Three.js.

References findScale(), csound::fundamentalDomainByPredicate(), scaleActualMaxima, scaleActualMinima, scaleActualRanges, and sort().

6.66.3.66 toString()

```
std::string csound::Score::toString ( )  [virtual]
```

References dump(), and csound::fundamentalDomainByPredicate().

6.66.3.67 transform()

```
void csound::Score::transform (
            const Eigen::MatrixXd & transformation )  [virtual]
```

Multiply each event in this by the transformation.

References csound::fundamentalDomainByPredicate().

6.66.3.68 voicelead() [1/2]

```
void csound::Score::voicelead (
            size_t beginSource,
            size_t endSource,
            size_t beginTarget,
            size_t endTarget,
            const std::vector< double > & targetPitches,
            double lowest,
            double range,
            bool avoidParallelFifths,
            size_t divisionsPerOctave = 12 )  [virtual]
```

Performs voice-leading between the specified segments of the score within the specified range, using the specified target pitches.

The voice-leading is first the closest by taxicab norm, and then the simplest in motion, optionally avoiding parallel fifths. Only the pitches of the target notes are affected. If necessary, the number of pitches in the target chord is adjusted to match the source.

See: http://ruccas.org/pub/Gogins/music_atoms.pdf

References csound::fundamentalDomainByPredicate(), csound::System::getMessageLevel(), getPitches(), getVoicing(), csound::System::inform(), csound::System::INFORMATION_LEVEL, csound::Voicelead::nonBijectiveVoicelead(), csound::Voicelead::pcs(), csound::printChord(), setPitchClassSet(), setPitches(), and csound::Voicelead::uniquePcs().

6.66.3.69 voicelead() [2/2]

```
void csound::Score::voicelead (
            size_t beginSource,
            size_t endSource,
            size_t beginTarget,
            size_t endTarget,
            double lowest,
            double range,
            bool avoidParallelFifths,
            size_t divisionsPerOctave = 12 )  [virtual]
```

Performs voice-leading between the specified segments of the score within the specified range.

The voice-leading is first the closest by taxicab norm, and then the simplest in motion, optionally avoiding parallel fifths. Only the pitches of the target notes are affected. If necessary, the number of pitches in the target chord is adjusted to match the source.

See: http://ruccas.org/pub/Gogins/music_atoms.pdf

References csound::fundamentalDomainByPredicate(), csound::System::getMessageLevel(), getPitches(), getVoicing(), csound::System::inform(), csound::System::INFORMATION_LEVEL, csound::Voicelead::nonBijectiveVoicelead(), csound::Voicelead::pcs(), csound::printChord(), setPitches(), and csound::Voicelead::uniquePcs().

Referenced by csound::VoiceleadingNode::apply(), setKL(), setQL(), voicelead_pitches(), and voicelead_segments().

6.66.3.70 voicelead_pitches()

```
void csound::Score::voicelead_pitches (
            size_t beginSource,
            size_t endSource,
            size_t beginTarget,
            size_t endTarget,
            const std::vector< double > & targetPitches,
            double lowest,
            double range,
            bool avoidParallelFifths,
            size_t divisionsPerOctave = 12 )  [virtual]
```

References csound::fundamentalDomainByPredicate(), and voicelead().

6.66.3.71 voicelead_segments()

```
void csound::Score::voicelead_segments (
            size_t beginSource,
            size_t endSource,
            size_t beginTarget,
            size_t endTarget,
            double lowest,
            double range,
            bool avoidParallelFifths,
            size_t divisionsPerOctave = 12 )  [virtual]
```

References csound::fundamentalDomainByPredicate(), and voicelead().

6.66.4 Field Documentation

6.66.4.1 csound_score_header

```
std::string csound::Score::csound_score_header
```

Arbitrary text that is prepended to the Csound score.

Should normally be Csound comments or "f" statements, or pre-composed Csound events.

Referenced by appendToCsoundScoreHeader(), getBlueScore(), getCsoundScore(), getCsoundScoreHeader(), initialize(), and setCsoundScoreHeader().

6.66.4.2 elements

```
T std::vector< T >::elements  [inherited]
```

STL member.

6.66.4.3 gains

```
std::map<int, double> csound::Score::gains
```

Referenced by arrange(), arrange(), arrange_all(), getBlueScore(), getCsoundScore(), and removeArrangement().

6.66.4.4 midifile

```
MidiFile csound::Score::midifile
```

6.66.4.5 pans

```
std::map<int, double> csound::Score::pans
```

Referenced by arrange(), arrange_all(), getBlueScore(), getCsoundScore(), and removeArrangement().

6.66.4.6 reassignments

```
std::map<int, double> csound::Score::reassignments
```

Referenced by arrange(), arrange(), arrange(), arrange_all(), getBlueScore(), getCsoundScore(), and removeArrangement().

6.66.4.7 rescaleMinima

```
std::vector<bool> csound::Score::rescaleMinima
```

Referenced by csound::Rescale::getRescale(), getRescaleMinima(), initialize(), csound::Rescale::Rescale(), rescale(), rescale(), csound::Rescale::setRescale(), and csound::Rescale::transform().

6.66.4.8 rescaleRanges

```
std::vector<bool> csound::Score::rescaleRanges
```

Referenced by csound::Rescale::getRescale(), getRescaleRanges(), initialize(), csound::Rescale::Rescale(), rescale(), rescale(), csound::Rescale::setRescale(), and csound::Rescale::transform().

6.66.4.9 scaleActualMaxima

```
Event csound::Score::scaleActualMaxima
```

Referenced by findScale(), and toJson().

6.66.4.10 scaleActualMinima

```
Event csound::Score::scaleActualMinima
```

Referenced by findScale(), csound::Lindenmayer::generateLocally(), getScaleActualMinima(), rescale(), toJson(), csound::VoiceleadingNode::transform(), csound::Koch::traverse(), and csound::Lindenmayer::updateActual().

6.66.4.11 scaleActualRanges

```
Event csound::Score::scaleActualRanges
```

Referenced by findScale(), csound::Lindenmayer::generateLocally(), getScaleActualRanges(), rescale(), toJson(), and csound::Lindenmayer::updateActual().

6.66.4.12 scaleTargetMinima

```
Event csound::Score::scaleTargetMinima
```

Referenced by csound::Rescale::getRescale(), getScaleTargetMinima(), initialize(), csound::ImageToScore2::pixel_to_event(), rescale(), rescale(), csound::Rescale::setRescale(), and csound::Rescale::transform().

6.66.4.13 scaleTargetRanges

```
Event csound::Score::scaleTargetRanges
```

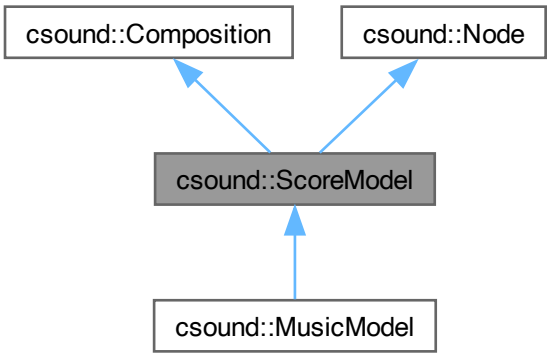
Referenced by csound::Rescale::getRescale(), getScaleTargetRanges(), initialize(), csound::ImageToScore2::pixel_to_event(), rescale(), rescale(), csound::Rescale::setRescale(), and csound::Rescale::transform().

6.67 csound::ScoreModel Class Reference

Base class for compositions that use the principle of a music graph to generate a score.

```
#include <ScoreModel.hpp>
```

Inheritance diagram for csound::ScoreModel:



Public Member Functions

- virtual void addChild (Node *node)

  Adds an immediate child Node to this.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

  Clears the score.
- virtual void clearOutputSoundfileName ()
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual int generate ()

Generates a score based on a music graph defined by the child nodes of this.

- virtual void generate (Score &score_from_this)

  Optionally generate notes into the score.
- virtual void generateAllNames ()

  Generates all filenames and other text based on required stem, output_directory, filename extension, and metadata.
- virtual std::string getAlbum () const
- virtual std::string getArtist () const
- virtual std::string getAuthor () const
- virtual std::string getBasename () const

  Returns the complete basename of the file, i.e., the output directory plus the stem.
- virtual std::string getCdSoundfileFilepath () const

  Returns a soundfile name for a CD audio track based on the filename of this, by appending ".cd.wav" to the filename.
- virtual Node * getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual bool getConformPitches () const

  Returns whether or not the pitches in generated scores will be conformed to the nearest equally tempered pitch.
- virtual std::string getCopyright () const
- virtual double getDuration () const

  Returns the duration to which all times and durations of all events will be rescaled.
- virtual std::string getFileFilepath () const

  Returns the complete basename of the file, i.e., the output directory plus the filename.
- virtual std::string getFilename () const

  Returns the stem of this, which is used as a base for derived filenames (soundfile, MIDI file, etc.).
- virtual std::string getFomusfileFilepath () const

  Returns a MusicXML filename based on the filename of this, by appending ".fms" to the filename.
- virtual std::string getLicense () const
- virtual std::string getLilypondfileFilepath () const

  Returns a MusicXML filename based on the filename of this, by appending ".ly" to the filename.
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual std::string getMidifileFilepath () const

  Returns a MIDI filename based on the filename of this, by appending ".mid" to the filename.
- virtual std::string getMp3SoundfileFilepath () const

  Returns a soundfile name for an MP3 file based on the filename of this, by appending ".mp3" to the filename.
- virtual std::string getMusicXmlfileFilepath () const

  Returns a MusicXML filename based on the filename of this, by appending ".xml" to the filename.
- virtual std::string getNormalizedSoundfileFilepath () const

  Returns a soundfile name based on the filename of this, by appending ".norm.wav" to the filename.
- virtual std::string getOutputDirectory () const

  Returns the directory in which to place the output files of this.
- virtual std::string getOutputSoundfileFilepath () const

  Returns a soundfile name based on the filename of this, by appending ".wav" to the filename, which is the default, or a non-default ouput name which need not be a file but must be set using setOutputSoundfileName().
- virtual std::string getPerformanceRightsOrganization () const
- virtual Score & getScore ()

  Return the self-contained Score.
- virtual intptr_t getThis ()

  Returns the address of this as a long integer.

- virtual Node * getThisNode ()

  Returns the address of this as a Node pointer.
- virtual bool getTieOverlappingNotes () const

  Returns whether or not overlapping notes in generated scores are replaced by one note.
- virtual std::string getTimestamp () const

  Returns the time the score was generated.
- virtual std::string getTitle () const
- virtual double getTonesPerOctave () const

  Returns the number of equally tempered intervals per octave (the default is 12, 0 means non-equally tempered).
- virtual std::string getYear () const
- virtual void initialize ()
- virtual int normalizeOutputSoundfile (double levelDb=-3.0)

  Assuming the score has been rendered, uses sox to translate the output soundfile to a normalized soundfile.
- virtual int perform ()

  Performs the current score to create an output soundfile, which should be tagged with author, timestamp, copyright, title, and optionally album.
- virtual int performAll ()

  Convenience function that calls performMaster(), and translateMaster().
- virtual int performMaster ()

  Convenience function that calls saveMidi(), saveMusicXML(), and perform().
- virtual int processArgs (const std::vector< std::string > &args)

  Pass the invoking program's command-line arguments to processArgs() and it will perform with possibly back-end-dependent options.
- virtual int processArgv (int argc, const char **argv)

  Pass the invoking program's command-line arguments to processArgs() and it will perform with possibly back-end-dependent options.
- virtual int render ()

  Convenience function that calls clear(), generate(), perform().
- virtual int renderAll ()

  Convenience function that calls clear(), generate(), performAll().
- ScoreModel ()
- virtual void setAlbum (std::string value)
- virtual void setArtist (std::string value)
- virtual void setAuthor (std::string value)
- virtual void setConformPitches (bool conformPitches)

  Sets whether or not the pitches in generated scores will be conformed to the nearest equally tempered pitch.
- virtual void setCopyright (std::string value)
- virtual void setDuration (double seconds)

  At the end of processing, if the defined duration is not zero, the times and durations of all events are rescaled to the defined duration.
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void setFilename (std::string filename)

  Sets the filename of this – basically, the title of the composition.
- virtual void setLicense (std::string value)
- virtual void setOutputDirectory (std::string directory)

  Sets the directory in which to place the output files of this.
- virtual void setOutputSoundfileName (std::string name)

Sets a non-default output name (could be an audio device not a file).

- virtual void setPerformanceRightsOrganization (std::string value)
- virtual void setScore (Score &score)

  Sets the score in this to the indicated score.
- virtual void setTieOverlappingNotes (bool tieOverlappingNotes)

  Sets whether or not overlapping notes in generated scores are replaced by one note.
- virtual void setTitle (std::string value)
- virtual void setTonesPerOctave (double tonesPerOctave)

  Sets the number of equally tempered intervals per octave (the default is 12, 0 means non-equally tempered).
- virtual void setYear (std::string value)
- virtual int tagFile (std::string filename) const
- virtual void transform (Score &score_from_children)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual int translateMaster ()

  Convenience function that calls rescaleOutputSoundfile(), translateToCdAudio(), and translateToMp3().
- virtual int translateToCdAudio (double levelDb=-3.0)

  Assuming the score has been rendered, uses sox to translate the output soundfile to normalized CD-audio format.
- virtual int translateToMp3 (double bitrate=256.01, double levelDb=-3.0)

  Assuming the score has been rendered, uses sox and LAME to translate the output soundfile to normalized MP3 format.
- virtual int translateToMp4 ()

  Assuming the score has been rendered, uses sox and ffmpeg to translate the output soundfile to a normalized mp4 video suitable for uploading to YouTube.
- virtual int translateToNotation (const std::vector< std::string > partNames=std::vector< std::string >(), std::string header="")

  Saves the generated score in Fomus format and uses Fomus and Lilypond to translate that to a PDF of music notation.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.
- virtual void write (const char *text)

  Write as if to stderr.
- virtual ~ScoreModel ()

Static Public Member Functions

- static std::string generateFilename ()

  Generates a versioned filename.
- static std::string makeTimestamp ()

  Returns the current locale time as a string.

Data Fields

- std::vector< Node * > children

  Child Nodes, if any.

Protected Attributes

- std::string album

  Optional metadata.
- std::string artist

  Required metadata.
- std::string author

  Required metadata.
- std::string base_filepath

  Generated.
- Score baseScore
- std::string bext_description

  Generated.
- std::string bext_orig_ref

  Generated.
- std::string bext_originator

  Generated.
- std::string cd_quality_filepath

  Generated.
- bool conformPitches
- std::string copyright

  Required metadata.
- double duration
- std::string flac_filepath

  Generated.
- std::string label

  Generated.
- std::string license

  Required metadata.
- Eigen::MatrixXd localCoordinates
- std::string master_filepath

  Generated.
- std::string midi_filepath

  Generated.
- std::string mp3_filepath

  Generated.
- std::string mp4_filepath

  Generated.
- std::string normalized_master_filepath

  Generated.
- std::string notes

  Optional metadata, defaults to "Electroacoustic Music.".
- std::string output_directory

  Required.
- std::string output_filename
- std::string performance_rights_organization

  Optional metadata.
- Score & score

- std::string spectrogram_filepath

  Generated.
- std::string stem

  Required.
- bool tieOverlappingNotes
- std::string timestamp

  Generated.
- double tonesPerOctave
- std::string track

  Optional metadata.
- std::string year

  Required metadata.

6.67.1 Detailed Description

Base class for compositions that use the principle of a music graph to generate a score.

A music graph is a directed acyclic graph of nodes including empty nodes, nodes that contain only child nodes, score nodes, event generator nodes, event transformer nodes, and others. Each node is associated with a local transformation of coordinate system in music space using a 12 x 12 homogeneous matrix. To generate the score, the music graph is traversed depth first, and each node postconcatenates its local transformation of coordinate system with the coordinate system of its parent to derive a new local coordinate system, which is applied to all child events.

6.67.2 Constructor & Destructor Documentation

6.67.2.1 ScoreModel()

```
csound::ScoreModel::ScoreModel ( )
```

6.67.2.2 ~ScoreModel()

```
csound::ScoreModel::~ScoreModel ( )  [virtual]
```

6.67.3 Member Function Documentation

6.67.3.1 addChild()

```
virtual void csound::ScoreModel::addChild (
            Node * node )  [inline], [virtual]
```

Adds an immediate child Node to this.

Reimplemented from csound::Node.

Referenced by main().

6.67.3.2 childCount()

```
virtual size_t csound::ScoreModel::childCount ( ) const  [inline], [virtual]
```

Returns the number of immediate children of this.

Reimplemented from csound::Node.

6.67.3.3 clear()

```
void csound::ScoreModel::clear ( )  [virtual]
```

Clears the score.

Reimplemented from csound::Composition.

Reimplemented in csound::MusicModel.

References csound::Composition::clear(), and csound::Node::clear().

6.67.3.4 clearOutputSoundfileName()

```
void csound::Composition::clearOutputSoundfileName ( )  [virtual], [inherited]
```

References csound::Composition::output_filename.

6.67.3.5 createTransform()

```
virtual Eigen::MatrixXd csound::ScoreModel::createTransform ( )  [inline], [virtual]
```

Returns the identity matrix for score space.

Reimplemented from csound::Node.

6.67.3.6 element()

```
virtual double & csound::ScoreModel::element (
            size_t row,
            size_t column )  [inline], [virtual]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented from csound::Node.

6.67.3.7 generate() [1/2]

```
int csound::ScoreModel::generate ( )  [virtual]
```

Generates a score based on a music graph defined by the child nodes of this.

Reimplemented from csound::Composition.

Reimplemented in csound::MusicModel.

References csound::Node::children, csound::Composition::duration, csound::Composition::getConformPitches(), getLocalCoordinates(), csound::Composition::getTieOverlappingNotes(), csound::Composition::getTonesPerOctave(), csound::System::message(), csound::Score::process(), csound::Composition::score, csound::Score::setDuration(), csound::Score::sort(), csound::Score::temper(), csound::Score::tieOverlappingNotes(), and traverse().

6.67.3.8 generate() [2/2]

```
virtual void csound::ScoreModel::generate (
            Score & score_from_this )  [inline], [virtual]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented from csound::Node.

6.67.3.9 generateAllNames()

```
void csound::Composition::generateAllNames ( )  [virtual], [inherited]
```

Generates all filenames and other text based on required stem, output_directory, filename extension, and metadata.

References csound::Composition::album, csound::Composition::artist, csound::Composition::author, csound::Composition::base_filepath, csound::Composition::bext_description, csound::Composition::bext_orig_ref, csound::Composition::bext_originator, csound::Composition::cd_quality_filepath, csound::Composition::copyright, csound::Composition::flac_filepath, csound::fundamentalDomainByPredicate(), csound::Composition::getLicense(), csound::Composition::getOutputDirectory(), csound::Composition::getTitle(), csound::System::inform(), csound::Composition::label, csound::Composition::makeTimestamp(), csound::Composition::master_filepath, csound::Composition::midi_filepath, csound::Composition::mp3_filepath, csound::Composition::mp4_filepath, csound::Composition::normalized_master_filepath, csound::Composition::notes, csound::Composition::output_directory, csound::Composition::performance_rights_organization, csound::Composition::spectrogram_filep csound::Composition::stem, csound::Composition::timestamp, csound::Composition::track, and csound::Composition::year.

Referenced by csound::MusicModel::csoundArgv(), csound::Composition::processArgs(), and csound::MusicModel::processArgs().

6.67.3.10 generateFilename()

```
std::string csound::Composition::generateFilename ( )  [static], [inherited]
```

Generates a versioned filename.

References csound::fundamentalDomainByPredicate(), and csound::Composition::makeTimestamp().

6.67.3.11 getAlbum()

```
std::string csound::Composition::getAlbum ( ) const  [virtual], [inherited]
```

References csound::Composition::album.

Referenced by csound::Composition::tagFile(), and csound::Composition::translateToMp3().

6.67.3.12 getArtist()

```
std::string csound::Composition::getArtist ( ) const  [virtual], [inherited]
```

References csound::Composition::artist.

Referenced by csound::Composition::tagFile(), and csound::Composition::translateToNotation().

6.67.3.13 getAuthor()

```
std::string csound::Composition::getAuthor ( ) const  [virtual], [inherited]
```

References csound::Composition::author.

Referenced by csound::Composition::tagFile(), and csound::Composition::translateToMp3().

6.67.3.14 getBasename()

```
std::string csound::Composition::getBasename ( ) const  [virtual], [inherited]
```

Returns the complete basename of the file, i.e., the output directory plus the stem.

References csound::Composition::base_filepath.

Referenced by csound::Composition::getFomusfileFilepath(), and csound::Composition::getLilypondfileFilepath().

6.67.3.15 getCdSoundfileFilepath()

```
std::string csound::Composition::getCdSoundfileFilepath ( ) const  [virtual], [inherited]
```

Returns a soundfile name for a CD audio track based on the filename of this, by appending ".cd.wav" to the filename.

References csound::Composition::cd_quality_filepath.

Referenced by csound::Composition::translateToCdAudio(), and csound::Composition::translateToMp3().

6.67.3.16 getChild()

```
virtual Node * csound::ScoreModel::getChild (
            size_t index )  [inline], [virtual]
```

Returns the immediate child of this at the index.

Reimplemented from csound::Node.

6.67.3.17 getConformPitches()

```
bool csound::Composition::getConformPitches ( ) const  [virtual], [inherited]
```

Returns whether or not the pitches in generated scores will be conformed to the nearest equally tempered pitch.

References csound::Composition::conformPitches.

Referenced by generate().

6.67.3.18 getCopyright()

```
std::string csound::Composition::getCopyright ( ) const  [virtual], [inherited]
```

References csound::Composition::copyright.

Referenced by csound::Composition::tagFile(), csound::Composition::translateToMp3(), and csound::Composition::translateToMp4().

6.67.3.19 getDuration()

```
double csound::Composition::getDuration ( ) const  [virtual], [inherited]
```

Returns the duration to which all times and durations of all events will be rescaled.

If the duration is 0, no rescaling is performed.

References csound::Composition::duration.

6.67.3.20 getFileFilepath()

```
std::string csound::Composition::getFileFilepath ( ) const  [virtual], [inherited]
```

Returns the complete basename of the file, i.e., the output directory plus the filename.

References csound::Composition::base_filepath.

6.67.3.21 getFilename()

```
std::string csound::Composition::getFilename ( ) const  [virtual], [inherited]
```

Returns the stem of this, which is used as a base for derived filenames (soundfile, MIDI file, etc.).

References csound::Composition::stem.

6.67.3.22 getFomusfileFilepath()

```
std::string csound::Composition::getFomusfileFilepath ( ) const  [virtual], [inherited]
```

Returns a MusicXML filename based on the filename of this, by appending ".fms" to the filename.

References csound::Composition::getBasename().

Referenced by csound::Composition::translateToNotation().

6.67.3.23 getLicense()

```
std::string csound::Composition::getLicense ( ) const  [virtual], [inherited]
```

References csound::Composition::license.

Referenced by csound::Composition::generateAllNames(), and csound::Composition::tagFile().

6.67.3.24 getLilypondfileFilepath()

```
std::string csound::Composition::getLilypondfileFilepath ( ) const  [virtual], [inherited]
```

Returns a MusicXML filename based on the filename of this, by appending ".ly" to the filename.

References csound::Composition::getBasename().

6.67.3.25 getLocalCoordinates()

```
virtual Eigen::MatrixXd csound::ScoreModel::getLocalCoordinates ( ) const  [inline], [virtual]
```

Returns the local transformation of coordinate system.

Reimplemented from csound::Node.

Referenced by csound::MusicModel::generate(), and generate().

6.67.3.26 getMidifileFilepath()

```
std::string csound::Composition::getMidifileFilepath ( ) const  [virtual], [inherited]
```

Returns a MIDI filename based on the filename of this, by appending ".mid" to the filename.

References csound::Composition::midi_filepath.

Referenced by csound::MusicModel::csoundArgv(), csound::MusicModel::generate(), and csound::MusicModel::processArgs().

6.67.3.27 getMp3SoundfileFilepath()

```
std::string csound::Composition::getMp3SoundfileFilepath ( ) const  [virtual], [inherited]
```

Returns a soundfile name for an MP3 file based on the filename of this, by appending ".mp3" to the filename.

References csound::Composition::mp3_filepath.

Referenced by csound::Composition::translateToMp3().

6.67.3.28 getMusicXmlfileFilepath()

```
std::string csound::Composition::getMusicXmlfileFilepath ( ) const  [virtual], [inherited]
```

Returns a MusicXML filename based on the filename of this, by appending ".xml" to the filename.

References csound::Composition::base_filepath.

6.67.3.29 getNormalizedSoundfileFilepath()

```
std::string csound::Composition::getNormalizedSoundfileFilepath ( ) const  [virtual], [inherited]
```

Returns a soundfile name based on the filename of this, by appending ".norm.wav" to the filename.

References csound::Composition::normalized_master_filepath.

Referenced by csound::Composition::normalizeOutputSoundfile(), and csound::MusicModel::processArgs().

6.67.3.30 getOutputDirectory()

```
std::string csound::Composition::getOutputDirectory ( ) const  [virtual], [inherited]
```

Returns the directory in which to place the output files of this.

References csound::fundamentalDomainByPredicate(), and csound::Composition::output_directory.

Referenced by csound::Composition::generateAllNames().

6.67.3.31 getOutputSoundfileFilepath()

```
std::string csound::Composition::getOutputSoundfileFilepath ( ) const  [virtual], [inherited]
```

Returns a soundfile name based on the filename of this, by appending ".wav" to the filename, which is the default, or a non-default ouput name which need not be a file but must be set using setOutputSoundfileName().

References csound::Composition::master_filepath, and csound::Composition::output_filename.

Referenced by csound::MusicModel::getCsoundCommand(), csound::Composition::normalizeOutputSoundfile(), csound::MusicModel::perform(), csound::MusicModel::processArgs(), csound::Composition::translateMaster(), and csound::Composition::translateToCdAudio().

6.67.3.32 getPerformanceRightsOrganization()

```
std::string csound::Composition::getPerformanceRightsOrganization ( ) const  [virtual], [inherited]
```

References csound::Composition::performance_rights_organization.

6.67.3.33 getScore()

```
Score & csound::Composition::getScore ( )  [virtual], [inherited]
```

Return the self-contained Score.

References csound::Composition::score.

Referenced by csound::MusicModel::csoundArgv(), and csound::MusicModel::processArgs().

6.67.3.34 getThis()

```
intptr_t csound::ScoreModel::getThis ( )  [virtual]
```

Returns the address of this as a long integer.

Reimplemented in csound::MusicModel.

References csound::fundamentalDomainByPredicate().

6.67.3.35 getThisNode()

```
Node * csound::ScoreModel::getThisNode ( )  [virtual]
```

Returns the address of this as a Node pointer.

Reimplemented in csound::MusicModel.

6.67.3.36 getTieOverlappingNotes()

```
bool csound::Composition::getTieOverlappingNotes ( ) const  [virtual], [inherited]
```

Returns whether or not overlapping notes in generated scores are replaced by one note.

References csound::Composition::tieOverlappingNotes.

Referenced by generate().

6.67.3.37 getTimestamp()

```
std::string csound::Composition::getTimestamp ( ) const  [virtual], [inherited]
```

Returns the time the score was generated.

References csound::Composition::timestamp.

Referenced by csound::Composition::tagFile().

6.67.3.38 getTitle()

```
std::string csound::Composition::getTitle ( ) const  [virtual], [inherited]
```

References csound::Composition::stem.

Referenced by csound::Composition::generateAllNames(), csound::Composition::tagFile(), csound::Composition::translateToMp3(), csound::Composition::translateToMp4(), and csound::Composition::translateToNotation().

6.67.3.39 getTonesPerOctave()

```
double csound::Composition::getTonesPerOctave ( ) const  [virtual], [inherited]
```

Returns the number of equally tempered intervals per octave (the default is 12, 0 means non-equally tempered).

References csound::Composition::tonesPerOctave.

Referenced by generate().

6.67.3.40 getYear()

```
std::string csound::Composition::getYear ( ) const  [virtual], [inherited]
```

References csound::Composition::year.

6.67.3.41 initialize()

```
void csound::ScoreModel::initialize ( )  [virtual]
```

Reimplemented in csound::MusicModel.

6.67.3.42 makeTimestamp()

```
std::string csound::Composition::makeTimestamp ( )  [static], [inherited]
```

Returns the current locale time as a string.

References csound::fundamentalDomainByPredicate().

Referenced by csound::Composition::generateAllNames(), and csound::Composition::generateFilename().

6.67.3.43 normalizeOutputSoundfile()

```
int csound::Composition::normalizeOutputSoundfile (
            double levelDb = -3.0 )  [virtual], [inherited]
```

Assuming the score has been rendered, uses sox to translate the output soundfile to a normalized soundfile.

References csound::fundamentalDomainByPredicate(), csound::Composition::getNormalizedSoundfileFilepath(), csound::Composition::getOutputSoundfileFilepath(), csound::System::inform(), and csound::Composition::tagFile().

Referenced by csound::Composition::translateMaster().

6.67.3.44 perform()

```
int csound::Composition::perform ( )  [virtual], [inherited]
```

Performs the current score to create an output soundfile, which should be tagged with author, timestamp, copyright, title, and optionally album.

The default implementation does nothing. Must be overridden in derived classes.

Reimplemented in csound::MusicModel.

Referenced by csound::Composition::performMaster(), and csound::Composition::render().

6.67.3.45 performAll()

```
int csound::Composition::performAll ( )  [virtual], [inherited]
```

Convenience function that calls performMaster(), and translateMaster().

References csound::fundamentalDomainByPredicate(), csound::System::inform(), csound::System::message(), csound::Composition::performMaster(), csound::System::startTiming(), csound::System::stopTiming(), and csound::Composition::translate

Referenced by csound::Composition::renderAll().

6.67.3.46 performMaster()

```
int csound::Composition::performMaster ( )  [virtual], [inherited]
```

Convenience function that calls saveMidi(), saveMusicXML(), and perform().

References csound::fundamentalDomainByPredicate(), csound::System::inform(), and csound::Composition::perform().

Referenced by csound::Composition::performAll().

6.67.3.47 processArgs()

```
int csound::Composition::processArgs (
            const std::vector< std::string > & args )  [virtual], [inherited]
```

Pass the invoking program's command-line arguments to processArgs() and it will perform with possibly back-end-dependent options.

Additional arguments can be added to the args before the call. Default implementation calls renderAll().

Reimplemented in csound::MusicModel.

References csound::Composition::generateAllNames(), and csound::Composition::renderAll().

Referenced by csound::Composition::processArgv().

6.67.3.48 processArgv()

```
int csound::Composition::processArgv (
            int argc,
            const char ** argv )  [virtual], [inherited]
```

Pass the invoking program's command-line arguments to processArgs() and it will perform with possibly back-end-dependent options.

Default implementation calls the std::string overload.

References csound::fundamentalDomainByPredicate(), and csound::Composition::processArgs().

Referenced by main().

6.67.3.49 render()

```
int csound::Composition::render ( )  [virtual], [inherited]
```

Convenience function that calls clear(), generate(), perform().

timestamp = makeTimestamp();

Reimplemented in csound::MusicModel.

References csound::Composition::clear(), csound::fundamentalDomainByPredicate(), csound::Composition::generate(), and csound::Composition::perform().

6.67.3.50 renderAll()

```
int csound::Composition::renderAll ( )  [virtual], [inherited]
```

Convenience function that calls clear(), generate(), performAll().

References csound::Composition::clear(), csound::fundamentalDomainByPredicate(), csound::Composition::generate(), and csound::Composition::performAll().

Referenced by csound::Composition::processArgs().

6.67.3.51 setAlbum()

```
void csound::Composition::setAlbum (
            std::string value )  [virtual], [inherited]
```

References csound::Composition::album.

Referenced by main().

6.67.3.52 setArtist()

```
void csound::Composition::setArtist (
            std::string value )  [virtual], [inherited]
```

References csound::Composition::artist.

6.67.3.53 setAuthor()

```
void csound::Composition::setAuthor (
            std::string value )  [virtual], [inherited]
```

References csound::Composition::author.

Referenced by main().

6.67.3.54 setConformPitches()

```
void csound::Composition::setConformPitches (
            bool conformPitches )  [virtual], [inherited]
```

Sets whether or not the pitches in generated scores will be conformed to the nearest equally tempered pitch.

References csound::Composition::conformPitches.

6.67.3.55 setCopyright()

```
void csound::Composition::setCopyright (
            std::string value )  [virtual], [inherited]
```

References csound::Composition::copyright.

6.67.3.56 setDuration()

```
void csound::Composition::setDuration (
            double seconds )  [virtual], [inherited]
```

At the end of processing, if the defined duration is not zero, the times and durations of all events are rescaled to the defined duration.

References csound::Composition::duration, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.67.3.57 setElement()

```
virtual void csound::ScoreModel::setElement (
            size_t row,
            size_t column,
            double value )  [inline], [virtual]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented from csound::Node.

6.67.3.58 setFilename()

```
void csound::Composition::setFilename (
            std::string filename )  [virtual], [inherited]
```

Sets the filename of this – basically, the title of the composition.

References csound::Composition::stem.

6.67.3.59 setLicense()

```
void csound::Composition::setLicense (
            std::string value )  [virtual], [inherited]
```

References csound::Composition::license.

6.67.3.60 setOutputDirectory()

```
void csound::Composition::setOutputDirectory (
            std::string directory )  [virtual], [inherited]
```

Sets the directory in which to place the output files of this.

The directory name must end with a directory separator.

References csound::fundamentalDomainByPredicate(), and csound::Composition::output_directory.

Referenced by csound::MusicModel::processArgs().

6.67.3.61 setOutputSoundfileName()

```
void csound::Composition::setOutputSoundfileName (
            std::string name )  [virtual], [inherited]
```

Sets a non-default output name (could be an audio device not a file).

References csound::Composition::output_filename.

6.67.3.62 setPerformanceRightsOrganization()

```
void csound::Composition::setPerformanceRightsOrganization (
            std::string value )  [virtual], [inherited]
```

References csound::Composition::performance_rights_organization.

Referenced by main().

6.67.3.63 setScore()

```
void csound::Composition::setScore (
            Score & score )  [virtual], [inherited]
```

Sets the score in this to the indicated score.

References csound::fundamentalDomainByPredicate(), and csound::Composition::score.

6.67.3.64 setTieOverlappingNotes()

```
void csound::Composition::setTieOverlappingNotes (
            bool tieOverlappingNotes )  [virtual], [inherited]
```

Sets whether or not overlapping notes in generated scores are replaced by one note.

References csound::fundamentalDomainByPredicate(), and csound::Composition::tieOverlappingNotes.

Referenced by main().

6.67.3.65 setTitle()

```
void csound::Composition::setTitle (
            std::string value )  [virtual], [inherited]
```

References csound::Composition::stem.

Referenced by main().

6.67.3.66 setTonesPerOctave()

```
void csound::Composition::setTonesPerOctave (
            double tonesPerOctave )  [virtual], [inherited]
```

Sets the number of equally tempered intervals per octave (the default is 12, 0 means non-equally tempered).

References csound::Composition::tonesPerOctave.

6.67.3.67 setYear()

```
void csound::Composition::setYear (
            std::string value )  [virtual], [inherited]
```

References csound::Composition::year.

Referenced by main().

6.67.3.68 tagFile()

```
int csound::Composition::tagFile (
            std::string filename ) const  [virtual], [inherited]
```

References csound::fundamentalDomainByPredicate(), csound::Composition::getAlbum(), csound::Composition::getArtist(), csound::Composition::getAuthor(), csound::Composition::getCopyright(), csound::Composition::getLicense(), csound::Composition::getTi csound::Composition::getTitle(), and csound::System::inform().

Referenced by csound::Composition::normalizeOutputSoundfile(), csound::Composition::translateMaster(), and csound::Composition::translateToCdAudio().

6.67.3.69 transform()

```
virtual void csound::ScoreModel::transform (
            Score & score_from_children )  [inline], [virtual]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented from csound::Node.

6.67.3.70 translateMaster()

```
int csound::Composition::translateMaster ( )  [virtual], [inherited]
```

Convenience function that calls rescaleOutputSoundfile(), translateToCdAudio(), and translateToMp3().

References csound::fundamentalDomainByPredicate(), csound::Composition::getOutputSoundfileFilepath(), csound::System::inform(), csound::System::message(), csound::Composition::normalizeOutputSoundfile(), csound::System::startTiming(), csound::System::stopTiming(), csound::Composition::tagFile(), csound::Composition::translateToCdAudio(), csound::Composition::transla and csound::Composition::translateToMp4().

Referenced by csound::Composition::performAll(), and csound::MusicModel::processArgs().

6.67.3.71 translateToCdAudio()

```
int csound::Composition::translateToCdAudio (
            double levelDb = -3.0 )  [virtual], [inherited]
```

Assuming the score has been rendered, uses sox to translate the output soundfile to normalized CD-audio format.

References csound::fundamentalDomainByPredicate(), csound::Composition::getCdSoundfileFilepath(), csound::Composition::getOutput csound::System::inform(), and csound::Composition::tagFile().

Referenced by csound::Composition::translateMaster().

6.67.3.72 translateToMp3()

```
int csound::Composition::translateToMp3 (
            double bitrate = 256.01,
            double levelDb = -3.0 )  [virtual], [inherited]
```

Assuming the score has been rendered, uses sox and LAME to translate the output soundfile to normalized MP3 format.

References csound::Composition::author, csound::fundamentalDomainByPredicate(), csound::Composition::getAlbum(), csound::Composition::getAuthor(), csound::Composition::getCdSoundfileFilepath(), csound::Composition::getCopyright(), csound::Composition::getMp3SoundfileFilepath(), csound::Composition::getTitle(), csound::System::inform(), and csound::Composition::year.

Referenced by csound::Composition::translateMaster().

6.67.3.73 translateToMp4()

```
int csound::Composition::translateToMp4 ( )  [virtual], [inherited]
```

Assuming the score has been rendered, uses sox and ffmpeg to translate the output soundfile to a normalized mp4 video suitable for uploading to YouTube.

References csound::Composition::album, csound::Composition::artist, csound::Composition::author, csound::Composition::cd_quality_file csound::Composition::copyright, csound::fundamentalDomainByPredicate(), csound::Composition::getCopyright(), csound::Composition::getTitle(), csound::System::inform(), csound::Composition::master_filepath, csound::Composition::mp4_filepath, csound::Composition::notes, csound::Composition::performance_rights_organization, csound::Composition::spectrogram_filepath, csound::Composition::stem, csound::Composition::track, and csound::Composition::year.

Referenced by csound::Composition::translateMaster().

6.67.3.74 translateToNotation()

```
int csound::Composition::translateToNotation (
            const std::vector< std::string > partNames = std::vector<std::string>(),
            std::string header = "" )  [virtual], [inherited]
```

Saves the generated score in Fomus format and uses Fomus and Lilypond to translate that to a PDF of music notation.

A meter of 4/4 and a tempo of MM 120 is assumed. A vector of part names may be supplied.

References csound::Composition::duration, csound::fundamentalDomainByPredicate(), csound::Composition::getArtist(), csound::Score::getDuration(), csound::Composition::getFomusfileFilepath(), csound::Composition::getTitle(), csound::iterator(), csound::Conversions::round(), csound::Composition::score, and csound::Score::sort().

Referenced by csound::MusicModel::processArgs().

6.67.3.75 traverse()

```
virtual void csound::ScoreModel::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [inline], [virtual]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented from csound::Node.

Referenced by csound::MusicModel::generate(), and generate().

6.67.3.76 write()

```
void csound::Composition::write (
            const char * text )  [virtual], [inherited]
```

Write as if to stderr.

References csound::fundamentalDomainByPredicate(), and csound::System::message().

6.67.4 Field Documentation

6.67.4.1 album

```
std::string csound::Composition::album  [protected], [inherited]
```

Optional metadata.

Referenced by csound::Composition::generateAllNames(), csound::Composition::getAlbum(), csound::Composition::setAlbum(), and csound::Composition::translateToMp4().

6.67.4.2 artist

```
std::string csound::Composition::artist  [protected], [inherited]
```

Required metadata.

Allows for performer, etc. to differ from author. Defaults to author.

Referenced by csound::Composition::generateAllNames(), csound::Composition::getArtist(), csound::Composition::setArtist(), and csound::Composition::translateToMp4().

6.67.4.3 author

```
std::string csound::Composition::author  [protected], [inherited]
```

Required metadata.

Referenced by csound::Composition::generateAllNames(), csound::Composition::getAuthor(), csound::Composition::setAuthor(), csound::Composition::translateToMp3(), and csound::Composition::translateToMp4().

6.67.4.4 base_filepath

```
std::string csound::Composition::base_filepath  [protected], [inherited]
```

Generated.

The dirname and stem of the output files.

Referenced by csound::Composition::generateAllNames(), csound::Composition::getBasename(), csound::Composition::getFileFilepath(), and csound::Composition::getMusicXmlfileFilepath().

6.67.4.5 baseScore

```
Score csound::Composition::baseScore  [protected], [inherited]
```

6.67.4.6 bext_description

```
std::string csound::Composition::bext_description  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames().

6.67.4.7 bext_orig_ref

```
std::string csound::Composition::bext_orig_ref  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames().

6.67.4.8 bext_originator

```
std::string csound::Composition::bext_originator  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames().

6.67.4.9 cd_quality_filepath

```
std::string csound::Composition::cd_quality_filepath  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames(), csound::Composition::getCdSoundfileFilepath(), and csound::Composition::translateToMp4().

6.67.4.10 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.67.4.11 conformPitches

```
bool csound::Composition::conformPitches  [protected], [inherited]
```

Referenced by csound::MusicModel::createCsoundScore(), csound::Composition::getConformPitches(), and csound::Composition::setConformPitches().

6.67.4.12 copyright

```
std::string csound::Composition::copyright  [protected], [inherited]
```

Required metadata.

Referenced by csound::Composition::generateAllNames(), csound::Composition::getCopyright(), csound::Composition::setCopyright(), and csound::Composition::translateToMp4().

6.67.4.13 duration

```
double csound::Composition::duration  [protected], [inherited]
```

Referenced by csound::MusicModel::generate(), generate(), csound::Composition::getDuration(), csound::Composition::setDuration(), and csound::Composition::translateToNotation().

6.67.4.14 flac_filepath

```
std::string csound::Composition::flac_filepath  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames().

6.67.4.15 label

```
std::string csound::Composition::label  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames().

6.67.4.16 license

```
std::string csound::Composition::license  [protected], [inherited]
```

Required metadata.

Defaults to Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International.

Referenced by csound::Composition::getLicense(), and csound::Composition::setLicense().

6.67.4.17 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.67.4.18 master_filepath

```
std::string csound::Composition::master_filepath  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames(), csound::Composition::getOutputSoundfileFilepath(), and csound::Composition::translateToMp4().

6.67.4.19 midi_filepath

```
std::string csound::Composition::midi_filepath  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames(), and csound::Composition::getMidifileFilepath().

6.67.4.20 mp3_filepath

```
std::string csound::Composition::mp3_filepath  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames(), and csound::Composition::getMp3SoundfileFilepath().

6.67.4.21 mp4_filepath

```
std::string csound::Composition::mp4_filepath  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames(), and csound::Composition::translateToMp4().

6.67.4.22 normalized_master_filepath

```
std::string csound::Composition::normalized_master_filepath  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames(), and csound::Composition::getNormalizedSoundfileFilepath().

6.67.4.23 notes

```
std::string csound::Composition::notes  [protected], [inherited]
```

Optional metadata, defaults to "Electroacoustic Music.".

Referenced by csound::Composition::generateAllNames(), and csound::Composition::translateToMp4().

6.67.4.24 output_directory

```
std::string csound::Composition::output_directory  [protected], [inherited]
```

Required.

The target directory of the output files. Defaults to the current working directory.

Referenced by csound::Composition::generateAllNames(), csound::Composition::getOutputDirectory(), and csound::Composition::setOutp

6.67.4.25 output_filename

```
std::string csound::Composition::output_filename  [protected], [inherited]
```

Referenced by csound::Composition::clearOutputSoundfileName(), csound::Composition::getOutputSoundfileFilepath(), and csound::Composition::setOutputSoundfileName().

6.67.4.26 performance_rights_organization

```
std::string csound::Composition::performance_rights_organization  [protected], [inherited]
```

Optional metadata.

