GUIDOLib

v.1.54

Generated by Doxygen 1.7.2

Thu Oct 23 2014 11:56:58

Contents

Chapter 1

The GUIDO Engine Library

1.1 Introduction

The GUIDOLib project aims at the development of a generic, portable library and API for the graphical rendering of musical scores. The library is based on the GUIDO Music Notation format as the underlying data format. It is an open source project covered by the Mozilla Public License.

The project has started in December 2002, based on the source code of the GUIDO NoteViewer developed by Kai Renz.

1.2 The GUIDO Music Notation

The GUIDO Music Notation format (GMN) is a general purpose formal language for representing score level music in a platform independent plain text and human readable way. It is based on a conceptually simple but powerful formalism: its design concentrates on general musical concepts (as opposed to graphical features). A key feature of the GUIDO design is adequacy which means that simple musical concepts should be represented in a simple way and only complex notions should require complex representations. This design is reflected by three specification levels: the basic, advanced and extended GUIDO specifications.

1.3 The GUIDO Engine

The GUIDO Engine operates on a memory representation of the GMN format: the GUIDO Abstract Representation (GAR). This representation is transformed step by step to produce graphical score pages. Two kinds of processing are first applied to the GAR:

- GAR to GAR transformations which represents a logical layout transformation: part of the layout (such as beaming for example) may be computed from the GAR as well as expressed in GAR,

- the GAR is converted into a GUIDO Semantic Normal Form (GSNF). The GSNF is a canonical form such that different semantically equivalent expressions have the same GSNF.

This GSNF is finally converted into a GUIDO Graphic Representation (GGR) that contains the necessary layout information and is directly used to draw the music score. This final step includes notably spacing and page breaking algorithms.

Note that although the GMN format allows for precise music formatting (in advanced GUIDO), the GUIDO Engine provides powerful automatic layout capabilities.

1.4 Main library services

The main services provided by the library are:

- Score layout: the library provides functions to parse a GMN file and to create the corresponding GAR and GGR.
- Score pages access: result of the score layout is a set of pages. The library provides the necessary to change the page size, to query a score pages count, the current page number or the page number corresponding to a given music date.
- The GUIDO Factory: the GUIDO Engine may be feeded with computer generated music using the GUIDO Factory. The GUIDO Factory API provides a set of functions to create a GAR from scratch and to convert it into a GGR.
- Mapping: along with the GGR, the GUIDO Engine maintains a tree of graphical music elements. Each element has a bounding box and a date. The library includes the necessary to browse this tree and to retrieve elements by date or position.

1.5 Graphic devices

First version of the GUIDO Engine was Windows dependent: a Windows HDC type (graphic device context handle) was provided to the GGR objects, that were directly calling Microsoft Windows APIs to draw graphics and text.

For the GUIDO Engine to be platform-independent, and to avoid being restricted to one graphical technology (even a cross-platform one such as pdf, eps or OpenGL ...), the choice has been made to provide a C++ object (called `VGDevice (`p. **??**`)`) to the GGR objects, in place of the previous windows HDC handle.

`VGDevice (`p. **??**`)` is a C++ pure virtual class that declares all methods required by the GGR objects to communicate their graphical operations. Implementations of `VGDevices` are provided by clients applications using derived classes so that neither GGR objects nor any part of the GUIDO Engine depends on a particular graphical implementation.

The main advantage is that `VGDevice (p. ??)` derived classes can implement any kind of graphic output: on-screen (platform specific, OpenGL), off-screen (raw bitmaps), files (pdf, svg, postscript), network streams...

`VGDevices` derived classes must provide standard graphic functions (Lines, Arcs, Boxes,Polygons, Text), coordinate transformations (zoom / scaling), and symbolic music symbols handlers (DrawSymbol method). `VGDevice (p. ??)` design makes a clear distinction between text characters and music symbols (although music symbols are generally glyphs in a music font). Music symbols are identified by font-independent constants. This mechanism provides `VGDevice (p. ??)` objects with a higher abstraction level than a pure graphic layer.