Referenced by csound::Composition::generateAllNames(), csound::Composition::getPerformanceRightsOrganization(), csound::Composition::setPerformanceRightsOrganization(), and csound::Composition::translateToMp4().

6.67.4.27 score

```
Score& csound::Composition::score  [protected], [inherited]
```

Referenced by csound::MusicModel::arrange(), csound::MusicModel::arrange(), csound::MusicModel::arrange(), csound::Composition::clear(), csound::MusicModel::createCsoundScore(), csound::MusicModel::generate(), generate(), csound::Composition::getScore(), csound::MusicModel::removeArrangement(), csound::Composition::setScore(), and csound::Composition::translateToNotation().

6.67.4.28 spectrogram_filepath

```
std::string csound::Composition::spectrogram_filepath  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames(), and csound::Composition::translateToMp4().

6.67.4.29 stem

```
std::string csound::Composition::stem  [protected], [inherited]
```

Required.

The stem must be a valid filename and also represents the title. All other names, text, and commands are generated from directory, stem, filename extensions, and required metadata.

Referenced by csound::Composition::generateAllNames(), csound::Composition::getFilename(), csound::Composition::getTitle(), csound::Composition::setFilename(), csound::Composition::setTitle(), and csound::Composition::translateToMp4().

6.67.4.30 tieOverlappingNotes

```
bool csound::Composition::tieOverlappingNotes  [protected], [inherited]
```

Referenced by csound::Composition::getTieOverlappingNotes(), and csound::Composition::setTieOverlappingNotes().

6.67.4.31 timestamp

```
std::string csound::Composition::timestamp  [protected], [inherited]
```

Generated.

Referenced by csound::Composition::generateAllNames(), and csound::Composition::getTimestamp().

6.67.4.32 tonesPerOctave

```
double csound::Composition::tonesPerOctave  [protected], [inherited]
```

Referenced by csound::MusicModel::createCsoundScore(), csound::Composition::getTonesPerOctave(), and csound::Composition::setTonesPerOctave().

6.67.4.33 track

```
std::string csound::Composition::track  [protected], [inherited]
```

Optional metadata.

Referenced by csound::Composition::generateAllNames(), and csound::Composition::translateToMp4().

6.67.4.34 year

```
std::string csound::Composition::year  [protected], [inherited]
```

Required metadata.

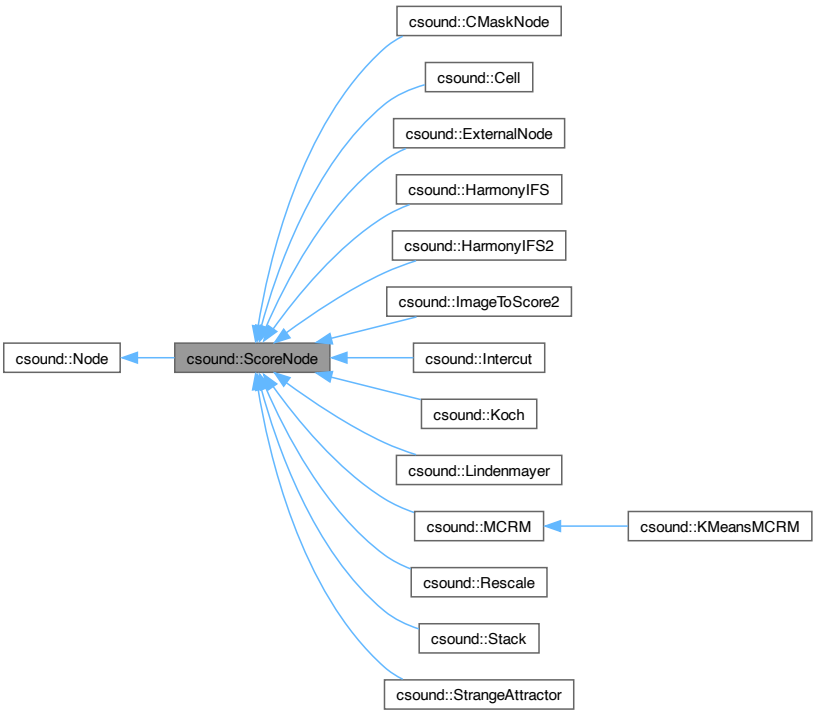
Referenced by csound::Composition::generateAllNames(), csound::Composition::getYear(), csound::Composition::setYear(), csound::Composition::translateToMp3(), and csound::Composition::translateToMp4().

6.68 csound::ScoreNode Class Reference

Node class that produces events from the contained score, which can be built up programmatically or imported from a standard MIDI file.

```
#include <ScoreNode.hpp>
```

Inheritance diagram for csound::ScoreNode:



Public Member Functions

- virtual void addChild (Node ∗node)

  Adds an immediate child Node to this.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &collectingScore)

  Optionally generate notes into the score.
- virtual Node ∗ getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual Score & getScore ()
- ScoreNode ()
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void transform (Score &score_from_children)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.
- virtual ∼ScoreNode ()

Data Fields

- std::vector< Node ∗ > children

  Child Nodes, if any.
- double duration

  If not 0, the score is rescaled to this duration.
- std::string importFilename

Protected Attributes

- Eigen::MatrixXd localCoordinates
- Score score

6.68.1 Detailed Description

Node class that produces events from the contained score, which can be built up programmatically or imported from a standard MIDI file.

6.68.2 Constructor & Destructor Documentation

6.68.2.1 ScoreNode()

```
csound::ScoreNode::ScoreNode ( )
```

6.68.2.2 ~ScoreNode()

```
csound::ScoreNode::~ScoreNode ( )  [virtual]
```

6.68.3 Member Function Documentation

6.68.3.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.68.3.2 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.68.3.3 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.68.3.4 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.68.3.5 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.68.3.6 generate()

```
void csound::ScoreNode::generate (
            Score & score_from_this )  [virtual]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented from csound::Node.

Reimplemented in csound::ExternalNode, and csound::MCRM.

References duration, csound::fundamentalDomainByPredicate(), csound::Score::getCsoundScoreHeader(), importFilename, csound::Score::load(), csound::Score::process(), score, csound::Score::setDuration(), and csound::Score::sort().

Referenced by csound::MCRM::generate().

6.68.3.7 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.68.3.8 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.68.3.9 getScore()

```
Score & csound::ScoreNode::getScore ( )  [virtual]
```

References score.

Referenced by main().

6.68.3.10 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.68.3.11 transform()

```
void csound::Node::transform (
            Score & score_from_children )  [virtual], [inherited]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented in csound::Cell, csound::CellRepeat, csound::CellAdd, csound::CellMultiply, csound::CellReflect, csound::CellSelect, csound::CellRemove, csound::CellChord, csound::CellRandom, csound::CellShuffle, csound::CounterpointNode, csound::RemoveDuplicates, csound::Transformer, csound::Random, csound::Rescale, csound::VoiceleadingNode, csound::LispTransformer, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.68.3.12 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord and csound::Node::transform().

6.68.4 Field Documentation

6.68.4.1 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.68.4.2 duration

```
double csound::ScoreNode::duration
```

If not 0, the score is rescaled to this duration.

Referenced by generate(), csound::ExternalNode::generateLocally(), and csound::Stack::getDuration().

6.68.4.3 importFilename

```
std::string csound::ScoreNode::importFilename
```

Referenced by generate().

6.68.4.4 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.68.4.5 score

```
Score csound::ScoreNode::score  [protected]
```

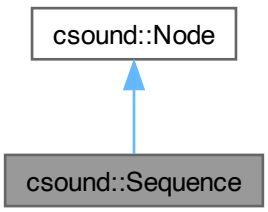
Referenced by csound::StrangeAttractor::evaluateAttractor(), csound::ExternalNode::generate(), generate(), csound::MCRM::generate(), csound::ExternalNode::generateLocally(), csound::ImageToScore2::generateLocally(), csound::Lindenmayer::generateLocally(), csound::Rescale::getRescale(), getScore(), csound::Lindenmayer::interpret(), csound::MCRM::iterate(), csound::StrangeAttractor::iterate_ csound::KMeansMCRM::means_to_notes(), csound::ImageToScore2::pixel_to_event(), csound::StrangeAttractor::render(), csound::Rescale::Rescale(), csound::Rescale::setRescale(), csound::Cell::transform(), csound::Rescale::transform(), csound::CMaskNode::translate_to_silence(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Lindenmayer::updateActual().

6.69 csound::Sequence Class Reference

Node that creates a temporal sequence of child nodes.

```
#include <Sequence.hpp>
```

Inheritance diagram for csound::Sequence:



Public Member Functions

- virtual void addChild (Node ∗node)

  Adds an immediate child Node to this.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &score_from_this)

  Optionally generate notes into the score.
- virtual Node ∗ getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- Sequence ()
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void transform (Score &score_from_children)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &globalCoordinates, Score &score)

  The notes produced by the first child node have a total duration.
- virtual ∼Sequence ()

Data Fields

- std::vector< Node ∗ > children

  Child Nodes, if any.

Protected Attributes

- Eigen::MatrixXd localCoordinates

6.69.1 Detailed Description

Node that creates a temporal sequence of child nodes.

6.69.2 Constructor & Destructor Documentation

6.69.2.1 Sequence()

```
csound::Sequence::Sequence ( )
```

6.69.2.2 ~Sequence()

```
csound::Sequence::~Sequence ( )  [virtual]
```

6.69.3 Member Function Documentation

6.69.3.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.69.3.2 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.69.3.3 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.69.3.4 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.69.3.5 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.69.3.6 generate()

```
void csound::Node::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented in csound::ExternalNode, csound::ScoreNode, csound::ChordLindenmayer, csound::MCRM, csound::Generator, csound::Random, csound::LispGenerator, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.69.3.7 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.69.3.8 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and traverse().

6.69.3.9 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.69.3.10 transform()

```
void csound::Node::transform (
            Score & score_from_children )  [virtual], [inherited]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented in csound::Cell, csound::CellRepeat, csound::CellAdd, csound::CellMultiply, csound::CellReflect, csound::CellSelect, csound::CellRemove, csound::CellChord, csound::CellRandom, csound::CellShuffle, csound::CounterpointNode, csound::RemoveDuplicates, csound::Transformer, csound::Random, csound::Rescale, csound::VoiceleadingNode, csound::LispTransformer, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.69.3.11 traverse()

```
void csound::Sequence::traverse (
            const Eigen::MatrixXd & globalCoordinates,
            Score & score )  [virtual]
```

The notes produced by the first child node have a total duration.

The notes produced by the second child node are shifted forward in time by that duration, and so on, to create a strict temporal sequence of child nodes.

Reimplemented from csound::Node.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::getLocalCoordinates(), csound::System::message(), and csound::Event::setTime().

6.69.4 Field Documentation

6.69.4.1 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and traverse().

6.69.4.2 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.70 csound::Shell Class Reference

Provide a shell in which Python scripts can be loaded, saved, and executed.

```
#include <Shell.hpp>
```

Public Member Functions

- virtual void clear ()
- virtual void close ()
- virtual std::string getFilename () const
- virtual std::string getMidiFilename () const
- virtual std::string getOutputSoundfileName () const
- virtual std::string getScript () const
- virtual void initialize ()
- virtual void load (std::string filename)
- virtual void loadAppend (std::string filename)
- virtual void main (int argc, char **argv)
- virtual void open ()
- virtual int runScript ()
- virtual int runScript (std::string script)
- virtual void save () const
- virtual void save (std::string filename) const
- virtual void setFilename (std::string filename)
- virtual void setScript (std::string text)
- Shell ()
- virtual void stop ()
- virtual ~Shell ()

Static Public Member Functions

- static std::string generateFilename ()

Protected Attributes

- std::string filename
- std::string script

Static Protected Attributes

- static void * pythonLibrary = 0
- static const char * pythonLibraryPathList [ ]

6.70.1 Detailed Description

Provide a shell in which Python scripts can be loaded, saved, and executed.

The Python library and API are dynamically loaded and do not reference Python.h, so if Python is not present, this module will still link and load, but not function.

6.70.2 Constructor & Destructor Documentation

6.70.2.1 Shell()

```
csound::Shell::Shell ( )
```

6.70.2.2 ~Shell()

```
csound::Shell::~Shell ( )  [virtual]
```

6.70.3 Member Function Documentation

6.70.3.1 clear()

```
void csound::Shell::clear ( )  [virtual]
```

References filename, and script.

Referenced by initialize(), and load().

6.70.3.2 close()

```
void csound::Shell::close ( )  [virtual]
```

6.70.3.3 generateFilename()

```
std::string csound::Shell::generateFilename ( )  [static]
```

References csound::fundamentalDomainByPredicate().

Referenced by initialize().

6.70.3.4 getFilename()

```
std::string csound::Shell::getFilename ( ) const  [virtual]
```

References filename.

Referenced by getMidiFilename(), getOutputSoundfileName(), and save().

6.70.3.5 getMidiFilename()

```
std::string csound::Shell::getMidiFilename ( ) const  [virtual]
```

References filename, csound::fundamentalDomainByPredicate(), and getFilename().

6.70.3.6 getOutputSoundfileName()

```
std::string csound::Shell::getOutputSoundfileName ( ) const  [virtual]
```

References filename, csound::fundamentalDomainByPredicate(), and getFilename().

6.70.3.7 getScript()

```
std::string csound::Shell::getScript ( ) const  [virtual]
```

References script.

6.70.3.8 initialize()

```
void csound::Shell::initialize ( )  [virtual]
```

References clear(), generateFilename(), and setFilename().

6.70.3.9 load()

```
void csound::Shell::load (
            std::string filename )  [virtual]
```

References clear(), filename, and loadAppend().

6.70.3.10 loadAppend()

```
void csound::Shell::loadAppend (
            std::string filename )  [virtual]
```

References filename, csound::fundamentalDomainByPredicate(), and script.

Referenced by load().

6.70.3.11 main()

```
void csound::Shell::main (
            int argc,
            char ** argv )  [virtual]
```

References csound::fundamentalDomainByPredicate(), csound::PyRun_SimpleString_, and csound::PySys_SetArgv_.

6.70.3.12 open()

```
void csound::Shell::open ( )  [virtual]
```

References csound::fundamentalDomainByPredicate(), csound::Py_Finalize_, csound::Py_Initialize_, csound::PyErr_Print_, csound::PyImport_ImportModule_, csound::PyLong_AsLong_, csound::PyObject_CallMethod_, csound::PyObject_GetAttrString_, csound::PyRun_SimpleFileEx_, csound::PyRun_SimpleString_, csound::PySys_SetArgv_, csound::pythonFuncWarning(), pythonLibrary, pythonLibraryPathList, and csound::System::warn().

6.70.3.13 runScript() [1/2]

```
int csound::Shell::runScript ( )  [virtual]
```

References runScript(), and script.

Referenced by runScript().

6.70.3.14 runScript() [2/2]

```
int csound::Shell::runScript (
            std::string script )  [virtual]
```

References csound::System::error(), csound::fundamentalDomainByPredicate(), csound::System::message(), csound::PyErr_Print_, and csound::PyRun_SimpleString_.

6.70.3.15 save() [1/2]

```
void csound::Shell::save ( ) const  [virtual]
```

References getFilename(), and save().

Referenced by save().

6.70.3.16 save() [2/2]

```
void csound::Shell::save (
            std::string filename ) const  [virtual]
```

References filename, csound::fundamentalDomainByPredicate(), and script.

6.70.3.17 setFilename()

```
void csound::Shell::setFilename (
            std::string filename )  [virtual]
```

References filename.

Referenced by initialize().

6.70.3.18 setScript()

```
void csound::Shell::setScript (
            std::string text )  [virtual]
```

References script.

6.70.3.19 stop()

```
void csound::Shell::stop ( )  [virtual]
```

6.70.4 Field Documentation

6.70.4.1 filename

```
std::string csound::Shell::filename  [protected]
```

Referenced by clear(), getFilename(), getMidiFilename(), getOutputSoundfileName(), load(), loadAppend(), save(), and setFilename().

6.70.4.2 pythonLibrary

```
void * csound::Shell::pythonLibrary = 0  [static], [protected]
```

Referenced by open().

6.70.4.3 pythonLibraryPathList

```
const char * csound::Shell::pythonLibraryPathList  [static], [protected]
```

Referenced by open().

6.70.4.4 script

```
std::string csound::Shell::script  [protected]
```

Referenced by clear(), getScript(), loadAppend(), runScript(), save(), and setScript().

6.71 csound::Soundfile Class Reference

Simple, basic read/write access, in sample frames, to PCM soundfiles.

```
#include <Soundfile.hpp>
```

Public Member Functions

- virtual void blank (double duration)

  Make the soundfile be so many seconds of silence.
- virtual int close ()

  Close the soundfile.
- virtual void cosineGrain (double centerTimeSeconds, double durationSeconds, double frequencyHz, double amplitude, double phaseOffsetRadians, double pan, bool synchronousPhase=true, bool buffer=false)

  Mix a cosine grain into the soundfile.
- virtual int create (std::string filename, int framesPerSecond=44100, int channelsPerFrame=2, int format=SF_FORMAT_WAV|SF_FO

  Create a new soundfile for writing and/or reading.
- virtual void error () const

  Print to stderr any current error status message.
- virtual int getChannelsPerFrame () const
- virtual int getFormat () const

  See sndfile.h for a descriptive list of format numbers.
- virtual int getFrames () const

  Return the number of sample frames in a just opened file, or just after calling updateHeader.
- virtual int getFramesPerSecond () const
- virtual void jonesParksGrain (double centerTimeSeconds, double durationSeconds, double beginningFrequencyHz, double centerFrequencyHz, double centerAmplitude, double centerPhaseOffsetRadians, double pan, bool synchronousPhase=true, bool buffer=false)

  Mix a Gaussian chirp into the soundfile.
- virtual int mixFrames (double *inputFrames, int samples, double *mixedFrames)

  Mix one or more samples, from a double array (in C++) or a binary string (in Python), into the existing signal in the soundfile.
- virtual void mixGrain ()

  Mix a grain that has already been computed into the soundfile.
- virtual int open (std::string filename)

  Open an existing soundfile for reading and/or writing.
- virtual int readFrame (double *outputFrame)

  Read one sample frame, and return it in a double array (in C++) or a sequence (in Python).
- virtual int readFrames (double *outputFrames, int samples)

  Read one or more samples, and return them in a double array (in C++) or a binary string (in Python).

- virtual int seek (int frames, int whence=0)

  Position the soundfile read/write pointer at the indicated sample frame.
- virtual double seekSeconds (double seconds, int whence=0)
- virtual void setChannelsPerFrame (int channelsPerFrame)
- virtual void setFormat (int format)

  See sndfile.h for a descriptive list of format numbers.
- virtual void setFramesPerSecond (int framesPerSecond)
- Soundfile ()
- virtual void updateHeader ()

  Update the soundfile header with the current file size, RIFF chunks, and so on.
- virtual int writeFrame (double *inputFrame)

  Write one sample frame, from a double array (in C++) or a sequence (in Python).
- virtual int writeFrames (double *inputFrames, int samples)

  Write one or more samples, from a double array (in C++) or a binary string (in Python).
- virtual ~Soundfile ()

Protected Member Functions

- virtual void initialize ()

6.71.1 Detailed Description

Simple, basic read/write access, in sample frames, to PCM soundfiles.

Reads and writes any format, but write defaults to WAV float format. This class is designed for Python wrapping with SWIG. See http://www.mega-nerd.com/libsndfile for more information on the underlying libsndfile library.

6.71.2 Constructor & Destructor Documentation

6.71.2.1 Soundfile()

```
csound::Soundfile::Soundfile ( )
```

References initialize().

6.71.2.2 ~Soundfile()

```
csound::Soundfile::~Soundfile ( )  [virtual]
```

References close().

6.71.3 Member Function Documentation

6.71.3.1 blank()

```
void csound::Soundfile::blank (
            double duration )  [virtual]
```

Make the soundfile be so many seconds of silence.

References csound::fundamentalDomainByPredicate(), getChannelsPerFrame(), getFramesPerSecond(), seekSeconds(), and updateHeader().

6.71.3.2 close()

```
int csound::Soundfile::close ( )  [virtual]
```

Close the soundfile.

Should be called once for every opened or created soundfile, although the class destructor will automatically close an open soundfile.

References csound::fundamentalDomainByPredicate(), and initialize().

Referenced by create(), open(), and ~Soundfile().

6.71.3.3 cosineGrain()

```
void csound::Soundfile::cosineGrain (
            double centerTimeSeconds,
            double durationSeconds,
            double frequencyHz,
            double amplitude,
            double phaseOffsetRadians,
            double pan,
            bool synchronousPhase = true,
            bool buffer = false )  [virtual]
```

Mix a cosine grain into the soundfile.

If the soundfile is stereo, the grain will be panned. If the synchronousPhase argument is true (the default value), then all grains of the same frequency will have synchronous phases, which can be useful in avoiding certain artifacts. For example, if cosine grains of the same frequency have synchronous phases, they can be overlapped by 1/2 their duration without artifacts to produce a continuous cosine tone.

If the buffer argument is true (the default is false), the grain is mixed into a buffer; this can be used to speed up writing grains that are arrangement in columns. To actually write the grain, call writeGrain().

The algorithm uses an efficient difference equation.

References csound::fundamentalDomainByPredicate(), csound::Conversions::get2PI(), getChannelsPerFrame(), getFramesPerSecond(), csound::Conversions::leftPan(), mixGrain(), csound::Conversions::rightPan(), and csound::Conversions::round().

6.71.3.4 create()

```
int csound::Soundfile::create (
            std::string filename,
            int framesPerSecond = 44100,
            int channelsPerFrame = 2,
            int format = SF_FORMAT_WAV | SF_FORMAT_FLOAT )  [virtual]
```

Create a new soundfile for writing and/or reading.

The default soundfile format is WAV PCM float samples at 44100 frames per second, stereo.

References close(), error(), and csound::fundamentalDomainByPredicate().

6.71.3.5 error()

```
void csound::Soundfile::error ( ) const  [virtual]
```

Print to stderr any current error status message.

References csound::fundamentalDomainByPredicate().

Referenced by create(), open(), seek(), and seekSeconds().

6.71.3.6 getChannelsPerFrame()

```
int csound::Soundfile::getChannelsPerFrame ( ) const  [virtual]
```

Referenced by blank(), cosineGrain(), and jonesParksGrain().

6.71.3.7 getFormat()

```
int csound::Soundfile::getFormat ( ) const  [virtual]
```

See sndfile.h for a descriptive list of format numbers.

6.71.3.8 getFrames()

```
int csound::Soundfile::getFrames ( ) const  [virtual]
```

Return the number of sample frames in a just opened file, or just after calling updateHeader.

6.71.3.9 getFramesPerSecond()

```
int csound::Soundfile::getFramesPerSecond ( ) const  [virtual]
```

Referenced by blank(), cosineGrain(), and jonesParksGrain().

6.71.3.10 initialize()

```
void csound::Soundfile::initialize ( )  [protected], [virtual]
```

Referenced by close(), and Soundfile().

6.71.3.11 jonesParksGrain()

```
void csound::Soundfile::jonesParksGrain (
            double centerTimeSeconds,
            double durationSeconds,
            double beginningFrequencyHz,
            double centerFrequencyHz,
            double centerAmplitude,
            double centerPhaseOffsetRadians,
            double pan,
            bool synchronousPhase = true,
            bool buffer = false )  [virtual]
```

Mix a Gaussian chirp into the soundfile.

If the soundfile is stereo, the grain will be panned. If the synchronousPhase argument is true (the default value), then all grains of the same frequency will have synchronous phases, which can be useful in avoiding certain artifacts.

If the buffer argument is true (the default is false), the grain is mixed into a buffer; this can be used to speed up writing grains that are arrangement in columns. To actually write the grain, call writeGrain().

The algorithm uses an efficient difference equation.

References csound::fundamentalDomainByPredicate(), csound::Conversions::get2PI(), getChannelsPerFrame(), getFramesPerSecond(), csound::Conversions::leftPan(), mixGrain(), and csound::Conversions::rightPan().

6.71.3.12 mixFrames()

```
int csound::Soundfile::mixFrames (
            double * inputFrames,
            int samples,
            double * mixedFrames )  [virtual]
```

Mix one or more samples, from a double array (in C++) or a binary string (in Python), into the existing signal in the soundfile.

The arrays or the strings must contain as many elements as there are samples (channels times frames) Channels are interleaved within frames. For efficiency, there is no checking of bounds or type in Python; the string must contain binary Float64.

References csound::fundamentalDomainByPredicate().

Referenced by mixGrain().

6.71.3.13 mixGrain()

```
void csound::Soundfile::mixGrain ( )  [virtual]
```

Mix a grain that has already been computed into the soundfile.

References mixFrames(), and seekSeconds().

Referenced by cosineGrain(), and jonesParksGrain().

6.71.3.14 open()

```
int csound::Soundfile::open (
            std::string filename )  [virtual]
```

Open an existing soundfile for reading and/or writing.

References close(), error(), and csound::fundamentalDomainByPredicate().

6.71.3.15 readFrame()

```
int csound::Soundfile::readFrame (
            double * outputFrame )  [virtual]
```

Read one sample frame, and return it in a double array (in C++) or a sequence (in Python).

The array or the sequence must already contain as many elements as there are channels. For efficiency, there is no bounds checking.

References csound::fundamentalDomainByPredicate().

6.71.3.16 readFrames()

```
int csound::Soundfile::readFrames (
            double * outputFrames,
            int samples )  [virtual]
```

Read one or more samples, and return them in a double array (in C++) or a binary string (in Python).

The array or the string must already contain as many elements as there are samples (channels times frames). Channels are interleaved within frames. For efficiency, there is no bounds checking; on return the string will contain binary Float64. In Python this function is not thread-safe, as a static buffer is used internally.

References csound::fundamentalDomainByPredicate().

6.71.3.17 seek()

```
int csound::Soundfile::seek (
            int frames,
            int whence = 0 )  [virtual]
```

Position the soundfile read/write pointer at the indicated sample frame.

Set whence to 0 for SEEK_SET, 1 for SEEK_CUR, 2 for SEEK_END. Calling with whence = SEEK_CUR and frames = 0 returns the current read/write pointer.

References error(), and csound::fundamentalDomainByPredicate().

6.71.3.18 seekSeconds()

```
double csound::Soundfile::seekSeconds (
            double seconds,
            int whence = 0 )  [virtual]
```

References error(), and csound::fundamentalDomainByPredicate().

Referenced by blank(), and mixGrain().

6.71.3.19 setChannelsPerFrame()

```
void csound::Soundfile::setChannelsPerFrame (
            int channelsPerFrame )  [virtual]
```

References csound::fundamentalDomainByPredicate().

6.71.3.20 setFormat()

```
void csound::Soundfile::setFormat (
            int format )  [virtual]
```

See sndfile.h for a descriptive list of format numbers.

References csound::fundamentalDomainByPredicate().

6.71.3.21 setFramesPerSecond()

```
void csound::Soundfile::setFramesPerSecond (
            int framesPerSecond )  [virtual]
```

References csound::fundamentalDomainByPredicate().

6.71.3.22 updateHeader()

```
void csound::Soundfile::updateHeader ( )  [virtual]
```

Update the soundfile header with the current file size, RIFF chunks, and so on.

References csound::fundamentalDomainByPredicate().

Referenced by blank().

6.71.3.23 writeFrame()

```
int csound::Soundfile::writeFrame (
            double * inputFrame )  [virtual]
```

Write one sample frame, from a double array (in C++) or a sequence (in Python).

The array or the sequence must contain as many elements as there are channels. For efficiency, there is no checking of bounds or type in Python; the string must contain Floats. In Python this function is not thread-safe, as a static buffer is used internally.

References csound::fundamentalDomainByPredicate().

6.71.3.24 writeFrames()

```
int csound::Soundfile::writeFrames (
            double * inputFrames,
            int samples )  [virtual]
```

Write one or more samples, from a double array (in C++) or a binary string (in Python).

The array or the string must contain as many elements as there are samples (channels times frames) Channels are interleaved within frames. For efficiency, there is no checking of bounds or type in Python; the string must contain binary Float64.

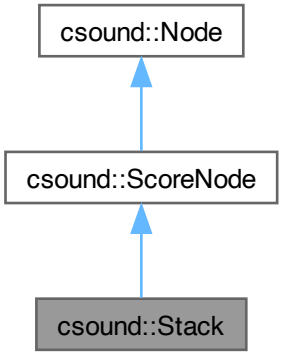
References csound::fundamentalDomainByPredicate().

6.72 csound::Stack Class Reference

The notes produced by each (not all) child node, are rescaled to all start at the same time, and last for the same duration; that of the 0th child, or a specified duration.

`#include <Cell.hpp>`

Inheritance diagram for csound::Stack:



Public Member Functions

- virtual void addChild (Node *node)
  Adds an immediate child Node to this.
- virtual size_t childCount () const
  Returns the number of immediate children of this.
- virtual void clear ()
  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()
  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)
  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &collectingScore)
  Optionally generate notes into the score.
- virtual Node * getChild (size_t index)
  Returns the immediate child of this at the index.
- virtual double getDuration () const
- virtual Eigen::MatrixXd getLocalCoordinates () const
  Returns the local transformation of coordinate system.
- virtual Score & getScore ()
- virtual void setDuration (double value)

- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- Stack ()
- virtual void transform (Score &score_from_children)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &globalCoordinates, Score &collectingScore)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.
- virtual ~Stack ()

Data Fields

- std::vector< Node * > children

  Child Nodes, if any.
- double duration

  If non-zero, then each the notes of each child node in turn are rescaled to fit within this duration; if zero, then the notes of each child node are rescaled to fit within the duration of the first (0th) node.
- std::string importFilename

Protected Attributes

- Eigen::MatrixXd localCoordinates
- Score score

6.72.1 Detailed Description

The notes produced by each (not all) child node, are rescaled to all start at the same time, and last for the same duration; that of the 0th child, or a specified duration.

6.72.2 Constructor & Destructor Documentation

6.72.2.1 Stack()

```
csound::Stack::Stack ( )
```

6.72.2.2 ~Stack()

```
csound::Stack::~Stack ( )  [virtual]
```

6.72.3 Member Function Documentation

6.72.3.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.72.3.2 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.72.3.3 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.72.3.4 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.72.3.5 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.72.3.6 generate()

```
void csound::ScoreNode::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented from csound::Node.

Reimplemented in csound::ExternalNode, and csound::MCRM.

References csound::ScoreNode::duration, csound::fundamentalDomainByPredicate(), csound::Score::getCsoundScoreHeader(), csound::ScoreNode::importFilename, csound::Score::load(), csound::Score::process(), csound::ScoreNode::score, csound::Score::setDuration(), and csound::Score::sort().

Referenced by csound::MCRM::generate().

6.72.3.7 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.72.3.8 getDuration()

```
virtual double csound::Stack::getDuration ( ) const  [inline], [virtual]
```

References csound::ScoreNode::duration.

6.72.3.9 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.72.3.10 getScore()

```
Score & csound::ScoreNode::getScore ( )  [virtual], [inherited]
```

References csound::ScoreNode::score.

Referenced by main().

6.72.3.11 setDuration()

```
virtual void csound::Stack::setDuration (
            double value )  [inline], [virtual]
```

6.72.3.12 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.72.3.13 transform()

```
void csound::Node::transform (
            Score & score_from_children )  [virtual], [inherited]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented in csound::Cell, csound::CellRepeat, csound::CellAdd, csound::CellMultiply, csound::CellReflect, csound::CellSelect, csound::CellRemove, csound::CellChord, csound::CellRandom, csound::CellShuffle, csound::CounterpointNode, csound::RemoveDuplicates, csound::Transformer, csound::Random, csound::Rescale, csound::VoiceleadingNode, csound::LispTransformer, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.72.3.14 traverse()

```
void csound::Stack::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented from csound::Node.

References csound::Score::append(), csound::Node::children, duration, csound::Node::getLocalCoordinates(), csound::System::message(), csound::ScoreNode::score, and csound::Score::setDuration().

6.72.4 Field Documentation

6.72.4.1 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.72.4.2 duration

```
double csound::Stack::duration
```

If non-zero, then each the notes of each child node in turn are rescaled to fit within this duration; if zero, then the notes of each child node are rescaled to fit within the duration of the first (0th) node.

Referenced by traverse().

6.72.4.3 importFilename

```
std::string csound::ScoreNode::importFilename  [inherited]
```

Referenced by csound::ScoreNode::generate().

6.72.4.4 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.72.4.5 score

```
Score csound::ScoreNode::score  [protected], [inherited]
```

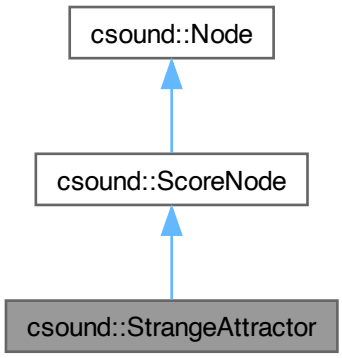
Referenced by csound::StrangeAttractor::evaluateAttractor(), csound::ExternalNode::generate(), csound::ScoreNode::generate(), csound::MCRM::generate(), csound::ExternalNode::generateLocally(), csound::ImageToScore2::generateLocally(), csound::Lindenmayer::generateLocally(), csound::Rescale::getRescale(), csound::ScoreNode::getScore(), csound::Lindenmayer::interpre csound::MCRM::iterate(), csound::StrangeAttractor::iterate_without_rendering(), csound::KMeansMCRM::means_to_notes(), csound::ImageToScore2::pixel_to_event(), csound::StrangeAttractor::render(), csound::Rescale::Rescale(), csound::Rescale::setRescale(csound::Cell::transform(), csound::Rescale::transform(), csound::CMaskNode::translate_to_silence(), csound::Intercut::traverse(), traverse(), csound::Koch::traverse(), and csound::Lindenmayer::updateActual().

6.73 csound::StrangeAttractor Class Reference

Generates notes by searching for a chaotic dynamical system defined by a polynomial equation or partial differential equation using Julien C.

```
#include <StrangeAttractor.hpp>
```

Inheritance diagram for csound::StrangeAttractor:



Public Member Functions

- virtual void addChild (Node *node)

  Adds an immediate child Node to this.
- virtual void calculateFractalDimension ()
- virtual void calculateLyupanovExponent ()
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual void codeRandomize ()
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual bool evaluateAttractor ()
- virtual void generate (Score &collectingScore)

  Optionally generate notes into the score.
- virtual void generateLocally ()
- virtual int getAttractorType () const
- virtual Node * getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual std::string getCode () const
- virtual void getCoefficients ()
- virtual void getDimensionAndOrder ()
- virtual int getDimensionCount () const
- virtual double getFractalDimension () const
- virtual size_t getIteration () const
- virtual size_t getIterationCount () const
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual double getLyupanovExponent () const
- virtual double getNormalizedW () const
- virtual double getNormalizedX () const
- virtual double getNormalizedY () const
- virtual double getNormalizedZ () const
- virtual Score & getScore ()
- virtual int getScoreType () const
- virtual double getW () const
- virtual double getX () const
- virtual double getY () const
- virtual double getZ () const
- virtual void initialize ()
- virtual void iterate ()
- virtual bool iterate_without_rendering ()

  Iterates an already found chaotic system one step, without rendering to the score.
- virtual void reinitialize ()
- virtual void render (int N, double X, double Y, double Z, double W)
- virtual void reset ()

- virtual bool searchForAttractor ()
- virtual void setAttractorType (int attractorType)

  Types: 1 = 1-dimensional polynomial map, 2 = 2-dimensional polynomial map, 3 = 3-dimensional polynomial map, 4 = 4-dimensional polynomial map, 5 = 3-dimensional ODE, 6 = 4-dimensional ODE, 7 through 12 = special functions.
- virtual void setCode (std::string code)
- virtual void setDimensionCount (int D)
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void setIteration (size_t iteration)
- virtual void setIterationCount (size_t iterationCount)
- virtual void setScoreType (int attractorType)
- virtual void setW (double X)
- virtual void setX (double X)
- virtual void setY (double X)
- virtual void setZ (double X)
- virtual void shuffleRandomNumbers ()
- virtual void specialFunctions ()
- StrangeAttractor ()
- virtual void transform (Score &score_from_children)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.
- virtual ~StrangeAttractor ()

Data Fields

- std::vector< Node * > children

  Child Nodes, if any.
- std::string importFilename
- Random randomNode

Protected Attributes

- std::vector< double > A
- double AL
- std::string code
- double COSAL
- int D
- double D2
- double D2MAX
- int DD
- double decibels
- double DF
- double DL2
- double DLW
- double DLX
- double DLY

- double DLZ
- double DUM
- double duration
- double DW
- double DX
- double DY
- double DZ
- double EPS
- double F
- std::string filename
- int I
- int I1
- int I2
- int I3
- int I4
- int I5
- double instrument
- int J
- double L
- Eigen::MatrixXd localCoordinates
- double LSUM
- int M
- double MX
- double MY
- int N
- double N1
- double N2
- double NL
- int NMAX
- int O
- double octave
- int ODE
- int OMAX
- int P
- double pitchClassSet
- int PREV
- double PT
- double RAN
- double RS
- Score score
- int scoreType
- double SH
- double SINAL
- double SW
- int T
- double TIA
- double time
- double TT
- int TWOD
- std::vector< double > V
- double W

- double WE
- double WMAX
- double WMIN
- double WNEW
- double WP
- std::vector< double > WS
- double WSAVE
- double x
- double X
- double XA
- double XE
- double XH
- double XL
- double XMAX
- double XMIN
- std::vector< double > XN
- double XNEW
- double XP
- std::vector< double > XS
- double XSAVE
- double XW
- std::vector< double > XY
- double XZ
- double Y
- double YA
- double YE
- double YH
- double YL
- double YMAX
- double YMIN
- double YNEW
- double YP
- std::vector< double > YS
- double YSAVE
- double YW
- double YZ
- double Z
- double ZA
- double ZE
- double ZMAX
- double ZMIN
- double ZNEW
- double ZP
- std::vector< double > ZS
- double ZSAVE

6.73.1 Detailed Description

Generates notes by searching for a chaotic dynamical system defined by a polynomial equation or partial differential equation using Julien C.

Sprott's Lyupanov exponent search, or by translating a known chaotic dynamical system into music, by interpreting each iteration of the system as a note. The time of the note can be represented either as the order of iteration, or as a dimension of the attractor. See Julien C. Sprott's book "Strange Attractors".

6.73.2 Constructor & Destructor Documentation

6.73.2.1 StrangeAttractor()

```
csound::StrangeAttractor::StrangeAttractor ( )
```

References csound::Random::createDistribution(), initialize(), randomNode, and reset().

6.73.2.2 ~StrangeAttractor()

```
csound::StrangeAttractor::~StrangeAttractor ( )  [virtual]
```

6.73.3 Member Function Documentation

6.73.3.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.73.3.2 calculateFractalDimension()

```
void csound::StrangeAttractor::calculateFractalDimension ( )  [virtual]
```

References D2, D2MAX, DW, DX, DY, DZ, F, csound::fundamentalDomainByPredicate(), J, N, N1, N2, P, randomNode, csound::Random::sample(), TWOD, WMAX, WMIN, WNEW, WS, XMAX, XMIN, XNEW, XS, YMAX, YMIN, YNEW, YS, ZMAX, ZMIN, ZNEW, and ZS.

Referenced by evaluateAttractor(), iterate_without_rendering(), and searchForAttractor().

6.73.3.3 calculateLyupanovExponent()

```
void csound::StrangeAttractor::calculateLyupanovExponent ( )  [virtual]
```

References DF, DL2, DLW, DLX, DLY, DLZ, EPS, csound::fundamentalDomainByPredicate(), iterate(), L, LSUM, N, NL, ODE, RS, W, WE, WNEW, WSAVE, X, XE, XNEW, XSAVE, Y, YE, YNEW, YSAVE, Z, ZE, ZNEW, and ZSAVE.

Referenced by evaluateAttractor(), iterate_without_rendering(), and searchForAttractor().

6.73.3.4 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.73.3.5 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.73.3.6 codeRandomize()

```
void csound::StrangeAttractor::codeRandomize ( )  [virtual]
```

References code, D, csound::System::debug(), csound::fundamentalDomainByPredicate(), getDimensionAndOrder(), I, M, O, ODE, OMAX, randomNode, and csound::Random::sample().

Referenced by reset(), and searchForAttractor().

6.73.3.7 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.73.3.8 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.73.3.9 evaluateAttractor()

```
bool csound::StrangeAttractor::evaluateAttractor ( )  [virtual]
```

References calculateFractalDimension(), calculateLyupanovExponent(), iterate(), N, NMAX, reinitialize(), render(), csound::ScoreNode::score, W, WNEW, X, XNEW, Y, YNEW, Z, and ZNEW.

Referenced by generateLocally().