Existing implementations of VGDevices:

```
GDeviceOSX:             MacOS X Quartz implementation.
GDeviceWin32:           Windows (gdi)
GDeviceWin32GDIPlus:    Windows (gdiplus)
CairoDevice:            Linux   (cairo)
GDeviceQt:              platform independent - Qt based device
```

VGDevice (p. **??**) implementations not maintained:

```
GDeviceWin2000:         Windows 2000 / XP(gdi+)
GDeviceGL:              OpenGL implementation
GDeviceGTK:             Linux GTK implementation
GDevicePostScript:      EPS files (encapsulated postscript)
GDeviceWx:              wxWindows DC implementation.
```

Chapter 2

Sample code

This section provides examples of the GUIDO library use.

2.1 Displaying a score

```
#include <iostream>

#include "GUIDOEngine.h"
#include "GDevicePostScript.h"

using namespace std;
// ---------------------------------------------------------------------------
//  This example of code is intended to show the minimum steps to read a
//  GMN file and to display it.
//  Take care that for simplicity, this example doesn't check return codes
// ---------------------------------------------------------------------------
void DrawGMNFile (const char * filename)
{
    ARHandler arh;
    // use a GDevicePostScript with a 72 dpi resolution
    GDevicePostScript dev ((int)(21*72/2.54), (int)(29.7*72/2.54), "outfile.eps",
       "", "");

    // declare a data structure for engine initialisation
    // we'll make use of the default graphic device (embedded in the library)
    // Guido font is guido2 and text font is times
    GuidoInitDesc gd = { &dev, 0, "guido2", "Times" };
    GuidoOnDrawDesc desc;              // declare a data structure for drawing

    GuidoInit (&gd);                   // Initialise the Guido Engine first

    // and parse the GMN file to get a GGR handle directly stored in the drawing
      struct
    GuidoParseFile (filename, &arh);
    GuidoAR2GR (arh, 0, &desc.handle);
    desc.hdc = &dev;                    // we'll draw on the postscript device
    desc.page = 1;                      // draw the first page only
    desc.updateRegion.erase = true;     // and draw everything
    desc.scrollx = desc.scrolly = 0;    // from the upper left page corner
    desc.sizex = desc.sizey = 500;      // size of the drawing region
    GuidoOnDraw (&desc);
}

int main(int argc, char **argv)
{
    const char * file = argv[1];
    DrawGMNFile (file);
    return 0;
}
```

2.2 Building music using the GUIDO Factory

```
#include "GUIDOFactory.h"

// ---------------------------------------------------------------------------
//  Guido Factory example of use
//  Take care that for simplicity, this example doesn’t check return codes
// ---------------------------------------------------------------------------
void TestGuidoFactory()
{
    ARHandler ar; GRHandler gr;
    ARFactoryHandler *f;

    GuidoFactoryOpen(&f);               // open the GUIDO Factory first

    GuidoFactoryOpenMusic(f);           // and create a new score
    GuidoFactoryOpenVoice(f);           // open a new voice

    GuidoFactoryOpenEvent(f, "c" );     // open a new note
    GuidoFactoryCloseEvent(f);          // close the note which is now part of th
      e opened voice

    GuidoFactoryOpenEvent(f, "e" );     // open another new note
    GuidoFactoryAddSharp(f);            // add a sharp to the opened note
    GuidoFactoryCloseEvent(f);          // and close the note which is now part o
      f the opened voice

    GuidoFactoryCloseVoice(f);          // close the voice which is now part of t
      he opened score


    GuidoFactoryOpenVoice(f);           // open a second voice
    GuidoFactoryOpenEvent(f, "a" );     // open a new voice
    GuidoFactoryAddFlat(f);             // add a flat to the opened note
    GuidoFactoryCloseEvent(f);          // and close the note which is now part o
      f the second voice

    GuidoFactoryOpenEvent(f, "d" );     // open another new note
    GuidoFactoryCloseEvent(f);          // and close the note which is now part o
      f the second voice

    GuidoFactoryCloseVoice(f);          // close the voice which is now part of t
      he opened score

    // - Extract AR and GR
    ar = GuidoFactoryCloseMusic(f);     // extract the GAR handle
    GuidoAR2GR( ar, 0, &gr );           // and the GGR handle
    GuidoFactoryClose(f);               // close the GUIDO Factory which won’t be
       used any more
    if( gr )                            // check for the handle validity
    {
        // ... Do something with the GGR handle (i.e: draw)
    }
}
```

Chapter 3

Bug List

Member GuidoFindEventPage (p. **??**)**(CGRHandler inHandleGR, const GuidoDate** (p. **??**) **&date)** returns
page + 1 when input date falls on the last system.

Member GuidoFindPageAt (p. **??**)**(CGRHandler inHandleGR, const GuidoDate** (p. **??**) **&date)** returns
page + 1 when input date falls on the last system.

Chapter 4

Module Documentation

4.1 MIDI support

Classes

- struct **Guido2MidiParams**

  The **GuidoInitDesc** *(*p. **??***) data structure contains all information required by* **GuidoInit()** *(*p. **??***)*

Typedefs

- typedef struct **Guido2MidiParams** **Guido2MidiParams**

  The **GuidoInitDesc** *(*p. **??***) data structure contains all information required by* **GuidoInit()** *(*p. **??***)*

Functions

- **GuidoErrCode** **GuidoAR2MIDIFile** (const **ARHandler** ar, const char ∗filename, const **Guido2MidiParams** ∗params)

  Export to a MIDI file.

4.1.1 Typedef Documentation

4.1.1.1 typedef struct Guido2MidiParams Guido2MidiParams

The **GuidoInitDesc** (p. **??**) data structure contains all information required by **GuidoInit()** (p. **??**)

4.1.2 Function Documentation

4.1.2.1 GuidoErrCode GuidoAR2MIDIFile (const ARHandler *ar,* const char ∗ *filename,* const Guido2MidiParams ∗ *params*)

Export to a MIDI file.

Parameters

ar,:	Guido AR handler
filename,:	the output file name
params,:	conversions parameters.

Returns

a Guido error code.

4.2 Error codes

Enumerations

- enum **GuidoErrCode** {
  guidoNoErr = 0, **guidoErrParse** = -1, **guidoErrMemory** = -2, **guidoErrFileAccess** = -3,
  guidoErrUserCancel = -4, **guidoErrNoMusicFont** = -5, **guidoErrNoTextFont** = -6, **guidoEr-rBadParameter** = -7,
  guidoErrInvalidHandle = -8, **guidoErrNotInitialized** = -9, **guidoErrActionFailed** = -10 }

  The guido error codes list.

4.2.1 Enumeration Type Documentation

4.2.1.1 enum GuidoErrCode

The guido error codes list.

This is the list of possible error codes returned by GuidoEngine and GuidoFactory APIs

Enumerator:

guidoNoErr	null is used to denote no error
guidoErrParse	error while parsing the Guido format
guidoErrMemory	memory allocation error
guidoErrFileAccess	error while reading or writing a file
guidoErrUserCancel	the user cancelled the action
guidoErrNoMusicFont	the music font is not available