6.73.3.10 generate()

```
void csound::ScoreNode::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented from csound::Node.

Reimplemented in csound::ExternalNode, and csound::MCRM.

References csound::ScoreNode::duration, csound::fundamentalDomainByPredicate(), csound::Score::getCsoundScoreHeader(), csound::ScoreNode::importFilename, csound::Score::load(), csound::Score::process(), csound::ScoreNode::score, csound::Score::setDuration(), and csound::Score::sort().

Referenced by csound::MCRM::generate().

6.73.3.11 generateLocally()

```
void csound::StrangeAttractor::generateLocally ( )  [virtual]
```

References evaluateAttractor(), and N.

6.73.3.12 getAttractorType()

```
int csound::StrangeAttractor::getAttractorType ( ) const  [virtual]
```

References D, and ODE.

6.73.3.13 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.73.3.14 getCode()

```
std::string csound::StrangeAttractor::getCode ( ) const  [virtual]
```

References code.

6.73.3.15 getCoefficients()

```
void csound::StrangeAttractor::getCoefficients ( )  [virtual]
```

References A, code, getDimensionAndOrder(), I, and M.

Referenced by reinitialize().

6.73.3.16 getDimensionAndOrder()

```
void csound::StrangeAttractor::getDimensionAndOrder ( )  [virtual]
```

References code, D, csound::fundamentalDomainByPredicate(), I, M, O, ODE, and specialFunctions().

Referenced by codeRandomize(), and getCoefficients().

6.73.3.17 getDimensionCount()

```
int csound::StrangeAttractor::getDimensionCount ( ) const  [virtual]
```

References D.

6.73.3.18 getFractalDimension()

```
double csound::StrangeAttractor::getFractalDimension ( ) const  [virtual]
```

References F.

6.73.3.19 getIteration()

```
size_t csound::StrangeAttractor::getIteration ( ) const  [virtual]
```

References N.

6.73.3.20 getIterationCount()

```
size_t csound::StrangeAttractor::getIterationCount ( ) const  [virtual]
```

References NMAX.

6.73.3.21 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.73.3.22 getLyupanovExponent()

```
double csound::StrangeAttractor::getLyupanovExponent ( ) const  [virtual]
```

References L.

6.73.3.23 getNormalizedW()

```
double csound::StrangeAttractor::getNormalizedW ( ) const  [virtual]
```

References csound::fundamentalDomainByPredicate(), W, WMAX, and WMIN.

6.73.3.24 getNormalizedX()

```
double csound::StrangeAttractor::getNormalizedX ( ) const  [virtual]
```

References csound::fundamentalDomainByPredicate(), X, XMAX, and XMIN.

6.73.3.25 getNormalizedY()

```
double csound::StrangeAttractor::getNormalizedY ( ) const  [virtual]
```

References csound::fundamentalDomainByPredicate(), Y, YMAX, and YMIN.

6.73.3.26 getNormalizedZ()

```
double csound::StrangeAttractor::getNormalizedZ ( ) const  [virtual]
```

References csound::fundamentalDomainByPredicate(), Z, ZMAX, and ZMIN.

6.73.3.27 getScore()

```
Score & csound::ScoreNode::getScore ( )  [virtual], [inherited]
```

References csound::ScoreNode::score.

Referenced by main().

6.73.3.28 getScoreType()

```
int csound::StrangeAttractor::getScoreType ( ) const  [virtual]
```

References scoreType.

6.73.3.29 getW()

```
double csound::StrangeAttractor::getW ( ) const  [virtual]
```

References W.

6.73.3.30 getX()

```
double csound::StrangeAttractor::getX ( ) const  [virtual]
```

References X.

6.73.3.31 getY()

```
double csound::StrangeAttractor::getY ( ) const  [virtual]
```

References Y.

6.73.3.32 getZ()

```
double csound::StrangeAttractor::getZ ( ) const  [virtual]
```

References Z.

6.73.3.33 initialize()

```
void csound::StrangeAttractor::initialize ( )  [virtual]
```

References A, D, EPS, N, ODE, OMAX, PREV, setIterationCount(), V, WS, XN, XS, XY, YS, and ZS.

Referenced by reset(), and StrangeAttractor().

6.73.3.34 iterate()

```
void csound::StrangeAttractor::iterate ( )  [virtual]
```

References A, D, EPS, I, I1, I2, I3, I4, I5, M, N, O, ODE, P, PREV, specialFunctions(), W, WMAX, WMIN, WNEW, WS, X, XMAX, XMIN, XN, XNEW, XP, XS, XY, Y, YMAX, YMIN, YNEW, YP, YS, Z, ZMAX, ZMIN, ZNEW, and ZS.

Referenced by calculateLyupanovExponent(), evaluateAttractor(), iterate_without_rendering(), and searchForAttractor().

6.73.3.35 iterate_without_rendering()

```
bool csound::StrangeAttractor::iterate_without_rendering ( )  [virtual]
```

Iterates an already found chaotic system one step, without rendering to the score.

This is useful for driving external score generators. Returns true if the system has settled onto the attractor, or false otherwise.

References calculateFractalDimension(), calculateLyupanovExponent(), iterate(), N, reinitialize(), csound::ScoreNode::score, W, WNEW, X, XNEW, Y, YNEW, Z, and ZNEW.

6.73.3.36 reinitialize()

```
void csound::StrangeAttractor::reinitialize ( )  [virtual]
```

References D, getCoefficients(), LSUM, N, N1, N2, NL, P, TWOD, W, WE, WMAX, WMIN, X, XE, XMAX, XMIN, Y, YE, YMAX, YMIN, Z, ZE, ZMAX, and ZMIN.

Referenced by evaluateAttractor(), iterate_without_rendering(), and searchForAttractor().

6.73.3.37 render()

```
void csound::StrangeAttractor::render (
            int N,
            double X,
            double Y,
            double Z,
            double W )  [virtual]
```

References csound::Score::append(), D, decibels, duration, csound::fundamentalDomainByPredicate(), instrument, N, octave, csound::Conversions::octaveToMidi(), pitchClassSet, randomNode, csound::Random::sample(), csound::ScoreNode::score, scoreType, time, W, x, X, Y, and Z.

Referenced by evaluateAttractor().

6.73.3.38 reset()

```
void csound::StrangeAttractor::reset ( )  [virtual]
```

References codeRandomize(), and initialize().

Referenced by StrangeAttractor().

6.73.3.39 searchForAttractor()

```
bool csound::StrangeAttractor::searchForAttractor ( )  [virtual]
```

References calculateFractalDimension(), calculateLyupanovExponent(), codeRandomize(), iterate(), L, N, NMAX, reinitialize(), W, WNEW, X, XNEW, Y, YNEW, Z, and ZNEW.

6.73.3.40 setAttractorType()

```
void csound::StrangeAttractor::setAttractorType (
            int attractorType )  [virtual]
```

Types: 1 = 1-dimensional polynomial map, 2 = 2-dimensional polynomial map, 3 = 3-dimensional polynomial map, 4 = 4-dimensional polynomial map, 5 = 3-dimensional ODE, 6 = 4-dimensional ODE, 7 through 12 = special functions.

References D, csound::fundamentalDomainByPredicate(), and ODE.

6.73.3.41 setCode()

```
void csound::StrangeAttractor::setCode (
            std::string code )  [virtual]
```

References code.

6.73.3.42 setDimensionCount()

```
void csound::StrangeAttractor::setDimensionCount (
            int D )  [virtual]
```

References D.

6.73.3.43 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.73.3.44 setIteration()

```
void csound::StrangeAttractor::setIteration (
            size_t iteration )  [virtual]
```

References csound::fundamentalDomainByPredicate(), and N.

6.73.3.45 setIterationCount()

```
void csound::StrangeAttractor::setIterationCount (
            size_t iterationCount )  [virtual]
```

References csound::fundamentalDomainByPredicate(), and NMAX.

Referenced by initialize().

6.73.3.46 setScoreType()

```
void csound::StrangeAttractor::setScoreType (
            int attractorType )  [virtual]
```

References csound::fundamentalDomainByPredicate(), and scoreType.

6.73.3.47 setW()

```
void csound::StrangeAttractor::setW (
            double X )  [virtual]
```

References csound::fundamentalDomainByPredicate(), and W.

6.73.3.48 setX()

```
void csound::StrangeAttractor::setX (
            double X )  [virtual]
```

References csound::fundamentalDomainByPredicate(), and X.

6.73.3.49 setY()

```
void csound::StrangeAttractor::setY (
            double X )  [virtual]
```

References csound::fundamentalDomainByPredicate(), and Y.

6.73.3.50 setZ()

```
void csound::StrangeAttractor::setZ (
            double X )  [virtual]
```

References csound::fundamentalDomainByPredicate(), and Z.

6.73.3.51 shuffleRandomNumbers()

```
void csound::StrangeAttractor::shuffleRandomNumbers ( )  [virtual]
```

References csound::fundamentalDomainByPredicate(), J, RAN, randomNode, csound::Random::sample(), and V.

6.73.3.52 specialFunctions()

```
void csound::StrangeAttractor::specialFunctions ( )  [virtual]
```

References A, AL, COSAL, DUM, EPS, csound::fundamentalDomainByPredicate(), csound::Conversions::get2PI(), M, N, NMAX, ODE, SINAL, WNEW, X, XNEW, Y, YNEW, Z, and ZNEW.

Referenced by getDimensionAndOrder(), and iterate().

6.73.3.53 transform()

```
void csound::Node::transform (
            Score & score_from_children )  [virtual], [inherited]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented in csound::Cell, csound::CellRepeat, csound::CellAdd, csound::CellMultiply, csound::CellReflect, csound::CellSelect, csound::CellRemove, csound::CellChord, csound::CellRandom, csound::CellShuffle, csound::CounterpointNode, csound::RemoveDuplicates, csound::Transformer, csound::Random, csound::Rescale, csound::VoiceleadingNode, csound::LispTransformer, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.73.3.54 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord and csound::Node::transform().

6.73.4 Field Documentation

6.73.4.1 A

```
std::vector<double> csound::StrangeAttractor::A  [protected]
```

Referenced by getCoefficients(), initialize(), iterate(), and specialFunctions().

6.73.4.2 AL

```
double csound::StrangeAttractor::AL  [protected]
```

Referenced by specialFunctions().

6.73.4.3 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.73.4.4 code

```
std::string csound::StrangeAttractor::code  [protected]
```

Referenced by codeRandomize(), getCode(), getCoefficients(), getDimensionAndOrder(), and setCode().

6.73.4.5 COSAL

```
double csound::StrangeAttractor::COSAL  [protected]
```

Referenced by specialFunctions().

6.73.4.6 D

```
int csound::StrangeAttractor::D  [protected]
```

Referenced by codeRandomize(), getAttractorType(), getDimensionAndOrder(), getDimensionCount(), initialize(), iterate(), reinitialize(), render(), setAttractorType(), and setDimensionCount().

6.73.4.7 D2

```
double csound::StrangeAttractor::D2  [protected]
```

Referenced by calculateFractalDimension().

6.73.4.8 D2MAX

```
double csound::StrangeAttractor::D2MAX  [protected]
```

Referenced by calculateFractalDimension().

6.73.4.9 DD

```
int csound::StrangeAttractor::DD  [protected]
```

6.73.4.10 decibels

```
double csound::StrangeAttractor::decibels  [protected]
```

Referenced by render().

6.73.4.11 DF

```
double csound::StrangeAttractor::DF  [protected]
```

Referenced by calculateLyupanovExponent().

6.73.4.12 DL2

```
double csound::StrangeAttractor::DL2  [protected]
```

Referenced by calculateLyupanovExponent().

6.73.4.13 DLW

```
double csound::StrangeAttractor::DLW  [protected]
```

Referenced by calculateLyupanovExponent().

6.73.4.14 DLX

```
double csound::StrangeAttractor::DLX  [protected]
```

Referenced by calculateLyupanovExponent().

6.73.4.15 DLY

```
double csound::StrangeAttractor::DLY  [protected]
```

Referenced by calculateLyupanovExponent().

6.73.4.16 DLZ

```
double csound::StrangeAttractor::DLZ  [protected]
```

Referenced by calculateLyupanovExponent().

6.73.4.17 DUM

```
double csound::StrangeAttractor::DUM  [protected]
```

Referenced by specialFunctions().

6.73.4.18 duration

```
double csound::StrangeAttractor::duration  [protected]
```

Referenced by render().

6.73.4.19 DW

```
double csound::StrangeAttractor::DW  [protected]
```

Referenced by calculateFractalDimension().

6.73.4.20 DX

```
double csound::StrangeAttractor::DX  [protected]
```

Referenced by calculateFractalDimension().

6.73.4.21 DY

```
double csound::StrangeAttractor::DY  [protected]
```

Referenced by calculateFractalDimension().

6.73.4.22 DZ

```
double csound::StrangeAttractor::DZ  [protected]
```

Referenced by calculateFractalDimension().

6.73.4.23 EPS

```
double csound::StrangeAttractor::EPS  [protected]
```

Referenced by calculateLyupanovExponent(), initialize(), iterate(), and specialFunctions().

6.73.4.24 F

```
double csound::StrangeAttractor::F  [protected]
```

Referenced by calculateFractalDimension(), and getFractalDimension().

6.73.4.25 filename

```
std::string csound::StrangeAttractor::filename  [protected]
```

6.73.4.26 I

```
int csound::StrangeAttractor::I  [protected]
```

Referenced by codeRandomize(), getCoefficients(), getDimensionAndOrder(), and iterate().

6.73.4.27 I1

```
int csound::StrangeAttractor::I1  [protected]
```

Referenced by iterate().

6.73.4.28 I2

```
int csound::StrangeAttractor::I2  [protected]
```

Referenced by iterate().

6.73.4.29 I3

```
int csound::StrangeAttractor::I3  [protected]
```

Referenced by iterate().

6.73.4.30 I4

```
int csound::StrangeAttractor::I4  [protected]
```

Referenced by iterate().

6.73.4.31 I5

```
int csound::StrangeAttractor::I5  [protected]
```

Referenced by iterate().

6.73.4.32 importFilename

```
std::string csound::ScoreNode::importFilename  [inherited]
```

Referenced by csound::ScoreNode::generate().

6.73.4.33 instrument

```
double csound::StrangeAttractor::instrument  [protected]
```

Referenced by render().

6.73.4.34 J

```
int csound::StrangeAttractor::J  [protected]
```

Referenced by calculateFractalDimension(), and shuffleRandomNumbers().

6.73.4.35 L

```
double csound::StrangeAttractor::L  [protected]
```

Referenced by calculateLyupanovExponent(), getLyupanovExponent(), and searchForAttractor().

6.73.4.36 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.73.4.37 LSUM

```
double csound::StrangeAttractor::LSUM  [protected]
```

Referenced by calculateLyupanovExponent(), and reinitialize().

6.73.4.38 M

```
int csound::StrangeAttractor::M  [protected]
```

Referenced by codeRandomize(), getCoefficients(), getDimensionAndOrder(), iterate(), and specialFunctions().

6.73.4.39 MX

```
double csound::StrangeAttractor::MX  [protected]
```

6.73.4.40 MY

```
double csound::StrangeAttractor::MY  [protected]
```

6.73.4.41 N

```
int csound::StrangeAttractor::N  [protected]
```

Referenced by calculateFractalDimension(), calculateLyupanovExponent(), evaluateAttractor(), generateLocally(), getIteration(), initialize(), iterate(), iterate_without_rendering(), reinitialize(), render(), searchForAttractor(), setIteration(), and specialFunctions().

6.73.4.42 N1

```
double csound::StrangeAttractor::N1  [protected]
```

Referenced by calculateFractalDimension(), and reinitialize().

6.73.4.43 N2

```
double csound::StrangeAttractor::N2  [protected]
```

Referenced by calculateFractalDimension(), and reinitialize().

6.73.4.44 NL

```
double csound::StrangeAttractor::NL  [protected]
```

Referenced by calculateLyupanovExponent(), and reinitialize().

6.73.4.45 NMAX

```
int csound::StrangeAttractor::NMAX  [protected]
```

Referenced by evaluateAttractor(), getIterationCount(), searchForAttractor(), setIterationCount(), and specialFunctions().

6.73.4.46 O

```
int csound::StrangeAttractor::O  [protected]
```

Referenced by codeRandomize(), getDimensionAndOrder(), and iterate().

6.73.4.47 octave

```
double csound::StrangeAttractor::octave  [protected]
```

Referenced by render().

6.73.4.48 ODE

```
int csound::StrangeAttractor::ODE  [protected]
```

Referenced by calculateLyupanovExponent(), codeRandomize(), getAttractorType(), getDimensionAndOrder(), initialize(), iterate(), setAttractorType(), and specialFunctions().

6.73.4.49 OMAX

```
int csound::StrangeAttractor::OMAX  [protected]
```

Referenced by codeRandomize(), and initialize().

6.73.4.50 P

```
int csound::StrangeAttractor::P  [protected]
```

Referenced by calculateFractalDimension(), iterate(), and reinitialize().

6.73.4.51 pitchClassSet

```
double csound::StrangeAttractor::pitchClassSet  [protected]
```

Referenced by render().

6.73.4.52 PREV

```
int csound::StrangeAttractor::PREV  [protected]
```

Referenced by initialize(), and iterate().

6.73.4.53 PT

```
double csound::StrangeAttractor::PT  [protected]
```

6.73.4.54 RAN

```
double csound::StrangeAttractor::RAN  [protected]
```

Referenced by shuffleRandomNumbers().

6.73.4.55 randomNode

```
Random csound::StrangeAttractor::randomNode
```

Referenced by calculateFractalDimension(), codeRandomize(), render(), shuffleRandomNumbers(), and StrangeAttractor().

6.73.4.56 RS

```
double csound::StrangeAttractor::RS  [protected]
```

Referenced by calculateLyupanovExponent().

6.73.4.57 score

```
Score csound::ScoreNode::score  [protected], [inherited]
```

Referenced by evaluateAttractor(), csound::ExternalNode::generate(), csound::ScoreNode::generate(), csound::MCRM::generate(), csound::ExternalNode::generateLocally(), csound::ImageToScore2::generateLocally(), csound::Lindenmayer::generateLocally(), csound::Rescale::getRescale(), csound::ScoreNode::getScore(), csound::Lindenmayer::interpret(), csound::MCRM::iterate(), iterate_without_rendering(), csound::KMeansMCRM::means_to_notes(), csound::ImageToScore2::pixel_to_event(), render(), csound::Rescale::Rescale(), csound::Rescale::setRescale(), csound::Cell::transform(), csound::Rescale::transform(), csound::CMaskNode::translate_to_silence(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Lindenmayer::updateActual().

6.73.4.58 scoreType

```
int csound::StrangeAttractor::scoreType  [protected]
```

Referenced by getScoreType(), render(), and setScoreType().

6.73.4.59 SH

```
double csound::StrangeAttractor::SH  [protected]
```

6.73.4.60 SINAL

```
double csound::StrangeAttractor::SINAL  [protected]
```

Referenced by specialFunctions().

6.73.4.61 SW

```
double csound::StrangeAttractor::SW  [protected]
```

6.73.4.62 T

```
int csound::StrangeAttractor::T  [protected]
```

6.73.4.63 TIA

```
double csound::StrangeAttractor::TIA  [protected]
```

6.73.4.64 time

```
double csound::StrangeAttractor::time  [protected]
```

Referenced by render().

6.73.4.65 TT

```
double csound::StrangeAttractor::TT  [protected]
```

6.73.4.66 TWOD

```
int csound::StrangeAttractor::TWOD  [protected]
```

Referenced by calculateFractalDimension(), and reinitialize().

6.73.4.67 V

```
std::vector<double> csound::StrangeAttractor::V  [protected]
```

Referenced by initialize(), and shuffleRandomNumbers().

6.73.4.68 W

```
double csound::StrangeAttractor::W  [protected]
```

Referenced by calculateLyupanovExponent(), evaluateAttractor(), getNormalizedW(), getW(), iterate(), iterate_without_rendering(), reinitialize(), render(), searchForAttractor(), and setW().

6.73.4.69 WE

```
double csound::StrangeAttractor::WE  [protected]
```

Referenced by calculateLyupanovExponent(), and reinitialize().

6.73.4.70 WMAX

```
double csound::StrangeAttractor::WMAX  [protected]
```

Referenced by calculateFractalDimension(), getNormalizedW(), iterate(), and reinitialize().

6.73.4.71 WMIN

```
double csound::StrangeAttractor::WMIN  [protected]
```

Referenced by calculateFractalDimension(), getNormalizedW(), iterate(), and reinitialize().

6.73.4.72 WNEW

```
double csound::StrangeAttractor::WNEW  [protected]
```

Referenced by calculateFractalDimension(), calculateLyupanovExponent(), evaluateAttractor(), iterate(), iterate_without_rendering(), searchForAttractor(), and specialFunctions().

6.73.4.73 WP

```
double csound::StrangeAttractor::WP  [protected]
```

6.73.4.74 WS

```
std::vector<double> csound::StrangeAttractor::WS  [protected]
```

Referenced by calculateFractalDimension(), initialize(), and iterate().

6.73.4.75 WSAVE

```
double csound::StrangeAttractor::WSAVE  [protected]
```

Referenced by calculateLyupanovExponent().

6.73.4.76 x

```
double csound::StrangeAttractor::x  [protected]
```

Referenced by render().

6.73.4.77 X

```
double csound::StrangeAttractor::X  [protected]
```

Referenced by calculateLyupanovExponent(), evaluateAttractor(), getNormalizedX(), getX(), iterate(), iterate_without_rendering(), reinitialize(), render(), searchForAttractor(), setX(), and specialFunctions().

6.73.4.78 XA

```
double csound::StrangeAttractor::XA  [protected]
```

6.73.4.79 XE

```
double csound::StrangeAttractor::XE  [protected]
```

Referenced by calculateLyupanovExponent(), and reinitialize().

6.73.4.80 XH

```
double csound::StrangeAttractor::XH  [protected]
```

6.73.4.81 XL

```
double csound::StrangeAttractor::XL  [protected]
```

6.73.4.82 XMAX

```
double csound::StrangeAttractor::XMAX  [protected]
```

Referenced by calculateFractalDimension(), getNormalizedX(), iterate(), and reinitialize().

6.73.4.83 XMIN

```
double csound::StrangeAttractor::XMIN  [protected]
```

Referenced by calculateFractalDimension(), getNormalizedX(), iterate(), and reinitialize().

6.73.4.84 XN

```
std::vector<double> csound::StrangeAttractor::XN  [protected]
```

Referenced by initialize(), and iterate().

6.73.4.85 XNEW

```
double csound::StrangeAttractor::XNEW  [protected]
```

Referenced by calculateFractalDimension(), calculateLyupanovExponent(), evaluateAttractor(), iterate(), iterate_without_rendering(), searchForAttractor(), and specialFunctions().

6.73.4.86 XP

```
double csound::StrangeAttractor::XP  [protected]
```

Referenced by iterate().

6.73.4.87 XS

```
std::vector<double> csound::StrangeAttractor::XS  [protected]
```

Referenced by calculateFractalDimension(), initialize(), and iterate().

6.73.4.88 XSAVE

```
double csound::StrangeAttractor::XSAVE  [protected]
```

Referenced by calculateLyupanovExponent().

6.73.4.89 XW

```
double csound::StrangeAttractor::XW  [protected]
```

6.73.4.90 XY

```
std::vector<double> csound::StrangeAttractor::XY  [protected]
```

Referenced by initialize(), and iterate().

6.73.4.91 XZ

```
double csound::StrangeAttractor::XZ  [protected]
```

6.73.4.92 Y

```
double csound::StrangeAttractor::Y  [protected]
```

Referenced by calculateLyupanovExponent(), evaluateAttractor(), getNormalizedY(), getY(), iterate(), iterate_without_rendering(), reinitialize(), render(), searchForAttractor(), setY(), and specialFunctions().

6.73.4.93 YA

```
double csound::StrangeAttractor::YA  [protected]
```

6.73.4.94 YE

```
double csound::StrangeAttractor::YE  [protected]
```

Referenced by calculateLyupanovExponent(), and reinitialize().

6.73.4.95 YH

```
double csound::StrangeAttractor::YH  [protected]
```

6.73.4.96 YL

```
double csound::StrangeAttractor::YL  [protected]
```

6.73.4.97 YMAX

```
double csound::StrangeAttractor::YMAX  [protected]
```

Referenced by calculateFractalDimension(), getNormalizedY(), iterate(), and reinitialize().

6.73.4.98 YMIN

```
double csound::StrangeAttractor::YMIN  [protected]
```

Referenced by calculateFractalDimension(), getNormalizedY(), iterate(), and reinitialize().

6.73.4.99 YNEW

```
double csound::StrangeAttractor::YNEW  [protected]
```

Referenced by calculateFractalDimension(), calculateLyupanovExponent(), evaluateAttractor(), iterate(), iterate_without_rendering(), searchForAttractor(), and specialFunctions().

6.73.4.100 YP

```
double csound::StrangeAttractor::YP  [protected]
```

Referenced by iterate().

6.73.4.101 YS

```
std::vector<double> csound::StrangeAttractor::YS  [protected]
```

Referenced by calculateFractalDimension(), initialize(), and iterate().

6.73.4.102 YSAVE

```
double csound::StrangeAttractor::YSAVE  [protected]
```

Referenced by calculateLyupanovExponent().

6.73.4.103 YW

```
double csound::StrangeAttractor::YW  [protected]
```

6.73.4.104 YZ

```
double csound::StrangeAttractor::YZ  [protected]
```

6.73.4.105 Z

```
double csound::StrangeAttractor::Z  [protected]
```

Referenced by calculateLyupanovExponent(), evaluateAttractor(), getNormalizedZ(), getZ(), iterate(), iterate_without_rendering(), reinitialize(), render(), searchForAttractor(), setZ(), and specialFunctions().

6.73.4.106 ZA

```
double csound::StrangeAttractor::ZA  [protected]
```

6.73.4.107 ZE

```
double csound::StrangeAttractor::ZE  [protected]
```

Referenced by calculateLyupanovExponent(), and reinitialize().

6.73.4.108 ZMAX

```
double csound::StrangeAttractor::ZMAX  [protected]
```

Referenced by calculateFractalDimension(), getNormalizedZ(), iterate(), and reinitialize().

6.73.4.109 ZMIN

```
double csound::StrangeAttractor::ZMIN  [protected]
```

Referenced by calculateFractalDimension(), getNormalizedZ(), iterate(), and reinitialize().

6.73.4.110 ZNEW

```
double csound::StrangeAttractor::ZNEW  [protected]
```

Referenced by calculateFractalDimension(), calculateLyupanovExponent(), evaluateAttractor(), iterate(), iterate_without_rendering(), searchForAttractor(), and specialFunctions().

6.73.4.111 ZP

```
double csound::StrangeAttractor::ZP  [protected]
```

6.73.4.112 ZS

```
std::vector<double> csound::StrangeAttractor::ZS  [protected]
```

Referenced by calculateFractalDimension(), initialize(), and iterate().

6.73.4.113 ZSAVE

```
double csound::StrangeAttractor::ZSAVE  [protected]
```

Referenced by calculateLyupanovExponent().

6.74 csound::System Class Reference

Abstraction layer for a minimal set of system services.

```
#include <System.hpp>
```

Public Types

- enum Level { ERROR_LEVEL = 1 , WARNING_LEVEL = 2 , INFORMATION_LEVEL = 4 , DEBUGGING_LEVEL = 8 }

Static Public Member Functions

- static void beep ()

  Make some sort of noticeable sound.
- static void closeLibrary (void *library)

  Closes a shared library.
- static void * createThread (void(*threadRoutine)(void *threadData), void *data, int priority)

  Creates a new thread.
- static void * createThreadLock ()

  Creates a thread lock.
- static void debug (const char *format,...)

  Prints a message if the DEBUGGING_LEVEL flag is set.
- static void debug (CSOUND *csound, const char *format,...)

  Prints a message if the DEBUGGING_LEVEL flag is set.
- static void debug_text (std::string text)
- static void destroyThreadLock (void *lock)

  Destroys a thread lock.
- static void error (const char *format,...)

  Prints a message if the ERROR_LEVEL flag is set.
- static void error (CSOUND *csound, const char *format,...)

  Prints a message if the ERROR_LEVEL flag is set.
- static void error_text (std::string text)
- static int execute (const char *command)

  Execute a system command or program.
- static std::vector< std::string > getDirectoryNames (std::string directoryName)

  Lists directory names in a directory; useful for locating plugins.
- static std::vector< std::string > getFilenames (std::string directoryName)

  Lists filenames in a directory; useful for locating plugins.
- static FILE * getLogfile ()

  Return the stream, if any, used for printing messages to.
- static MessageCallbackType getMessageCallback ()

  Return the message callback, or null if none.
- static int getMessageLevel ()

  Returns current system message level.
- static std::string getSharedLibraryExtension ()

  Returns the standard filename extension for a shared library, such as "dll" or "so".
- static void * getSymbol (void *library, std::string name)

  Returns the address of a symbol (function or object) in a shared library; useful for loading plugin functions.
- static void * getUserdata ()

  Returns userdata for message printing.
- static void inform (const char *format,...)

  Prints a message if the INFORMATION_LEVEL flag is set.
- static void inform (CSOUND *csound, const char *format,...)

  Prints a message if the INFORMATION_LEVEL flag is set.
- static void inform_text (std::string text)
- static void message (const char *format, va_list valist)

  Prints a message.

- static void message (const char *format,...)

  Prints a message.
- static void message (CSOUND *csound, const char *format, va_list valist)

  Prints a message.
- static void message (CSOUND *csound, const char *format,...)

  Prints a message.
- static void message (CSOUND *csound, int attribute, const char *format, va_list valist)

  Unconditionally prints a message.
- static void message (CSOUND *csound, int level, const char *format,...)

  Prints a message.
- static void message (std::string text)

  Prints a message.
- static void message_text (std::string text)
- static void notifyThreadLock (void *lock)

  Releases a thread lock.
- static int openLibrary (void **library, std::string filename)

  Opens a shared library; useful for loading plugins.
- static void parsePathname (const std::string pathname, std::string &drive, std::string &base, std::string &file, std::string &extension)

  Parses a filename into its component parts, which are returned in the arguments.
- static void setLogfile (FILE *logfile)

  Set a stream for printing messages to (in addition to callback, stderr, etc.).
- static void setMessageCallback (MessageCallbackType messageCallback_)

  Sets message callback.
- static int setMessageLevel (int messageLevel)

  Sets message level, returns old message level.
- static void setUserdata (void *userdata)

  Sets userdata for message printing.
- static int shellOpen (const char *filename, const char *command="open")

  Open a file using the operating system shell.
- static void sleep (double milliseconds)

  Sleep the indicated number of milliseconds.
- static clock_t startTiming ()

  Starts timing.
- static double stopTiming (clock_t startedAt)

  Stop timing, and return elapsed seonds.
- static void waitThreadLock (void *lock, size_t timeoutMilliseconds=0)

  Waits on a thread lock.
- static void warn (const char *format,...)

  Prints a message if the WARNNING_LEVEL flag is set.
- static void warn (CSOUND *csound, const char *format,...)

  Prints a message if the WARNNING_LEVEL flag is set.
- static void warn_text (std::string text)
- static void yieldThread ()

  Yields to the next waiting thread.

6.74.1 Detailed Description

Abstraction layer for a minimal set of system services.

6.74.2 Member Enumeration Documentation

6.74.2.1 Level

```
enum csound::System::Level
```

Enumerator

ERROR_LEVEL	
WARNING_LEVEL	
INFORMATION_LEVEL	
DEBUGGING_LEVEL	

6.74.3 Member Function Documentation

6.74.3.1 beep()

```
static void csound::System::beep ( )  [static]
```

Make some sort of noticeable sound.

6.74.3.2 closeLibrary()

```
SILENCE_PUBLIC void csound::System::closeLibrary (
            void * library )  [static]
```

Closes a shared library.

References csound::fundamentalDomainByPredicate().

6.74.3.3 createThread()

```
static void * csound::System::createThread (
            void(*)(void *threadData) threadRoutine,
            void * data,
            int priority )  [static]
```

Creates a new thread.

6.74.3.4 createThreadLock()

```
static void * csound::System::createThreadLock ( )  [static]
```

Creates a thread lock.

Referenced by csound::ThreadLock::open().

6.74.3.5 debug() [1/2]

```
SILENCE_PUBLIC void csound::System::debug (
            const char * format,
             ... )  [static]
```

Prints a message if the DEBUGGING_LEVEL flag is set.

References DEBUGGING_LEVEL, csound::fundamentalDomainByPredicate(), message(), csound::message_level(), and csound::user_data().

6.74.3.6 debug() [2/2]

```
SILENCE_PUBLIC void csound::System::debug (
            CSOUND * csound,
            const char * format,
             ... )  [static]
```

Prints a message if the DEBUGGING_LEVEL flag is set.

References DEBUGGING_LEVEL, csound::fundamentalDomainByPredicate(), message(), and csound::message_level().

Referenced by csound::ChordLindenmayer::arithmetic(), csound::ChordLindenmayer::arithmetic(), csound::ChordLindenmayer::chordOpe
csound::StrangeAttractor::codeRandomize(), csound::LispGenerator::generate(), csound::ChordLindenmayer::generateLindenmayerSyste
csound::ExternalNode::generateLocally(), csound::ImageToScore2::generateLocally(), csound::ChordLindenmayer::modalityOperation(),
csound::ChordLindenmayer::noteOperation(), csound::ChordLindenmayer::noteOrientationOperation(), csound::ChordLindenmayer::noteS
csound::parseIndex(), csound::MidiEvent::readIn(), csound::Lindenmayer::rewrite(), csound::ChordLindenmayer::scaleDegreeOperation(),
csound::ChordLindenmayer::scaleOperation(), csound::ChordLindenmayer::scoreOperation(), csound::Score::setScale(),
csound::ChordLindenmayer::turtleOperation(), and csound::ChordLindenmayer::voicingOperation().

6.74.3.7 debug_text()

```
static void csound::System::debug_text (
            std::string text )  [inline], [static]
```

6.74.3.8 destroyThreadLock()

```
static void csound::System::destroyThreadLock (
            void * lock )  [static]
```

Destroys a thread lock.

Referenced by csound::ThreadLock::close().

6.74.3.9 error() [1/2]

```
SILENCE_PUBLIC void csound::System::error (
            const char * format,
             ... ) [static]
```

Prints a message if the ERROR_LEVEL flag is set.

References ERROR_LEVEL, csound::fundamentalDomainByPredicate(), message(), csound::message_level(), and csound::user_data().

6.74.3.10 error() [2/2]

```
SILENCE_PUBLIC void csound::System::error (
            CSOUND * csound,
            const char * format,
             ... ) [static]
```

Prints a message if the ERROR_LEVEL flag is set.

References ERROR_LEVEL, csound::fundamentalDomainByPredicate(), message(), and csound::message_level().

Referenced by equals(), csound::ExternalNode::generateLocally(), csound::Lindenmayer::interpret(), csound::Score::load(), csound::ImageToScore2::processImage(), csound::MidiEvent::readIn(), csound::Shell::runScript(), and csound::Score::save().

6.74.3.11 error_text()

```
static void csound::System::error_text (
            std::string text ) [inline], [static]
```

6.74.3.12 execute()

```
static int csound::System::execute (
            const char * command ) [static]
```

Execute a system command or program.

6.74.3.13 getDirectoryNames()

```
static std::vector< std::string > csound::System::getDirectoryNames (
            std::string directoryName ) [static]
```

Lists directory names in a directory; useful for locating plugins.

6.74.3.14 getFilenames()

```
static std::vector< std::string > csound::System::getFilenames (
            std::string directoryName )  [static]
```

Lists filenames in a directory; useful for locating plugins.

6.74.3.15 getLogfile()

```
SILENCE_PUBLIC FILE * csound::System::getLogfile ( )  [static]
```

Return the stream, if any, used for printing messages to.

References csound::log_file().

6.74.3.16 getMessageCallback()

```
SILENCE_PUBLIC MessageCallbackType csound::System::getMessageCallback ( )  [static]
```

Return the message callback, or null if none.

References csound::message_callback().

6.74.3.17 getMessageLevel()

```
SILENCE_PUBLIC int csound::System::getMessageLevel ( )  [static]
```

Returns current system message level.

References csound::message_level().

Referenced by csound::VoiceleadingNode::apply(), csound::ImageToScore2::generateLocally(), csound::printChord(), csound::printChord(), csound::Score::voicelead(), and csound::Score::voicelead().

6.74.3.18 getSharedLibraryExtension()

```
static std::string csound::System::getSharedLibraryExtension ( )  [static]
```

Returns the standard filename extension for a shared library, such as "dll" or "so".

6.74.3.19 getSymbol()

```
SILENCE_PUBLIC void * csound::System::getSymbol (
            void * library,
            std::string name )  [static]
```

Returns the address of a symbol (function or object) in a shared library; useful for loading plugin functions.

References csound::fundamentalDomainByPredicate().

6.74.3.20 getUserdata()

```
SILENCE_PUBLIC void * csound::System::getUserdata ( )  [static]
```

Returns userdata for message printing.

References csound::user_data().

6.74.3.21 inform() [1/2]

```
SILENCE_PUBLIC void csound::System::inform (
            const char * format,
             ... )  [static]
```

Prints a message if the INFORMATION_LEVEL flag is set.

References csound::fundamentalDomainByPredicate(), INFORMATION_LEVEL, message(), csound::message_level(), and csound::user_data().

6.74.3.22 inform() [2/2]

```
SILENCE_PUBLIC void csound::System::inform (
            CSOUND * csound,
            const char * format,
             ... )  [static]
```

Prints a message if the INFORMATION_LEVEL flag is set.

References csound::fundamentalDomainByPredicate(), INFORMATION_LEVEL, message(), and csound::message_level().

Referenced by csound::VoiceleadingNode::apply(), csound::MusicModel::createCsoundScore(), csound::MusicModel::csoundArgv(), csound::KMeansMCRM::deterministic_algorithm(), csound::LispGenerator::generate(), csound::Composition::generateAllNames(), csound::ChordLindenmayer::generateLindenmayerSystem(), csound::ChordLindenmayer::generateLocally(), csound::ExternalNode::gene csound::ImageToScore2::generateLocally(), csound::KMeansMCRM::generateLocally(), csound::Score::getPitches(), csound::Score::getVoicing(), csound::initialize_ecl(), csound::KMeansMCRM::iterate(), csound::Score::load(), main(), csound::KMeansMCRM::means_to_notes(), csound::Composition::normalizeOutputSoundfile(), csound::Composition::performAll(), csound::Composition::performMaster(), csound::printChord(), csound::Score::process(), csound::MusicModel::processArgs(), csound::ImageToScore2::processImage(), csound::KMeansMCRM::random_algorithm(), csound::Chunk::read(), csound::MidiEvent::readIn(), csound::Score::save(), csound::Score::setPT(), csound::Score::setPTV(), csound::Score::setQ(), csound::MusicModel::stop(), csound::Composition::tagFile(), Counterpoint::toCsoundScore(), csound::CellRepeat::transform(), csound::VoiceleadingNode::transform(), csound::LispTransformer::transform(), csound::Composition::translateMaster(), csound::Composition::translateToCdAudio(), csound::Composition::translateToMp3(), csound::Composition::translateToMp4(), csound::Score::voicelead(), and csound::Score::voicelead().

6.74.3.23 inform_text()

```
static void csound::System::inform_text (
            std::string text )  [inline], [static]
```

6.74.3.24 message() [1/7]

```
SILENCE_PUBLIC void csound::System::message (
            const char * format,
            va_list valist )  [static]
```

Prints a message.

References csound::fundamentalDomainByPredicate(), message(), csound::message_level(), and csound::user_data().

6.74.3.25 message() [2/7]

```
SILENCE_PUBLIC void csound::System::message (
            const char * format,
             ... )  [static]
```

Prints a message.

References csound::fundamentalDomainByPredicate(), message(), csound::message_level(), and csound::user_data().

6.74.3.26 message() [3/7]

```
SILENCE_PUBLIC void csound::System::message (
            CSOUND * csound,
            const char * format,
            va_list valist )  [static]
```

Prints a message.

References csound::fundamentalDomainByPredicate(), and message().

6.74.3.27 message() [4/7]

```
SILENCE_PUBLIC void csound::System::message (
            CSOUND * csound,
            const char * format,
             ... )  [static]
```

Prints a message.

References csound::fundamentalDomainByPredicate(), message(), and csound::message_level().

Referenced by debug(), debug(), error(), error(), fail(), csound::MusicModel::generate(), csound::ScoreModel::generate(), inform(), inform(), main(), main(), Counterpoint::message(), message(), message(), message(), message(), message(), message(), pass(), csound::MusicModel::perform(), csound::Composition::performAll(), printSet(), csound::Shell::runScript(), summary(), test_nrL(), test_nrP(), test_nrR(), test_pitv(), test_pitv(), testEquivalenceRelation(), testEquivalenceRelations(), testNormalsAndEquivalents(), csound::Cell::transform(), csound::CounterpointNode::transform(), csound::Composition::translateMaster(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), csound::Sequence::traverse(), warn(), warn(), and csound::Composition::write().

6.74.3.28 message() [5/7]

```
SILENCE_PUBLIC void csound::System::message (
            CSOUND * csound,
            int attribute,
            const char * format,
            va_list valist )  [static]
```

Unconditionally prints a message.

This is the lowest-level message function that calls the message callback, if one has been set.

References csound::fundamentalDomainByPredicate(), csound::log_file(), and csound::message_callback().

6.74.3.29 message() [6/7]

```
SILENCE_PUBLIC void csound::System::message (
            CSOUND * csound,
            int level,
            const char * format,
             ... )  [static]
```

Prints a message.

References csound::fundamentalDomainByPredicate(), message(), and csound::message_level().

6.74.3.30 message() [7/7]

```
SILENCE_PUBLIC void csound::System::message (
            std::string text )  [static]
```

Prints a message.

References csound::fundamentalDomainByPredicate(), and message().

6.74.3.31 message_text()

```
static void csound::System::message_text (
            std::string text )  [inline], [static]
```

6.74.3.32 notifyThreadLock()

```
static void csound::System::notifyThreadLock (
            void * lock )  [static]
```

Releases a thread lock.

Referenced by csound::ThreadLock::endWait().

6.74.3.33 openLibrary()

```
SILENCE_PUBLIC int csound::System::openLibrary (
            void ** library,
            std::string filename )  [static]
```

Opens a shared library; useful for loading plugins.

References csound::fundamentalDomainByPredicate().

6.74.3.34 parsePathname()

```
static void csound::System::parsePathname (
            const std::string pathname,
            std::string & drive,
            std::string & base,
            std::string & file,
            std::string & extension )  [static]
```

Parses a filename into its component parts, which are returned in the arguments.

On Unix and Linux, "drive" is always empty.

6.74.3.35 setLogfile()

```
SILENCE_PUBLIC void csound::System::setLogfile (
            FILE * logfile )  [static]
```

Set a stream for printing messages to (in addition to callback, stderr, etc.).

References csound::fundamentalDomainByPredicate(), and csound::log_file().

6.74.3.36 setMessageCallback()

```
SILENCE_PUBLIC void csound::System::setMessageCallback (
            MessageCallbackType messageCallback_ )  [static]
```

Sets message callback.

References csound::fundamentalDomainByPredicate(), and csound::message_callback().

6.74.3.37 setMessageLevel()

```
SILENCE_PUBLIC int csound::System::setMessageLevel (
            int messageLevel )  [static]
```

Sets message level, returns old message level.

References csound::fundamentalDomainByPredicate(), and csound::message_level().

Referenced by main(), and main().

6.74.3.38 setUserdata()

```
SILENCE_PUBLIC void csound::System::setUserdata (
            void * userdata )  [static]
```

Sets userdata for message printing.

References csound::fundamentalDomainByPredicate(), and csound::user_data().

6.74.3.39 shellOpen()

```
static int csound::System::shellOpen (
            const char * filename,
            const char * command = "open" )  [static]
```

Open a file using the operating system shell.

6.74.3.40 sleep()

```
static void csound::System::sleep (
            double milliseconds )  [static]
```

Sleep the indicated number of milliseconds.

6.74.3.41 startTiming()

```
SILENCE_PUBLIC clock_t csound::System::startTiming ( )  [static]
```

Starts timing.

References csound::fundamentalDomainByPredicate().

Referenced by csound::KMeansMCRM::deterministic_algorithm(), csound::KMeansMCRM::generateLocally(), csound::KMeansMCRM::means_to_notes(), csound::MusicModel::perform(), csound::Composition::performAll(), csound::KMeansMCRM::random_algorithm(), and csound::Composition::translateMaster().

6.74.3.42 stopTiming()

```
SILENCE_PUBLIC double csound::System::stopTiming (
            clock_t startedAt )  [static]
```

Stop timing, and return elapsed seonds.

References csound::fundamentalDomainByPredicate().

Referenced by csound::KMeansMCRM::deterministic_algorithm(), csound::KMeansMCRM::generateLocally(), csound::KMeansMCRM::means_to_notes(), csound::MusicModel::perform(), csound::Composition::performAll(), csound::KMeansMCRM::random_algorithm(), and csound::Composition::translateMaster().

6.74.3.43 waitThreadLock()

```
static void csound::System::waitThreadLock (
            void * lock,
            size_t timeoutMilliseconds = 0 )  [static]
```

Waits on a thread lock.

Zero timeout means infinite timeout.

Referenced by csound::ThreadLock::startWait().

6.74.3.44 warn() [1/2]

```
SILENCE_PUBLIC void csound::System::warn (
            const char * format,
             ... )  [static]
```

Prints a message if the WARNNING_LEVEL flag is set.

References csound::fundamentalDomainByPredicate(), message(), csound::message_level(), csound::user_data(), and WARNING_LEVEL.

6.74.3.45 warn() [2/2]

```
SILENCE_PUBLIC void csound::System::warn (
            CSOUND * csound,
            const char * format,
             ... )  [static]
```

Prints a message if the WARNNING_LEVEL flag is set.

References csound::fundamentalDomainByPredicate(), message(), csound::message_level(), and WARNING_LEVEL.

Referenced by csound::Shell::open(), csound::pythonFuncWarning(), and csound::Chunk::read().

6.74.3.46 warn_text()

```
static void csound::System::warn_text (
            std::string text )  [inline], [static]
```

6.74.3.47 yieldThread()

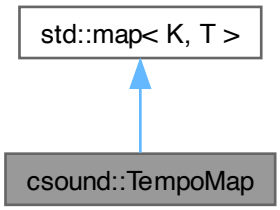
```
SILENCE_PUBLIC void csound::System::yieldThread ( )  [static]
```

Yields to the next waiting thread.

6.75 csound::TempoMap Class Reference

`#include <Midifile.hpp>`

Inheritance diagram for csound::TempoMap:



Public Member Functions

- double getCurrentSecondsPerTick (int tick)

Data Fields

- T elements
  STL member.
- K keys
  STL member.

6.75.1 Member Function Documentation

6.75.1.1 getCurrentSecondsPerTick()

```
double csound::TempoMap::getCurrentSecondsPerTick (
            int tick )
```

References csound::fundamentalDomainByPredicate().

6.75.2 Field Documentation

6.75.2.1 elements

```
T std::map< K, T >::elements  [inherited]
```

STL member.

6.75.2.2 keys

```
K std::map< K, T >::keys  [inherited]
```

STL member.

6.76 csound::ThreadLock Class Reference

Encapsulates a thread monitor, such as a Windows event handle.

```
#include <System.hpp>
```

Public Member Functions

- virtual void close ()
  Destroys the monitor.
- virtual void endWait ()
  Releases one thread that is waiting on the monitor.
- virtual bool isOpen ()
  Returns whether the monitor is open.
- virtual void open ()
  Creates and initializes the monitor.
- virtual void startWait (size_t timeoutMilliseconds=0)
  Waits until the monitor is notified by another thread.
- ThreadLock ()
- virtual ~ThreadLock ()

6.76.1 Detailed Description

Encapsulates a thread monitor, such as a Windows event handle.

6.76.2 Constructor & Destructor Documentation

6.76.2.1 ThreadLock()

```
SILENCE_PUBLIC csound::ThreadLock::ThreadLock ( )
```

6.76.2.2 ~ThreadLock()

```
SILENCE_PUBLIC csound::ThreadLock::~ThreadLock ( )  [virtual]
```

References close().