guidoErrNoTextFont the text font is not available

guidoErrBadParameter bad parameter used as argument

guidoErrInvalidHandle invalid handler used

guidoErrNotInitialized required initialisation has not been performed

guidoErrActionFailed the action failed

4.3 Building abstract and graphic representations

Functions

- **GuidoErrCode GuidoInit** (**GuidoInitDesc** ∗desc)
- void **GuidoShutdown** ()
- **GuidoErrCode GuidoParseFile** (const char ∗filename, **ARHandler** ∗ar)
- **GuidoErrCode GuidoParseString** (const char ∗str, **ARHandler** ∗ar)
- **GuidoErrCode GuidoAR2GR** (**ARHandler** ar, const **GuidoLayoutSettings** ∗settings, **GRHandler** ∗gr)
- **GuidoErrCode GuidoAR2RProportional** (**ARHandler** ar, int width, int height, const **GuidoDate** &start, const **GuidoDate** &end, bool drawdur, **VGDevice** ∗dev)
- **GuidoErrCode GuidoUpdateGR** (**GRHandler** gr, const **GuidoLayoutSettings** ∗settings)
- void **GuidoFreeAR** (**ARHandler** ar)
- void **GuidoFreeGR** (**GRHandler** gr)
- const char ∗ **GuidoGetErrorString** (**GuidoErrCode** errCode)
- int **GuidoGetParseErrorLine** ()
- void **GuidoGetDefaultLayoutSettings** (**GuidoLayoutSettings** ∗settings)

4.3.1 Detailed Description

The Guido Engine operates on two kinds of music representation:

- an abstract representation that corresponds to the Guido Music Notation elements. An important part of the layout - the logical layout, that decides on the stems direction for example - is applied to the abstract representation.

- a graphic representation that corresponds to a graphic instantiation of an abstract representation. Algorithms such as spacing and line breaking are applied to the graphic representation.

The functions described in this section are intended to build abstract and graphic representations.

4.3.2 Function Documentation

4.3.2.1 GuidoErrCode GuidoInit (GuidoInitDesc ∗ *desc*)

Initialises the Guido Engine. Must be called before any attempt to read a Guido file or to use the Guido Factory

Parameters

desc	the graphic environment description.

Returns

a Guido error code.

WARNING: the caller must ensure desc maintains a constant reference on a valid **VGDevice** (p. **??**), because Guido keeps it internally (to calculate fonts, etc.)

4.3.2.2 void GuidoShutdown ( )

Guido Engine shutdown

Actually release the font allocated by the engine. Anyway, the fonts are release when the client application exit but the function provides control over the time of the release.

4.3.2.3 GuidoErrCode GuidoParseFile (const char ∗ *filename,* ARHandler ∗ *ar*)

Parses a Guido Music Notation (.gmn) file and builds the corresponding abstract representation.

Parameters

filename	the file to parse.
ar	on output: a Guido opaque handle to an abstract music representation. It's the caller responsability to free the handle using GuidoFreeAR.

Returns

a Guido error code.

4.3.2.4 GuidoErrCode GuidoParseString (const char ∗ *str,* ARHandler ∗ *ar*)

Parses a buffer and builds the corresponding abstract representation. The buffer if expected to contain gmn code.

Parameters

str	the null terminated buffer to parse.
ar	on output: a Guido opaque handle to an abstract music representation. It's the caller responsability to free the handle using GuidoFreeAR.

Returns

a Guido error code.

4.3.2.5 GuidoErrCode GuidoAR2GR (ARHandler *ar,* const GuidoLayoutSettings ∗ *settings,* GRHandler ∗ *gr*)

Transforms a Guido abstract representation into a Guido graphic representation. The engine applies layout algorithms according to the settings given as argument.

Note

You can safely free the AR after the transformation.

Parameters

ar	the handler to the abstract representation.
settings	a pointer to the settings for the graphic layout. If null, default settings are applied.
gr	on output: a Guido opaque handle to a graphic music representation It's the caller responsability to free the handle using GuidoFreeGR.

Returns

a Guido error code.

4.3.2.6 GuidoErrCode GuidoAR2RProportional (ARHandler *ar,* int *width,* int *height,* const GuidoDate & *start,* const GuidoDate & *end,* bool *drawdur,* VGDevice ∗ *dev*)

Transforms a Guido abstract representation into a simplified proportional representation.

Parameters

ar	the handler to the abstract representation.
width	the drawing area width.
height	the drawing area height.
start	start date of the time zone to be displayed.
end	end date of the time zone to be displayed, when 0, the score duration is used.
drawdur	control duration lines drawing.
dev	a graphic device.

Returns

a Guido error code.

4.3.2.7 GuidoErrCode GuidoUpdateGR (GRHandler *gr,* const GuidoLayoutSettings ∗ *settings*)

Transforms a MIDI file into a simplified proportional representation.

Parameters

midifile	the MIDI file name
width	the drawing area width.
height	the drawing area height.
start	start date of the time zone to be displayed.
end	end date of the time zone to be displayed, when 0, the score duration is used.
drawdur	control duration lines drawing.
dev	a graphic device.

Returns

a Guido error code.

Transforms a MIDI file into a piano roll representation.

Parameters

midifile	the MIDI file name
width	the drawing area width.
height	the drawing area height.
start	start date of the time zone to be displayed.
end	end date of the time zone to be displayed, when 0, the score duration is used.
dev	a graphic device.

Returns

a Guido error code.

Applies new layout settings to an existing Guido graphic representation.

Parameters

gr	the handler to the graphic representation.
settings	a pointer to the settings for the graphic layout. If null, default settings are applied.

Returns

a Guido error code.

4.3.2.8 void GuidoFreeAR (ARHandler *ar*)

Releases a Guido abstract representation.

Parameters

ar	the handler to the abstract representation.

4.3.2.9 void GuidoFreeGR (GRHandler *gr*)

Releases a Guido graphic representation.

Parameters

gr	the handler to the graphic representation.

4.3.2.10 const char* GuidoGetErrorString (GuidoErrCode *errCode*)

Gives a textual description of a Guido error code.

Parameters

errCode	a Guido error code.

Returns

a string describing the error.

4.3.2.11 int GuidoGetParseErrorLine ( )

Gives the line of a Guido script where the last parse error has occured.

Returns

a line number.

4.3.2.12 void GuidoGetDefaultLayoutSettings (GuidoLayoutSettings * *settings*)

Gives the default values of the layout settings.

Parameters

settings	on output, a pointer to the settings to be filled with default values.

4.4 Browsing music pages

Functions

- int **GuidoCountVoices** (**CARHandler** inHandleAR)

Gives the number of score pages of the graphic representation.

- int **GuidoGetPageCount** (**CGRHandler** inHandleGR)

    Gives the number of score pages of the graphic representation.

- int **GuidoGetSystemCount** (**CGRHandler** inHandleGR, int page)

    Gives the number of systems on a given page.

- **GuidoErrCode GuidoDuration** (**CGRHandler** inHandleGR, **GuidoDate** ∗date)

    Returns the music duration of a score.

- int **GuidoFindEventPage** (**CGRHandler** inHandleGR, const **GuidoDate** &date)

    Finds the page which has an event (note or rest) at a given date.

- int **GuidoFindPageAt** (**CGRHandler** inHandleGR, const **GuidoDate** &date)

    Finds the page which contain a given date.

- **GuidoErrCode GuidoGetPageDate** (**CGRHandler** inHandleGR, int pageNum, **GuidoDate** ∗date)

    Gives the time location of a Page.

4.4.1 Detailed Description

The Guido Engine produces pages of music and therefore, the graphic representation consists in a collection of pages. The following functions are intended to access these pages by page number as well as by date. Page numbers start at 1.

4.4.2 Function Documentation

4.4.2.1 int GuidoCountVoices (CARHandler *inHandleAR*)

Gives the number of score pages of the graphic representation.

Parameters

inHandleAR	a Guido opaque handle to a AR structure.

Returns

the number of voices or a guido error code.

4.4.2.2 int GuidoGetPageCount (CGRHandler *inHandleGR*)

Gives the number of score pages of the graphic representation.

Parameters

inHandleGR	a Guido opaque handle to a GR structure.

Returns

a number of pages or a guido error code.

4.4.2.3 int GuidoGetSystemCount (CGRHandler *inHandleGR,* int *page*)

Gives the number of systems on a given page.

Parameters

inHandleGR	a Guido opaque handle to a GR structure.
page	a page number (starts at 1).

Returns

the systems count on the given page or a guido error code.

4.4.2.4 GuidoErrCode GuidoDuration (CGRHandler *inHandleGR,* GuidoDate ∗ *date*)

Returns the music duration of a score.

The duration is expressed as a fractional value where 1 represents a whole note.