6.76.3 Member Function Documentation

6.76.3.1 close()

```
SILENCE_PUBLIC void csound::ThreadLock::close ( )  [virtual]
```

Destroys the monitor.

References csound::System::destroyThreadLock().

Referenced by ~ThreadLock().

6.76.3.2 endWait()

```
SILENCE_PUBLIC void csound::ThreadLock::endWait ( )  [virtual]
```

Releases one thread that is waiting on the monitor.

References csound::System::notifyThreadLock().

6.76.3.3 isOpen()

```
SILENCE_PUBLIC bool csound::ThreadLock::isOpen ( )  [virtual]
```

Returns whether the monitor is open.

6.76.3.4 open()

```
SILENCE_PUBLIC void csound::ThreadLock::open ( )  [virtual]
```

Creates and initializes the monitor.

The monitor is in a non-notified or unsignaled state.

References csound::System::createThreadLock().

6.76.3.5 startWait()

```
SILENCE_PUBLIC void csound::ThreadLock::startWait (
            size_t timeoutMilliseconds = 0 )  [virtual]
```

Waits until the monitor is notified by another thread.

Zero timeout means infinite timeout.

References csound::fundamentalDomainByPredicate(), and csound::System::waitThreadLock().

6.77 csound::TimeAfterComparator Struct Reference

Public Member Functions

- bool operator() (const Event &event)
- TimeAfterComparator (double time_)

Data Fields

- double time

6.77.1 Constructor & Destructor Documentation

6.77.1.1 TimeAfterComparator()

```
csound::TimeAfterComparator::TimeAfterComparator (
            double time_ )  [inline]
```

6.77.2 Member Function Documentation

6.77.2.1 operator()()

```
bool csound::TimeAfterComparator::operator() (
            const Event & event )  [inline]
```

References csound::fundamentalDomainByPredicate(), and time.

6.77.3 Field Documentation

6.77.3.1 time

```
double csound::TimeAfterComparator::time
```

Referenced by operator()().

6.78 csound::TimeAtComparator Struct Reference

Public Member Functions

- bool operator() (const Event &event)
- TimeAtComparator (double time_)

Data Fields

- double time

6.78.1 Constructor & Destructor Documentation

6.78.1.1 TimeAtComparator()

```
csound::TimeAtComparator::TimeAtComparator (
            double time_ )  [inline]
```

6.78.2 Member Function Documentation

6.78.2.1 operator()()

```
bool csound::TimeAtComparator::operator() (
            const Event & event )  [inline]
```

References csound::fundamentalDomainByPredicate(), and time.

6.78.3 Field Documentation

6.78.3.1 time

```
double csound::TimeAtComparator::time
```

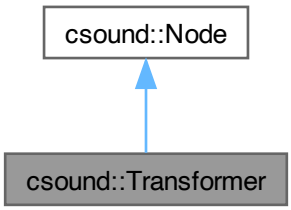
Referenced by operator()().

6.79 csound::Transformer Class Reference

Node that uses any callable to implement Node::transform.

`#include <Node.hpp>`

Inheritance diagram for csound::Transformer:



Public Member Functions

- virtual void addChild (Node ∗node)

  Adds an immediate child Node to this.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &score_from_this)

  Optionally generate notes into the score.
- virtual Node ∗ getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void transform (Score &score)

  Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

Data Fields

- std::function< void(csound::Score &) callable)
- std::vector< Node ∗ > children

  Child Nodes, if any.

Protected Attributes

- Eigen::MatrixXd localCoordinates

6.79.1 Detailed Description

Node that uses any callable to implement Node::transform.

This is particularly useful as the callable may be a closure that refers to objects outside of the music graph.

6.79.2 Member Function Documentation

6.79.2.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.79.2.2 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.79.2.3 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.79.2.4 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.79.2.5 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.79.2.6 generate()

```
void csound::Node::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented in csound::ExternalNode, csound::ScoreNode, csound::ChordLindenmayer, csound::MCRM, csound::Generator, csound::Random, csound::LispGenerator, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.79.2.7 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.79.2.8 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.79.2.9 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.79.2.10 transform()

```
virtual void csound::Transformer::transform (
            Score & score_from_children )  [inline], [virtual]
```

Optionally transform any or all notes produced by child nodes of this, which are in the score and in the global coordinate system.

The default implementation does nothing. Additional notes may also be generated.

Reimplemented from csound::Node.

6.79.2.11 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord and csound::Node::transform().

6.79.3 Field Documentation

6.79.3.1 callable

```
std::function<void(csound::Score &) csound::Transformer::callable)
```

6.79.3.2 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.79.3.3 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.80 csound::Turtle Struct Reference

```
#include <ChordLindenmayer.hpp>
```

Public Member Functions

- virtual std::string __str__ () const
- void initialize ()
- bool operator< (const Turtle &other) const
- Turtle & operator= (const Turtle &other)
- Turtle ()
- Turtle (const Turtle &other)
- virtual ~Turtle ()

Data Fields

- Chord chord
- Chord modality
- Event note
- Event orientation
- double rangeBass
- double rangeSize
- Scale scale
- int scaleDegree
- Event step
- double voicing

6.80.1 Constructor & Destructor Documentation

6.80.1.1 Turtle() [1/2]

```
csound::Turtle::Turtle ( )  [inline]
```

6.80.1.2 ~Turtle()

```
virtual csound::Turtle::~Turtle ( )  [inline], [virtual]
```

6.80.1.3 Turtle() [2/2]

```
csound::Turtle::Turtle (
            const Turtle & other )  [inline]
```

6.80.2 Member Function Documentation

6.80.2.1 __str__()

```
virtual std::string csound::Turtle::__str__ ( ) const  [inline], [virtual]
```

References csound::chord(), csound::note(), csound::printChord(), csound::scale(), and csound::Event::toString().

6.80.2.2 initialize()

```
void csound::Turtle::initialize ( )  [inline]
```

References csound::chord(), csound::chordForName(), csound::note(), csound::scale(), and csound::scaleForName().

Referenced by csound::ChordLindenmayer::initialize().

6.80.2.3 operator<()

```
bool csound::Turtle::operator< (
            const Turtle & other ) const  [inline]
```

References chord, csound::chord(), modality, note, csound::note(), orientation, rangeBass, rangeSize, scale, csound::scale(), scaleDegree, step, and voicing.

6.80.2.4 operator=()

```
Turtle & csound::Turtle::operator= (
            const Turtle & other )  [inline]
```

References chord, csound::chord(), modality, note, csound::note(), orientation, rangeBass, rangeSize, scale, csound::scale(), scaleDegree, step, and voicing.

6.80.3 Field Documentation

6.80.3.1 chord

```
Chord csound::Turtle::chord
```

Referenced by csound::ChordLindenmayer::chordOperation(), csound::ChordLindenmayer::getTurtleChord(), operator<(), operator=(), csound::ChordLindenmayer::scaleDegreeOperation(), csound::ChordLindenmayer::scaleOperation(), csound::ChordLindenmayer::scoreOperation(), and csound::ChordLindenmayer::setTurtleChord().

6.80.3.2 modality

```
Chord csound::Turtle::modality
```

Referenced by csound::ChordLindenmayer::chordOperation(), csound::ChordLindenmayer::getTurtleModality(), csound::ChordLindenmayer::modalityOperation(), operator<(), operator=(), and csound::ChordLindenmayer::setTurtleModality().

6.80.3.3 note

```
Event csound::Turtle::note
```

Referenced by csound::ChordLindenmayer::chordOperation(), csound::ChordLindenmayer::noteOperation(), operator<(), operator=(), and csound::ChordLindenmayer::scoreOperation().

6.80.3.4 orientation

```
Event csound::Turtle::orientation
```

Referenced by csound::ChordLindenmayer::noteOperation(), csound::ChordLindenmayer::noteOrientationOperation(), operator<(), and operator=().

6.80.3.5 rangeBass

```
double csound::Turtle::rangeBass
```

Referenced by operator<(), and operator=().

6.80.3.6 rangeSize

```
double csound::Turtle::rangeSize
```

Referenced by csound::ChordLindenmayer::chordOperation(), csound::ChordLindenmayer::equivalence(), operator<(), and operator=().

6.80.3.7 scale

```
Scale csound::Turtle::scale
```

Referenced by csound::ChordLindenmayer::getTurtleScale(), operator<(), operator=(), csound::ChordLindenmayer::scaleDegreeOperatio csound::ChordLindenmayer::scaleOperation(), csound::ChordLindenmayer::scoreOperation(), and csound::ChordLindenmayer::setTurtleS

6.80.3.8 scaleDegree

```
int csound::Turtle::scaleDegree
```

Referenced by csound::ChordLindenmayer::getTurtleScaleDegree(), operator<(), operator=(), csound::ChordLindenmayer::scaleDegreeC and csound::ChordLindenmayer::setTurtleScaleDegree().

6.80.3.9 step

```
Event csound::Turtle::step
```

Referenced by csound::ChordLindenmayer::noteOperation(), csound::ChordLindenmayer::noteStepOperation(), operator<(), and operator=().

6.80.3.10 voicing

```
double csound::Turtle::voicing
```

Referenced by csound::ChordLindenmayer::chordOperation(), operator<(), operator=(), and csound::ChordLindenmayer::voicingOperatio

6.81 csound::Voicelead Class Reference

This class contains facilities for voiceleading, harmonic progression, and identifying chord types.

```
#include <Voicelead.hpp>
```

Static Public Member Functions

- static bool addOctave (const std::vector< double > &lowestVoicing, std::vector< double > &newVoicing, size_t maximumPitch, size_t divisionsPerOctave)

  Add an octave to a voicing; can be iterated to enumerate the voicings of a chord.
- static bool areParallel (const std::vector< double > &chord1, const std::vector< double > &chord2)
- static std::vector< double > chordToPTV (const std::vector< double > &chord, size_t lowestPitch, size_t highestPitch, size_t divisionsPerOctave=12)

  Return the voiced chord for the prime chord index P, transposition T, and voicing index V within the specified range for the indicated number of tones per octave.
- static const std::vector< double > & closer (const std::vector< double > &source, const std::vector< double > &destination1, const std::vector< double > &destination2, bool avoidParallels)

  Return the closer, first by smoothness then by simplicity., of the voiceleadings between source and either destination1 or destination2, optionally avoiding parallel fifths.
- static const std::vector< double > closest (const std::vector< double > &source, const std::vector< std::vector< double > > &destinations, bool avoidParallels)

  Return the closest voiceleading within the specified range, first by smoothness then by simplicity, between the source chord any of the destination chords, optionally avoiding parallel fifths.
- static double closestPitch (double pitch, const std::vector< double > &pitches)

  Return the pitch in pitches that is closest to the specified pitch.
- static double conformToPitchClassSet (double pitch, const std::vector< double > &pcs, size_t divisionsPer↩ Octave=12)

  Return the pitch that results from making the minimum adjustment to the pitch-class of the pitch argument that is required to make its pitch-class the same as one of the pitch-classes in the pitch-class set argument.
- static double cToM (double C, size_t divisionsPerOctaven=12)

  Return M = sum over pitch-classes of (2 ^ pitch-class) (multiplicative monoid for pitch-class sets) for C = (sum over pitch-classes of (pitch-class ^ 2)) - 1 (additive cyclic group for non-empty pitch-class sets).
- static double cToP (double C, size_t divisionsPerOctave=12)

  Return C = (sum over pitch-classes of (pitch-class ^ 2)) - 1 (additive cyclic group for non-empty pitch-class sets) for P = index of prime chords.
- static double euclideanDistance (const std::vector< double > &chord1, const std::vector< double > &chord2)

  Return the Euclidean distance between two chords, which must have the same number of voices.
- static double I (double p, double n)

  Return the pitch-class inversion of pitch p by n semitones.
- static std::vector< double > I_vector (const std::vector< double > &c, double n)

  Return the pitch-class inversion of chord c by n semitones.
- static bool Iform (const std::vector< double > &X, const std::vector< double > &Y, double g=1.0)

  Return whether chord Y is an inverted form of chord X; g is the generator of inversions.
- static void initializePrimeChordsForDivisionsPerOctave (size_t divisionsPerOctave)
- static std::vector< std::vector< double > > inversions (const std::vector< double > &chord)

  Return as many inversions of the pitch-classes in the chord as there are voices in the chord.
- static std::vector< double > invert (const std::vector< double > &chord)

  Invert by rotating the chord and adding an octave to its last pitch.
- static std::vector< double > K (const std::vector< double > &c)

  Invert chord c by exchange.
- static double mToC (double M, size_t divisionsPerOctave)

  Return C = (sum over pitch-classes of (pitch-class ^ 2)) - 1 (additive cyclic group for non-empty pitch-class sets) for M = sum over pitch-classes of (2 ^ pitch-class) (multiplicative monoid for pitch-class sets).
- static std::vector< double > mToPitchClassSet (double pcn, size_t divisionsPerOctave=12)

Convert a pitch-class set number M = sum over pitch-classes of (2 ^ pitch-class) to a pitch-class set chord.

- static double nameToC (std::string name, size_t divisionsPerOctave_)

  Return C = (sum over pitch-classes of (pitch-class ^ 2)) - 1 (additive cyclic group for pitch-class sets) for the named pitch-class set.

- static std::vector< std::vector< double > > nonBijectiveVoicelead (const std::vector< double > &sourceChord, const std::vector< double > &targetPitchClassSet, size_t divisionsPerOctave=12)

  Return the closest crossing-free, non-bijective voiceleading from the source chord to the pitch-classes in the target chord, using Dimitri Tymoczko's linear programming algorithm.

- static std::vector< double > normalChord (const std::vector< double > &chord)

  Return the normal chord: that inversion of the pitch-classes in the chord which is closest to the orthogonal axis of the Tonnetz for that chord.

- static std::vector< double > orderedPcs (const std::vector< double > &chord, size_t divisionsPerOctave=12)

  Return a copy of the chord where each pitch is replaced by its corresponding pitch-class.

- static std::vector< double > pAndTtoPitchClassSet (double prime, double transposition, size_t divisionsPer↩ Octave=12)

  Convert a prime chord number and transposition to a pitch-class set.

- static double pc (double pitch, size_t divisionsPerOctave=12)

  Return the pitch-class of the pitch.

- static std::vector< double > pcs (const std::vector< double > &chord, size_t divisionsPerOctave=12)

  Return the chord as the list of its pitch-classes.

- static double pitchClassSetToM (const std::vector< double > &chord, size_t divisionsPerOctave=12)

  Convert a chord to a pitch-class set number M = sum over pitch-classes of (2 ^ pitch-class).

- static std::vector< double > pitchClassSetToPandT (const std::vector< double > &pcs, size_t divisionsPer↩ Octave=12)

  Convert a pitch-class set to a prime chord number and a transposition.

- static std::vector< double > primeChord (const std::vector< double > &chord)

  Return the prime chord: that inversion of the pitch-classes in the chord which is closest to the orthogonal axis of the Tonnetz for that chord, transposed so that its lowest pitch is at the origin.

- static double pToC (double Z, size_t divisionsPerOctave=12)

  Return P = index of prime chords for C = (sum over pitch-classes of (pitch-class ^ 2)) - 1 (additive cyclic group for non-empty pitch-class sets).

- static std::vector< double > pToPrimeChord (double P, size_t divisionsPerOctave=12)

  Return the prime chord for the index P.

- static std::vector< double > ptvToChord (size_t P, size_t T, size_t V_, size_t lowest, size_t range, size_t divisionsPerOctave=12)

  Return the voiced chord for the prime chord index P, transposition T, and voicing index V within the specified range for the indicated number of tones per octave.

- static std::vector< double > Q (const std::vector< double > &c, double n, const std::vector< double > &s, double g=1.0)

  Contextually transpose chord c with respect to chord s by n semitones; g is the generator of transpositions.

- static std::vector< double > recursiveVoicelead (const std::vector< double > &source, const std::vector< double > &targetPitchClassSet, double lowest, double range, bool avoidParallels, size_t divisionsPerOctave=12)

  Return the closest voiceleading within the specified range, first by smoothness then by simplicity, between the source chord and the target pitch-class set, optionally avoiding parallel fifths.

- static std::vector< double > rotate (const std::vector< double > &chord)

  Return the chord with the first note rotated to the last note.

- static std::vector< std::vector< double > > rotations (const std::vector< double > &chord)

  Return the set of all rotations of the chord.

- static const std::vector< double > & simpler (const std::vector< double > &source, const std::vector< double > &destination1, const std::vector< double > &destination2, bool avoidParallels)

Return the simpler (fewer motions) of the voiceleadings between source chord and either destination1 or destination2, optionally avoiding parallel fifths.

- static double smoothness (const std::vector< double > &chord1, const std::vector< double > &chord2)

  Return the smoothness (distance by taxicab or L1 norm) of the voiceleading between chord1 and chord2.
- static std::vector< double > sortByAscendingDistance (const std::vector< double > &chord, size_t divisions↩ PerOctave=12)

  Return a copy of the chord sorted by ascending distance from its first pitch-class.
- static double T (double p, double n)

  Return the pitch-class transposition of pitch p by n semitones.
- static std::vector< double > T_vector (const std::vector< double > &c, double n)

  Return the pitch-class transposition of chord c by n semitones.
- static bool Tform (const std::vector< double > &X, const std::vector< double > &Y, double g=1.0)

  Return whether chord Y is a transposed form of chord X; g is the generator of transpositions.
- static std::vector< double > toOrigin (const std::vector< double > &chord)

  Return the chord transposed so its lowest pitch is at the origin.
- static std::vector< double > transpose (const std::vector< double > &chord, double semitones)

  Return the chord transposed by the indicated number of semitones.
- static std::vector< double > uniquePcs (const std::vector< double > &chord, size_t divisionsPerOctave=12)

  Return the chord as the list of its pitch-classes.
- static std::vector< double > voicelead (const std::vector< double > &source, const std::vector< double > &targetPitchClassSet, double lowest, double range, bool avoidParallels, size_t divisionsPerOctave=12)

  Return the closest voiceleading within the specified range, first by smoothness then by simplicity, between the source chord and the target pitch-class set, optionally avoiding parallel fifths.
- static std::vector< double > voiceleading (const std::vector< double > &chord1, const std::vector< double > &chord2)

  Return the voice-leading vector (difference) between chord1 and chord2.
- static std::vector< std::vector< double > > voicings (const std::vector< double > &chord, double lowest, double highest, size_t divisionsPerOctave)

  Return an enumeration of all voicings of the chord that are greater than or equal to the lowest pitch, and less than the highest pitch, by adding octaves.
- static std::vector< double > wrap (const std::vector< double > &chord, size_t lowestPitch, size_t highestPitch, size_t divisionsPerOctave=12)

  Wrap chord tones that exceed the highest pitch around to the bottom of the range orbifold.

Static Public Attributes

- static const double semitonesPerOctave = double(12)

  Size of the octave in semitones.

6.81.1 Detailed Description

This class contains facilities for voiceleading, harmonic progression, and identifying chord types.

See: http://ruccas.org/pub/Gogins/music_atoms.pdf

6.81.2 Member Function Documentation

6.81.2.1 addOctave()

```
bool csound::Voicelead::addOctave (
            const std::vector< double > & lowestVoicing,
            std::vector< double > & newVoicing,
            size_t maximumPitch,
            size_t divisionsPerOctave )  [static]
```

Add an octave to a voicing; can be iterated to enumerate the voicings of a chord.

The lowest voicing must initially be set equal to the original voicing. The algorithm treats a chord as a 'numeral' that increments with a radix equal to the number of octaves in the total range of pitches. Returns an empty voicing if adding an octave would create a voicing that exceeds the maximum pitch, i.e. when the highest-order voice needs to 'carry.'

References csound::debug, and csound::fundamentalDomainByPredicate().

Referenced by chordToPTV(), ptvToChord(), and voicings().

6.81.2.2 areParallel()

```
bool csound::Voicelead::areParallel (
            const std::vector< double > & chord1,
            const std::vector< double > & chord2 )  [static]
```

References csound::debug, csound::fundamentalDomainByPredicate(), and voiceleading().

Referenced by closer().

6.81.2.3 chordToPTV()

```
std::vector< double > csound::Voicelead::chordToPTV (
            const std::vector< double > & chord,
            size_t lowestPitch,
            size_t highestPitch,
            size_t divisionsPerOctave = 12 )  [static]
```

Return the voiced chord for the prime chord index P, transposition T, and voicing index V within the specified range for the indicated number of tones per octave.

The algorithm finds the zero voicing (the lowest octave transposition of the normal chord of the chord that is no lower than the lowest pitch, which has voicing index V = 0) and the zero iterator (the lowest (in all voices) unordered voicing of the chord that is no lower (in all voices) than the lowest pitch, which has enumeration index = 0). Thus, the V of a voicing equals the enumeration index of that voicing minus the enumeration index of the zero voicing. The algorithm enumerates the voicings until the chord is matched.

References addOctave(), csound::fundamentalDomainByPredicate(), csound::iterator(), normalChord(), pcs(), pitchClassSetToPandT(), and csound::sort().

Referenced by csound::Score::getPTV().

6.81.2.4 closer()

```
const std::vector< double > & csound::Voicelead::closer (
            const std::vector< double > & source,
            const std::vector< double > & destination1,
            const std::vector< double > & destination2,
            bool avoidParallels )  [static]
```

Return the closer, first by smoothness then by simplicity., of the voiceleadings between source and either destination1 or destination2, optionally avoiding parallel fifths.

References areParallel(), csound::fundamentalDomainByPredicate(), simpler(), and smoothness().

Referenced by closest(), and csound::recursiveVoicelead_().

6.81.2.5 closest()

```
const std::vector< double > csound::Voicelead::closest (
            const std::vector< double > & source,
            const std::vector< std::vector< double > > & destinations,
            bool avoidParallels )  [static]
```

Return the closest voiceleading within the specified range, first by smoothness then by simplicity, between the source chord any of the destination chords, optionally avoiding parallel fifths.

References closer(), and csound::fundamentalDomainByPredicate().

Referenced by voicelead().

6.81.2.6 closestPitch()

```
double csound::Voicelead::closestPitch (
            double pitch,
            const std::vector< double > & pitches )  [static]
```

Return the pitch in pitches that is closest to the specified pitch.

References csound::fundamentalDomainByPredicate().

Referenced by conformToPitchClassSet(), and csound::Score::setPitches().

6.81.2.7 conformToPitchClassSet()

```
double csound::Voicelead::conformToPitchClassSet (
            double pitch,
            const std::vector< double > & pcs,
            size_t divisionsPerOctave = 12 )  [static]
```

Return the pitch that results from making the minimum adjustment to the pitch-class of the pitch argument that is required to make its pitch-class the same as one of the pitch-classes in the pitch-class set argument.

I.e., "round up or down" to make the pitch fit into a chord or scale.

References closestPitch(), csound::fundamentalDomainByPredicate(), pc(), pcs(), and csound::round().

Referenced by csound::Score::setPitchClassSet(), and csound::Score::setVoicing().

6.81.2.8 cToM()

```
double csound::Voicelead::cToM (
            double C,
            size_t divisionsPerOctaven = 12 )  [static]
```

Return M = sum over pitch-classes of (2 ^ pitch-class) (multiplicative monoid for pitch-class sets) for C = (sum over pitch-classes of (pitch-class ^ 2)) - 1 (additive cyclic group for non-empty pitch-class sets).

References csound::fundamentalDomainByPredicate().

Referenced by csound::VoiceleadingNode::apply(), cToP(), and initializePrimeChordsForDivisionsPerOctave().

6.81.2.9 cToP()

```
double csound::Voicelead::cToP (
            double C,
            size_t divisionsPerOctave = 12 )  [static]
```

Return C = (sum over pitch-classes of (pitch-class ^ 2)) - 1 (additive cyclic group for non-empty pitch-class sets) for P = index of prime chords.

If an exact match is not found the closest match is returned.

References cToM(), csound::fundamentalDomainByPredicate(), initializePrimeChordsForDivisionsPerOctave(), mToPitchClassSet(), csound::pForPrimeChordsForDivisionsPerOctave, and primeChord().

Referenced by pitchClassSetToPandT().

6.81.2.10 euclideanDistance()

```
double csound::Voicelead::euclideanDistance (
            const std::vector< double > & chord1,
            const std::vector< double > & chord2 )  [static]
```

Return the Euclidean distance between two chords, which must have the same number of voices.

References csound::fundamentalDomainByPredicate().

Referenced by normalChord().

6.81.2.11 I()

```
double csound::Voicelead::I (
            double p,
            double n )  [static]
```

Return the pitch-class inversion of pitch p by n semitones.

References csound::fundamentalDomainByPredicate(), and pc().

Referenced by I_vector().

6.81.2.12 I_vector()

```
std::vector< double > csound::Voicelead::I_vector (
            const std::vector< double > & c,
            double n )  [static]
```

Return the pitch-class inversion of chord c by n semitones.

References csound::fundamentalDomainByPredicate(), I(), and csound::sort().

Referenced by Iform(), and K().

6.81.2.13 Iform()

```
bool csound::Voicelead::Iform (
            const std::vector< double > & X,
            const std::vector< double > & Y,
            double g = 1.0 )  [static]
```

Return whether chord Y is an inverted form of chord X; g is the generator of inversions.

References csound::fundamentalDomainByPredicate(), I_vector(), and pcs().

Referenced by Q().

6.81.2.14 initializePrimeChordsForDivisionsPerOctave()

```
void csound::Voicelead::initializePrimeChordsForDivisionsPerOctave (
            size_t divisionsPerOctave )  [static]
```

References csound::cForPForDivisionsPerOctave, csound::chord(), cToM(), csound::fundamentalDomainByPredicate(), mToPitchClassSet(), normalChord(), csound::pForCForDivisionsPerOctave, csound::pForPrimeChordsForDivisionsPerOctave, csound::primeChordsForDivisionsPerOctave, and toOrigin().

Referenced by cToP(), pToC(), and pToPrimeChord().

6.81.2.15 inversions()

```
std::vector< std::vector< double > > csound::Voicelead::inversions (
            const std::vector< double > & chord )  [static]
```

Return as many inversions of the pitch-classes in the chord as there are voices in the chord.

References csound::chord(), csound::fundamentalDomainByPredicate(), invert(), and pcs().

Referenced by normalChord().

6.81.2.16 invert()

```
std::vector< double > csound::Voicelead::invert (
            const std::vector< double > & chord )  [static]
```

Invert by rotating the chord and adding an octave to its last pitch.

References csound::chord(), and csound::fundamentalDomainByPredicate().

Referenced by inversions(), and recursiveVoicelead().

6.81.2.17 K()

```
std::vector< double > csound::Voicelead::K (
            const std::vector< double > & c )  [static]
```

Invert chord c by exchange.

References csound::fundamentalDomainByPredicate(), and I_vector().

Referenced by csound::Score::setK(), csound::Score::setKL(), and csound::Score::setKV().

6.81.2.18 mToC()

```
double csound::Voicelead::mToC (
            double M,
            size_t divisionsPerOctave )  [static]
```

Return C = (sum over pitch-classes of (pitch-class ^ 2)) - 1 (additive cyclic group for non-empty pitch-class sets) for M = sum over pitch-classes of (2 ^ pitch-class) (multiplicative monoid for pitch-class sets).

References csound::fundamentalDomainByPredicate().

Referenced by nameToC(), and pitchClassSetToPandT().

6.81.2.19 mToPitchClassSet()

```
std::vector< double > csound::Voicelead::mToPitchClassSet (
            double pcn,
            size_t divisionsPerOctave = 12 )  [static]
```

Convert a pitch-class set number M = sum over pitch-classes of (2 ^ pitch-class) to a pitch-class set chord.

References csound::fundamentalDomainByPredicate(), pcs(), and csound::round().

Referenced by csound::VoiceleadingNode::apply(), cToP(), and initializePrimeChordsForDivisionsPerOctave().

6.81.2.20 nameToC()

```
double csound::Voicelead::nameToC (
            std::string name,
            size_t divisionsPerOctave_ )  [static]
```

Return C = (sum over pitch-classes of (pitch-class ^ 2)) - 1 (additive cyclic group for pitch-class sets) for the named pitch-class set.

References csound::fundamentalDomainByPredicate(), mToC(), and csound::Conversions::nameToM().

Referenced by csound::VoiceleadingNode::C_name(), csound::VoiceleadingNode::CL_name(), and csound::VoiceleadingNode::CV_name

6.81.2.21 nonBijectiveVoicelead()

```
std::vector< std::vector< double > > csound::Voicelead::nonBijectiveVoicelead (
            const std::vector< double > & sourceChord,
            const std::vector< double > & targetPitchClassSet,
            size_t divisionsPerOctave = 12 )  [static]
```

Return the closest crossing-free, non-bijective voiceleading from the source chord to the pitch-classes in the target chord, using Dimitri Tymoczko's linear programming algorithm.

Because voices can be doubled, the source chord is returned along with result. The algorithm does not avoid parallel motions, and does not maintain the original order of the voices. The return value contains the original chord, the voiceleading vector, and the resulting chord, in that order.

References csound::createMatrix(), csound::fundamentalDomainByPredicate(), orderedPcs(), rotations(), and sortByAscendingDistance().

Referenced by csound::Score::voicelead(), and csound::Score::voicelead().

6.81.2.22 normalChord()

```
std::vector< double > csound::Voicelead::normalChord (
            const std::vector< double > & chord )  [static]
```

Return the normal chord: that inversion of the pitch-classes in the chord which is closest to the orthogonal axis of the Tonnetz for that chord.

Similar to, but not identical with, "normal form."

References csound::chord(), euclideanDistance(), csound::fundamentalDomainByPredicate(), inversions(), normalChord(), and toOrigin().

Referenced by chordToPTV(), initializePrimeChordsForDivisionsPerOctave(), normalChord(), pitchClassSetToPandT(), primeChord(), and ptvToChord().

6.81.2.23 orderedPcs()

```
std::vector< double > csound::Voicelead::orderedPcs (
            const std::vector< double > & chord,
            size_t divisionsPerOctave = 12 )  [static]
```

Return a copy of the chord where each pitch is replaced by its corresponding pitch-class.

The voices remain in their original order.

References csound::chord(), csound::fundamentalDomainByPredicate(), and pc().

Referenced by nonBijectiveVoicelead().

6.81.2.24 pAndTtoPitchClassSet()

```
std::vector< double > csound::Voicelead::pAndTtoPitchClassSet (
            double prime,
            double transposition,
            size_t divisionsPerOctave = 12 )  [static]
```

Convert a prime chord number and transposition to a pitch-class set.

References csound::fundamentalDomainByPredicate(), pc(), pToPrimeChord(), and T().

Referenced by ptvToChord(), and csound::Score::setPT().

6.81.2.25 pc()

```
double csound::Voicelead::pc (
            double pitch,
            size_t divisionsPerOctave = 12 )  [static]
```

Return the pitch-class of the pitch.

Pitch is measured in semitones, and the octave is always 12 semitones, so the pitch-class is the pitch modulo 12. If the pitch is an integral number of semitones, and the number of divisions per octave is also 12, then the pitch-class of a pitch is an integer. If the pitch is not an integral number of semitones, or the number of divisions per octave is not 12, then the pitch-class is not necessarily an integer.

References csound::fundamentalDomainByPredicate().

Referenced by csound::AscendingDistanceComparator::ascendingDistance(), conformToPitchClassSet(), csound::Score::getVoicing(), I(), orderedPcs(), pAndTtoPitchClassSet(), pcs(), pitchClassSetToM(), csound::Score::setVoicing(), T(), T_vector(), and uniquePcs().

6.81.2.26 pcs()

```
std::vector< double > csound::Voicelead::pcs (
            const std::vector< double > & chord,
            size_t divisionsPerOctave = 12 )  [static]
```

Return the chord as the list of its pitch-classes.

Although the list is nominally unordered, it is returned sorted in ascending order. Note that pitch-classes may be doubled.

References csound::chord(), csound::debug, csound::fundamentalDomainByPredicate(), and pc().

Referenced by chordToPTV(), conformToPitchClassSet(), Iform(), inversions(), mToPitchClassSet(), ptvToChord(), recursiveVoicelead(), rotations(), csound::Score::setVoicing(), Tform(), csound::Score::voicelead(), csound::Score::voicelead(), and voicings().

6.81.2.27 pitchClassSetToM()

```
double csound::Voicelead::pitchClassSetToM (
            const std::vector< double > & chord,
            size_t divisionsPerOctave = 12 )  [static]
```

Convert a chord to a pitch-class set number M = sum over pitch-classes of (2 ^ pitch-class).

These numbers form a multiplicative monoid. Arithmetic on this monoid can perform many harmonic and other manipulations of pitch.

References csound::chord(), csound::fundamentalDomainByPredicate(), and pc().

Referenced by pitchClassSetToPandT().

6.81.2.28 pitchClassSetToPandT()

```
std::vector< double > csound::Voicelead::pitchClassSetToPandT (
            const std::vector< double > & pcs,
            size_t divisionsPerOctave = 12 )  [static]
```

Convert a pitch-class set to a prime chord number and a transposition.

Note that the prime chord numbers, and transpositions, each form an additive cyclic group.

References cToP(), csound::fundamentalDomainByPredicate(), mToC(), normalChord(), pitchClassSetToM(), and toOrigin().

Referenced by csound::VoiceleadingNode::apply(), chordToPTV(), csound::Score::getPT(), csound::Score::setKV(), and csound::Score::setQV().

6.81.2.29 primeChord()

```
std::vector< double > csound::Voicelead::primeChord (
            const std::vector< double > & chord )  [static]
```

Return the prime chord: that inversion of the pitch-classes in the chord which is closest to the orthogonal axis of the Tonnetz for that chord, transposed so that its lowest pitch is at the origin.

Similar to, but not identical with, "prime form."

References csound::chord(), normalChord(), and toOrigin().

Referenced by cToP().

6.81.2.30 pToC()

```
double csound::Voicelead::pToC (
            double Z,
            size_t divisionsPerOctave = 12 )  [static]
```

Return P = index of prime chords for C = (sum over pitch-classes of (pitch-class ^ 2)) - 1 (additive cyclic group for non-empty pitch-class sets).

If an exact match is not found the closest match is returned.

References csound::cForPForDivisionsPerOctave, csound::fundamentalDomainByPredicate(), initializePrimeChordsForDivisionsPerOctav
and csound::primeChordsForDivisionsPerOctave.

6.81.2.31 pToPrimeChord()

```
std::vector< double > csound::Voicelead::pToPrimeChord (
            double P,
            size_t divisionsPerOctave = 12 )  [static]
```

Return the prime chord for the index P.

References csound::fundamentalDomainByPredicate(), initializePrimeChordsForDivisionsPerOctave(), csound::primeChordsForDivisionsI
and csound::round().

Referenced by pAndTtoPitchClassSet().

6.81.2.32 ptvToChord()

```
std::vector< double > csound::Voicelead::ptvToChord (
            size_t P,
            size_t T,
            size_t V_,
            size_t lowest,
            size_t range,
            size_t divisionsPerOctave = 12 )  [static]
```

Return the voiced chord for the prime chord index P, transposition T, and voicing index V within the specified range for the indicated number of tones per octave.

The algorithm finds the zero voicing (the lowest octave transposition of the normal chord of the chord that is no lower than the lowest pitch, which has voicing index V = 0) and the zero iterator (the lowest (in all voices) unordered voicing of the chord that is no lower (in all voices) than the lowest pitch, which has enumeration index = 0). Thus, V of a voicing equals the enumeration index of that voicing minus the enumeration index of the zero voicing. The algorithm enumerates the voicings, and thus V, until V is matched. If V is greater than the maximum V, its modulus is used.

References addOctave(), csound::fundamentalDomainByPredicate(), csound::iterator(), normalChord(), pAndTtoPitchClassSet(), pcs(), csound::sort(), and T().

Referenced by csound::Score::setPTV().

6.81.2.33 Q()

```
std::vector< double > csound::Voicelead::Q (
            const std::vector< double > & c,
            double n,
            const std::vector< double > & s,
            double g = 1.0 )  [static]
```

Contextually transpose chord c with respect to chord s by n semitones; g is the generator of transpositions.

References csound::fundamentalDomainByPredicate(), Iform(), T_vector(), and Tform().

Referenced by csound::Score::setQ(), csound::Score::setQL(), and csound::Score::setQV().

6.81.2.34 recursiveVoicelead()

```
std::vector< double > csound::Voicelead::recursiveVoicelead (
            const std::vector< double > & source,
            const std::vector< double > & targetPitchClassSet,
            double lowest,
            double range,
            bool avoidParallels,
            size_t divisionsPerOctave = 12 )  [static]
```

Return the closest voiceleading within the specified range, first by smoothness then by simplicity, between the source chord and the target pitch-class set, optionally avoiding parallel fifths.

Bijective voiceleading first by closeness, then by simplicity, with optional avoidance of parallel fifths.

The algorithm uses a brute-force search through all unordered chords, which are recursively enumerated, fitting the target pitch-class set within the specified range. Although the time complexity is exponential, the algorithm is still usable for non-real-time operations in most cases of musical interest.

References csound::debug, csound::fundamentalDomainByPredicate(), invert(), csound::iterator(), pcs(), csound::pitchRotations(), csound::recursiveVoicelead_(), and voiceleading().

6.81.2.35 rotate()

```
std::vector< double > csound::Voicelead::rotate (
            const std::vector< double > & chord )  [static]
```

Return the chord with the first note rotated to the last note.

References csound::chord(), and csound::fundamentalDomainByPredicate().

Referenced by csound::pitchRotations(), and rotations().

6.81.2.36 rotations()

```
std::vector< std::vector< double > > csound::Voicelead::rotations (
            const std::vector< double > & chord )  [static]
```

Return the set of all rotations of the chord.

References csound::chord(), csound::debug, csound::fundamentalDomainByPredicate(), pcs(), and rotate().

Referenced by nonBijectiveVoicelead().

6.81.2.37 simpler()

```
const std::vector< double > & csound::Voicelead::simpler (
            const std::vector< double > & source,
            const std::vector< double > & destination1,
            const std::vector< double > & destination2,
            bool avoidParallels )  [static]
```

Return the simpler (fewer motions) of the voiceleadings between source chord and either destination1 or destination2, optionally avoiding parallel fifths.

References csound::fundamentalDomainByPredicate(), and voiceleading().

Referenced by closer().

6.81.2.38 smoothness()

```
double csound::Voicelead::smoothness (
            const std::vector< double > & chord1,
            const std::vector< double > & chord2 )  [static]
```

Return the smoothness (distance by taxicab or L1 norm) of the voiceleading between chord1 and chord2.

References csound::fundamentalDomainByPredicate().

Referenced by closer(), and csound::createMatrix().

6.81.2.39 sortByAscendingDistance()

```
std::vector< double > csound::Voicelead::sortByAscendingDistance (
            const std::vector< double > & chord,
            size_t divisionsPerOctave = 12 )  [static]
```

Return a copy of the chord sorted by ascending distance from its first pitch-class.

References csound::chord(), and csound::fundamentalDomainByPredicate().

Referenced by nonBijectiveVoicelead().

6.81.2.40 T()

```
double csound::Voicelead::T (
            double p,
            double n )  [static]
```

Return the pitch-class transposition of pitch p by n semitones.

References csound::fundamentalDomainByPredicate(), and pc().

Referenced by pAndTtoPitchClassSet(), and ptvToChord().

6.81.2.41 T_vector()

```
std::vector< double > csound::Voicelead::T_vector (
            const std::vector< double > & c,
            double n )  [static]
```

Return the pitch-class transposition of chord c by n semitones.

References csound::fundamentalDomainByPredicate(), pc(), and csound::sort().

Referenced by Q(), and Tform().

6.81.2.42 Tform()

```
bool csound::Voicelead::Tform (
            const std::vector< double > & X,
            const std::vector< double > & Y,
            double g = 1.0 )  [static]
```

Return whether chord Y is a transposed form of chord X; g is the generator of transpositions.

References csound::fundamentalDomainByPredicate(), pcs(), and T_vector().

Referenced by Q().

6.81.2.43 toOrigin()

```
std::vector< double > csound::Voicelead::toOrigin (
            const std::vector< double > & chord )  [static]
```

Return the chord transposed so its lowest pitch is at the origin.

References csound::chord(), and csound::fundamentalDomainByPredicate().

Referenced by initializePrimeChordsForDivisionsPerOctave(), normalChord(), pitchClassSetToPandT(), and primeChord().

6.81.2.44 transpose()

```
std::vector< double > csound::Voicelead::transpose (
            const std::vector< double > & chord,
            double semitones )  [static]
```

Return the chord transposed by the indicated number of semitones.

References csound::chord(), and csound::fundamentalDomainByPredicate().

6.81.2.45 uniquePcs()

```
std::vector< double > csound::Voicelead::uniquePcs (
            const std::vector< double > & chord,
            size_t divisionsPerOctave = 12 )  [static]
```

Return the chord as the list of its pitch-classes.

Although the list is nominally unordered, it is returned sorted in ascending order. Note that pitch-classes are NOT doubled.

References csound::chord(), csound::fundamentalDomainByPredicate(), pc(), and csound::sort().

Referenced by csound::Score::getPT(), csound::Score::getVoicing(), csound::Score::setK(), csound::Score::setKL(), csound::Score::setKV(), csound::Score::setPT(), csound::Score::setPTV(), csound::Score::setQ(), csound::Score::setQL(), csound::Score::setQV(), csound::Score::voicelead(), and csound::Score::voicelead().

6.81.2.46 voicelead()

```
std::vector< double > csound::Voicelead::voicelead (
            const std::vector< double > & source_,
            const std::vector< double > & target_,
            double lowest,
            double range,
            bool avoidParallels,
            size_t divisionsPerOctave = 12 )  [static]
```

Return the closest voiceleading within the specified range, first by smoothness then by simplicity, between the source chord and the target pitch-class set, optionally avoiding parallel fifths.

Bijective voiceleading first by closeness, then by simplicity, with optional avoidance of parallel fifths.

The algorithm uses a brute-force search through all unordered chords, which are stored in a cache, fitting the target pitch-class set within the specified range. Although the time complexity is exponential, this is still usable for non-real-time operations in most cases of musical interest.

If source and target are the same, parallel fifths are not avoided.

References closest(), csound::debug, csound::fundamentalDomainByPredicate(), voiceleading(), and voicings().

6.81.2.47 voiceleading()

```
std::vector< double > csound::Voicelead::voiceleading (
            const std::vector< double > & chord1,
            const std::vector< double > & chord2 )  [static]
```

Return the voice-leading vector (difference) between chord1 and chord2.

References csound::fundamentalDomainByPredicate().

Referenced by areParallel(), csound::createMatrix(), recursiveVoicelead(), simpler(), and voicelead().

6.81.2.48 voicings()

```
std::vector< std::vector< double > > csound::Voicelead::voicings (
            const std::vector< double > & chord,
            double lowest,
            double highest,
            size_t divisionsPerOctave )  [static]
```

Return an enumeration of all voicings of the chord that are greater than or equal to the lowest pitch, and less than the highest pitch, by adding octaves.

Voicings are ordered, but note that normally in this module chords are considered to be unordered. Note that complex chords and/or wide ranges may require more memory than is available. The index of voicings V forms an additive cyclic group. Arithmetic on this group can perform many operations on the voices of the chord such as revoicing, arpeggiation, and so on.

References addOctave(), csound::fundamentalDomainByPredicate(), csound::iterator(), pcs(), csound::sort(), and voicings().

Referenced by voicelead(), and voicings().

6.81.2.49 wrap()

```
std::vector< double > csound::Voicelead::wrap (
            const std::vector< double > & chord,
            size_t lowestPitch,
            size_t highestPitch,
            size_t divisionsPerOctave = 12 )  [static]
```

Wrap chord tones that exceed the highest pitch around to the bottom of the range orbifold.

References csound::chord(), and csound::fundamentalDomainByPredicate().

6.81.3 Field Documentation

6.81.3.1 semitonesPerOctave

```
const double csound::Voicelead::semitonesPerOctave = double(12)  [static]
```

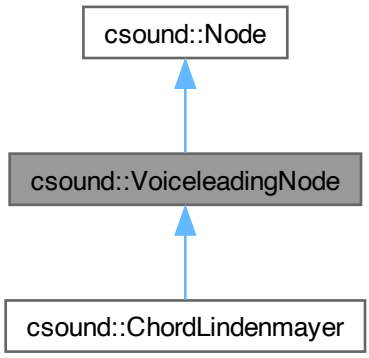
Size of the octave in semitones.

6.82 csound::VoiceleadingNode Class Reference

This node class imposes a sequence of one or more "voice-leading" operations upon the pitches of notes produced by children of this node, within a segment of the notes.

`#include <VoiceleadingNode.hpp>`

Inheritance diagram for csound::VoiceleadingNode:



Public Member Functions

- virtual void addChild (Node *node)

  Adds an immediate child Node to this.
- virtual void apply (Score &score, const VoiceleadingOperation &priorOperation, const VoiceleadingOperation &currentOperation)

  Apply the current voice-leading operation to the score, within the specified range of notes.
- void C (double time, double C_)

  Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified prime chord and transposition.
- void C_name (double time, std::string C_)

  Same as C, except the chord can be specified by jazz-type name (e.g.
- virtual size_t childCount () const

  Returns the number of immediate children of this.
- void chord (const csound::Chord &chord, double time)

  Apply the specified chord to the current segement.
- void chordVoiceleading (const csound::Chord &chord, double time, bool avoid_parallels)

  Apply the specified chord to the current segement, using the closest voice-leading from the pitches of the previous segment.
- void CL (double time, double C_, bool avoidParallels=true)

Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified chord; the voicing of the chord will be the smoothest voice-leading from the pitches of the previous chord.

- void CL_name (double time, std::string C_, bool avoidParallels=true)

  Same as CL, except the chord is specified by jazz-type name (e.g.
- virtual void clear ()

  Recursively clears all child Nodes of this.
- virtual Eigen::MatrixXd createTransform ()

  Returns the identity matrix for score space.
- void CV (double time, double C_, double V_)

  Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified prime chord, transpostion, and voicing.
- void CV_name (double time, std::string C_, double V_)

  Same as CV, except the chord is specified by jazz-type name (e.g.
- virtual double & element (size_t row, size_t column)

  Returns a reference to the indicated element of the local transformation of coordinate system.
- virtual void generate (Score &score_from_this)

  Optionally generate notes into the score.
- virtual Node * getChild (size_t index)

  Returns the immediate child of this at the index.
- virtual Eigen::MatrixXd getLocalCoordinates () const

  Returns the local transformation of coordinate system.
- virtual std::vector< double > getModality () const
- void K (double time)

  Find the C of the previous segment, and contextually invert it; apply the resulting C to the current segment.
- void KL (double time, bool avoidParallels=true)

  Find the C of the previous segment, and contextually invert it; apply the resulting C to the current segment, using the closest voiceleading from the pitches of the previous segment.
- void KV (double time, double V_)

  Find the C of the previous segment, and contextually invert it; apply the resulting C to the current segment with voicing V.
- void L (double time, bool avoidParallels=true)

  Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the smoothest voice-leading from the pitches of the previous segment.
- void PT (double time, double P_, double T)

  Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified prime chord and transposition.
- void PTL (double time, double P_, double T, bool avoidParallels=true)

  Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified chord; the voicing of the chord will be the smoothest voice-leading from the pitches of the previous chord.
- void PTV (double time, double P_, double T, double V_)

  Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified prime chord, transpostion, and voicing.
- void Q (double time, double Q_)

  Find the C of the previous segment, and contextually transpose it; apply the resulting C to the current segment.
- void QL (double time, double Q_, bool avoidParallels=true)

  Find the C of the previous segment, and contextually transpose it; apply the resulting C to the current segment, using the specified octavewise revoicing.

- void QV (double time, double Q_, double V_)

  Find the C of the previous segment, and contextually transpose it; apply the resulting C to the current segment with voicing V.
- virtual void setElement (size_t row, size_t column, double value)

  Sets the indicated element of the local transformation of coordinate system.
- virtual void setModality (const std::vector< double > &pcs)
- virtual void transform (Score &score)

  Apply all of the voice-leading operations stored within this node to the score.
- virtual void traverse (const Eigen::MatrixXd &global_coordinates, Score &global_score)

  The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.
- void V (double time, double V_)

  Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified voicing of the chord.
- VoiceleadingNode ()
- virtual ~VoiceleadingNode ()

Data Fields

- bool avoidParallels

  If true (the default), voice-leadings will avoid parallel fifths.
- double base

  The lowest pitch of the range of voicings, as a MIDI key number (default = 36).
- std::vector< Node * > children

  Child Nodes, if any.
- size_t divisionsPerOctave

  The number of equally tempered divisions of the octave (default = 12).
- std::vector< double > modality

  Context for the K and Q operations; must have the same cardinality as the pitch-classes in use.
- std::map< double, VoiceleadingOperation > operations

  Voice-leading operations stored in order of starting time.
- double range

  The range of voicings, from the lowest to the highest pitch, as a MIDI key number (default = 60).
- bool rescaleTimes

Protected Attributes

- Eigen::MatrixXd localCoordinates

6.82.1 Detailed Description

This node class imposes a sequence of one or more "voice-leading" operations upon the pitches of notes produced by children of this node, within a segment of the notes.

These operations comprise: prime chord (P), transpose (T), unordered pitch-class set (C, equivalent to PT), contextual inversion (K), contextual transposition (Q), voicing (V) within a specified range of pitches, and voice-lead (L). The values of P, T, C, and V each form an additive cyclic group whose elements are defined by counting through all possible values in order. Note that C is not the same as "pitch-class set number" in the sense of M = sum over pitch-classes of (2 ^ pitch-class); it is rather one less than M. Not all combinations of operations are consistent. P requires T. PT cannot be used with C. V cannot be used with L. If neither PT nor C is specified, the existing C of the notes is used. K and Q require a previous section, and cannot be used with P, T, or C. The consistent combinations of operations are thus: PT, PTV, PTL, C, CV, CL, K, KV, KL, Q, QV, QL, V, and L.

6.82.2 Constructor & Destructor Documentation

6.82.2.1 VoiceleadingNode()

```
csound::VoiceleadingNode::VoiceleadingNode ( )
```

6.82.2.2 ~VoiceleadingNode()

```
csound::VoiceleadingNode::~VoiceleadingNode ( )  [virtual]
```

6.82.3 Member Function Documentation

6.82.3.1 addChild()

```
void csound::Node::addChild (
            Node * node )  [virtual], [inherited]
```

Adds an immediate child Node to this.

Reimplemented in csound::ScoreModel.

References csound::Node::children, and csound::fundamentalDomainByPredicate().

Referenced by main().

6.82.3.2 apply()

```
void csound::VoiceleadingNode::apply (
            Score & score,
            const VoiceleadingOperation & priorOperation,
            const VoiceleadingOperation & currentOperation )  [virtual]
```

Apply the current voice-leading operation to the score, within the specified range of notes.

If voice-leading proper is to be performed, the prior voice-leading operation is used to determine how to lead the voices.

References avoidParallels, base, csound::Voicelead::cToM(), divisionsPerOctave, csound::fundamentalDomainByPredicate(), csound::Score::getDuration(), csound::System::getMessageLevel(), csound::Score::getPTV(), csound::System::inform(), csound::System::INFORMATION_LEVEL, modality, csound::Voicelead::mToPitchClassSet(), csound::Voicelead::pitchClassSetToPandT(), csound::printChord(), range, csound::Score::setK(), csound::Score::setKL(), csound::Score::setKV(), csound::Score::setPitchClassSet(), csound::Score::setPT(), csound::Score::setPTV(), csound::Score::setQ(), csound::Score::setQL(), csound::Score::setQV(), and csound::Score::voicelead().

Referenced by transform().

6.82.3.3 C()

```
void csound::VoiceleadingNode::C (
            double time,
            double C_ )
```

Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified prime chord and transposition.

Note that C (equivalent to PT) specifies what musicians normally call a chord.

References operations.

Referenced by C_name().

6.82.3.4 C_name()

```
void csound::VoiceleadingNode::C_name (
            double time,
            std::string C_ )
```

Same as C, except the chord can be specified by jazz-type name (e.g.

EbM7) instead of C number.

References C(), divisionsPerOctave, and csound::Voicelead::nameToC().

6.82.3.5 childCount()

```
size_t csound::Node::childCount ( ) const  [virtual], [inherited]
```

Returns the number of immediate children of this.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.82.3.6 chord()

```
void csound::VoiceleadingNode::chord (
            const csound::Chord & chord,
            double time )
```

Apply the specified chord to the current segement.

References chord(), and operations.

Referenced by chord(), csound::ChordLindenmayer::chordOperation(), chordVoiceleading(), csound::ChordLindenmayer::scaleOperation(), csound::ChordLindenmayer::scoreOperation(), and csound::ChordLindenmayer::setTurtleChord().

6.82.3.7 chordVoiceleading()

```
void csound::VoiceleadingNode::chordVoiceleading (
            const csound::Chord & chord,
            double time,
            bool avoid_parallels = true )
```

Apply the specified chord to the current segement, using the closest voice-leading from the pitches of the previous segment.

References chord(), csound::fundamentalDomainByPredicate(), and operations.

Referenced by csound::ChordLindenmayer::scoreOperation().

6.82.3.8 CL()

```
void csound::VoiceleadingNode::CL (
            double time,
            double C_,
            bool avoidParallels = true )
```

Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified chord; the voicing of the chord will be the smoothest voice-leading from the pitches of the previous chord.

Optionally, parallel fifths can be avoided. Note that CL (equivalent to PTL) specifies what musicians normally call the voice-leading of a chord.

References avoidParallels, and operations.

Referenced by CL_name().

6.82.3.9 CL_name()

```
void csound::VoiceleadingNode::CL_name (
            double time,
            std::string C_,
            bool avoidParallels = true )
```

Same as CL, except the chord is specified by jazz-type name (e.g.

EbM7) instead of C number.

References avoidParallels, CL(), divisionsPerOctave, and csound::Voicelead::nameToC().

6.82.3.10 clear()

```
void csound::Node::clear ( )  [virtual], [inherited]
```

Recursively clears all child Nodes of this.

Reimplemented in csound::ChordLindenmayer, csound::Lindenmayer, csound::MusicModel, and csound::ScoreModel.

References csound::Node::children, csound::Node::clear(), and csound::fundamentalDomainByPredicate().

Referenced by csound::MusicModel::clear(), csound::Node::clear(), and csound::ScoreModel::clear().

6.82.3.11 createTransform()

```
Eigen::MatrixXd csound::Node::createTransform ( )  [virtual], [inherited]
```

Returns the identity matrix for score space.

Reimplemented in csound::ScoreModel.

References csound::Event::ELEMENT_COUNT, and csound::fundamentalDomainByPredicate().

Referenced by csound::Node::Node(), and csound::MCRM::resize().

6.82.3.12 CV()

```
void csound::VoiceleadingNode::CV (
            double time,
            double C_,
            double V_ )
```

Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified prime chord, transpostion, and voicing.

Note that CV (equivalent to PTV) specifies what musicians normally call the voicing, or octavewise inversion, of a chord.

References operations.

Referenced by CV_name().

6.82.3.13 CV_name()

```
void csound::VoiceleadingNode::CV_name (
            double time,
            std::string C_,
            double V_ )
```

Same as CV, except the chord is specified by jazz-type name (e.g.

EbM7) instead of C number.

References CV(), divisionsPerOctave, and csound::Voicelead::nameToC().

6.82.3.14 element()

```
double & csound::Node::element (
            size_t row,
            size_t column )  [virtual], [inherited]
```

Returns a reference to the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.82.3.15 generate()

```
void csound::Node::generate (
            Score & score_from_this )  [virtual], [inherited]
```

Optionally generate notes into the score.

The notes must be produced at the coordinate system with origin at zero, and are automatically transformed to the global coordinate system.

Reimplemented in csound::ExternalNode, csound::ScoreNode, csound::ChordLindenmayer, csound::MCRM, csound::Generator, csound::Random, csound::LispGenerator, and csound::ScoreModel.

Referenced by csound::Node::traverse().

6.82.3.16 getChild()

```
Node * csound::Node::getChild (
            size_t index )  [virtual], [inherited]
```

Returns the immediate child of this at the index.

Reimplemented in csound::ScoreModel.

References csound::Node::children.

6.82.3.17 getLocalCoordinates()

```
Eigen::MatrixXd csound::Node::getLocalCoordinates ( ) const  [virtual], [inherited]
```

Returns the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

Referenced by csound::Random::getRandomCoordinates(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.82.3.18 getModality()

```
std::vector< double > csound::VoiceleadingNode::getModality ( ) const  [virtual]
```

References modality.

6.82.3.19 K()

```
void csound::VoiceleadingNode::K (
            double time )
```

Find the C of the previous segment, and contextually invert it; apply the resulting C to the current segment.

Contextual inversion is that inversion of C in which the first two pitch-classes are exchanged. If the chords are major or minor triads, produces the relative minor or major.

References operations.

6.82.3.20 KL()

```
void csound::VoiceleadingNode::KL (
            double time,
            bool avoidParallels = true )
```

Find the C of the previous segment, and contextually invert it; apply the resulting C to the current segment, using the closest voiceleading from the pitches of the previous segment.

Contextual inversion is that inversion of C in which the first two pitch-classes are exchanged.

References avoidParallels, and operations.

6.82.3.21 KV()

```
void csound::VoiceleadingNode::KV (
            double time,
            double V_ )
```

Find the C of the previous segment, and contextually invert it; apply the resulting C to the current segment with voicing V.

Contextual inversion is that inversion of C in which the first two pitch-classes are exchanged.

References operations.

6.82.3.22 L()

```
void csound::VoiceleadingNode::L (
            double time,
            bool avoidParallels = true )
```

Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the smoothest voice-leading from the pitches of the previous segment.

Optionally, parallel fifths can be avoided. Note that L specifies what musicians normally call voice-leading.

References avoidParallels, and operations.

6.82.3.23 PT()

```
void csound::VoiceleadingNode::PT (
            double time,
            double P_,
            double T )
```

Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified prime chord and transposition.

Note that PT specifies what musicians normally call a chord, e.g. "E flat major ninth." However, chords do not have to be in twelve tone equal temperament.

References operations.

6.82.3.24 PTL()

```
void csound::VoiceleadingNode::PTL (
            double time,
            double P_,
            double T,
            bool avoidParallels = true )
```

Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified chord; the voicing of the chord will be the smoothest voice-leading from the pitches of the previous chord.

Optionally, parallel fifths can be avoided. Note that PTL specifies what musicians normally call the voice-leading of a chord.

References avoidParallels, and operations.

6.82.3.25 PTV()

```
void csound::VoiceleadingNode::PTV (
            double time,
            double P_,
            double T,
            double V_ )
```

Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified prime chord, transpostion, and voicing.

Note that PTV specifies what musicians normally call the voicing, or octavewise inversion, of a chord.

References operations.

6.82.3.26 Q()

```
void csound::VoiceleadingNode::Q (
            double time,
            double Q_ )
```

Find the C of the previous segment, and contextually transpose it; apply the resulting C to the current segment.

Contextual transposition transposes C up by Q if C is an I-form, and down by Q if C is a T-form.

References operations.

6.82.3.27 QL()

```
void csound::VoiceleadingNode::QL (
            double time,
            double Q_,
            bool avoidParallels = true )
```

Find the C of the previous segment, and contextually transpose it; apply the resulting C to the current segment, using the specified octavewise revoicing.

Contextual transposition transposes C up by Q if C is an I-form, and down by Q if C is a T-form.

References avoidParallels, and operations.

6.82.3.28 QV()

```
void csound::VoiceleadingNode::QV (
            double time,
            double Q_,
            double V_ )
```

Find the C of the previous segment, and contextually transpose it; apply the resulting C to the current segment with voicing V.

Contextual transposition transposes C up by Q if C is an I-form, and down by Q if C is a T-form.

References operations.

6.82.3.29 setElement()

```
void csound::Node::setElement (
            size_t row,
            size_t column,
            double value )  [virtual], [inherited]
```

Sets the indicated element of the local transformation of coordinate system.

Reimplemented in csound::ScoreModel.

References csound::Node::localCoordinates.

6.82.3.30 setModality()

```
void csound::VoiceleadingNode::setModality (
            const std::vector< double > & pcs )  [virtual]
```

References modality.

6.82.3.31 transform()

```
void csound::VoiceleadingNode::transform (
            Score & score )  [virtual]
```

Apply all of the voice-leading operations stored within this node to the score.

Enables voice-leading operations to be used outside the context of a music graph.

Reimplemented from csound::Node.

References apply(), csound::Score::findScale(), csound::fundamentalDomainByPredicate(), csound::Score::getDuration(), csound::Event::getTime(), csound::Score::indexAfterTime(), csound::Score::indexAtTime(), csound::System::inform(), operations, csound::VoiceleadingOperation::rescaledBeginTime, rescaleTimes, csound::Score::scaleActualMinima, and csound::Score::sort().

Referenced by csound::ChordLindenmayer::applyVoiceleadingOperations().

6.82.3.32 traverse()

```
void csound::Node::traverse (
            const Eigen::MatrixXd & global_coordinates,
            Score & global_score )  [virtual], [inherited]
```

The default implementation postconcatenates its own local coordinate system with the global coordinates, then passes the score and the product of coordinate systems to each child, thus performing a depth-first traversal of the music graph.

In case a derived class needs to apply a different local transformation to each child node's notes, this method must be overridden. After child nodes have been traversed, notes generated by the child nodes are passed to the transform method of this, and the resulting notes appended to the gobal score; then an empty score is passed to the generate method of this, and the resulting notes appended to the global score.

Reimplemented in csound::ScoreModel, csound::Intercut, csound::Stack, csound::Koch, and csound::Sequence.

References csound::Node::children, csound::fundamentalDomainByPredicate(), csound::Node::generate(), csound::Node::getLocalCoord and csound::Node::transform().

6.82.3.33 V()

```
void csound::VoiceleadingNode::V (
            double time,
            double V_ )
```

Beginning at the specified time and continuing to the beginning of the next operation or the end of the score, whichever comes first, conform notes produced by this node or its children to the specified voicing of the chord.

Note that V specifies what musicians normally call the voicing or octavewise inversion of the chord.

References operations.

6.82.4 Field Documentation

6.82.4.1 avoidParallels

```
bool csound::VoiceleadingNode::avoidParallels
```

If true (the default), voice-leadings will avoid parallel fifths.

Referenced by apply(), CL(), CL_name(), KL(), L(), PTL(), and QL().

6.82.4.2 base

```
double csound::VoiceleadingNode::base
```

The lowest pitch of the range of voicings, as a MIDI key number (default = 36).

Referenced by apply().

6.82.4.3 children

```
std::vector<Node *> csound::Node::children  [inherited]
```

Child Nodes, if any.

Referenced by csound::Node::addChild(), csound::Node::childCount(), csound::Node::clear(), csound::MusicModel::generate(), csound::ScoreModel::generate(), csound::Node::getChild(), csound::Node::traverse(), csound::Intercut::traverse(), csound::Stack::traverse(), csound::Koch::traverse(), and csound::Sequence::traverse().

6.82.4.4 divisionsPerOctave

```
size_t csound::VoiceleadingNode::divisionsPerOctave
```

The number of equally tempered divisions of the octave (default = 12).

Note that the octave is always size 12. The size of a division of the octave is then 1 in 12-tone equal temperament, 0.5 in 24-tone equal temperament, 1.33333 in 9-tone equal temperament, and so on.

Referenced by apply(), C_name(), CL_name(), and CV_name().

6.82.4.5 localCoordinates

```
Eigen::MatrixXd csound::Node::localCoordinates  [protected], [inherited]
```

Referenced by csound::Node::element(), csound::Node::getLocalCoordinates(), csound::Node::Node(), and csound::Node::setElement().

6.82.4.6 modality

```
std::vector<double> csound::VoiceleadingNode::modality
```

Context for the K and Q operations; must have the same cardinality as the pitch-classes in use.

Referenced by apply(), getModality(), setModality(), and csound::ChordLindenmayer::setTurtleModality().

6.82.4.7 operations

```
std::map<double, VoiceleadingOperation> csound::VoiceleadingNode::operations
```

Voice-leading operations stored in order of starting time.

Referenced by C(), chord(), chordVoiceleading(), CL(), CV(), K(), KL(), KV(), L(), PT(), PTL(), PTV(), Q(), QL(), QV(), csound::ChordLindenmayer::scoreOperation(), transform(), and V().

6.82.4.8 range

```
double csound::VoiceleadingNode::range
```

The range of voicings, from the lowest to the highest pitch, as a MIDI key number (default = 60).

Referenced by apply().

6.82.4.9 rescaleTimes

```
bool csound::VoiceleadingNode::rescaleTimes
```

Referenced by transform().

6.83 csound::VoiceleadingOperation Class Reference

Utility class for storing voice-leading operations within a VoiceleadNode for future application.

```
#include <VoiceleadingNode.hpp>
```

Public Member Functions

- VoiceleadingOperation ()

  Utility class for storing voice-leading operations.
- virtual ~VoiceleadingOperation ()

Data Fields

- bool avoidParallels
- size_t begin

  The index of the first event to which the operation is applied.
- double beginTime

  The operation begins at this time, and continues until just before the beginning of the next operation, or the end of the score, whichever comes first.
- double C_

  Pitch-set class, or DBL_MAX if no operation.
- Chord chord

  Actual instance of csound::Chord.
- size_t end

  One past the index of the last event to which the operation is applied.
- double endTime

  The operation ends before this time.
- double K_

  Inversion by interchange.
- bool L_

  If true, perform the closest voice-leading from the prior operation.
- double P_

  Prime chord, or DBL_MAX if no operation.
- double Q_

  Contextual transposition.
- double rescaledBeginTime

  Times may need to be rescaled to match the duration of the score.
- double rescaledEndTime

  Times may need to be rescaled to match the duration of the score.
- double T_

  Transposition, or DBL_MAX if no operation.
- double V_

  Voicing, or DBL_MAX if no operation.

6.83.1 Detailed Description

Utility class for storing voice-leading operations within a VoiceleadNode for future application.

6.83.2 Constructor & Destructor Documentation

6.83.2.1 VoiceleadingOperation()

```
csound::VoiceleadingOperation::VoiceleadingOperation ( )
```

Utility class for storing voice-leading operations.

6.83.2.2 ~VoiceleadingOperation()

```
csound::VoiceleadingOperation::~VoiceleadingOperation ( )  [virtual]
```

6.83.3 Field Documentation

6.83.3.1 avoidParallels

```
bool csound::VoiceleadingOperation::avoidParallels
```

6.83.3.2 begin

```
size_t csound::VoiceleadingOperation::begin
```

The index of the first event to which the operation is applied.

6.83.3.3 beginTime

```
double csound::VoiceleadingOperation::beginTime
```

The operation begins at this time, and continues until just before the beginning of the next operation, or the end of the score, whichever comes first.

6.83.3.4 C_

```
double csound::VoiceleadingOperation::C_
```

Pitch-set class, or DBL_MAX if no operation.

6.83.3.5 chord

```
Chord csound::VoiceleadingOperation::chord
```

Actual instance of csound::Chord.

6.83.3.6 end

```
size_t csound::VoiceleadingOperation::end
```

One past the index of the last event to which the operation is applied.

6.83.3.7 endTime

```
double csound::VoiceleadingOperation::endTime
```

The operation ends before this time.

6.83.3.8 K_

```
double csound::VoiceleadingOperation::K_
```

Inversion by interchange.

6.83.3.9 L_

```
bool csound::VoiceleadingOperation::L_
```

If true, perform the closest voice-leading from the prior operation.

6.83.3.10 P_

```
double csound::VoiceleadingOperation::P_
```

Prime chord, or DBL_MAX if no operation.

6.83.3.11 Q_

```
double csound::VoiceleadingOperation::Q_
```

Contextual transposition.

6.83.3.12 rescaledBeginTime

```
double csound::VoiceleadingOperation::rescaledBeginTime
```

Times may need to be rescaled to match the duration of the score.

Referenced by csound::VoiceleadingNode::transform().

6.83.3.13 rescaledEndTime

```
double csound::VoiceleadingOperation::rescaledEndTime
```

Times may need to be rescaled to match the duration of the score.

6.83.3.14 T_

```
double csound::VoiceleadingOperation::T_
```

Transposition, or DBL_MAX if no operation.

6.83.3.15 V_

```
double csound::VoiceleadingOperation::V_
```

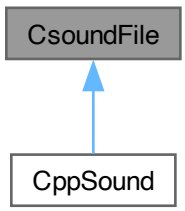
Voicing, or DBL_MAX if no operation.

6.84 CsoundFile Class Reference

Manages a Csound Structured Data (CSD) file with facilities for creating an arrangement of selected instruments in the orchestra, and for programmatically building score files.

```
#include <CsoundFile.hpp>
```

Inheritance diagram for CsoundFile:



Public Member Functions

- virtual void addArrangement (std::string instrument)
- virtual void addNote (double p1, double p2, double p3)
- virtual void addNote (double p1, double p2, double p3, double p4)
- virtual void addNote (double p1, double p2, double p3, double p4, double p5)
- virtual void addNote (double p1, double p2, double p3, double p4, double p5, double p6)
- virtual void addNote (double p1, double p2, double p3, double p4, double p5, double p6, double p7)
- virtual void addNote (double p1, double p2, double p3, double p4, double p5, double p6, double p7, double p8)
- virtual void addNote (double p1, double p2, double p3, double p4, double p5, double p6, double p7, double p8, double p9)
- virtual void addNote (double p1, double p2, double p3, double p4, double p5, double p6, double p7, double p8, double p9, double p10)
- virtual void addNote (double p1, double p2, double p3, double p4, double p5, double p6, double p7, double p8, double p9, double p10, double p11)
- virtual void addScoreLine (const std::string line)
- CsoundFile ()
- virtual int exportArrangement (std::ostream &stream) const
- virtual int exportArrangementForPerformance (std::ostream &stream) const
- virtual int exportArrangementForPerformance (std::string filename) const
- virtual int exportCommand (std::ostream &stream) const
- virtual bool exportForPerformance () const
- virtual int exportMidifile (std::ostream &stream) const
- virtual int exportOrchestra (std::ostream &stream) const
- virtual int exportScore (std::ostream &stream) const
- virtual std::string generateFilename ()
- virtual std::string getArrangement (int index) const
- virtual int getArrangementCount () const
- virtual std::string getCommand () const
- virtual std::string getCSD () const
- virtual std::string getFilename () const
- virtual std::string getInstrument (int number) const
- virtual bool getInstrument (int number, std::string &definition) const
- virtual std::string getInstrument (std::string name) const
- virtual bool getInstrument (std::string name, std::string &definition) const
- virtual std::string getInstrumentBody (int number) const
- virtual std::string getInstrumentBody (std::string name) const
- virtual int getInstrumentCount () const
- virtual std::map< int, std::string > getInstrumentNames () const
- virtual double getInstrumentNumber (std::string name) const
- virtual std::string getMidiFilename () const
- virtual std::string getOrcFilename () const
- virtual std::string getOrchestra () const
- virtual std::string getOrchestraHeader () const
- virtual std::string getOutputSoundfileName () const
- virtual std::string getScoFilename () const
- virtual std::string getScore () const
- virtual int importArrangement (std::istream &stream)
- virtual int importCommand (std::istream &stream)
- virtual int importFile (std::istream &stream)
- virtual int importFile (std::string filename)

Imports the indicated file, which can be a Csound unified file (.csd), Csound orchestra (.orc), Csound score (.sco), standard MIDI file (.mid), or MusicXML v2 (.xml) file.

- virtual int importMidifile (std::istream &stream)
- virtual int importOrchestra (std::istream &stream)
- virtual int importScore (std::istream &stream)
- virtual void insertArrangement (int index, std::string instrument)
- virtual int load (std::istream &stream)
- virtual int load (std::string filename)

  Clears all contents of this, then imports the indicated file, which can be a Csound unified file (.csd), Csound orchestra (.orc), Csound score (.sco), standard MIDI file (.mid), or MusicXML v2 (.xml) file.
- virtual bool loadOrcLibrary (const char *filename=0)
- virtual void removeAll ()
- virtual void removeArrangement ()
- virtual void removeArrangement (int index)
- virtual void removeCommand ()
- virtual void removeMidifile ()
- virtual void removeOrchestra ()
- virtual void removeScore ()
- virtual int save (std::ostream &stream) const
- virtual int save (std::string filename) const
- virtual void setArrangement (int index, std::string instrument)
- virtual void setCommand (std::string commandLine)
- virtual void setCSD (std::string xml)
- virtual void setFilename (std::string name)
- virtual void setOrchestra (std::string orchestra)
- virtual void setScore (std::string score)
- virtual ~CsoundFile ()

Data Fields

- std::vector< std::string > arrangement
- std::string libraryFilename

  Patch library and arrangement.

Protected Attributes

- std::vector< std::string > args
- std::vector< char * > argv
- std::string command

  CsOptions.
- std::string filename

  What are we storing, anyway?
- std::vector< unsigned char > midifile

  CsMidi.
- std::string orchestra

  CsInstruments.
- std::string score

  CsScore.

6.84.1 Detailed Description

Manages a Csound Structured Data (CSD) file with facilities for creating an arrangement of selected instruments in the orchestra, and for programmatically building score files.

6.84.2 Constructor & Destructor Documentation

6.84.2.1 CsoundFile()

```
CsoundFile::CsoundFile ( )
```

References removeAll().

6.84.2.2 ~CsoundFile()

```
virtual CsoundFile::~CsoundFile ( )  [inline], [virtual]
```

6.84.3 Member Function Documentation

6.84.3.1 addArrangement()

```
void CsoundFile::addArrangement (
            std::string instrument )  [virtual]
```

References arrangement.

6.84.3.2 addNote() [1/9]

```
void CsoundFile::addNote (
            double p1,
            double p2,
            double p3 )  [virtual]
```

References addScoreLine().

6.84.3.3 addNote() [2/9]

```
void CsoundFile::addNote (
            double p1,
            double p2,
            double p3,
            double p4 )  [virtual]
```

References addScoreLine().

6.84.3.4 addNote() [3/9]

```
void CsoundFile::addNote (
            double p1,
            double p2,
            double p3,
            double p4,
            double p5 )  [virtual]
```

References addScoreLine().

6.84.3.5 addNote() [4/9]

```
void CsoundFile::addNote (
            double p1,
            double p2,
            double p3,
            double p4,
            double p5,
            double p6 )  [virtual]
```

References addScoreLine().

6.84.3.6 addNote() [5/9]

```
void CsoundFile::addNote (
            double p1,
            double p2,
            double p3,
            double p4,
            double p5,
            double p6,
            double p7 )  [virtual]
```

References addScoreLine().

6.84.3.7 addNote() [6/9]

```
void CsoundFile::addNote (
            double p1,
            double p2,
            double p3,
            double p4,
            double p5,
            double p6,
            double p7,
            double p8 )  [virtual]
```

References addScoreLine().

6.84.3.8 addNote() [7/9]

```
void CsoundFile::addNote (
            double p1,
            double p2,
            double p3,
            double p4,
            double p5,
            double p6,
            double p7,
            double p8,
            double p9 )  [virtual]
```

References addScoreLine().

6.84.3.9 addNote() [8/9]

```
void CsoundFile::addNote (
            double p1,
            double p2,
            double p3,
            double p4,
            double p5,
            double p6,
            double p7,
            double p8,
            double p9,
            double p10 )  [virtual]
```

References addScoreLine().

6.84.3.10 addNote() [9/9]

```
void CsoundFile::addNote (
            double p1,
            double p2,
            double p3,
            double p4,
            double p5,
            double p6,
            double p7,
            double p8,
            double p9,
            double p10,
            double p11 )  [virtual]
```

References addScoreLine().

6.84.3.11 addScoreLine()

```
void CsoundFile::addScoreLine (
            const std::string line )  [virtual]
```

References score.

Referenced by addNote(), addNote(), addNote(), addNote(), addNote(), addNote(), addNote(), addNote(), addNote(), and csound::MusicModel::createCsoundScore().

6.84.3.12 exportArrangement()

```
int CsoundFile::exportArrangement (
            std::ostream & stream ) const  [virtual]
```

References arrangement.

Referenced by save().

6.84.3.13 exportArrangementForPerformance() [1/2]

```
int CsoundFile::exportArrangementForPerformance (
            std::ostream & stream ) const  [virtual]
```

References arrangement, exportOrchestra(), getInstrument(), getOrcFilename(), getOrchestraHeader(), and parseInstrument().

6.84.3.14 exportArrangementForPerformance() [2/2]

```
int CsoundFile::exportArrangementForPerformance (
            std::string filename ) const  [virtual]
```

References exportArrangementForPerformance(), and filename.

Referenced by exportArrangementForPerformance(), and exportForPerformance().

6.84.3.15 exportCommand()

```
int CsoundFile::exportCommand (
            std::ostream & stream ) const  [virtual]
```

References command.

Referenced by save().

6.84.3.16 exportForPerformance()

```
bool CsoundFile::exportForPerformance ( ) const  [virtual]
```

References exportArrangementForPerformance(), getMidiFilename(), getOrcFilename(), getScoFilename(), midifile, and save().

6.84.3.17 exportMidifile()

```
int CsoundFile::exportMidifile (
            std::ostream & stream ) const  [virtual]
```

References midifile.

Referenced by save(), and save().

6.84.3.18 exportOrchestra()

```
int CsoundFile::exportOrchestra (
            std::ostream & stream ) const  [virtual]
```

References orchestra.

Referenced by exportArrangementForPerformance(), save(), and save().

6.84.3.19 exportScore()

```
int CsoundFile::exportScore (
            std::ostream & stream ) const  [virtual]
```

References score.

Referenced by save(), and save().

6.84.3.20 generateFilename()

```
std::string CsoundFile::generateFilename ( )  [virtual]
```

References filename.

6.84.3.21 getArrangement()

```
std::string CsoundFile::getArrangement (
            int index ) const  [virtual]
```

References arrangement.

6.84.3.22 getArrangementCount()

```
int CsoundFile::getArrangementCount ( ) const  [virtual]
```

References arrangement.

6.84.3.23 getCommand()

```
std::string CsoundFile::getCommand ( ) const  [virtual]
```

References command.

Referenced by CppSound::compile(), csound::MusicModel::getCsoundCommand(), CppSound::perform(), and csound::MusicModel::perform().

6.84.3.24 getCSD()

```
std::string CsoundFile::getCSD ( ) const  [virtual]
```

References save().

Referenced by csound::MusicModel::perform().

6.84.3.25 getFilename()

```
std::string CsoundFile::getFilename ( ) const  [virtual]
```

References filename.

Referenced by CppSound::perform().

6.84.3.26 getInstrument() [1/4]

```
std::string CsoundFile::getInstrument (
            int number ) const  [virtual]
```

References getInstrument().

6.84.3.27 getInstrument() [2/4]

```
bool CsoundFile::getInstrument (
            int number,
            std::string & definition ) const  [virtual]
```

References findToken(), orchestra, and parseInstrument().

Referenced by exportArrangementForPerformance(), getInstrument(), getInstrument(), getInstrumentBody(), and getInstrumentBody().

6.84.3.28 getInstrument() [3/4]

```
std::string CsoundFile::getInstrument (
            std::string name ) const  [virtual]
```

References getInstrument().

6.84.3.29 getInstrument() [4/4]

```
bool CsoundFile::getInstrument (
            std::string name,
            std::string & definition ) const  [virtual]
```

References findToken(), orchestra, parseInstrument(), and trim().

6.84.3.30 getInstrumentBody() [1/2]

```
std::string CsoundFile::getInstrumentBody (
            int number ) const  [virtual]
```

References getInstrument(), and parseInstrument().

6.84.3.31 getInstrumentBody() [2/2]

```
std::string CsoundFile::getInstrumentBody (
            std::string name ) const  [virtual]
```

References getInstrument(), and parseInstrument().

6.84.3.32 getInstrumentCount()

```
int CsoundFile::getInstrumentCount ( ) const  [virtual]
```

References findToken(), orchestra, and parseInstrument().

6.84.3.33 getInstrumentNames()

```
std::map< int, std::string > CsoundFile::getInstrumentNames ( ) const  [virtual]
```

References findToken(), orchestra, and parseInstrument().

6.84.3.34 getInstrumentNumber()

```
double CsoundFile::getInstrumentNumber (
            std::string name ) const  [virtual]
```

References findToken(), orchestra, parseInstrument(), and trim().

Referenced by csound::MusicModel::arrange(), csound::MusicModel::arrange(), and csound::MusicModel::arrange().

6.84.3.35 getMidiFilename()

```
std::string CsoundFile::getMidiFilename ( ) const  [virtual]
```

References args, argv, command, and scatterArgs().

Referenced by exportForPerformance().

6.84.3.36 getOrcFilename()

```
std::string CsoundFile::getOrcFilename ( ) const  [virtual]
```

References args, argv, command, and scatterArgs().

Referenced by exportArrangementForPerformance(), and exportForPerformance().

6.84.3.37 getOrchestra()

```
std::string CsoundFile::getOrchestra ( ) const  [virtual]
```

References orchestra.

Referenced by CppSound::compile(), and csound::MusicModel::getCsoundOrchestra().

6.84.3.38 getOrchestraHeader()

```
std::string CsoundFile::getOrchestraHeader ( ) const  [virtual]
```

References findToken(), and orchestra.

Referenced by exportArrangementForPerformance().

6.84.3.39 getOutputSoundfileName()

```
std::string CsoundFile::getOutputSoundfileName ( ) const  [virtual]
```

Reimplemented in CppSound.

6.84.3.40 getScoFilename()

```
std::string CsoundFile::getScoFilename ( ) const  [virtual]
```

References args, argv, command, and scatterArgs().

Referenced by exportForPerformance().

6.84.3.41 getScore()

```
std::string CsoundFile::getScore ( ) const  [virtual]
```

References score.

Referenced by CppSound::compile().

6.84.3.42 importArrangement()

```
int CsoundFile::importArrangement (
            std::istream & stream )  [virtual]
```

References arrangement, getline(), removeArrangement(), and trim().

Referenced by importFile().

6.84.3.43 importCommand()

```
int CsoundFile::importCommand (
            std::istream & stream )  [virtual]
```

References command, and getline().

Referenced by importFile().

6.84.3.44 importFile() [1/2]

```
int CsoundFile::importFile (
            std::istream & stream )  [virtual]
```

References getline(), importArrangement(), importCommand(), importMidifile(), importOrchestra(), and importScore().

6.84.3.45 importFile() [2/2]

```
int CsoundFile::importFile (
            std::string filename )  [virtual]
```

Imports the indicated file, which can be a Csound unified file (.csd), Csound orchestra (.orc), Csound score (.sco), standard MIDI file (.mid), or MusicXML v2 (.xml) file.

The data that is read replaces existing data of that type, but leaves other types of data untouched.

The MusicXML notes become instrument number + 1, time in seconds, duration in seconds, MIDI key number, and MIDI velocity number.

References filename, importFile(), importMidifile(), importOrchestra(), importScore(), and score.

Referenced by importFile(), load(), and load().

6.84.3.46 importMidifile()

```
int CsoundFile::importMidifile (
            std::istream & stream )  [virtual]
```

References getline(), and midifile.

Referenced by importFile(), and importFile().

6.84.3.47 importOrchestra()

```
int CsoundFile::importOrchestra (
            std::istream & stream )  [virtual]
```

References getline(), and orchestra.

Referenced by importFile(), importFile(), and loadOrcLibrary().

6.84.3.48 importScore()

```
int CsoundFile::importScore (
            std::istream & stream )  [virtual]
```

References getline(), and score.

Referenced by importFile(), and importFile().

6.84.3.49 insertArrangement()

```
void CsoundFile::insertArrangement (
            int index,
            std::string instrument )  [virtual]
```

References arrangement.

6.84.3.50 load() [1/2]

```
int CsoundFile::load (
            std::istream & stream )  [virtual]
```

References importFile(), and removeAll().

6.84.3.51 load() [2/2]

```
int CsoundFile::load (
            std::string filename )  [virtual]
```

Clears all contents of this, then imports the indicated file, which can be a Csound unified file (.csd), Csound orchestra (.orc), Csound score (.sco), standard MIDI file (.mid), or MusicXML v2 (.xml) file.

The MusicXML notes become instrument number + 1, time in seconds, duration in seconds, MIDI key number, and MIDI velocity number.

References filename, importFile(), and removeAll().

Referenced by setCSD().

6.84.3.52 loadOrcLibrary()

```
bool CsoundFile::loadOrcLibrary (
            const char * filename = 0 )  [virtual]
```

References filename, importOrchestra(), and removeOrchestra().

6.84.3.53 removeAll()

```
void CsoundFile::removeAll ( )  [virtual]
```

References arrangement, command, filename, orchestra, removeMidifile(), and score.

Referenced by CsoundFile(), load(), and load().

6.84.3.54 removeArrangement() [1/2]

```
void CsoundFile::removeArrangement ( )  [virtual]
```

References arrangement.

Referenced by importArrangement().

6.84.3.55 removeArrangement() [2/2]

```
void CsoundFile::removeArrangement (
            int index )  [virtual]
```

References arrangement.

6.84.3.56 removeCommand()

```
void CsoundFile::removeCommand ( )  [virtual]
```

References command.

6.84.3.57 removeMidifile()

```
void CsoundFile::removeMidifile ( )  [virtual]
```

References midifile.

Referenced by removeAll().

6.84.3.58 removeOrchestra()

```
void CsoundFile::removeOrchestra ( )  [virtual]
```

References orchestra.

Referenced by loadOrcLibrary().

6.84.3.59 removeScore()

```
void CsoundFile::removeScore ( )  [virtual]
```

References score.

Referenced by csound::MusicModel::clear(), csound::MusicModel::createCsoundScore(), and csound::MusicModel::generate().

6.84.3.60 save() [1/2]

```
int CsoundFile::save (
            std::ostream & stream ) const  [virtual]
```

References arrangement, exportArrangement(), exportCommand(), exportMidifile(), exportOrchestra(), exportScore(), and midifile.

6.84.3.61 save() [2/2]

```
int CsoundFile::save (
            std::string filename ) const  [virtual]
```

References exportMidifile(), exportOrchestra(), exportScore(), filename, and save().

Referenced by exportForPerformance(), getCSD(), and save().

6.84.3.62 setArrangement()

```
void CsoundFile::setArrangement (
            int index,
            std::string instrument )  [virtual]
```

References arrangement.

6.84.3.63 setCommand()

```
void CsoundFile::setCommand (
            std::string commandLine )  [virtual]
```

References command.

Referenced by csound::MusicModel::perform(), and csound::MusicModel::setCsoundCommand().

6.84.3.64 setCSD()

```
void CsoundFile::setCSD (
            std::string xml )  [virtual]
```

References load().

6.84.3.65 setFilename()

```
void CsoundFile::setFilename (
            std::string name )  [virtual]
```

References filename.

6.84.3.66 setOrchestra()

```
void CsoundFile::setOrchestra (
            std::string orchestra )  [virtual]
```

References orchestra.

Referenced by csound::MusicModel::setCsoundOrchestra().

6.84.3.67 setScore()

```
void CsoundFile::setScore (
            std::string score )  [virtual]
```

References score.

6.84.4 Field Documentation

6.84.4.1 args

```
std::vector<std::string> CsoundFile::args  [protected]
```

Referenced by CppSound::compile(), getMidiFilename(), getOrcFilename(), getScoFilename(), and CppSound::perform().

6.84.4.2 argv

```
std::vector<char *> CsoundFile::argv  [protected]
```

Referenced by CppSound::compile(), CppSound::compile(), getMidiFilename(), getOrcFilename(), getScoFilename(), and CppSound::perform().

6.84.4.3 arrangement

```
std::vector<std::string> CsoundFile::arrangement
```

Referenced by addArrangement(), exportArrangement(), exportArrangementForPerformance(), getArrangement(), getArrangementCount(), importArrangement(), insertArrangement(), removeAll(), removeArrangement(), removeArrangement(), save(), and setArrangement().

6.84.4.4 command

```
std::string CsoundFile::command  [protected]
```

CsOptions.

Referenced by exportCommand(), getCommand(), getMidiFilename(), getOrcFilename(), getScoFilename(), importCommand(), CppSound::perform(), removeAll(), removeCommand(), and setCommand().

6.84.4.5 filename

```
std::string CsoundFile::filename  [protected]
```

What are we storing, anyway?

Referenced by exportArrangementForPerformance(), generateFilename(), getFilename(), importFile(), load(), loadOrcLibrary(), CppSound::perform(), removeAll(), save(), and setFilename().

6.84.4.6 libraryFilename

```
std::string CsoundFile::libraryFilename
```

Patch library and arrangement.

6.84.4.7 midifile

```
std::vector<unsigned char> CsoundFile::midifile  [protected]
```

CsMidi.

Referenced by exportForPerformance(), exportMidifile(), importMidifile(), removeMidifile(), and save().

6.84.4.8 orchestra

```
std::string CsoundFile::orchestra  [protected]
```

CsInstruments.

Referenced by exportOrchestra(), getInstrument(), getInstrument(), getInstrumentCount(), getInstrumentNames(), getInstrumentNumber(), getOrchestra(), getOrchestraHeader(), importOrchestra(), removeAll(), removeOrchestra(), and setOrchestra().

6.84.4.9 score

```
std::string CsoundFile::score  [protected]
```

CsScore.

Referenced by addScoreLine(), exportScore(), getScore(), importFile(), importScore(), removeAll(), removeScore(), and setScore().

6.85 CsoundFile_ Struct Reference

Data Fields

- std::string options
- std::string orchestra
- std::vector< std::string > score

6.85.1 Field Documentation

6.85.1.1 options

```
std::string CsoundFile_::options
```

Referenced by csoundCsdSave(), and csoundCsdSetOptions().

6.85.1.2 orchestra

```
std::string CsoundFile_::orchestra
```

Referenced by csoundCsdSave().

6.85.1.3 score

```
std::vector<std::string> CsoundFile_::score
```

Referenced by csoundCsdSave().

6.86 OrchestraNode Class Reference

```
#include <OrchestraNode.hpp>
```

Public Member Functions

- virtual void addSource (OrchestraNode *orchestraNode)
- virtual OrchestraNode * getSource (size_it index)
- virtual size_t getSourceCount () const
- Timebase * getTimebase ()
- OrchestraNode ()
- virtual void removeAllSources ()
- virtual void setSource (size_t index, OrchestraNode *source)
- setTimebase (Timebase *timebase)
- virtual ~OrchestraNode ()

6.86.1 Constructor & Destructor Documentation

6.86.1.1 OrchestraNode()

```
OrchestraNode::OrchestraNode ( )
```

6.86.1.2 ~OrchestraNode()

```
virtual OrchestraNode::~OrchestraNode ( )  [virtual]
```

6.86.2 Member Function Documentation

6.86.2.1 addSource()

```
virtual void OrchestraNode::addSource (
            OrchestraNode * orchestraNode )  [virtual]
```

6.86.2.2 getSource()

```
virtual OrchestraNode * OrchestraNode::getSource (
            size_it index )  [virtual]
```

6.86.2.3 getSourceCount()

```
virtual size_t OrchestraNode::getSourceCount ( ) const  [virtual]
```

6.86.2.4 getTimebase()

```
Timebase * OrchestraNode::getTimebase ( )
```

6.86.2.5 removeAllSources()

```
virtual void OrchestraNode::removeAllSources ( )  [virtual]
```

6.86.2.6 setSource()

```
virtual void OrchestraNode::setSource (
            size_t index,
            OrchestraNode * source )  [virtual]
```

6.86.2.7 setTimebase()

```
OrchestraNode::setTimebase (
            Timebase * timebase )
```

Chapter 7

File Documentation

7.1 /Users/michaelgogins/csound-ac/CsoundAC/Cell.cpp File Reference

```
#include <array>
#include "Cell.hpp"
#include "ChordSpaceBase.hpp"
#include "System.hpp"
```

Namespaces

- namespace csound

  C S O U N D.

7.2 /Users/michaelgogins/csound-ac/CsoundAC/Cell.hpp File Reference

```
#include "Platform.hpp"
#include <limits>
#include <map>
#include "Random.hpp"
#include "ScoreNode.hpp"
#include "ChordSpace.hpp"
#include <Eigen/Dense>
```

Data Structures

- class csound::Cell

  Score node that simplifies building up structures of motivic cells, and incrementally transforming them, as in Minimalism.
- class csound::CellAdd

  The indicated factor is added to the indicated dimension of each note produced by the child nodes of this, beginning at the start index and proceeding up to but not including the end index, at the specified stride.
- class csound::CellChord

  Notes produced by the child nodes of this are conformed to the chord, starting at the indicated start index, up to but not including the end index, at the indicated stride.
- class csound::CellMultiply

  The indicated dimension of each note produced by the child nodes of this, beginning at the start index and proceeding up to but not including the end index, at the specified stride, is multiplied by the indicated factor.
- class csound::CellRandom

  Notes produced by the child nodes of this, starting at the indicated start index, up to but not including the indicated end index, at the indicated stride, have added to them a random variable from the indicated distribution, rescaled to the indicated minimum and range.
- class csound::CellReflect

  The indicated dimension of each note produced by the child nodes of this, beginning at the start index and proceeding up to but not including the end index, at the specified stride, is reflected (i.e.
- class csound::CellRemove

  Notes are removed from the notes produced by the child nodes of this, beginning at the indicated start index, up to but not including the end index, at the indicated stride.
- class csound::CellRepeat

  All notes produced by child nodes are repeated for the specified number of iterations, beginning at the start index and proceeding up to but not including the end index, at the specified stride.
- class csound::CellSelect

  The notes produced by the child nodes of this are returned as sampled from the indicated start index, up to but not including the indicated end index, at the indicated stride.
- class csound::CellShuffle

  Notes produced by the child nodes of this, starting at the indicated start index, up to but not including the indicated end index, at the indicated stride, are randomly shuffled as to time.
- class csound::Intercut

  The notes produced by each child node are intercut to produce the notes produced by this; e.g.
- class csound::Koch

  All notes produced by child[N - 1] are rescaled and stacked on top of each note produced by child[N - 2], and so on.
- class csound::Stack

  The notes produced by each (not all) child node, are rescaled to all start at the same time, and last for the same duration; that of the 0th child, or a specified duration.

Namespaces

- namespace csound

  C S O U N D.

7.3 /Users/michaelgogins/csound-ac/CsoundAC/ChordLindenmayer.cpp File Reference

```
#include "ChordLindenmayer.hpp"
#include <random>
#include <stdio.h>
```

Namespaces

- namespace csound

  C S O U N D.

Macros

- #define DEBUGGING 1
- #define INDEX_DEBUGGING 0

Functions

- static void csound::addVoice (Chord &chord)
- static int csound::equivalentDegree (const Scale &scale, int degree)
- static int csound::getIndex (const std::string &dimension)

  Returns a zero-based numerical index for a string dimension name (for Events) or voice number (for Chords).
- static bool csound::getIndex (int &index, const std::string &dimension)
- static bool csound::parseIndex (int &index, const std::string &target)
- bool csound::parseVector (std::vector< double > &elements, std::string text)
- static double csound::real (const std::string &number)
- static void csound::removeVoice (Chord &chord)

Variables

- static std::mt19937_64 csound::twister

7.3.1 Macro Definition Documentation

7.3.1.1 DEBUGGING

```
#define DEBUGGING 1
```

7.3.1.2 INDEX_DEBUGGING

```
#define INDEX_DEBUGGING 0
```

7.4 /Users/michaelgogins/csound-ac/CsoundAC/ChordLindenmayer.hpp File Reference

```
#include "Platform.hpp"
#include "ChordSpace.hpp"
#include "Conversions.hpp"
#include "Event.hpp"
#include "Score.hpp"
#include "Node.hpp"
#include "Voicelead.hpp"
#include "VoiceleadingNode.hpp"
#include "System.hpp"
#include <sstream>
#include <stack>
#include <string>
#include <map>
#include <vector>
#include <Eigen/Dense>
```

Data Structures

- class csound::ChordLindenmayer

  A Lindenmayer system consists of a turtle representing a position in musical space, that is, a note; commands for moving the turtle or writing its state into a musical score; an axiom or initial set of commands; and zero or more rules for replacing commands with arbitrary sequences of commands.
- struct csound::Turtle

Namespaces

- namespace csound

  C S O U N D.

Functions

- void SILENCE_PUBLIC csound::printChord (std::ostream &stream, std::string label, const std::vector< double > &chord)

7.5 /Users/michaelgogins/csound-ac/CsoundAC/ChordSpace.cpp File Reference

```
#include "Platform.hpp"
#include <algorithm>
#include <boost/math/special_functions/ulp.hpp>
#include <cfloat>
#include "ChordSpace.hpp"
```

```
#include <climits>
#include <cmath>
#include <csignal>
#include <cstdarg>
#include <Eigen/Dense>
#include "Event.hpp"
#include <functional>
#include <iostream>
#include <iterator>
#include <map>
#include <random>
#include "Score.hpp"
#include <set>
#include <sstream>
#include <vector>
```

Namespaces

- namespace csound

  C S O U N D.

Macros

- #define EIGEN_INITIALIZE_MATRICES_BY_ZERO

Functions

- SILENCE_PUBLIC std::vector< Chord > csound::allOfEquivalenceClass (int voice_count, std::string equivalence_class, double range, double g, int sector, bool printme)
- SILENCE_PUBLIC void csound::apply (Score &score, const Chord &chord, double startTime, double endTime, bool octaveEquivalence)
- SILENCE_PUBLIC void csound::conformToChord (Event &event, const Chord &chord)
- SILENCE_PUBLIC void csound::conformToChord_equivalence (Event &event, const Chord &chord, bool octaveEquivalence)

  If the Event is a note, moves its pitch to the closest pitch of the chord.
- SILENCE_PUBLIC Chord csound::gather (Score &score, double startTime, double endTime)

  Returns a chord containing all the pitches of the score beginning at or later than the start time, and up to but not including the end time.
- SILENCE_PUBLIC void csound::insert (Score &score, const Chord &chord, double time_)
- SILENCE_PUBLIC void csound::insert (Score &score, const Chord &chord, double time_, bool voice_is_instrument)

  Inserts the notes of the chord into the score at the specified time.
- Event csound::note (const Chord &chord, int voice, double time_, double duration_=DBL_MAX, double channel_=DBL_MAX, double velocity_=DBL_MAX, double pan_=DBL_MAX)

  Creates a complete "note on" Event for the indicated voice of the chord.
- Score csound::notes (const Chord &chord, double time_, double duration_=DBL_MAX, double channel_=DBL_MAX, double velocity_=DBL_MAX, double pan_=DBL_MAX)

  Returns an individual note for each voice of the chord.
- SILENCE_PUBLIC void csound::numerics_information (double a, double b, int epsilons, int ulps)
- SILENCE_PUBLIC std::vector< Event * > csound::slice (Score &score, double startTime, double endTime)

  Returns a slice of the Score starting at the start time and extending up to but not including the end time.
- SILENCE_PUBLIC void csound::toScore (const Chord &chord, Score &score, double time_, bool voiceIsInstrument)

7.5.1 Macro Definition Documentation

7.5.1.1 EIGEN_INITIALIZE_MATRICES_BY_ZERO

```
#define EIGEN_INITIALIZE_MATRICES_BY_ZERO
```

7.6 /Users/michaelgogins/csound-ac/CsoundAC/ChordSpace.hpp File Reference

This library implements a geometric approach to some common operations on chords in neo-Riemannian music theory for use in score generating procedures:

```
#include "Platform.hpp"
#include <algorithm>
#include <boost/math/special_functions/ulp.hpp>
#include <cfloat>
#include "ChordSpaceBase.hpp"
#include <climits>
#include <cmath>
#include <csignal>
#include <cstdarg>
#include <Eigen/Dense>
#include "Event.hpp"
#include <functional>
#include <iostream>
#include <iterator>
#include <map>
#include <random>
#include "Score.hpp"
#include <set>
#include <sstream>
#include <vector>
```

Data Structures

- class csound::ChordScore

  Score equipped with chords.

Namespaces

- namespace csound

  C S O U N D.

Macros

- #define EIGEN_INITIALIZE_MATRICES_BY_ZERO

Functions

- SILENCE_PUBLIC std::vector< Chord > csound::allOfEquivalenceClass (int voice_count, std::string equivalence_class, double range, double g, int sector, bool printme)
- SILENCE_PUBLIC void csound::apply (Score &score, const Chord &chord, double startTime, double endTime, bool octaveEquivalence)
- SILENCE_PUBLIC void csound::conformToChord (Event &event, const Chord &chord)
- SILENCE_PUBLIC void csound::conformToChord_equivalence (Event &event, const Chord &chord, bool octaveEquivalence)

  If the Event is a note, moves its pitch to the closest pitch of the chord.
- SILENCE_PUBLIC Chord csound::gather (Score &score, double startTime, double endTime)

  Returns a chord containing all the pitches of the score beginning at or later than the start time, and up to but not including the end time.
- SILENCE_PUBLIC void csound::insert (Score &score, const Chord &chord, double time_)
- SILENCE_PUBLIC void csound::insert (Score &score, const Chord &chord, double time_, bool voice_is_instrument)

  Inserts the notes of the chord into the score at the specified time.
- Event csound::note (const Chord &chord, int voice, double time_, double duration_=DBL_MAX, double channel_=DBL_MAX, double velocity_=DBL_MAX, double pan_=DBL_MAX)

  Creates a complete "note on" Event for the indicated voice of the chord.
- Score csound::notes (const Chord &chord, double time_, double duration_=DBL_MAX, double channel_=DBL_MAX, double velocity_=DBL_MAX, double pan_=DBL_MAX)

  Returns an individual note for each voice of the chord.
- SILENCE_PUBLIC void csound::numerics_information (double a, double b, int epsilons, int ulps)
- SILENCE_PUBLIC std::vector< Event * > csound::slice (Score &score, double startTime, double endTime)

  Returns a slice of the Score starting at the start time and extending up to but not including the end time.
- SILENCE_PUBLIC void csound::toScore (const Chord &chord, Score &score, double time_, bool voiceIsInstrument)

Variables

- class SILENCE_PUBLIC csound::ChordScore

7.6.1 Detailed Description

This library implements a geometric approach to some common operations on chords in neo-Riemannian music theory for use in score generating procedures:

- Identifying whether a chord belongs to some equivalence class of music theory, or sending a chord to its equivalent within a representative ("normal") fundamental domain of some equivalence relation. The equivalence relations are octave (O), permutational (P), transpositional, (T), inversional (I), and their compounds OP, OPT (set-class or chord type), and OPTI (similar to prime form), among others.
- Causing chord progressions to move strictly within an orbifold that reoresents some equivalence class.
- Implementing chord progressions based on the L, P, R, D, K, and Q operations of neo-Riemannian theory (thus implementing some aspects of "harmony").
- Implementing chord progressions performed within a more abstract equivalence class by means of the closest voice-leading within a less abstract equivalence class (thus implementing some fundamentals of "counterpoint").
- Implementing "functional" or "Roman numeral" operations performed using scales and scale degrees (thus implementing many fundamentals of "pragmatic music theory").

7.6.2 Definitions

Pitch is the perception of a distinct sound frequency. It is a logarithmic perception; octaves, which sound 'equivalent' in some sense, represent doublings or halvings of frequency.

Pitches and intervals are represented as real numbers. Middle C is 60 and the octave is 12. Our usual system of 12-tone equal temperament, as well as MIDI key numbers, are completely represented by the whole numbers; any and all other pitches can be represented simply by using fractions.

A voice is a distinct sound that is heard as having a pitch.

A chord is simply a set of voices heard at the same time, represented here as a point in a chord space having one dimension of pitch for each voice in the chord.

A scale is a chord with a tonic pitch-class as its first and lowest voice, all other voices being pitch-classes sorted in ascending order.

For the purposes of algorithmic composition, a score can be considered to be a sequence of more or less fleeting chords.

7.6.3 Equivalence Relations and Classes

An equivalence relation identifies different elements of a set as belonging to the same class. For example the octave is an equivalence relation that identifies C1, C2, and C3 as belonging to the equivalence class C. Operations that send elements to their equivalents induce quotient spaces or orbifolds, where the equivalence operation identifies points on one facet of the orbifold with points on an opposing facet. The fundamental domain of the equivalence relation is the space consisting of the orbifold and its surface.

Plain chord space has no equivalence relation. Ordered chords are represented as vectors in parentheses (p1, ..., pN). Unordered chords are represented as sorted vectors in braces {p1, ..., pN}. Unordering is itself an equivalence relation – permutational equivalence.

The following equivalence relations apply to pitches and chords, and exist in different orbifolds. Equivalence relations can be combined (Callendar, Quinn, and Tymoczko, "Generalized Voice-Leading Spaces," *Science* 320, 2008), and the more equivalence relations are combined, the more abstract is the resulting orbifold compared to the parent space.

In most cases, a chord space can be divided into a number, possibly infinite, of geometrically equivalent fundamental domains for the same equivalence relation. Therefore, here we use the notion of 'representative' or 'normal' fundamental domain. For example, the representative fundamental domain of unordered sequences, out of all possible orderings, consists of all sequences in their ordinary sorted order. It is important, in the following, to identify representative fundamental domains that combine properly, e.g. such that the representative fundamental domain of OP / the representative fundamental domain of PI equals the representative fundamental domain of OPI. And this in turn may require accounting for duplicate elements of the representative fundamental domain caused by reflections or singularities in the orbifold (e.g. on vertices, edges, or facets shared by fundamental domains with a cyclical structure), or by doubled pitches in a chord.

C Cardinality equivalence, e.g. {1, 1, 2} == {1, 2}. *Not* assuming cardinality equivalence ensures that there is a prototypic metric in plain chord space that is inherited by all child chord spaces. Cardinality equivalence is never assumed here, because we are working in chord spaces of fixed dimensionality; e.g. we represent the note middle C not only as {60}, but also as {60, 60, ..., 60}.

O Octave equivalence. The fundamental domain is defined by the pitches in a chord spanning the range of an octave or less, and summing to an octave or less.

P Permutational equivalence. The fundamental domain is defined by a "wedge" of plain chord space in which the voices of a chord are always sorted by pitch.

T Transpositional equivalence, e.g. {1, 2} == {7, 8}. The fundamental domain is defined as a hyperplane in chord space at right angles to the diagonal of unison chords. Represented by the chord always having a sum of pitches equal to 0.

Tg Transpositional equivalence; the pitches of the chord are sent to the ceilings of the pitches in the first chord whose sum is equal to or greater than 0, i.e., rounded up to equal temperament.

I Inversional equivalence. Care is needed to distinguish the mathematician's sense of 'invert', which means 'pitch-space inversion' or 'reflect in a point', from the musician's sense of 'invert', which varies according to context but in practice often means 'registral inversion' or 'revoice by adding an octave to the lowest tone of a chord.' Here, we use 'invert' and 'inversion' in the mathematician's sense, and we use the terms 'revoice' and 'voicing' for the musician's 'invert' and 'inversion'. Here, the inversion of a chord is its reflection in a hyperplane (the inversion flat) that divides a fundamental domain of pitch.

PI Inversional equivalence with permutational equivalence. The 'inversion flat' of unordered chord space is a hyperplane consisting of all those unordered chords that are invariant under inversion. A fundamental domain is defined by any half space bounded by a hyperplane containing the inversion flat.

OP Octave equivalence with permutational equivalence. Tymoczko's orbifold for chords; i.e. chords with a fixed number of voices in a harmonic context. The fundamental domain is defined as a hyperprism one octave long with as many sides as voices and the ends identified by octave equivalence and one cyclical permutation of voices, modulo the unordering. In OP for trichords in 12TET, the augmented triads run up the middle of the prism, the major and minor triads are in 6 alternating columns around the augmented triads, the two-pitch chords form the 3 sides, and the one-pitch chords form the 3 edges that join the sides.

OPT The layer of the OP prism as close as possible to the origin, modulo the number of voices. Chord type. Note that CM and Cm are different OPT. Because the OP prism is canted down from the origin, at least one pitch in each OPT chord (excepting the origin itself) is negative. For n dimensions there are n OPT fundamental domains centering on the maximally even chord and generated by rotation about the maximally even chord, equivalently octavewise revoicing, more or less the same as the musician's sense of "chord inversion."

OPTT The same as OPT, but with chords rounded up within equal temperament; equivalent to "chord type."

OPI The OP prism modulo inversion, i.e. 1/2 of the OP prism. The representative fundamental consits of those chords having inversional equivalence.

OPTI The OPT layer modulo inversion, i.e. 1/2 of the OPT layer. Set-class. Note that minor and major triads are are the same OPTI.

OPTTI The same as OPTI, but with chords rounded up within equal temperament; equivalent to "set class."

7.6.4 Operations

Each of the above equivalence relations is, of course, an operation that sends chords outside some fundamental domain to chords inside that fundamental domain. We define the following additional operations:

T(p, x) Translate p by x.

I(p [, x]) Reflect p in x, by default the origin.

P Send a major triad to the minor triad with the same root, or vice versa (Riemann's parallel transformation).

L Send a major triad to the minor triad one major third higher, or vice versa (Riemann's Leittonwechsel or leading-tone exchange transformation).

R Send a major triad to the minor triad one minor third lower, or vice versa (Riemann's relative transformation).

D Send a triad to the next triad a perfect fifth lower (dominant transformation).

P, L, and R have been extended as follows, see Fiore and Satyendra, "Generalized Contextual Groups", *Music Theory Online* 11, August 2008:

K(c) Interchange by inversion; `K(c) := I(c, c[1] + c[2])`. This is a generalized form of P; for major and minor triads, it is exactly the same as P, but it also works with other chord types.

Q(c, n, m) Contexual transposition; `Q(c, n, m) := T(c, n)` if c is a T-form of m, or `T(c, -n)` if c is an I-form of M. Not a generalized form of L or R; but, like them, K and Q generate the T-I group.

7.6.5 Macro Definition Documentation

7.6.5.1 EIGEN_INITIALIZE_MATRICES_BY_ZERO

```
#define EIGEN_INITIALIZE_MATRICES_BY_ZERO
```

7.7 /Users/michaelgogins/csound-ac/CsoundAC/ChordSpaceBase.hpp File Reference

This library implements a geometric approach to some common operations on chords in neo-Riemannian music theory for use in score generating procedures:

```
#include "Platform.hpp"
#include "System.hpp"
#include <algorithm>
#include <boost/algorithm/string.hpp>
#include <boost/math/special_functions/ulp.hpp>
#include <cfloat>
#include <climits>
#include <cmath>
#include <csignal>
#include <cstdarg>
#include <Eigen/Dense>
#include <functional>
#include <iostream>
#include <iterator>
#include <map>
#include <random>
#include <set>
#include <sstream>
#include <vector>
```

Data Structures

- class csound::Chord

  Chords consist of simultaneously sounding pitches.
- struct csound::compare_by_normal_form
- struct csound::compare_by_normal_order
- struct csound::compare_by_op
- struct csound::HyperplaneEquation
- class csound::PITV

  This class implements a cyclic additive group for all chords under cardinality, permutational, and range equivalence.
- class csound::Scale

  Scale as a class; must be created with the name of the scale.
- struct csound::SCOPED_DEBUGGING

Namespaces

- namespace csound

  C S O U N D.

Macros

- #define CHORD_SPACE_DEBUG if (CHORD_SPACE_DEBUGGING() == true) csound::System::message
- #define EIGEN_INITIALIZE_MATRICES_BY_ZERO

Typedefs

- typedef Eigen::Matrix< double, Eigen::Dynamic, Eigen::Dynamic > csound::Matrix
- typedef Eigen::Matrix< double, Eigen::Dynamic, 1 > csound::Vector

Enumerations

- enum csound::EQUIVALENCE_RELATIONS {
  csound::EQUIVALENCE_RELATION_r = 0 , csound::EQUIVALENCE_RELATION_R , csound::EQUIVALENCE_RELATION_P
  , csound::EQUIVALENCE_RELATION_T ,
  csound::EQUIVALENCE_RELATION_Tg , csound::EQUIVALENCE_RELATION_I , csound::EQUIVALENCE_RELATION_RP
  , csound::EQUIVALENCE_RELATION_RT ,
  csound::EQUIVALENCE_RELATION_RPT , csound::EQUIVALENCE_RELATION_RPTg , csound::EQUIVALENCE_RELATION_RF
  , csound::EQUIVALENCE_RELATION_RTI ,
  csound::EQUIVALENCE_RELATION_RTgI , csound::EQUIVALENCE_RELATION_RPTI , csound::EQUIVALENCE_RELATION_RP
  }

  Enums for all defined equivalence relations, used to specialize template functions.

Functions

- SILENCE_PUBLIC void csound::add_chord (std::string, const Chord &chord)
- SILENCE_PUBLIC void csound::add_scale (std::string, const Scale &scale)
- SILENCE_PUBLIC double csound::C4 ()
- SILENCE_PUBLIC Chord csound::chord (const Chord &scale, int scale_degree, int chord_voices, int interval=3)

  Returns the chord, in scale order, for the specified degree of the scale.
- static SILENCE_PUBLIC bool & csound::CHORD_SPACE_DEBUGGING ()

  Returns the current state of the chord space debugging flag as a reference, which can be an lvalue or an rvalue.
- static SILENCE_PUBLIC std::string csound::chord_space_version ()
- SILENCE_PUBLIC const Chord & csound::chordForName (std::string name)
- SILENCE_PUBLIC std::map< std::string, Chord > & csound::chordsForNames ()
- SILENCE_PUBLIC double csound::closestPitch (double pitch, const Chord &chord)

  Returns the pitch in the chord that is closest to the indicated pitch.
- SILENCE_PUBLIC double csound::conformToPitchClassSet (double pitch, const Chord &pitch_class_set)

  Conforms the pitch to the pitch-class set, but in its original register.
- SILENCE_PUBLIC double csound::distance_to_points (const Chord &chord, const std::vector< Chord > &sector_vertices)

  Returns the sum of the distances of the chord to each of the vertices of the indicated sector of a cyclical region.
- SILENCE_PUBLIC double csound::epc (double pitch)

  Returns the equivalent of the pitch under pitch-class equivalence, i.e.
- SILENCE_PUBLIC bool csound::eq_tolerance (double a, double b, int epsilons=20, int ulps=200)

  This is the basis of all other numeric comparisons that take floating-point limits into account.
- template<int EQUIVALENCE_RELATION>
  SILENCE_PUBLIC Chord csound::equate (const Chord &chord)
- template<int EQUIVALENCE_RELATION>
  SILENCE_PUBLIC Chord csound::equate (const Chord &chord, double range)
- template<int EQUIVALENCE_RELATION>
  SILENCE_PUBLIC Chord csound::equate (const Chord &chord, double range, double g, int opt_sector)

  Template function that returns the chord sent to a fundamental domain of specialized equivalence relation, which in some cases may be defined by the indicated range, generator of transposition g, and sector of the cyclical region of OPT fundamental domains.
- template<> SILENCE_PUBLIC Chord csound::equate< EQUIVALENCE_RELATION_I > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC Chord csound::equate< EQUIVALENCE_RELATION_P > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC Chord csound::equate< EQUIVALENCE_RELATION_r > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC Chord csound::equate< EQUIVALENCE_RELATION_R > (const Chord &chord, double range_, double g, int opt_sector)
- template<> SILENCE_PUBLIC Chord csound::equate< EQUIVALENCE_RELATION_RP > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC Chord csound::equate< EQUIVALENCE_RELATION_RPI > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC Chord csound::equate< EQUIVALENCE_RELATION_RPT > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC Chord csound::equate< EQUIVALENCE_RELATION_RPTg > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC Chord csound::equate< EQUIVALENCE_RELATION_RPTgI > (const Chord &chord, double range, double g, int opt_sector)

- template<> SILENCE_PUBLIC Chord csound::equate< EQUIVALENCE_RELATION_RPTI > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC Chord csound::equate< EQUIVALENCE_RELATION_T > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC Chord csound::equate< EQUIVALENCE_RELATION_Tg > (const Chord &chord, double range, double g, int opt_sector)
- SILENCE_PUBLIC double csound::euclidean (const csound::Chord &a, const csound::Chord &b)

  Returns the Euclidean distance between the two chords.
- SILENCE_PUBLIC double csound::factorial (double n)
- void csound::fill (std::string rootName, double rootPitch, std::string typeName, std::string typePitches, bool is_scale=false)
- template<int EQUIVALENCE_RELATION>
  SILENCE_PUBLIC std::vector< csound::Chord > csound::fundamentalDomainByPredicate (int voiceN, double range, double g, int sector, bool printme)

  Returns a set of chords in sector 0 of the cyclical region, sorted by normal order, for the indicated equivalence relation.
- template<int EQUIVALENCE_RELATION>
  SILENCE_PUBLIC std::vector< Chord > csound::fundamentalDomainByPredicate (int voiceN, double range, double g=1., int sector=0, bool printme=false)

  Returns a set of chords in sector 0 of the cyclical region, sorted by normal order, for the indicated equivalence relation.
- template<int EQUIVALENCE_RELATION>
  SILENCE_PUBLIC std::vector< csound::Chord > csound::fundamentalDomainByTransformation (int voiceN, double range, double g, int sector)

  Returns a set of chords in sector 0 of the cyclical region, sorted by normal order, for the indicated equivalence relation.
- template<int EQUIVALENCE_RELATION>
  SILENCE_PUBLIC std::vector< Chord > csound::fundamentalDomainByTransformation (int voiceN, double range, double g=1., int sector=0)

  Returns a set of chords in sector 0 of the cyclical region, sorted by normal order, for the indicated equivalence relation.
- SILENCE_PUBLIC bool csound::ge_tolerance (double a, double b, int epsilons=20, int ulps=200)
- SILENCE_PUBLIC bool csound::gt_tolerance (double a, double b, int epsilons=20, int ulps=200)
- SILENCE_PUBLIC HyperplaneEquation csound::hyperplane_equation_from_random_inversion_flat (int dimensions, bool transpositional_equivalence, int opt_sector)
- SILENCE_PUBLIC HyperplaneEquation csound::hyperplane_equation_from_singular_value_decomposition (const std::vector< Chord > &points_, bool make_eT)
- SILENCE_PUBLIC double csound::I (double pitch, double center=0.0)

  Returns the pitch reflected in the center, which may be any pitch.
- SILENCE_PUBLIC int csound::indexForOctavewiseRevoicing (const Chord &chord, double range)

  Returns the index of the octavewise revoicing that this chord is, relative to its OP equivalent, within the indicated range.
- SILENCE_PUBLIC int csound::indexForOctavewiseRevoicing (const Chord &origin, const Chord &chord, double range)

  Returns the index of the octavewise revoicing that this chord is, counting up from the origin, within the indicated range.
- void csound::initializeNames ()
- SILENCE_PUBLIC std::map< Chord, Chord > & csound::inverse_prime_forms_for_chords ()

  Cache inverse prime forms for chords for speed.
- SILENCE_PUBLIC Chord csound::iterator (int voiceN, double first)

  Returns a chord with the specified number of voices all set to a first pitch, useful as an iterator.
- SILENCE_PUBLIC bool csound::le_tolerance (double a, double b, int epsilons=20, int ulps=200)
- SILENCE_PUBLIC bool csound::lt_tolerance (double a, double b, int epsilons=20, int ulps=200)
- SILENCE_PUBLIC double csound::MIDDLE_C ()
- SILENCE_PUBLIC Chord csound::midpoint (const Chord &a, const Chord &b)

  Returns the chord that is the midpoint between two chords, which must have the same number of voices.

- SILENCE_PUBLIC double csound::modulo (double dividend, double divisor)

  Returns the remainder of the dividend divided by the divisor, according to the Euclidean definition.
- SILENCE_PUBLIC std::string csound::nameForChord (const Chord &chord)

  Returns the first valid name for the Chord.
- SILENCE_PUBLIC std::string csound::nameForPitchClass (double pitch)

  Returns the name of the pitch-class of the pitch.
- SILENCE_PUBLIC std::string csound::nameForScale (const Scale &scale)

  Returns the first valid name for the Scale.
- SILENCE_PUBLIC std::vector< std::string > csound::namesForChord (const Chord &chord)

  Returns all enharmonic names for the Chord, if any exists.
- SILENCE_PUBLIC std::multimap< Chord, std::string > & csound::namesForChords ()
- SILENCE_PUBLIC std::vector< std::string > csound::namesForScale (const Scale &scale)

  Returns all enharmonic names for the Scale, if any exists.
- SILENCE_PUBLIC std::multimap< Scale, std::string > & csound::namesForScales ()
- SILENCE_PUBLIC bool csound::next (Chord &iterator_, const Chord &minimum, double range, double g=1.)

  Increment a chord voicewise through chord space, from a low point on the unison diagonal through a high point on the unison diagonal.
- SILENCE_PUBLIC std::map< Chord, Chord > & csound::normal_forms_for_chords ()

  Cache prime forms for chords for speed.
- SILENCE_PUBLIC double csound::OCTAVE ()

  The size of the octave, defined to be consistent with 12 tone equal temperament and MIDI.
- SILENCE_PUBLIC Chord csound::octavewiseRevoicing (const Chord &chord, int revoicingNumber_, double range)

  Returns the nth octavewise revoicing of the chord that is generated by iterating revoicings within the indicated range.
- SILENCE_PUBLIC int csound::octavewiseRevoicings (const Chord &chord, double range=OCTAVE())

  Returns the full set of octavewise revoicings of the chord within the indicated range.
- SILENCE_PUBLIC bool csound::operator< (const Chord &a, const Chord &b)
- SILENCE_PUBLIC bool csound::operator<= (const Chord &a, const Chord &b)
- SILENCE_PUBLIC bool csound::operator== (const Chord &a, const Chord &b)
- SILENCE_PUBLIC bool csound::operator> (const Chord &a, const Chord &b)
- SILENCE_PUBLIC bool csound::operator>= (const Chord &a, const Chord &b)
- SILENCE_PUBLIC bool csound::parallelFifth (const Chord &a, const Chord &b)

  Returns whether the voiceleading between chords a and b contains a parallel fifth.
- SILENCE_PUBLIC const std::map< std::string, double > & csound::pitchClassesForNames ()
- SILENCE_PUBLIC double csound::pitchClassForName (std::string name)
- template<int EQUIVALENCE_RELATION>
  SILENCE_PUBLIC bool csound::predicate (const Chord &chord)
- template<int EQUIVALENCE_RELATION>
  SILENCE_PUBLIC bool csound::predicate (const Chord &chord, double range)
- template<int EQUIVALENCE_RELATION>
  SILENCE_PUBLIC bool csound::predicate (const Chord &chord, double range, double g, int opt_sector)

  Template function returning whether or not the chord is within the specialized fundamental domain, which may in some cases be defined by the indicated range, generator of transposition g, and sector of the cyclical region of OPT fundamental domains.
- template<int EQUIVALENCE_RELATION>
  SILENCE_PUBLIC bool csound::predicate (const Chord &chord, double range, int sector)
- template<> SILENCE_PUBLIC bool csound::predicate< EQUIVALENCE_RELATION_I > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC bool csound::predicate< EQUIVALENCE_RELATION_P > (const Chord &chord, double range, double g, int opt_sector)

- template<> SILENCE_PUBLIC bool csound::predicate< EQUIVALENCE_RELATION_R > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC bool csound::predicate< EQUIVALENCE_RELATION_r > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC bool csound::predicate< EQUIVALENCE_RELATION_RP > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC bool csound::predicate< EQUIVALENCE_RELATION_RPI > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC bool csound::predicate< EQUIVALENCE_RELATION_RPT > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC bool csound::predicate< EQUIVALENCE_RELATION_RPTg > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC bool csound::predicate< EQUIVALENCE_RELATION_RPTgI > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC bool csound::predicate< EQUIVALENCE_RELATION_RPTI > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC bool csound::predicate< EQUIVALENCE_RELATION_T > (const Chord &chord, double range, double g, int opt_sector)
- template<> SILENCE_PUBLIC bool csound::predicate< EQUIVALENCE_RELATION_Tg > (const Chord &chord, double range, double g, int opt_sector)
- SILENCE_PUBLIC std::map< Chord, Chord > & csound::prime_forms_for_chords ()

  Cache normal forms for chords for speed.
- SILENCE_PUBLIC const char * csound::print_chord (const Chord &chord)

  Returns a string representation of the pitches in the chord, along with the sectors of the cyclical regions of the OPT and OPTI fundamental domains to which the chord belongs.
- static std::string csound::print_opti_sectors (const Chord &chord)
- SILENCE_PUBLIC Chord csound::reflect_by_householder (const Chord &chord)

  Computes the Householder reflector matrix and applies it to the chord.
- SILENCE_PUBLIC Chord csound::reflect_in_central_diagonal (const Chord &chord)
- SILENCE_PUBLIC Chord csound::reflect_in_central_point (const Chord &chord)
- SILENCE_PUBLIC Chord csound::reflect_in_inversion_flat (const Chord &chord, int opt_sector)
- SILENCE_PUBLIC Chord csound::reflect_in_unison_diagonal (const Chord &chord)
- SILENCE_PUBLIC Vector csound::reflect_vector (const Vector &point, const Vector &unit_normal_vector, double constant_term)

  Returns the point reflected in the hyperplane defined by the unit normal vector and constant term.
- SILENCE_PUBLIC Vector csound::reflect_vectorx (const Vector &v, const Vector &u, double c)
- SILENCE_PUBLIC Chord csound::scale (std::string name)

  Returns the named chord as a scale, that is, starting with the chord in OP, and sorting it from the tonic pitch-class on up.
- SILENCE_PUBLIC const Scale & csound::scaleForName (std::string name)
- SILENCE_PUBLIC std::map< std::string, Scale > & csound::scalesForNames ()
- static SILENCE_PUBLIC bool & csound::SCOPED_DEBUGGING_FLAG ()

  Returns the current state of the chord space scoped debugging flag as a reference, which can be an lvalue or an rvalue.
- static SILENCE_PUBLIC bool csound::SET_CHORD_SPACE_DEBUGGING (bool enabled)
- static SILENCE_PUBLIC bool csound::SET_SCOPED_DEBUGGING (bool enabled)
- SILENCE_PUBLIC std::vector< std::string > csound::split (std::string)
- SILENCE_PUBLIC double csound::T (double pitch, double semitones)

  Returns the pitch transposed by semitones, which may be any scalar.
- SILENCE_PUBLIC std::string csound::toString (const Matrix &mat)
- SILENCE_PUBLIC Chord csound::transpose_degrees (const Chord &scale, const Chord &original_chord, int transposition_degrees, int interval=3)

Returns the chord, in scale order, transposed within the scale by the indicated number of scale degrees, which can be positive or negative.

- SILENCE_PUBLIC std::set< Chord > & csound::unique_chords ()
- SILENCE_PUBLIC std::set< Scale > & csound::unique_scales ()
- SILENCE_PUBLIC Chord csound::voiceleading (const Chord &a, const Chord &b)

  Returns the voice-leading between chords a and b, i.e.
- SILENCE_PUBLIC Chord csound::voiceleadingCloser (const Chord &source, const Chord &d1, const Chord &d2, bool avoidParallels=false)

  Returns which of the voiceleadings (source to d1, source to d2) is the closer (first smoother, then simpler), optionally avoiding parallel fifths.
- SILENCE_PUBLIC Chord csound::voiceleadingClosestRange (const Chord &source, const Chord &destination, double range, bool avoidParallels)

  Returns the voicing of the destination which has the closest voice-leading from the source within the range, optionally avoiding parallel fifths.
- SILENCE_PUBLIC Chord csound::voiceleadingSimpler (const Chord &source, const Chord &d1, const Chord &d2, bool avoidParallels=false)

  Returns which of the voiceleadings (source to d1, source to d2) is the simpler (fewest moves), optionally avoiding parallel fifths.
- SILENCE_PUBLIC Chord csound::voiceleadingSmoother (const Chord &source, const Chord &d1, const Chord &d2, bool avoidParallels=false, double range=OCTAVE())

  Returns which of the voiceleadings (source to d1, source to d2) is the smoother (shortest moves), optionally avoiding parallel fifths.
- SILENCE_PUBLIC double csound::voiceleadingSmoothness (const Chord &a, const Chord &b)

  Returns the smoothness of the voiceleading between chords a and b by L1 norm.

Variables

- class SILENCE_PUBLIC csound::Chord
- static std::mt19937 csound::mersenne_twister
- static const char * csound::namesForEquivalenceRelations [ ]
- class SILENCE_PUBLIC csound::PITV
- class SILENCE_PUBLIC csound::Scale

7.7.1 Detailed Description

This library implements a geometric approach to some common operations on chords in neo-Riemannian music theory for use in score generating procedures:

- Identifying whether a chord belongs to some equivalence class of music theory, or sending a chord to its equivalent within a representative ("normal") fundamental domain of some equivalence relation. The equivalence relations are octave (O), permutational (P), transpositional, (T), inversional (I), and their compounds OP, OPT (set-class or chord type), and OPTI (similar to prime form), among others.
- Causing chord progressions to move strictly within an orbifold that reoresents some equivalence class.
- Implementing chord progressions based on the L, P, R, D, K, and Q operations of neo-Riemannian theory (thus implementing some aspects of "harmony").
- Implementing chord progressions performed within a more abstract equivalence class by means of the closest voice-leading within a less abstract equivalence class (thus implementing some fundamentals of "counterpoint").
- Implementing "functional" or "Roman numeral" operations performed using scales and scale degrees (thus implementing many fundamentals of "pragmatic music theory").

7.7.2 Definitions

Pitch is the perception of a distinct sound frequency. It is a logarithmic perception; octaves, which sound 'equivalent' in some sense, represent doublings or halvings of frequency.

Pitches and intervals are represented as real numbers. Middle C is 60 and the octave is 12. Our usual system of 12-tone equal temperament, as well as MIDI key numbers, are completely represented by the whole numbers; any and all other pitches can be represented simply by using fractions.

A voice is a distinct sound that is heard as having a pitch.

A chord is simply a set of voices heard at the same time, represented here as a point in a chord space having one dimension of pitch for each voice in the chord.

A scale is a chord with a tonic pitch-class as its first and lowest voice, all other voices being pitch-classes sorted in ascending order.

For the purposes of algorithmic composition, a score can be considered to be a sequence of more or less fleeting chords.

7.7.3 Equivalence Relations and Classes

An equivalence relation identifies different elements of a set as belonging to the same class. For example the octave is an equivalence relation that identifies C1, C2, and C3 as belonging to the equivalence class C. Operations that send elements to their equivalents induce quotient spaces or orbifolds, where the equivalence operation identifies points on one facet of the orbifold with points on an opposing facet. The fundamental domain of the equivalence relation is the space consisting of the orbifold and its surface.

Plain chord space has no equivalence relation. Ordered chords are represented as vectors in parentheses (p1, ..., pN). Unordered chords are represented as sorted vectors in braces {p1, ..., pN}. Unordering is itself an equivalence relation – permutational equivalence.

The following equivalence relations apply to pitches and chords, and exist in different orbifolds. Equivalence relations can be combined (Callendar, Quinn, and Tymoczko, "Generalized Voice-Leading Spaces," *Science* 320, 2008), and the more equivalence relations are combined, the more abstract is the resulting orbifold compared to the parent space.

In most cases, a chord space can be divided into a number, possibly infinite, of geometrically equivalent fundamental domains for the same equivalence relation. Therefore, here we use the notion of 'representative' or 'normal' fundamental domain. For example, the representative fundamental domain of unordered sequences, out of all possible orderings, consists of all sequences in their ordinary sorted order. It is important, in the following, to identify representative fundamental domains that combine properly, e.g. such that the representative fundamental domain of OP / the representative fundamental domain of PI equals the representative fundamental domain of OPI. And this in turn may require accounting for duplicate elements of the representative fundamental domain caused by reflections or singularities in the orbifold (e.g. on vertices, edges, or facets shared by fundamental domains with a cyclical structure), or by doubled pitches in a chord.

C Cardinality equivalence, e.g. {1, 1, 2} == {1, 2}. *Not* assuming cardinality equivalence ensures that there is a prototype metric in plain chord space that is inherited by all child chord spaces. Cardinality equivalence is never assumed here, because we are working in chord spaces of fixed dimensionality; e.g. we represent the note middle C not only as {60}, but also as {60, 60, ..., 60}.

O Octave equivalence. The fundamental domain is defined by the pitches in a chord spanning the range of an octave or less, and summing to an octave or less.

P Permutational equivalence. The fundamental domain is defined by a "wedge" of plain chord space in which the voices of a chord are always sorted by pitch.

T Transpositional equivalence, e.g. {1, 2} == {7, 8}. The fundamental domain is defined as a hyperplane in chord space at right angles to the diagonal of unison chords. Represented by the chord always having a sum of pitches equal to 0.

Tg Transpositional equivalence; the pitches of the chord are sent to the ceilings of the pitches in the first chord whose sum is equal to or greater than 0, i.e., rounded up to equal temperament.

I Inversional equivalence. Care is needed to distinguish the mathematician's sense of 'invert', which means 'pitch-space inversion' or 'reflect in a point', from the musician's sense of 'invert', which varies according to context but in practice often means 'registral inversion' or 'revoice by adding an octave to the lowest tone of a chord.' Here, we use 'invert' and 'inversion' in the mathematician's sense, and we use the terms 'revoice' and 'voicing' for the musician's 'invert' and 'inversion'. Here, the inversion of a chord is its reflection in a hyperplane (the inversion flat) that divides a fundamental domain of pitch.

PI Inversional equivalence with permutational equivalence. The 'inversion flat' of unordered chord space is a hyperplane consisting of all those unordered chords that are invariant under inversion. A fundamental domain is defined by any half space bounded by a hyperplane containing the inversion flat.

OP Octave equivalence with permutational equivalence. Tymoczko's orbifold for chords; i.e. chords with a fixed number of voices in a harmonic context. The fundamental domain is defined as a hyperprism one octave long with as many sides as voices and the ends identified by octave equivalence and one cyclical permutation of voices, modulo the unordering. In OP for trichords in 12TET, the augmented triads run up the middle of the prism, the major and minor triads are in 6 alternating columns around the augmented triads, the two-pitch chords form the 3 sides, and the one-pitch chords form the 3 edges that join the sides.

OPT The layer of the OP prism as close as possible to the origin, modulo the number of voices. Chord type. Note that CM and Cm are different OPT. Because the OP prism is canted down from the origin, at least one pitch in each OPT chord (excepting the origin itself) is negative. For n dimensions there are n OPT fundamental domains centering on the maximally even chord and generated by rotation about the maximally even chord, equivalently octavewise revoicing, more or less the same as the musician's sense of "chord inversion."

OPTT The same as OPT, but with chords rounded up within equal temperament; equivalent to "chord type."

OPI The OP prism modulo inversion, i.e. 1/2 of the OP prism. The representative fundamental consits of those chords having inversional equivalence.

OPTI The OPT layer modulo inversion, i.e. 1/2 of the OPT layer. Set-class. Note that minor and major triads are are the same OPTI.

OPTTI The same as OPTI, but with chords rounded up within equal temperament; equivalent to "set class."

7.7.4 Operations

Each of the above equivalence relations is, of course, an operation that sends chords outside some fundamental domain to chords inside that fundamental domain. We define the following additional operations:

T(p, x) Translate p by x.

I(p [, x]) Reflect p in x, by default the origin.

P Send a major triad to the minor triad with the same root, or vice versa (Riemann's parallel transformation).

L Send a major triad to the minor triad one major third higher, or vice versa (Riemann's Leittonwechsel or leading-tone exchange transformation).

R Send a major triad to the minor triad one minor third lower, or vice versa (Riemann's relative transformation).

D Send a triad to the next triad a perfect fifth lower (dominant transformation).

P, L, and R have been extended as follows, see Fiore and Satyendra, "Generalized Contextual Groups", *Music Theory Online* 11, August 2008:

K(c) Interchange by inversion; `K(c) := I(c, c[1] + c[2])`. This is a generalized form of P; for major and minor triads, it is exactly the same as P, but it also works with other chord types.

Q(c, n, m) Contexual transposition; `Q(c, n, m) := T(c, n)` if c is a T-form of m, or `T(c, -n)` if c is an I-form of M. Not a generalized form of L or R; but, like them, K and Q generate the T-I group.

7.7.5 Macro Definition Documentation

7.7.5.1 CHORD_SPACE_DEBUG

```
#define CHORD_SPACE_DEBUG if (CHORD_SPACE_DEBUGGING() == true) csound::System::message
```

Referenced by csound::Chord::ceiling(), csound::eq_tolerance(), csound::equate< EQUIVALENCE_RELATION_R >(), csound::fill(), csound::fundamentalDomainByPredicate(), csound::fundamentalDomainByTransformation(), csound::hyperplane_equation_ csound::indexForOctavewiseRevoicing(), csound::Chord::initialize_sectors(), csound::initializeNames(), csound::numerics_information(), csound::octavewiseRevoicing(), csound::octavewiseRevoicings(), csound::Chord::opti_domain_sectors(), csound::predicate< EQUIVALE csound::predicate< EQUIVALENCE_RELATION_T >(), csound::reflect_by_householder(), csound::reflect_vector(), csound::Scale::relative_tonicizations_for_scale_types(), csound::scale(), csound::Scale::tonicizations(), csound::Scale::transpose(), csound::transpose_degrees(), and csound::Scale::transpose_to_degree().

7.7.5.2 EIGEN_INITIALIZE_MATRICES_BY_ZERO

```
#define EIGEN_INITIALIZE_MATRICES_BY_ZERO
```

7.8 /Users/michaelgogins/csound-ac/CsoundAC/ChordSpaceTest.cpp File Reference

```
#include "ChordSpace.hpp"
#include <algorithm>
#include <iostream>
#include <cstdlib>
#include <cstdio>
#include <string>
```

Typedefs

- typedef csound::Chord(∗ equate_t) (const csound::Chord &, double, double, int)
- typedef std::vector< csound::Chord >(∗ fundamentalDomainByEquate_t) (int, double, double, int)
- typedef std::vector< csound::Chord >(∗ fundamentalDomainByPredicate_t) (int, double, double, int, bool)
- typedef Eigen::Matrix< double, Eigen::Dynamic, Eigen::Dynamic > Matrix
- typedef bool(∗ predicate_t) (const csound::Chord &, double, double, int)
- typedef Eigen::Matrix< double, Eigen::Dynamic, 1 > Vector

Functions

- static bool equals (const csound::HyperplaneEquation &a, const csound::HyperplaneEquation &b)
- static bool fail (std::string message)
- static void Hyperplane_Equation_for_Test_Points ()
- int main (int argc, char ∗∗argv)
- static bool pass (std::string message)
- static void printSet (std::string name, const std::vector< csound::Chord > &chords)
- static void setDifference (const std::string &a_name, std::vector< csound::Chord > &A, const std::string &b_↩ name, std::vector< csound::Chord > &B, std::vector< csound::Chord > &difference)

  Puts the set difference of A \ B, if any, into difference.
- static void summary ()
- static bool test (bool passes, std::string message)
- static void test_eq_tolerance ()
- static void test_nrL ()
- static void test_nrP ()
- static void test_nrR ()
- static void test_pitv (const csound::PITV &pitv_, std::string chordName)
- static void test_pitv (int initialVoiceCount, int finalVoiceCount)
- static bool testEquivalenceRelation (std::string equivalenceRelation, int voiceCount, double range, double g)
- static bool testEquivalenceRelations (int voiceCount, double range, double g)
- static bool testNormalsAndEquivalents (std::string equivalence, std::vector< csound::Chord > &made_↩ equivalents, std::vector< csound::Chord > &found_equivalents, double range, double g)

Variables

- std::map< std::string, equate_t > equatesForEquivalenceRelations
- std::map< std::string, std::set< std::string > > equivalenceRelationsForCompoundEquivalenceRelations
- std::vector< std::string > equivalenceRelationsToTest = {"RP", "RPT", "RPTg", "RPTI", "RPTgI"}
- static int exitAfterFailureCount = 5
- static int failureCount = 0
- static bool failureExits = false
- std::map< std::string, fundamentalDomainByEquate_t > fundamentalDomainByEquateForEquivalenceRelations
- std::map< std::string, fundamentalDomainByPredicate_t > fundamentalDomainByPredicateForEquivalenceRelations
- static int passCount = 0
- std::map< std::string, predicate_t > predicatesForEquivalenceRelations
- static bool printPass = true
- static bool printPitv = true
- static int testCount = 0
- static int testSector = 0

7.8.1 Typedef Documentation

7.8.1.1 equate_t

```
typedef csound::Chord(* equate_t) (const csound::Chord &, double, double, int)
```

7.8.1.2 fundamentalDomainByEquate_t

```
typedef std::vector< csound::Chord >(* fundamentalDomainByEquate_t) (int, double, double, int)
```

7.8.1.3 fundamentalDomainByPredicate_t

```
typedef std::vector< csound::Chord >(* fundamentalDomainByPredicate_t) (int, double, double, int,
bool)
```

7.8.1.4 Matrix

```
typedef Eigen::Matrix<double, Eigen::Dynamic, Eigen::Dynamic> Matrix
```

7.8.1.5 predicate_t

```
typedef bool(* predicate_t) (const csound::Chord &, double, double, int)
```

7.8.1.6 Vector

```
typedef Eigen::Matrix<double, Eigen::Dynamic, 1> Vector
```

7.8.2 Function Documentation

7.8.2.1 equals()

```
static bool equals (
            const csound::HyperplaneEquation & a,
            const csound::HyperplaneEquation & b )  [static]
```

References csound::HyperplaneEquation::constant_term, csound::eq_tolerance(), csound::System::error(), and csound::HyperplaneEquation::unit_normal_vector.

Referenced by Hyperplane_Equation_for_Test_Points().

7.8.2.2 fail()

```
static bool fail (
            std::string message )  [static]
```

References exitAfterFailureCount, failureCount, failureExits, csound::System::message(), passCount, and testCount.

Referenced by test().

7.8.2.3 Hyperplane_Equation_for_Test_Points()

```
static void Hyperplane_Equation_for_Test_Points ( )  [static]
```

References csound::HyperplaneEquation::constant_term, equals(), test(), and csound::HyperplaneEquation::unit_normal_vector.

Referenced by main().

7.8.2.4 main()

```
int main (
            int argc,
            char ** argv )
```

return 0;

return 0;

return 0;

return 0;

csound::SCOPED_DEBUGGING scoped_debugging;

References csound::allOfEquivalenceClass(), csound::Chord::ceiling(), csound::Chord::center(), csound::Chord, csound::chord_space_version(), csound::chordForName(), csound::conformToPitchClassSet(), csound::Chord::contains(), csound::Chord::count(), csound::Chord::cycle(), csound::Chord::distanceToOrigin(), csound::Chord::distanceToUnisonDiagonal(), csound::Chord::eI(), csound::Chord::eO(), csound::Chord::eP(), csound::epc(), csound::Chord::epcs(), equatesForEquivalenceRelations, equivalenceRelationsForCompoundEquivalenceRelations, csound::Chord::eT(), csound::Chord::et(), csound::Chord::eTT(), csound::fill(), csound::Chord::floor(), fundamentalDomainByEquateForEquivalenceRelations, csound::fundamentalDomainByPredicate(), fundamentalDomainByPredicateForEquivalenceRelations, Hyperplane_Equation_for_Test_Points(), csound::Chord::I(), csound::Chord::information(), csound::Chord::information_debug(), csound::PITV::initialize(), csound::Chord::iseI(), csound::Chord::iseO(), csound::Chord::iseP(), csound::Chord::isepcs(), csound::Chord::iseT(), csound::Chord::iset(), csound::Chord::iseTT(), csound::PITV::list(), csound::Chord::maximumInterval(), csound::System::message(), csound::Chord::min(), csound::Chord::minimumInterval(), csound::modulo(), csound::OCTAVE(), csound::Chord::permutations(), predicatesForEquivalenceRelations, printSet(), csound::Chord::reflect(), csound::reflect_by_householder(), csound::Chord::resize(), setDifference(), csound::System::setMessageLevel(), csound::Chord::setPitch(), summary(), csound::Chord::T(), test_eq_tolerance(), test_nrL(), test_nrP(), test_nrR(), test_pitv(), testEquivalenceRelations(), testSector, and csound::Chord::toString().

7.8.2.5 pass()

```
static bool pass (
            std::string message )  [static]
```

References failureCount, csound::System::message(), passCount, printPass, and testCount.

Referenced by test().

7.8.2.6 printSet()

```
static void printSet (
            std::string name,
            const std::vector< csound::Chord > & chords )  [static]
```

References csound::System::message().

Referenced by main().

7.8.2.7 setDifference()

```
static void setDifference (
            const std::string & a_name,
            std::vector< csound::Chord > & A,
            const std::string & b_name,
            std::vector< csound::Chord > & B,
            std::vector< csound::Chord > & difference )  [static]
```

Puts the set difference of A \ B, if any, into difference.

std::cerr << "less" << std::endl;

std::cerr << "not less" << std::endl;

References csound::Chord::eOPTT(), csound::Chord::normal_form(), and csound::Chord::toString().

Referenced by main().

7.8.2.8 summary()

```
static void summary ( )  [static]
```

References failureCount, csound::System::message(), passCount, and testCount.

Referenced by main().

7.8.2.9 test()

```
static bool test (
            bool passes,
            std::string message )  [static]
```

References fail(), and pass().

Referenced by Hyperplane_Equation_for_Test_Points(), test_nrL(), test_nrP(), test_nrR(), test_pitv(), test_pitv(), testEquivalenceRelation(), and testNormalsAndEquivalents().

7.8.2.10 test_eq_tolerance()

```
static void test_eq_tolerance ( )  [static]
```

References csound::eq_tolerance(), csound::ge_tolerance(), csound::gt_tolerance(), and csound::numerics_information().

Referenced by main().

7.8.2.11 test_nrL()

```
static void test_nrL ( )  [static]
```

References csound::chordForName(), csound::System::message(), and test().

Referenced by main().

7.8.2.12 test_nrP()

```
static void test_nrP ( )  [static]
```

References csound::chordForName(), csound::System::message(), and test().

Referenced by main().

7.8.2.13 test_nrR()

```
static void test_nrR ( )  [static]
```

References csound::chordForName(), csound::System::message(), and test().

Referenced by main().

7.8.2.14 test_pitv() [1/2]

```
static void test_pitv (
            const csound::PITV & pitv_,
            std::string chordName )  [static]
```

References csound::chordForName(), csound::Chord::eOP(), csound::Chord::eOPTTI(), csound::PITV::fromChord(), csound::Chord::getPitch(), csound::Chord::I(), csound::Chord::information(), csound::System::message(), printPitv, csound::Chord::setPitch(), test(), csound::PITV::toChord(), and csound::PITV::toChord_vector().

Referenced by main().

7.8.2.15 test_pitv() [2/2]

```
static void test_pitv (
            int initialVoiceCount,
            int finalVoiceCount )  [static]
```

References csound::PITV::countI, csound::PITV::countP, csound::PITV::countT, csound::PITV::countV, csound::Chord::equals(), csound::PITV::fromChord(), csound::Chord::information(), csound::PITV::initialize(), csound::PITV::list(), csound::System::message(), printPass, printPitv, test(), and csound::PITV::toChord().

7.8.2.16 testEquivalenceRelation()

```
static bool testEquivalenceRelation (
            std::string equivalenceRelation,
            int voiceCount,
            double range,
            double g )  [static]
```

References fundamentalDomainByEquateForEquivalenceRelations, fundamentalDomainByPredicateForEquivalenceRelations, csound::System::message(), test(), testNormalsAndEquivalents(), and testSector.

Referenced by testEquivalenceRelations().

7.8.2.17 testEquivalenceRelations()

```
static bool testEquivalenceRelations (
            int voiceCount,
            double range,
            double g )  [static]
```

References equivalenceRelationsToTest, csound::System::message(), and testEquivalenceRelation().

Referenced by main().

7.8.2.18 testNormalsAndEquivalents()

```
static bool testNormalsAndEquivalents (
            std::string equivalence,
            std::vector< csound::Chord > & made_equivalents,
            std::vector< csound::Chord > & found_equivalents,
            double range,
            double g )  [static]
```

References equatesForEquivalenceRelations, csound::System::message(), predicatesForEquivalenceRelations, and test().

Referenced by testEquivalenceRelation().

7.8.3 Variable Documentation

7.8.3.1 equatesForEquivalenceRelations

```
std::map<std::string, equate_t> equatesForEquivalenceRelations
```

Referenced by main(), and testNormalsAndEquivalents().

7.8.3.2 equivalenceRelationsForCompoundEquivalenceRelations

```
std::map<std::string, std::set<std::string> > equivalenceRelationsForCompoundEquivalenceRelations
```

Referenced by main().

7.8.3.3 equivalenceRelationsToTest

```
std::vector<std::string> equivalenceRelationsToTest = {"RP", "RPT", "RPTg", "RPTI", "RPTgI"}
```

Referenced by testEquivalenceRelations().

7.8.3.4 exitAfterFailureCount

```
int exitAfterFailureCount = 5  [static]
```

Referenced by fail().

7.8.3.5 failureCount

```
int failureCount = 0  [static]
```

Referenced by fail(), pass(), and summary().

7.8.3.6 failureExits

```
bool failureExits = false  [static]
```

Referenced by fail().

7.8.3.7 fundamentalDomainByEquateForEquivalenceRelations

```
std::map<std::string, fundamentalDomainByEquate_t> fundamentalDomainByEquateForEquivalence↩
Relations
```

Referenced by main(), and testEquivalenceRelation().

7.8.3.8 fundamentalDomainByPredicateForEquivalenceRelations

```
std::map<std::string, fundamentalDomainByPredicate_t> fundamentalDomainByPredicateForEquivalence↩
Relations
```

Referenced by main(), and testEquivalenceRelation().

7.8.3.9 passCount

```
int passCount = 0  [static]
```

Referenced by fail(), pass(), and summary().

7.8.3.10 predicatesForEquivalenceRelations

```
std::map<std::string, predicate_t> predicatesForEquivalenceRelations
```

Referenced by main(), and testNormalsAndEquivalents().

7.8.3.11 printPass

```
bool printPass = true  [static]
```

Referenced by pass(), and test_pitv().

7.8.3.12 printPitv

```
bool printPitv = true  [static]
```

Referenced by test_pitv(), and test_pitv().

7.8.3.13 testCount

```
int testCount = 0  [static]
```

Referenced by fail(), pass(), and summary().

7.8.3.14 testSector

```
int testSector = 0  [static]
```

Referenced by main(), and testEquivalenceRelation().

7.9 /Users/michaelgogins/csound-ac/CsoundAC/CMaskNode.hpp File Reference

```
#include "Platform.hpp"
#include "Conversions.hpp"
#include "Event.hpp"
#include "Score.hpp"
#include "ScoreNode.hpp"
#include <algorithm>
#include <cerrno>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <fstream>
#include <iostream>
#include <sstream>
#include <string>
#include <utility>
#include <vector>
#include "../dependencies/cmask/source/globals.h"
#include "../dependencies/cmask/source/event.h"
#include "../dependencies/cmask/source/field.h"
#include "../dependencies/cmask/source/fileio.h"
#include "../dependencies/cmask/source/gen.h"
#include "../dependencies/cmask/source/items.h"
#include "../dependencies/cmask/source/mask.h"
#include "../dependencies/cmask/source/parser.h"
#include "../dependencies/cmask/source/quant.h"
#include "../dependencies/cmask/source/tables.h"
#include "../dependencies/cmask/source/utils.h"
#include "../dependencies/cmask/source/cmask.cpp"
#include "../dependencies/cmask/source/event.cpp"
#include "../dependencies/cmask/source/field.cpp"
#include "../dependencies/cmask/source/fileio.cpp"
#include "../dependencies/cmask/source/gen.cpp"
#include "../dependencies/cmask/source/items.cpp"
```

```
#include "../dependencies/cmask/source/mask.cpp"
#include "../dependencies/cmask/source/parser.cpp"
#include "../dependencies/cmask/source/quant.cpp"
#include "../dependencies/cmask/source/tables.cpp"
#include "../dependencies/cmask/source/utils.cpp"
```

Data Structures

- class csound::CMaskNode

  Uses the CMask library for tendency masks to generate events as a Csound score in the format determined by the CMask parameters text.

Namespaces

- namespace cmask
- namespace csound

  C S O U N D.

Macros

- #define NL "\n"

7.9.1 Macro Definition Documentation

7.9.1.1 NL

```
#define NL "\n"
```

7.10 /Users/michaelgogins/csound-ac/CsoundAC/Composition.cpp File Reference

```
#include "Composition.hpp"
#include "System.hpp"
#include <algorithm>
#include <cstdlib>
#include <unistd.h>
#include <map>
```

Namespaces

- namespace csound

  C S O U N D.

7.11 /Users/michaelgogins/csound-ac/CsoundAC/Composition.hpp File Reference

```
#include "Platform.hpp"
#include "Score.hpp"
```

Data Structures

- class csound::Composition

  Base class for user-defined musical compositions.

Namespaces

- namespace csound

  C S O U N D.

7.12 /Users/michaelgogins/csound-ac/CsoundAC/Conversions.cpp File Reference

```
#include "Conversions.hpp"
#include <sstream>
#include <cstring>
#include <cstdlib>
```

Namespaces

- namespace csound

  C S O U N D.

Variables

- static bool csound::initialized__ = Conversions::initialize()

7.13 /Users/michaelgogins/csound-ac/CsoundAC/Conversions.hpp File Reference

```
#include "Platform.hpp"
#include <cmath>
#include <string>
#include <cstdio>
#include <map>
#include <vector>
```

Data Structures

- class csound::Conversions

  Conversions to and from various music and signal processing units.

Namespaces

- namespace csound

  C S O U N D.

7.14 /Users/michaelgogins/csound-ac/CsoundAC/Counterpoint.cpp File Reference

```
#include "CppSound.hpp"
#include "Counterpoint.hpp"
#include "System.hpp"
#include <vector>
#include <iostream>
#include <fstream>
```

7.15 /Users/michaelgogins/csound-ac/CsoundAC/Counterpoint.hpp File Reference

```
#include "Platform.hpp"
#include <string>
#include <cstdarg>
#include <stdio.h>
#include <stdlib.h>
#include <Eigen/Dense>
#include <random>
#include "Random.hpp"
```

Data Structures

- class Counterpoint

7.16 /Users/michaelgogins/csound-ac/CsoundAC/CounterpointMain.cpp File Reference

```
#include "CppSound.hpp"
#include "Counterpoint.hpp"
#include "System.hpp"
#include <vector>
#include <iostream>
#include <fstream>
```

Functions

- int main (int argc, char **argv)

7.16.1 Function Documentation

7.16.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

References Counterpoint::Aeolian, Counterpoint::AnySpecies(), Counterpoint::counterpoint(), Counterpoint::Dorian, Counterpoint::fillCantus(), Counterpoint::FillRhyPat(), csound::System::inform(), Counterpoint::Lydian, Counterpoint::Mixolydian, Counterpoint::Phrygian, Counterpoint::toCsoundScore(), and Counterpoint::vbs.

7.17 /Users/michaelgogins/csound-ac/CsoundAC/CounterpointNode.cpp File Reference

```
#include "CppSound.hpp"
#include "CounterpointNode.hpp"
#include "System.hpp"
#include "Conversions.hpp"
```

Namespaces

- namespace csound
  C S O U N D.

7.18 /Users/michaelgogins/csound-ac/CsoundAC/CounterpointNode.hpp File Reference

```
#include "Platform.hpp"
#include "Node.hpp"
#include "Counterpoint.hpp"
#include <cmath>
```

Data Structures

- class csound::CounterpointNode
  Uses Bill Schottstaedt's species counterpoint generator code to either (a) generate a counterpoint in species 1, 2, or 3 for a cantus firmus selected from notes generated by child nodes, or (b) attempt to correct the voice leading for species 1, 2, or 3 counterpoint in notes generated by child nodes.

Namespaces

- namespace csound
  C S O U N D.

7.19 /Users/michaelgogins/csound-ac/CsoundAC/CppSound.cpp File Reference

```
#include "CppSound.hpp"
#include <cstdio>
#include <cstring>
#include <ctime>
```

Macros

- #define __BUILDING_LIBCSOUND

Functions

- int argdecode (CSOUND *csound, int argc, const char **argv_)
  #include <csoundCore.h>

7.19.1 Macro Definition Documentation

7.19.1.1 __BUILDING_LIBCSOUND

```
#define __BUILDING_LIBCSOUND
```

7.19.2 Function Documentation

7.19.2.1 argdecode()

```
int argdecode (
            CSOUND * csound,
            int argc,
            const char ** argv_ )
```

#include <csoundCore.h>

7.20 /Users/michaelgogins/csound-ac/CsoundAC/CppSound.hpp File Reference

```
#include "float-version.h"
#include "csound.hpp"
#include "CsoundFile.hpp"
#include <string>
#include <vector>
```

Data Structures

- class CppSound

Macros

- #define __MYFLT_DEF
- #define MYFLT float

7.20.1 Macro Definition Documentation

7.20.1.1 __MYFLT_DEF

```
#define __MYFLT_DEF
```

7.20.1.2 MYFLT

```
#define MYFLT float
```

Referenced by CppSound::compile().

7.21 /Users/michaelgogins/csound-ac/CsoundAC/CsoundFile.cpp File Reference

```
#include "CsoundFile.hpp"
#include <algorithm>
#include <cctype>
#include <ctime>
#include <iterator>
#include <sstream>
#include <sys/types.h>
#include <sys/stat.h>
#include <csound.h>
#include <string>
#include <string.h>
#include <vector>
```

Functions

- int findToken (std::string text, std::string token, int position)
- void PUBLIC gatherArgs (int argc, const char **argv, std::string &commandLine)
- bool getline (std::istream &stream, std::string &buffer)

  Considerably more efficient than std::getline.
- bool isToken (std::string text, int position, std::string token)
- bool PUBLIC parseInstrument (const std::string &definition, std::string &preNumber, std::string &id, std::string &name, std::string &postNumber)

  Returns true if definition is a valid Csound instrument definition block.
- void PUBLIC scatterArgs (const std::string line, std::vector< std::string > &args, std::vector< char * > &argv)
- std::string PUBLIC & trim (std::string &value)
- std::string PUBLIC & trimQuotes (std::string &value)

Variables

- char staticBuffer [0x1000]

7.21.1 Function Documentation

7.21.1.1 findToken()

```
int findToken (
            std::string text,
            std::string token,
            int position )
```

References isToken().

Referenced by CsoundFile::getInstrument(), CsoundFile::getInstrument(), CsoundFile::getInstrumentCount(), CsoundFile::getInstrumentNames(), CsoundFile::getInstrumentNumber(), and CsoundFile::getOrchestraHeader().

7.21.1.2 gatherArgs()

```
void PUBLIC gatherArgs (
            int argc,
            const char ** argv,
            std::string & commandLine )
```

7.21.1.3 getline()

```
bool getline (
            std::istream & stream,
            std::string & buffer )
```

Considerably more efficient than std::getline.

References staticBuffer.

Referenced by CsoundFile::importArrangement(), CsoundFile::importCommand(), CsoundFile::importFile(), CsoundFile::importMidifile(), CsoundFile::importOrchestra(), and CsoundFile::importScore().

7.21.1.4 isToken()

```
bool isToken (
            std::string text,
            int position,
            std::string token )
```

Referenced by findToken().

7.21.1.5 parseInstrument()

```
bool PUBLIC parseInstrument (
            const std::string & definition,
            std::string & preNumber,
            std::string & id,
            std::string & name,
            std::string & postNumber )
```

Returns true if definition is a valid Csound instrument definition block.

Also returns the part before the instr number, the instr number, the name (all text after the first comment on the same line as the instr number), and the part after the instr number, all by reference.

References trim().

Referenced by CsoundFile::exportArrangementForPerformance(), CsoundFile::getInstrument(), CsoundFile::getInstrument(), CsoundFile::getInstrumentBody(), CsoundFile::getInstrumentBody(), CsoundFile::getInstrumentCount(), CsoundFile::getInstrumentName and CsoundFile::getInstrumentNumber().

7.21.1.6 scatterArgs()

```
void PUBLIC scatterArgs (
            const std::string line,
            std::vector< std::string > & args,
            std::vector< char * > & argv )
```

Referenced by CppSound::compile(), CsoundFile::getMidiFilename(), CsoundFile::getOrcFilename(), CsoundFile::getScoFilename(), and CppSound::perform().

7.21.1.7 trim()

```
std::string PUBLIC & trim (
            std::string & value )
```

Referenced by CsoundFile::getInstrument(), CsoundFile::getInstrumentNumber(), CsoundFile::importArrangement(), and parseInstrument().

7.21.1.8 trimQuotes()

```
std::string PUBLIC & trimQuotes (
            std::string & value )
```

7.21.2 Variable Documentation

7.21.2.1 staticBuffer

```
char staticBuffer[0x1000]
```

Referenced by getline().

7.22 /Users/michaelgogins/csound-ac/CsoundAC/CsoundFile.hpp File Reference

```
#include <iostream>
#include <string>
#include <vector>
#include <map>
#include <fstream>
#include <sstream>
#include <stdlib.h>
```

Data Structures

- class CsoundFile

  Manages a Csound Structured Data (CSD) file with facilities for creating an arrangement of selected instruments in the orchestra, and for programmatically building score files.

Macros

- #define PUBLIC

Functions

- void PUBLIC gatherArgs (int argc, const char **argv, std::string &commandLine)
- bool PUBLIC parseInstrument (const std::string &definition, std::string &preNumber, std::string &id, std::string &name, std::string &postNumber)

  Returns true if definition is a valid Csound instrument definition block.
- void PUBLIC scatterArgs (const std::string commandLine, std::vector< std::string > &args, std::vector< char * > &argv)
- std::string PUBLIC & trim (std::string &value)
- std::string PUBLIC & trimQuotes (std::string &value)

7.22.1 Macro Definition Documentation

7.22.1.1 PUBLIC

```
#define PUBLIC
```

7.22.2 Function Documentation

7.22.2.1 gatherArgs()

```
void PUBLIC gatherArgs (
            int argc,
            const char ** argv,
            std::string & commandLine )
```

7.22.2.2 parseInstrument()

```
bool PUBLIC parseInstrument (
            const std::string & definition,
            std::string & preNumber,
            std::string & id,
            std::string & name,
            std::string & postNumber )
```

Returns true if definition is a valid Csound instrument definition block.

Also returns the part before the instr number, the instr number, the name (all text after the first comment on the same line as the instr number), and the part after the instr number, all by reference.

References trim().

Referenced by CsoundFile::exportArrangementForPerformance(), CsoundFile::getInstrument(), CsoundFile::getInstrument(), CsoundFile::getInstrumentBody(), CsoundFile::getInstrumentBody(), CsoundFile::getInstrumentCount(), CsoundFile::getInstrumentName and CsoundFile::getInstrumentNumber().

7.22.2.3 scatterArgs()

```
void PUBLIC scatterArgs (
            const std::string commandLine,
            std::vector< std::string > & args,
            std::vector< char * > & argv )
```

Referenced by CppSound::compile(), CsoundFile::getMidiFilename(), CsoundFile::getOrcFilename(), CsoundFile::getScoFilename(), and CppSound::perform().

7.22.2.4 trim()

```
std::string PUBLIC & trim (
            std::string & value )
```

Referenced by CsoundFile::getInstrument(), CsoundFile::getInstrumentNumber(), CsoundFile::importArrangement(), and parseInstrument().

7.22.2.5 trimQuotes()

```
std::string PUBLIC & trimQuotes (
            std::string & value )
```

7.23 /Users/michaelgogins/csound-ac/CsoundAC/CsoundProducer.hpp File Reference

```
#include <csound_threaded.hpp>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <fstream>
#include <iostream>
#include <map>
#include <string>
```

Data Structures

- class csound::CsoundProducer

  Optionally adds metadata, performs post-processing, translates to various soundfile formats as automatic steps in the Csound rendering of a composition to a soundfile.

Namespaces

- namespace csound

  C S O U N D.

Functions

- static void csound::PostProcess (std::map< std::string, std::string > &tags, std::string filename, CsoundThreaded *csound)

  Uses ffmpeg to translate a soundfile to a normalized output file, an MP3 file, a CD audio file, a FLAC soundfile, and an MP4 video file suitable for posting to YouTube.

7.24 /Users/michaelgogins/csound-ac/CsoundAC/CsoundProducerTest.cpp File Reference

```
#include "CsoundProducer.hpp"
#include <iostream>
```

Functions

- int main (int argc, const char **argv)

Variables

- const char csd_text [ ]

7.24.1 Function Documentation

7.24.1.1 main()

```
int main (
            int argc,
            const char ** argv )
```

References csd_text.

7.24.2 Variable Documentation

7.24.2.1 csd_text

```
const char csd_text[]
```

Referenced by csound::MusicModel::cppsoundCompileCsdText(), and main().

7.25 /Users/michaelgogins/csound-ac/CsoundAC/dkm.hpp File Reference

```
#include <algorithm>
#include <array>
#include <cassert>
#include <cstddef>
#include <cstdint>
#include <random>
#include <tuple>
#include <type_traits>
#include <vector>
```

7.26 /Users/michaelgogins/csound-ac/CsoundAC/dkm_utils.hpp File Reference

```
#include <algorithm>
#include <array>
#include <tuple>
#include <vector>
#include "dkm.hpp"
```

7.27 /Users/michaelgogins/csound-ac/CsoundAC/ecl-test.cpp File Reference

```
#include <cstdio>
#include <ecl/ecl.h>
#include <iostream>
#include <string>
```

Functions

- cl_object evaluate_form (const std::string &form)
- int main (int argc, char **argv)

7.27.1 Function Documentation

7.27.1.1 evaluate_form()

```
cl_object evaluate_form (
            const std::string & form )
```

Referenced by main().

7.27.1.2 main()

```
int main (
            int argc,
            char ** argv )
```

References evaluate_form().

7.28 /Users/michaelgogins/csound-ac/CsoundAC/Event.cpp File Reference

```
#include "Event.hpp"
#include "Midifile.hpp"
```

Namespaces

- namespace csound
  C S O U N D.

Functions

- bool csound::getCorrectNegativeDurations ()
- bool csound::operator< (const Event &a, const Event &b)
- void csound::setCorrectNegativeDurations (bool do_correct)

7.29 /Users/michaelgogins/csound-ac/CsoundAC/Event.hpp File Reference

```
#include "Platform.hpp"
#include "Conversions.hpp"
#include <algorithm>
#include <Eigen/Dense>
#include <functional>
#include <iostream>
#include <map>
#include <sstream>
#include <string>
#include <utility>
#include <vector>
```

Data Structures

- class csound::Event

Namespaces

- namespace csound
  C S O U N D.

Functions

- bool csound::getCorrectNegativeDurations ()
- bool csound::operator< (const Event &a, const Event &b)
- void csound::setCorrectNegativeDurations (bool do_correct)

7.30 /Users/michaelgogins/csound-ac/CsoundAC/Exception.hpp File Reference

```
#include "Platform.hpp"
#include <string>
```

Data Structures

- class csound::Exception

  Base class for C++ exceptions in the Silence system.

Namespaces

- namespace csound

  C S O U N D.

7.31 /Users/michaelgogins/csound-ac/CsoundAC/ExternalNode.cpp File Reference

```
#include "Event.hpp"
#include "ExternalNode.hpp"
#include <cstdio>
#include <stdio.h>
#include <cstring>
#include <iostream>
#include <fstream>
#include <boost/process.hpp>
#include <boost/tokenizer.hpp>
#include "System.hpp"
```

Namespaces

- namespace csound

  C S O U N D.

Functions

- static void csound::parse_line (std::string line, Score &score)

7.32 /Users/michaelgogins/csound-ac/CsoundAC/ExternalNode.hpp File Reference

```
#include "Platform.hpp"
#include "Node.hpp"
#include "Score.hpp"
#include "ScoreNode.hpp"
#include <Eigen/Dense>
```

Data Structures

- class csound::ExternalNode

  ExternalNode runs a stored script with a specified command line, and imports Csound "i" statements printed by the script to stdout as CsoundAC Event objects in a CsoundAC Score.

Namespaces

- namespace csound

  C S O U N D.

7.33 /Users/michaelgogins/csound-ac/CsoundAC/ExternalNodeTest.cpp File Reference

```
#include <Silence.hpp>
```

Functions

- int main (int argc, const char **argv)

Variables

- auto script

7.33.1 Function Documentation

7.33.1.1 main()

```
int main (
            int argc,
            const char ** argv )
```

References csound::ExternalNode::generateLocally(), csound::Score::getCsoundScore(), csound::ScoreNode::getScore(), script, csound::ExternalNode::setCommand(), csound::System::setMessageLevel(), and csound::ExternalNode::setScript().

7.33.2 Variable Documentation

7.33.2.1 script

```
auto script
```

Initial value:

```
= R"(
import math

c = .98
y = 0.5
bass = 36
range_ = 60
for i in range(100):
    y1 = c * y * (1 - y) * 4
    y = y1
    midi_key = math.floor(bass + (y * range_))
    insno = 1
    time_ = i / 8.
    duration = .5
    midi_velocity = 60.
    print("i ", insno, time_, duration, midi_key, midi_velocity)
)"
```

Referenced by main().

7.34 /Users/michaelgogins/csound-ac/CsoundAC/filebuilding.cpp File Reference

```
#include "filebuilding.h"
#include <stdlib.h>
#include <string>
#include <vector>
#include <map>
```

Data Structures

- struct CsoundFile_

Functions

- PUBLIC void csoundCsdAddEvent10 (CSOUND *csound, double p1, double p2, double p3, double p4, double p5, double p6, double p7, double p8, double p9, double p10)

  Append an 'i' event to the CsScore element of the internal CSD file.
- PUBLIC void csoundCsdAddEvent11 (CSOUND *csound, double p1, double p2, double p3, double p4, double p5, double p6, double p7, double p8, double p9, double p10, double p11)

  Append an 'i' event to the CsScore element of the internal CSD file.
- PUBLIC void csoundCsdAddEvent3 (CSOUND *csound, double p1, double p2, double p3)

  Append an 'i' event to the CsScore element of the internal CSD file.
- PUBLIC void csoundCsdAddEvent4 (CSOUND *csound, double p1, double p2, double p3, double p4)

Append an 'i' event to the CsScore element of the internal CSD file.

- PUBLIC void csoundCsdAddEvent5 (CSOUND *csound, double p1, double p2, double p3, double p4, double p5)

  Append an 'i' event to the CsScore element of the internal CSD file.
- PUBLIC void csoundCsdAddEvent6 (CSOUND *csound, double p1, double p2, double p3, double p4, double p5, double p6)

  Append an 'i' event to the CsScore element of the internal CSD file.
- PUBLIC void csoundCsdAddEvent7 (CSOUND *csound, double p1, double p2, double p3, double p4, double p5, double p6, double p7)

  Append an 'i' event to the CsScore element of the internal CSD file.
- PUBLIC void csoundCsdAddEvent8 (CSOUND *csound, double p1, double p2, double p3, double p4, double p5, double p6, double p7, double p8)

  Append an 'i' event to the CsScore element of the internal CSD file.
- PUBLIC void csoundCsdAddEvent9 (CSOUND *csound, double p1, double p2, double p3, double p4, double p5, double p6, double p7, double p8, double p9)

  Append an 'i' event to the CsScore element of the internal CSD file.
- PUBLIC void csoundCsdAddScoreLine (CSOUND *csound, char *line)

  Append a line of text to the CsScore element of the internal CSD file.
- PUBLIC int csoundCsdCompile (CSOUND *csound, char *filename)

  Convenience function that saves the internal CSD file to the indicated filename, which must end in '.csd, then performs the file.
- PUBLIC void csoundCsdCreate (CSOUND *csound)

  Enables Python interface.
- PUBLIC const char * csoundCsdGetOptions (CSOUND *csound)

  Return the CsOptions element of the internal CSD file.
- PUBLIC const char * csoundCsdGetOrchestra (CSOUND *csound)

  Return the CsInstruments element of the internal CSD file.
- PUBLIC int csoundCsdPerform (CSOUND *csound, char *filename)

  Convenience function that saves the internal CSD file to the indicated filename, which must end in '.csd, then compiles the file for later performance.
- PUBLIC int csoundCsdSave (CSOUND *csound, char *filename)

  Save the internal CSD file to the indicated filename, which must end in '.csd'.
- PUBLIC void csoundCsdSetOptions (CSOUND *csound, char *options)

  Set the CsOptions element of the internal CSD file.
- PUBLIC void csoundCsdSetOrchestra (CSOUND *csound, char *orchestra)

  Set the CsInstruments element of the internal CSD file.
- PUBLIC void csoundNewCSD (char *path)
- PUBLIC int csoundPerformCsd (CSOUND *csound, char *csdFilename)

  Compiles and renders a Csound performance, as directed by the supplied CSD file, in one pass.
- PUBLIC int csoundPerformLoop (CSOUND *cs)
- uintptr_t perfthread (void *data)

Variables

- static std::map< CSOUND *, CsoundFile_ > files

7.34.1 Function Documentation

7.34.1.1 csoundCsdAddEvent10()

```
PUBLIC void csoundCsdAddEvent10 (
            CSOUND * csound,
            double p1,
            double p2,
            double p3,
            double p4,
            double p5,
            double p6,
            double p7,
            double p8,
            double p9,
            double p10 )
```

Append an 'i' event to the CsScore element of the internal CSD file.

References files.

7.34.1.2 csoundCsdAddEvent11()

```
PUBLIC void csoundCsdAddEvent11 (
            CSOUND * csound,
            double p1,
            double p2,
            double p3,
            double p4,
            double p5,
            double p6,
            double p7,
            double p8,
            double p9,
            double p10,
            double p11 )
```

Append an 'i' event to the CsScore element of the internal CSD file.

References files.

7.34.1.3 csoundCsdAddEvent3()

```
PUBLIC void csoundCsdAddEvent3 (
            CSOUND * csound,
            double p1,
            double p2,
            double p3 )
```

Append an 'i' event to the CsScore element of the internal CSD file.

References files.

7.34.1.4 csoundCsdAddEvent4()

```
PUBLIC void csoundCsdAddEvent4 (
             CSOUND * csound,
             double p1,
             double p2,
             double p3,
             double p4 )
```

Append an 'i' event to the CsScore element of the internal CSD file.

References files.

7.34.1.5 csoundCsdAddEvent5()

```
PUBLIC void csoundCsdAddEvent5 (
             CSOUND * csound,
             double p1,
             double p2,
             double p3,
             double p4,
             double p5 )
```

Append an 'i' event to the CsScore element of the internal CSD file.

References files.

7.34.1.6 csoundCsdAddEvent6()

```
PUBLIC void csoundCsdAddEvent6 (
             CSOUND * csound,
             double p1,
             double p2,
             double p3,
             double p4,
             double p5,
             double p6 )
```

Append an 'i' event to the CsScore element of the internal CSD file.

References files.

7.34.1.7 csoundCsdAddEvent7()

```
PUBLIC void csoundCsdAddEvent7 (
             CSOUND * csound,
             double p1,
             double p2,
             double p3,
             double p4,
             double p5,
             double p6,
             double p7 )
```

Append an 'i' event to the CsScore element of the internal CSD file.

References files.

7.34.1.8 csoundCsdAddEvent8()

```
PUBLIC void csoundCsdAddEvent8 (
             CSOUND * csound,
             double p1,
             double p2,
             double p3,
             double p4,
             double p5,
             double p6,
             double p7,
             double p8 )
```

Append an 'i' event to the CsScore element of the internal CSD file.

References files.

7.34.1.9 csoundCsdAddEvent9()

```
PUBLIC void csoundCsdAddEvent9 (
             CSOUND * csound,
             double p1,
             double p2,
             double p3,
             double p4,
             double p5,
             double p6,
             double p7,
             double p8,
             double p9 )
```

Append an 'i' event to the CsScore element of the internal CSD file.

References files.

7.34.1.10 csoundCsdAddScoreLine()

```
PUBLIC void csoundCsdAddScoreLine (
             CSOUND * csound,
             char * line )
```

Append a line of text to the CsScore element of the internal CSD file.

References files.

7.34.1.11 csoundCsdCompile()

```
PUBLIC int csoundCsdCompile (
             CSOUND * csound,
             char * filename )
```

Convenience function that saves the internal CSD file to the indicated filename, which must end in '.csd, then performs the file.

References csoundCsdSave().

7.34.1.12 csoundCsdCreate()

```
PUBLIC void csoundCsdCreate (
             CSOUND * csound )
```

Enables Python interface.

Initialize an internal CSD file.

References files.

7.34.1.13 csoundCsdGetOptions()

```
PUBLIC const char * csoundCsdGetOptions (
             CSOUND * csound )
```

Return the CsOptions element of the internal CSD file.

References files.

7.34.1.14 csoundCsdGetOrchestra()

```
PUBLIC const char * csoundCsdGetOrchestra (
             CSOUND * csound )
```

Return the CsInstruments element of the internal CSD file.

References files.

7.34.1.15 csoundCsdPerform()

```
PUBLIC int csoundCsdPerform (
             CSOUND * csound,
             char * filename )
```

Convenience function that saves the internal CSD file to the indicated filename, which must end in '.csd, then compiles the file for later performance.

References csoundCsdSave(), and csoundPerformCsd().

7.34.1.16 csoundCsdSave()

```
PUBLIC int csoundCsdSave (
             CSOUND * csound,
             char * filename )
```

Save the internal CSD file to the indicated filename, which must end in '.csd'.

References files, CsoundFile_::options, CsoundFile_::orchestra, and CsoundFile_::score.

Referenced by csoundCsdCompile(), and csoundCsdPerform().

7.34.1.17 csoundCsdSetOptions()

```
PUBLIC void csoundCsdSetOptions (
             CSOUND * csound,
             char * options )
```

Set the CsOptions element of the internal CSD file.

References files, and CsoundFile_::options.

7.34.1.18 csoundCsdSetOrchestra()

```
PUBLIC void csoundCsdSetOrchestra (
             CSOUND * csound,
             char * orchestra )
```

Set the CsInstruments element of the internal CSD file.

References files.

7.34.1.19 csoundNewCSD()

```
PUBLIC void csoundNewCSD (
             char * path )
```

References perfthread().

7.34.1.20 csoundPerformCsd()

```
PUBLIC int csoundPerformCsd (
             CSOUND * csound,
             char * csdFilename )
```

Compiles and renders a Csound performance, as directed by the supplied CSD file, in one pass.

Returns 0 for success.

Referenced by csoundCsdPerform().

7.34.1.21 csoundPerformLoop()

```
PUBLIC int csoundPerformLoop (
             CSOUND * cs )
```

References perfthread().

7.34.1.22 perfthread()

```
uintptr_t perfthread (
             void * data )
```

Referenced by csoundNewCSD(), and csoundPerformLoop().

7.34.2 Variable Documentation

7.34.2.1 files

```
std::map<CSOUND *, CsoundFile_> files  [static]
```

Referenced by csoundCsdAddEvent10(), csoundCsdAddEvent11(), csoundCsdAddEvent3(), csoundCsdAddEvent4(), csoundCsdAddEvent5(), csoundCsdAddEvent6(), csoundCsdAddEvent7(), csoundCsdAddEvent8(), csoundCsdAddEvent9(), csoundCsdAddScoreLine(), csoundCsdCreate(), csoundCsdGetOptions(), csoundCsdGetOrchestra(), csoundCsdSave(), csoundCsdSetOptions(), and csoundCsdSetOrchestra().

7.35 /Users/michaelgogins/csound-ac/CsoundAC/filebuilding.h File Reference

Csound API functions to create, build up, and save CSD files.

```
#include "csound.h"
#include "sysdep.h"
#include "text.h"
#include <stdarg.h>
```

Macros

- #define PUBLIC

Functions

- PUBLIC void csoundCsdAddEvent10 (CSOUND *csound, double p1, double p2, double p3, double p4, double p5, double p6, double p7, double p8, double p9, double p10)

  Append an 'i' event to the CsScore element of the internal CSD file.
- PUBLIC void csoundCsdAddEvent11 (CSOUND *csound, double p1, double p2, double p3, double p4, double p5, double p6, double p7, double p8, double p9, double p10, double p11)

  Append an 'i' event to the CsScore element of the internal CSD file.
- PUBLIC void csoundCsdAddEvent3 (CSOUND *csound, double p1, double p2, double p3)

  Append an 'i' event to the CsScore element of the internal CSD file.
- PUBLIC void csoundCsdAddEvent4 (CSOUND *csound, double p1, double p2, double p3, double p4)

  Append an 'i' event to the CsScore element of the internal CSD file.
- PUBLIC void csoundCsdAddEvent5 (CSOUND *csound, double p1, double p2, double p3, double p4, double p5)

  Append an 'i' event to the CsScore element of the internal CSD file.
- PUBLIC void csoundCsdAddEvent6 (CSOUND *csound, double p1, double p2, double p3, double p4, double p5, double p6)

  Append an 'i' event to the CsScore element of the internal CSD file.
- PUBLIC void csoundCsdAddEvent7 (CSOUND *csound, double p1, double p2, double p3, double p4, double p5, double p6, double p7)

  Append an 'i' event to the CsScore element of the internal CSD file.
- PUBLIC void csoundCsdAddEvent8 (CSOUND *csound, double p1, double p2, double p3, double p4, double p5, double p6, double p7, double p8)

  Append an 'i' event to the CsScore element of the internal CSD file.
- PUBLIC void csoundCsdAddEvent9 (CSOUND *csound, double p1, double p2, double p3, double p4, double p5, double p6, double p7, double p8, double p9)

  Append an 'i' event to the CsScore element of the internal CSD file.
- PUBLIC void csoundCsdAddScoreLine (CSOUND *csound, char *line)

  Append a line of text to the CsScore element of the internal CSD file.
- PUBLIC int csoundCsdCompile (CSOUND *csound, char *filename)

  Convenience function that saves the internal CSD file to the indicated filename, which must end in '.csd, then performs the file.
- PUBLIC void csoundCsdCreate (CSOUND *csound)

  Enables Python interface.
- PUBLIC const char * csoundCsdGetOptions (CSOUND *csound)

  Return the CsOptions element of the internal CSD file.
- PUBLIC const char * csoundCsdGetOrchestra (CSOUND *csound)

  Return the CsInstruments element of the internal CSD file.
- PUBLIC int csoundCsdPerform (CSOUND *csound, char *filename)

  Convenience function that saves the internal CSD file to the indicated filename, which must end in '.csd, then compiles the file for later performance.
- PUBLIC int csoundCsdSave (CSOUND *csound, char *filename)

  Save the internal CSD file to the indicated filename, which must end in '.csd'.
- PUBLIC void csoundCsdSetOptions (CSOUND *csound, char *options)

  Set the CsOptions element of the internal CSD file.
- PUBLIC void csoundCsdSetOrchestra (CSOUND *csound, char *orchestra)

  Set the CsInstruments element of the internal CSD file.
- PUBLIC int csoundPerformCsd (CSOUND *, char *csdFilename)

  Compiles and renders a Csound performance, as directed by the supplied CSD file, in one pass.

7.35.1 Detailed Description

Csound API functions to create, build up, and save CSD files.

Author

Michael Gogins

Purpose

The purpose of these functions is to make it easier for clients of the Csound API to programmatically build up CSD files, including set instrument definitions, set options, and especially append score statements.

There are also convenience functions to compile and perform the saved CSD file.

7.35.2 Macro Definition Documentation

7.35.2.1 PUBLIC

```
#define PUBLIC
```

7.35.3 Function Documentation

7.35.3.1 csoundCsdAddEvent10()

```
PUBLIC void csoundCsdAddEvent10 (
             CSOUND * csound,
             double p1,
             double p2,
             double p3,
             double p4,
             double p5,
             double p6,
             double p7,
             double p8,
             double p9,
             double p10 )
```

Append an 'i' event to the CsScore element of the internal CSD file.

References files.

7.35.3.2 csoundCsdAddEvent11()

```
PUBLIC void csoundCsdAddEvent11 (
             CSOUND * csound,
             double p1,
             double p2,
             double p3,
             double p4,
             double p5,
             double p6,
             double p7,
             double p8,
             double p9,
             double p10,
             double p11 )
```

Append an 'i' event to the CsScore element of the internal CSD file.

References files.

7.35.3.3 csoundCsdAddEvent3()

```
PUBLIC void csoundCsdAddEvent3 (
             CSOUND * csound,
             double p1,
             double p2,
             double p3 )
```

Append an 'i' event to the CsScore element of the internal CSD file.

References files.

7.35.3.4 csoundCsdAddEvent4()

```
PUBLIC void csoundCsdAddEvent4 (
             CSOUND * csound,
             double p1,
             double p2,
             double p3,
             double p4 )
```

Append an 'i' event to the CsScore element of the internal CSD file.

References files.

7.35.3.5 csoundCsdAddEvent5()

```
PUBLIC void csoundCsdAddEvent5 (
             CSOUND * csound,
             double p1,
             double p2,
             double p3,
             double p4,
             double p5 )
```

Append an 'i' event to the CsScore element of the internal CSD file.

References files.

7.35.3.6 csoundCsdAddEvent6()

```
PUBLIC void csoundCsdAddEvent6 (
             CSOUND * csound,
             double p1,
             double p2,
             double p3,
             double p4,
             double p5,
             double p6 )
```

Append an 'i' event to the CsScore element of the internal CSD file.

References files.

7.35.3.7 csoundCsdAddEvent7()

```
PUBLIC void csoundCsdAddEvent7 (
             CSOUND * csound,
             double p1,
             double p2,
             double p3,
             double p4,
             double p5,
             double p6,
             double p7 )
```

Append an 'i' event to the CsScore element of the internal CSD file.

References files.

7.35.3.8 csoundCsdAddEvent8()

```
PUBLIC void csoundCsdAddEvent8 (
             CSOUND * csound,
             double p1,
             double p2,
             double p3,
             double p4,
             double p5,
             double p6,
             double p7,
             double p8 )
```

Append an 'i' event to the CsScore element of the internal CSD file.

References files.

7.35.3.9 csoundCsdAddEvent9()

```
PUBLIC void csoundCsdAddEvent9 (
             CSOUND * csound,
             double p1,
             double p2,
             double p3,
             double p4,
             double p5,
             double p6,
             double p7,
             double p8,
             double p9 )
```

Append an 'i' event to the CsScore element of the internal CSD file.

References files.

7.35.3.10 csoundCsdAddScoreLine()

```
PUBLIC void csoundCsdAddScoreLine (
             CSOUND * csound,
             char * line )
```

Append a line of text to the CsScore element of the internal CSD file.

References files.

7.35.3.11 csoundCsdCompile()

```
PUBLIC int csoundCsdCompile (
             CSOUND * csound,
             char * filename )
```

Convenience function that saves the internal CSD file to the indicated filename, which must end in '.csd, then performs the file.

References csoundCsdSave().

7.35.3.12 csoundCsdCreate()

```
PUBLIC void csoundCsdCreate (
             CSOUND * csound )
```

Enables Python interface.

Initialize an internal CSD file.

References files.

7.35.3.13 csoundCsdGetOptions()

```
PUBLIC const char * csoundCsdGetOptions (
             CSOUND * csound )
```

Return the CsOptions element of the internal CSD file.

References files.

7.35.3.14 csoundCsdGetOrchestra()

```
PUBLIC const char * csoundCsdGetOrchestra (
             CSOUND * csound )
```

Return the CsInstruments element of the internal CSD file.

References files.

7.35.3.15 csoundCsdPerform()

```
PUBLIC int csoundCsdPerform (
             CSOUND * csound,
             char * filename )
```

Convenience function that saves the internal CSD file to the indicated filename, which must end in '.csd, then compiles the file for later performance.

References csoundCsdSave(), and csoundPerformCsd().

7.35.3.16 csoundCsdSave()

```
PUBLIC int csoundCsdSave (
             CSOUND * csound,
             char * filename )
```

Save the internal CSD file to the indicated filename, which must end in '.csd'.

References files, CsoundFile_::options, CsoundFile_::orchestra, and CsoundFile_::score.

Referenced by csoundCsdCompile(), and csoundCsdPerform().

7.35.3.17 csoundCsdSetOptions()

```
PUBLIC void csoundCsdSetOptions (
             CSOUND * csound,
             char * options )
```

Set the CsOptions element of the internal CSD file.

References files, and CsoundFile_::options.

7.35.3.18 csoundCsdSetOrchestra()

```
PUBLIC void csoundCsdSetOrchestra (
             CSOUND * csound,
             char * orchestra )
```

Set the CsInstruments element of the internal CSD file.

References files.

7.35.3.19 csoundPerformCsd()

```
PUBLIC int csoundPerformCsd (
             CSOUND * csound,
             char * csdFilename )
```

Compiles and renders a Csound performance, as directed by the supplied CSD file, in one pass.

Returns 0 for success.

Referenced by csoundCsdPerform().

7.36 /Users/michaelgogins/csound-ac/CsoundAC/float-version.h File Reference

7.37 /Users/michaelgogins/csound-ac/CsoundAC/gcg_duality.cpp File Reference

```
#include "ChordSpace.hpp"
#include <algorithm>
#include <iostream>
#include <cstdlib>
#include <cstdio>
#include <string>
```

Typedefs

- typedef Eigen::Matrix< double, Eigen::Dynamic, Eigen::Dynamic > Matrix
- typedef Eigen::Matrix< double, Eigen::Dynamic, 1 > Vector

Functions

- static bool is_k_dual (const csound::Chord &chord_)
- int main (int argc, char **argv)
- static void print_dualities (std::vector< csound::Chord > &chords)

Variables

- static bool printPitv = true
- static int testSector = 0

7.37.1 Typedef Documentation

7.37.1.1 Matrix

```
typedef Eigen::Matrix<double, Eigen::Dynamic, Eigen::Dynamic> Matrix
```

7.37.1.2 Vector

```
typedef Eigen::Matrix<double, Eigen::Dynamic, 1> Vector
```

7.37.2 Function Documentation

7.37.2.1 is_k_dual()

```
static bool is_k_dual (
            const csound::Chord & chord_ )  [static]
```

References csound::Chord::eOP(), csound::Chord::eOPTT(), csound::Chord::K(), csound::Chord::name(), and csound::Chord::toString().

Referenced by main(), and print_dualities().

7.37.2.2 main()

```
int main (
            int argc,
            char ** argv )
```

References csound::chord_space_version(), csound::fundamentalDomainByPredicate(), is_k_dual(), print_dualities(), csound::scaleForName(), and testSector.

7.37.2.3 print_dualities()

```
static void print_dualities (
            std::vector< csound::Chord > & chords )  [static]
```

References is_k_dual().

Referenced by main().

7.37.3 Variable Documentation

7.37.3.1 printPitv

```
bool printPitv = true  [static]
```

7.37.3.2 testSector

```
int testSector = 0  [static]
```

Referenced by main().

7.38 /Users/michaelgogins/csound-ac/CsoundAC/HarmonyIFS.hpp File Reference

```
#include "Platform.hpp"
#include "ChordSpaceBase.hpp"
#include "ChordSpace.hpp"
#include "Conversions.hpp"
#include "Event.hpp"
#include "Score.hpp"
#include "ScoreNode.hpp"
#include "Node.hpp"
#include "System.hpp"
#include <sstream>
#include <stack>
#include <string>
#include <map>
#include <vector>
#include <Eigen/Dense>
```

Data Structures

- struct csound::HarmonyEvent

  Associates a Chord with an Event representing a musical note.
- class csound::HarmonyIFS

  HarmonyIFS is a class for doing algorithmic music composition by means of fractal interpolation functions.
- class csound::HarmonyInterpolationPoint

  *Represents an interpolation point with scaling factors for a fractal interpolation function in the **time-harmony subspace** of the score space.*
- class csound::HarmonyPoint

  Represents a point on a time line in a score space that has a time- harmony subspace.

Namespaces

- namespace csound

  C S O U N D.

Functions

- SILENCE_PUBLIC bool csound::interpolation_point_less (const HarmonyInterpolationPoint &a, const HarmonyInterpolationPoint &b)

7.39 /Users/michaelgogins/csound-ac/CsoundAC/HarmonyIFS2.hpp File Reference

```
#include "Platform.hpp"
#include "ChordSpaceBase.hpp"
#include "ChordSpace.hpp"
#include "Conversions.hpp"
#include "Event.hpp"
#include "Score.hpp"
#include "ScoreNode.hpp"
#include "Node.hpp"
#include "System.hpp"
#include <sstream>
#include <stack>
#include <string>
#include <map>
#include <vector>
#include <Eigen/Dense>
```

Data Structures

- class csound::HarmonyIFS2

  HarmonyIFS is a class for doing algorithmic music composition by means of fractal interpolation functions.
- class csound::HarmonyInterpolationPoint2

  *Represents an interpolation point with scaling factors for a fractal interpolation function in the **time-harmony subspace** of the score space.*
- class csound::HarmonyPoint2

  Represents a point on a time line in a score space that has a time- harmony subspace.

Namespaces

- namespace csound

  C S O U N D.

Functions

- SILENCE_PUBLIC bool csound::interpolation_point_less2 (const HarmonyInterpolationPoint2 &a, const HarmonyInterpolationPoint2 &b)

7.40 /Users/michaelgogins/csound-ac/CsoundAC/HarmonyIfs2Test.cpp File Reference

```
#include <Composition.hpp>
#include <HarmonyIFS2.hpp>
#include <cmath>
#include <Eigen/Dense>
#include <functional>
#include <memory>
#include <MusicModel.hpp>
#include <random>
#include <Rescale.hpp>
#include <ScoreNode.hpp>
#include <VoiceleadingNode.hpp>
#include <vector>
```

Functions

- int main (int argc, const char **argv)

7.40.1 Function Documentation

7.40.1.1 main()

```
int main (
            int argc,
            const char ** argv )
```

auto &score = harmony_ifs.getScore();

References csound::HarmonyIFS2::add_interpolation_point_as_chord(), csound::Node::addChild(), csound::ScoreModel::addChild(), csound::chordForName(), csound::HarmonyIFS2::generate_score_attractor(), csound::HarmonyIFS2::initialize(), csound::HarmonyIFS2::initialize_hutchinson_operator(), csound::Event::INSTRUMENT, csound::System::message(), csound::octavewiseRevoicing(), csound::Composition::processArgv(), csound::Composition::setAlbum(), csound::Composition::setAuthor(csound::MusicModel::setCsoundOrchestra(), csound::MusicModel::setCsoundScoreHeader(), csound::Composition::setDuration(), csound::System::setMessageLevel(), csound::Composition::setPerformanceRightsOrganization(), csound::Rescale::setRescale(), csound::Composition::setTitle(), csound::Composition::setYear(), and csound::Event::VELOCITY.

7.41 /Users/michaelgogins/csound-ac/CsoundAC/HarmonyIfsTest.cpp File Reference

```
#include <Composition.hpp>
#include <MCRM.hpp>
#include <cmath>
#include <Eigen/Dense>
```

```
#include <functional>
#include <memory>
#include <MusicModel.hpp>
#include <random>
#include <ScoreNode.hpp>
#include <VoiceleadingNode.hpp>
#include <vector>
```

Functions

- int main (int argc, const char **argv)

7.41.1 Function Documentation

7.41.1.1 main()

```
int main (
            int argc,
            const char ** argv )
```

auto &score = harmony_ifs.getScore();

References csound::HarmonyIFS::add_interpolation_point_as_chord(), csound::Node::addChild(), csound::ScoreModel::addChild(), csound::chordForName(), csound::HarmonyIFS::generate_score_attractor(), csound::HarmonyIFS::initialize(), csound::HarmonyIFS::initialize_hutchinson_operator(), csound::Event::INSTRUMENT, csound::System::message(), csound::Composition::processArgv(), csound::HarmonyIFS::set_transformation(), csound::Composition::setAlbum(), csound::Composition::setAuthor(), csound::MusicModel::setCsoundOrchestra(), csound::MusicModel::setCsoundScoreHeader(), csound::Composition::setDuration(), csound::System::setMessageLevel(), csound::Composition::setPerformanceRightsOrganization(), csound::Rescale::setRescale(), csound::Composition::setTitle(), csound::Composition::setYear(), and csound::Event::VELOCITY.

7.42 /Users/michaelgogins/csound-ac/CsoundAC/ImageToScore.cpp File Reference

```
#include "CppSound.hpp"
#include "ImageToScore.hpp"
#include "System.hpp"
#include <cmath>
#include <complex>
#include <set>
#include <functional>
#include <opencv2/imgcodecs.hpp>
```

Namespaces

- namespace csound

  C S O U N D.

7.43 /Users/michaelgogins/csound-ac/CsoundAC/ImageToScore.hpp File Reference

```
#include "Platform.hpp"
#include "Silence.hpp"
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/highgui/highgui.hpp>
```

Data Structures

- class csound::ImageToScore2

  Translates images files to scores.

Namespaces

- namespace csound

  C S O U N D.

Typedefs

- typedef ImageToScore2 csound::ImageToScore

  Only for backwards compatibility.

7.44 /Users/michaelgogins/csound-ac/CsoundAC/Lindenmayer.cpp File Reference

```
#include "CppSound.hpp"
#include "Lindenmayer.hpp"
#include <iostream>
#include <iomanip>
#include <ios>
#include <sstream>
#include <stdio.h>
```

Namespaces

- namespace csound

  C S O U N D.

7.45 /Users/michaelgogins/csound-ac/CsoundAC/Lindenmayer.hpp File Reference

```
#include "Platform.hpp"
#include "Silence.hpp"
#include <stack>
#include <string>
#include <map>
#include <vector>
#include <Eigen/Dense>
```

Data Structures

- class csound::Lindenmayer

  This class implements a Lindenmayer system in music space for a turtle that writes either notes into a score, or Jones-Parks grains into a memory soundfile.

Namespaces

- namespace csound

  C S O U N D.

7.46 /Users/michaelgogins/csound-ac/CsoundAC/Lisp.cpp File Reference

```
#include "Platform.hpp"
#include <limits>
#include <map>
#include <sstream>
#include "Lisp.hpp"
#include "Score.hpp"
#include "System.hpp"
#include <ecl/ecl.h>
#include <Eigen/Dense>
```

Namespaces

- namespace csound

  C S O U N D.

Macros

- #define LISP_H

Functions

- cl_object csound::evaluate_form (const std::string &form)

  Evaluates a SINGLE Lisp form.
- void csound::initialize_ecl (int argc, char **argv)

  This function must be called with the arc and argv from main() before any Lisp code is executed.
- cl_object csound::scoreToSeq (Score &score, std::string seq_name)

  Translates a Silence Score to Common Music seq.
- void csound::seqToScore (cl_object &seq_, Score &score)

  Translates a Common Music seq to a Silence Score.
- std::string csound::to_std_string (cl_object lisp_string)

  Translate a Lisp string to a C++ string.

7.46.1 Macro Definition Documentation

7.46.1.1 LISP_H

```
#define LISP_H
```

7.47 /Users/michaelgogins/csound-ac/CsoundAC/Lisp.hpp File Reference

```
#include "Platform.hpp"
#include <limits>
#include <map>
#include "Node.hpp"
#include <ecl/ecl.h>
#include <Eigen/Dense>
```

Data Structures

- struct csound::are_cl_objects< ... >
- struct csound::are_cl_objects< Head, Tail... >
- struct csound::is_cl_object< T >
- struct csound::is_cl_object< cl_object >
- class csound::LispGenerator

  Node that uses Lisp code to generate Events.
- class csound::LispNode

  Base class for Nodes that can use embedded Lisp code to generate or transform Events.
- class csound::LispTransformer

  Node that uses Lisp code to transform Events produced by child Nodes.

Namespaces

- namespace csound

  C S O U N D.

Functions

- template<typename... Params>
  void csound::defun (const std::string &name, cl_object fun(Params... params))
  Creates a DEFUN abstraction in C++.
- cl_object csound::evaluate_form (const std::string &form)
  Evaluates a SINGLE Lisp form.
- void csound::initialize_ecl (int argc, char **argv)
  This function must be called with the arc and argv from main() before any Lisp code is executed.
- cl_object csound::scoreToSeq (Score &score, std::string seq_name)
  Translates a Silence Score to Common Music seq.
- void csound::seqToScore (cl_object &seq_, Score &score)
  Translates a Common Music seq to a Silence Score.
- std::string csound::to_std_string (cl_object lisp_string)
  Translate a Lisp string to a C++ string.

7.48 /Users/michaelgogins/csound-ac/CsoundAC/LispNodeTest.cpp File Reference

```
#include <Composition.hpp>
#include <MCRM.hpp>
#include <Eigen/Dense>
#include <functional>
#include <memory>
#include <MusicModel.hpp>
#include <random>
#include <Lisp.hpp>
#include <VoiceleadingNode.hpp>
#include <vector>
```

Functions

- int main (int argc, const char **argv)
  All composition and synthesis code is defined in the main function.

7.48.1 Function Documentation

7.48.1.1 main()

```
int main (
            int argc,
            const char ** argv )
```

All composition and synthesis code is defined in the main function.

There is no need for any of this code to be in a separate file.

References csound::Node::addChild(), csound::ScoreModel::addChild(), csound::LispNode::appendTopLevelForm(), csound::evaluate_form(), csound::initialize_ecl(), csound::Composition::processArgv(), csound::Composition::setAlbum(), csound::Composition::setAuthor(), csound::MusicModel::setCsoundOrchestra(), csound::Composition::setDuration(), csound::Composition::setPerformanceRightsOrganization(), csound::Composition::setTieOverlappingNotes(), csound::Composition::setTi and csound::Composition::setYear().

7.49 /Users/michaelgogins/csound-ac/CsoundAC/MCRM.cpp File Reference

```
#include <functional>
#include <random>
#include "CppSound.hpp"
#include "dkm.hpp"
#include "MCRM.hpp"
```

Namespaces

- namespace csound

  C S O U N D.

Functions

- static Event csound::mean_to_note (const std::array< double, KMeansMCRM::MEASURE_DIMENSIONS > &mean)

7.50 /Users/michaelgogins/csound-ac/CsoundAC/MCRM.hpp File Reference

```
#include "Platform.hpp"
#include "Silence.hpp"
#include <vector>
```

Data Structures

- class csound::KMeansMCRM

  Uses k-means clustering to translate the accumulated samples that approximate the measure on the iterated function system implemented by the multiple copy reducing machine algorithm into a specified number of notes.
- class csound::MCRM

Namespaces

- namespace csound

  C S O U N D.

7.51 /Users/michaelgogins/csound-ac/CsoundAC/Midifile.cpp File Reference

```
#include "Midifile.hpp"
#include "System.hpp"
#include <algorithm>
#include <fstream>
#include <iostream>
#include <map>
#include <sstream>
#include <vector>
#include <cstring>
```

Namespaces

- namespace csound

  C S O U N D.

Functions

- bool csound::operator< (const MidiEvent &a, const MidiEvent &b)

7.52 /Users/michaelgogins/csound-ac/CsoundAC/Midifile.hpp File Reference

```
#include "Platform.hpp"
#include <algorithm>
#include <utility>
#include <fstream>
#include <iostream>
#include <map>
#include <string>
#include <vector>
```

Data Structures

- class csound::Chunk
- class csound::MidiEvent

  This class is used to store ALL Midi messages.
- struct csound::MidiEventComparator
- class csound::MidiFile

  Reads and writes format 0 and format 1 standard MIDI files.
- class csound::MidiHeader
- class csound::MidiTrack
- class csound::TempoMap

Namespaces

- namespace csound
  C S O U N D.

Typedefs

- typedef unsigned char csound::csound_u_char

Variables

- class SILENCE_PUBLIC csound::MidiFile

7.53 /Users/michaelgogins/csound-ac/CsoundAC/MusicModel.cpp File Reference

```
#include "MusicModel.hpp"
#include "Exception.hpp"
#include "Composition.hpp"
#include "System.hpp"
#include <cstdio>
#include <cstdlib>
#include <fstream>
#include <set>
#include <sstream>
#include <stdint.h>
```

Namespaces

- namespace csound
  C S O U N D.

7.54 /Users/michaelgogins/csound-ac/CsoundAC/MusicModel.hpp File Reference

```
#include "Platform.hpp"
#include "ScoreModel.hpp"
#include "CppSound.hpp"
#include "Node.hpp"
#include "Score.hpp"
#include <stdint.h>
```

Data Structures

- class csound::MusicModel

  A ScoreModel that uses Csound to render generated scores, via the CppSound class.

Namespaces

- namespace csound

  C S O U N D.

7.55 /Users/michaelgogins/csound-ac/CsoundAC/Node.cpp File Reference

```
#include "Node.hpp"
#include <set>
```

Namespaces

- namespace csound

  C S O U N D.

7.56 /Users/michaelgogins/csound-ac/CsoundAC/Node.hpp File Reference

```
#include "Platform.hpp"
#include "Score.hpp"
#include <vector>
#include <Eigen/Dense>
#include <functional>
```

Data Structures

- class csound::Generator

  Node that uses any callable to implement Node::generate.
- class csound::Node

  Base class for all music graph nodes in the Silence system.
- class csound::RemoveDuplicates

  Removes all duplicate events produced by the child nodes of this.
- class csound::Transformer

  Node that uses any callable to implement Node::transform.

Namespaces

- namespace csound
  C S O U N D.

Typedefs

- typedef Node * csound::NodePtr

7.57 /Users/michaelgogins/csound-ac/CsoundAC/OrchestraNode.hpp File Reference

Data Structures

- class OrchestraNode

7.58 /Users/michaelgogins/csound-ac/CsoundAC/Platform.hpp File Reference

Namespaces

- namespace csound
  C S O U N D.

Macros

- #define SILENCE_PUBLIC

7.58.1 Macro Definition Documentation

7.58.1.1 SILENCE_PUBLIC

```
#define SILENCE_PUBLIC
```

7.59 /Users/michaelgogins/csound-ac/CsoundAC/Random.cpp File Reference

```
#include "Random.hpp"
```

Namespaces

- namespace csound
  C S O U N D.

7.60 /Users/michaelgogins/csound-ac/CsoundAC/Random.hpp File Reference

```
#include "Platform.hpp"
#include "Node.hpp"
#include <random>
#include <cmath>
#include <cstdint>
```

Data Structures

- class csound::Random
  A random value will be sampled from the specified distribution, translated and scaled as specified, and set in the specified row and column of the local coordinates.

Namespaces

- namespace csound
  C S O U N D.

7.61 /Users/michaelgogins/csound-ac/CsoundAC/Rescale.cpp File Reference

```
#include "Rescale.hpp"
```

Namespaces

- namespace csound
  C S O U N D.

7.62 /Users/michaelgogins/csound-ac/CsoundAC/Rescale.hpp File Reference

```
#include "Platform.hpp"
#include "ScoreNode.hpp"
```

Data Structures

- class csound::Rescale

  Rescales all child events to fit a bounding hypercube in music space.

Namespaces

- namespace csound

  C S O U N D.

7.63 /Users/michaelgogins/csound-ac/CsoundAC/Score.cpp File Reference

```
#include "Conversions.hpp"
#include "CppSound.hpp"
#include "Midifile.hpp"
#include "Score.hpp"
#include "System.hpp"
#include "Voicelead.hpp"
#include <algorithm>
#include <cfloat>
#include <cstdarg>
#include <iostream>
#include <fstream>
#include <set>
#include <sstream>
#include "allegro.h"
```

Data Structures

- struct csound::TimeAfterComparator
- struct csound::TimeAtComparator

Namespaces

- namespace csound

  C S O U N D.

Functions

- static std::vector< double > csound::matchContextSize (const std::vector< double > context, const std::vector< double > pcs)
- static double csound::max (double a, double b)
- static double csound::min (double a, double b)
- void SILENCE_PUBLIC csound::printChord (std::ostream &stream, std::string label, const std::vector< double > &chord)
- void SILENCE_PUBLIC csound::printChord (std::string label, const std::vector< double > &chord)

7.64 /Users/michaelgogins/csound-ac/CsoundAC/Score.hpp File Reference

```
#include "Platform.hpp"
#include "Event.hpp"
#include "Midifile.hpp"
#include <iostream>
#include <vector>
```

Data Structures

- class csound::Score

  Base class for collections of events in music space.

Namespaces

- namespace csound

  C S O U N D.

7.65 /Users/michaelgogins/csound-ac/CsoundAC/ScoreModel.cpp File Reference

```
#include "ScoreModel.hpp"
#include "Exception.hpp"
#include "Composition.hpp"
#include "System.hpp"
```

Namespaces

- namespace csound

  C S O U N D.

7.66 /Users/michaelgogins/csound-ac/CsoundAC/ScoreModel.hpp File Reference

```
#include "Platform.hpp"
#include "Composition.hpp"
#include "Node.hpp"
#include "Score.hpp"
#include <stdint.h>
```

Data Structures

- class csound::ScoreModel

  Base class for compositions that use the principle of a music graph to generate a score.

Namespaces

- namespace csound

  C S O U N D.

7.67 /Users/michaelgogins/csound-ac/CsoundAC/ScoreNode.cpp File Reference

```
#include "ScoreNode.hpp"
```

Namespaces

- namespace csound

  C S O U N D.

7.68 /Users/michaelgogins/csound-ac/CsoundAC/ScoreNode.hpp File Reference

```
#include "Platform.hpp"
#include "Node.hpp"
#include "Score.hpp"
#include <Eigen/Dense>
```

Data Structures

- class csound::ScoreNode

  Node class that produces events from the contained score, which can be built up programmatically or imported from a standard MIDI file.

Namespaces

- namespace csound

  C S O U N D.

7.69 /Users/michaelgogins/csound-ac/CsoundAC/Sequence.cpp File Reference

```
#include "CppSound.hpp"
#include "Sequence.hpp"
#include "System.hpp"
```

Namespaces

- namespace csound

  C S O U N D.

7.70 /Users/michaelgogins/csound-ac/CsoundAC/Sequence.hpp File Reference

```
#include "Platform.hpp"
#include "ScoreNode.hpp"
```

Data Structures

- class csound::Sequence

  Node that creates a temporal sequence of child nodes.

Namespaces

- namespace csound

  C S O U N D.

7.71 /Users/michaelgogins/csound-ac/CsoundAC/Shell.cpp File Reference

```
#include "CppSound.hpp"
#include "Shell.hpp"
#include "System.hpp"
#include <iostream>
#include <fstream>
#include <ctime>
#include "csdl.h"
```

Namespaces

- namespace csound

  C S O U N D.

Functions

- static bool csound::pythonFuncWarning (void **pythonLibrary, const char *funcName)

Variables

- void(* csound::Py_Finalize_)(void)=0
- void(* csound::Py_Initialize_)(void)=0
- void(* csound::PyErr_Print_)(void)=0
- PyObject_ *(* csound::PyImport_ImportModule_)(char *)=0
- long(* csound::PyLong_AsLong_)(PyObject_ *)=0
- PyObject_ *(* csound::PyObject_CallMethod_)(PyObject_ *, char *, char *,...)=0
- PyObject_ *(* csound::PyObject_GetAttrString_)(PyObject_ *, char *)=0
- int(* csound::PyRun_SimpleFileEx_)(FILE *, const char *, int)=0
- int(* csound::PyRun_SimpleString_)(const char *)=0
- void(* csound::PySys_SetArgv_)(int, char **)=0

7.72 /Users/michaelgogins/csound-ac/CsoundAC/Shell.hpp File Reference

```
#include "Platform.hpp"
#include <string>
```

Data Structures

- class csound::Shell

  Provide a shell in which Python scripts can be loaded, saved, and executed.

Namespaces

- namespace csound

  C S O U N D.

Typedefs

- typedef void csound::PyObject_

7.73 /Users/michaelgogins/csound-ac/CsoundAC/Silence.hpp File Reference

```
#include "Platform.hpp"
#include <string>
#include <vector>
#include <map>
#include <Eigen/Dense>
#include "System.hpp"
#include "Conversions.hpp"
#include "Event.hpp"
#include "Midifile.hpp"
#include "Score.hpp"
#include "ChordSpaceBase.hpp"
#include "ChordSpace.hpp"
#include "Composition.hpp"
#include "Node.hpp"
#include "Counterpoint.hpp"
#include "CounterpointNode.hpp"
#include "ScoreNode.hpp"
#include "Cell.hpp"
#include "HarmonyIFS.hpp"
#include "HarmonyIFS2.hpp"
#include "Rescale.hpp"
#include "ScoreModel.hpp"
#include "MusicModel.hpp"
#include "Sequence.hpp"
#include "Random.hpp"
#include "ImageToScore.hpp"
#include "Soundfile.hpp"
#include "StrangeAttractor.hpp"
#include "Lindenmayer.hpp"
#include "MCRM.hpp"
#include "Voicelead.hpp"
#include "VoiceleadingNode.hpp"
#include "ChordLindenmayer.hpp"
#include "ExternalNode.hpp"
```

7.74 /Users/michaelgogins/csound-ac/CsoundAC/silencio.hpp File Reference

```
#include <Eigen/Core>
#include <vector>
```

7.75 /Users/michaelgogins/csound-ac/CsoundAC/Soundfile.cpp File Reference

```
#include "Soundfile.hpp"
#include "Conversions.hpp"
```

Namespaces

- namespace csound

  C S O U N D.

7.76 /Users/michaelgogins/csound-ac/CsoundAC/Soundfile.hpp File Reference

```
#include "Platform.hpp"
#include <sndfile.h>
#include <iostream>
#include <string>
#include <vector>
#include <cstring>
#include <complex>
#include <Eigen/Dense>
```

Data Structures

- class csound::Soundfile

  Simple, basic read/write access, in sample frames, to PCM soundfiles.

Namespaces

- namespace csound

  C S O U N D.

7.77 /Users/michaelgogins/csound-ac/CsoundAC/StrangeAttractor.cpp File Reference

```
#include "CppSound.hpp"
#include "StrangeAttractor.hpp"
#include "Conversions.hpp"
#include "Random.hpp"
#include "System.hpp"
#include <cmath>
```

Namespaces

- namespace csound
  C S O U N D.

7.78 /Users/michaelgogins/csound-ac/CsoundAC/StrangeAttractor.hpp File Reference

```
#include "Platform.hpp"
#include "Silence.hpp"
#include <string>
#include <vector>
#include <random>
#include <Eigen/Dense>
```

Data Structures

- class csound::StrangeAttractor
  Generates notes by searching for a chaotic dynamical system defined by a polynomial equation or partial differential equation using Julien C.

Namespaces

- namespace csound
  C S O U N D.

7.79 /Users/michaelgogins/csound-ac/CsoundAC/System.cpp File Reference

```
#include "System.hpp"
```

Namespaces

- namespace csound
  C S O U N D.

Functions

- SILENCE_PUBLIC FILE *& csound::log_file ()
- MessageCallbackType & csound::message_callback ()
- SILENCE_PUBLIC int csound::message_level (int verbosity)
- SILENCE_PUBLIC void *& csound::user_data ()

7.80 /Users/michaelgogins/csound-ac/CsoundAC/System.hpp File Reference

```
#include "Platform.hpp"
#include "CppSound.hpp"
#include <string.h>
#include <string>
#include <vector>
#include <cstdarg>
#include <ctime>
```

Data Structures

- class csound::Logger
- class csound::System

  Abstraction layer for a minimal set of system services.
- class csound::ThreadLock

  Encapsulates a thread monitor, such as a Windows event handle.

Namespaces

- namespace csound

  C S O U N D.

Typedefs

- typedef void(∗ csound::MessageCallbackType) (CSOUND ∗csound, int attribute, const char ∗format, va_list marker)

Functions

- SILENCE_PUBLIC FILE ∗& csound::log_file ()
- MessageCallbackType & csound::message_callback ()
- SILENCE_PUBLIC int csound::message_level (int verbosity)
- SILENCE_PUBLIC void ∗& csound::user_data ()

7.81 /Users/michaelgogins/csound-ac/CsoundAC/trace.cpp File Reference

```
#include "stdarg.h"
#include "stdio.h"
```

Functions

- void trace (char *format,...)

7.81.1 Function Documentation

7.81.1.1 trace()

```
void trace (
             char * format,
              ...  )
```

7.82 /Users/michaelgogins/csound-ac/CsoundAC/version.h File Reference

Macros

- #define CS_APISUBVER
- #define CS_APIVERSION
- #define CS_PACKAGE_DATE __DATE__
- #define CS_PACKAGE_NAME "Csound"
- #define CS_PACKAGE_STRING "Csound " VERSION
- #define CS_PACKAGE_TARNAME "csound"
- #define CS_PACKAGE_VERSION VERSION
- #define CS_PATCHLEVEL (0)
- #define CS_SUBVER (18)
- #define CS_VERSION (6)
- #define VERSION "6.18"

7.82.1 Macro Definition Documentation

7.82.1.1 CS_APISUBVER

```
#define CS_APISUBVER
```

Value:

```
0   /* for minor changes that will still allow
compatiblity with older hosts */
```

7.82.1.2 CS_APIVERSION

```
#define CS_APIVERSION
```

Value:

```
4   /* should be increased anytime a new version
contains changes that an older host will
not be able to handle -- most likely this
will be a change to an API function or
the CSOUND struct */
```

7.82.1.3 CS_PACKAGE_DATE

```
#define CS_PACKAGE_DATE __DATE__
```

7.82.1.4 CS_PACKAGE_NAME

```
#define CS_PACKAGE_NAME "Csound"
```

7.82.1.5 CS_PACKAGE_STRING

```
#define CS_PACKAGE_STRING "Csound " VERSION
```

7.82.1.6 CS_PACKAGE_TARNAME

```
#define CS_PACKAGE_TARNAME "csound"
```

7.82.1.7 CS_PACKAGE_VERSION

```
#define CS_PACKAGE_VERSION VERSION
```

7.82.1.8 CS_PATCHLEVEL

```
#define CS_PATCHLEVEL (0)
```

7.82.1.9 CS_SUBVER

```
#define CS_SUBVER (18)
```

7.82.1.10 CS_VERSION

```
#define CS_VERSION (6)
```

7.82.1.11 VERSION

```
#define VERSION "6.18"
```

7.83 /Users/michaelgogins/csound-ac/CsoundAC/Voicelead.cpp File Reference

```
#include "System.hpp"
#include "Voicelead.hpp"
#include <algorithm>
#include <cmath>
#include <ctime>
#include <iostream>
#include <map>
#include <vector>
#include <set>
```

Data Structures

- struct csound::AscendingDistanceComparator
- struct csound::MatrixCell

Namespaces

- namespace csound

  C S O U N D.

Functions

- std::vector< std::vector< MatrixCell > > csound::createMatrix (const std::vector< double > &sourceMultiset_, const std::vector< double > &targetMultiset_, const std::vector< double > &sourceChord_)
- void csound::inversions (const std::vector< double > &original, const std::vector< double > &iterator, size_t voice, double maximum, std::set< std::vector< double > > &chords, size_t divisionsPerOctave)
- const MatrixCell & csound::minimumCell (const MatrixCell &a, const MatrixCell &b, const MatrixCell &c)
- std::ostream & operator<< (std::ostream &stream, const std::vector< double > &chord)
- std::vector< std::vector< double > > csound::pitchRotations (const std::vector< double > &chord)
- void csound::recursiveVoicelead_ (const std::vector< double > &source, const std::vector< double > &original, const std::vector< double > &iterator, std::vector< double > &target, size_t voice, double maximum, bool avoid↩ Parallels, size_t divisionsPerOctave)
- double csound::round (double x)
- std::vector< double > csound::sort (const std::vector< double > &chord)

Variables

- std::map< size_t, std::map< double, double > > csound::cForPForDivisionsPerOctave
- static int csound::debug = 1
- std::map< size_t, std::map< double, double > > csound::pForCForDivisionsPerOctave
- std::map< size_t, std::map< std::vector< double >, double > > csound::pForPrimeChordsForDivisionsPerOctave
- std::map< size_t, std::vector< std::vector< double > > > csound::primeChordsForDivisionsPerOctave

7.83.1 Function Documentation

7.83.1.1 operator<<()

```
std::ostream & operator<< (
            std::ostream & stream,
            const std::vector< double > & chord )
```

7.84 /Users/michaelgogins/csound-ac/CsoundAC/Voicelead.hpp File Reference

```
#include "Platform.hpp"
#include "Event.hpp"
#include <vector>
```

Data Structures

- class csound::Voicelead

  This class contains facilities for voiceleading, harmonic progression, and identifying chord types.

Namespaces

- namespace csound

  C S O U N D.

7.85 /Users/michaelgogins/csound-ac/CsoundAC/VoiceleadingNode.cpp File Reference

```
#include "VoiceleadingNode.hpp"
#include "Conversions.hpp"
#include "Voicelead.hpp"
#include "System.hpp"
#include <cmath>
#include <cfloat>
#include <sstream>
```

Namespaces

- namespace csound

  C S O U N D.

Functions

- std::ostream & csound::operator<< (std::ostream &stream, const VoiceleadingOperation &operation)
- void SILENCE_PUBLIC csound::printChord (std::ostream &stream, std::string label, const std::vector< double > &chord)
- void SILENCE_PUBLIC csound::printChord (std::string label, const std::vector< double > &chord)

7.86 /Users/michaelgogins/csound-ac/CsoundAC/VoiceleadingNode.hpp File Reference

```
#include "Platform.hpp"
#include "Voicelead.hpp"
#include "Node.hpp"
#include "Score.hpp"
#include "ChordSpace.hpp"
```

Data Structures

- class csound::VoiceleadingNode

  This node class imposes a sequence of one or more "voice-leading" operations upon the pitches of notes produced by children of this node, within a segment of the notes.
- class csound::VoiceleadingOperation

  Utility class for storing voice-leading operations within a VoiceleadNode for future application.

Namespaces

- namespace csound

  C S O U N D.

Functions

- std::ostream & csound::operator<< (std::ostream &stream, const VoiceleadingOperation &operation)

7.87 /Users/michaelgogins/csound-ac/README.md File Reference