Parameters

inHandleGR	a Guido opaque handle to a GR structure.
date	on output: the duration expressed as a fractional value

Returns

a Guido error code.

4.4.2.5 int GuidoFindEventPage (CGRHandler *inHandleGR,* const GuidoDate & *date*)

Finds the page which has an event (note or rest) at a given date.

Bug

returns page + 1 when input date falls on the last system.

Parameters

inHandleGR	a Guido opaque handle to a GR structure.
date	the target date.

Returns

a page number if greater than 0, 0 if no page found,

4.4.2.6 int GuidoFindPageAt (CGRHandler *inHandleGR,* const GuidoDate & *date*)

Finds the page which contain a given date.

Bug

returns page + 1 when input date falls on the last system.

Parameters

inHandleGR	a Guido opaque handle to a GR structure.
date	the target date.

Returns

a page number if greater than 0, 0 if no page found,

4.4.2.7 GuidoErrCode GuidoGetPageDate (CGRHandler *inHandleGR,* int *pageNum,* GuidoDate ∗ *date*)

Gives the time location of a Page.

Parameters

inHandleGR	a Guido opaque handle to a GR structure.
pageNum	a page number (starts at 1).
date	on output: the page date if the page number is valid

Returns

a Guido error code.

4.5 Score drawing and pages formating

Functions

- **GuidoErrCode GuidoOnDraw** (**GuidoOnDrawDesc** ∗desc)

  Draws one page of score into a graphic device.

- **GuidoErrCode GuidoSVGExport** (const **GRHandler** handle, int page, std::ostream &out, const char ∗fontfile)

  Exports one page of score to SVG.

- **GuidoErrCode GuidoAbstractExport** (const **GRHandler** handle, int page, std::ostream &out)

  Exports an abstract representation of GUIDO draw commands.

- **GuidoErrCode GuidoBinaryExport** (const **GRHandler** handle, int page, std::ostream &out)

  Exports an representation of GUIDO draw commands in a data-reduced dsl.

- void **GuidoDrawBoundingBoxes** (int bbMap)

  Control bounding boxes drawing.

- int **GuidoGetDrawBoundingBoxes** ()

  Gives bounding boxes drawing state.

- void **GuidoGetPageFormat** (**CGRHandler** inHandleGR, int pageNum, **GuidoPageFormat** ∗format)

  Gives a score page format.

- void **GuidoSetDefaultPageFormat** (const **GuidoPageFormat** ∗format)

  Sets the default score page format.

- void **GuidoGetDefaultPageFormat** (**GuidoPageFormat** ∗format)

  Gives the default score page format.

- float **GuidoUnit2CM** (float val)

  Converts internal Guido units into centimeters.

- float **GuidoCM2Unit** (float val)

  Converts centimeters into internal Guido units.

- float **GuidoUnit2Inches** (float val)

  Converts internal Guido units into inches.

- float **GuidoInches2Unit** (float val)

  Converts inches into internal Guido units.

- **GuidoErrCode GuidoResizePageToMusic** (**GRHandler** inHandleGR)

  Resize the page sizes to the music size.

4.5.1 Detailed Description

The GuidoEngine makes use of internal units for graphic operations. The functions that query or set graphic dimensions always makes use of this internal unit. Conversion functions are provided to convert to standard units.

4.5.2 Function Documentation

4.5.2.1 GuidoErrCode GuidoOnDraw (GuidoOnDrawDesc ∗ *desc*)

Draws one page of score into a graphic device.

Parameters

desc	informations about what to draw and how to draw.

Returns

a Guido error code

4.5.2.2 GuidoErrCode GuidoSVGExport (const GRHandler *handle,* int *page,* std::ostream & *out,* const char ∗ *fontfile*)

Exports one page of score to SVG.

Parameters

page	the page number.
out	the output stream.
fontfile	path of the guido svg font file.

Returns

a Guido error code

4.5.2.3 GuidoErrCode GuidoAbstractExport (const GRHandler *handle,* int *page,* std::ostream & *out*)

Exports an abstract representation of GUIDO draw commands.

Parameters

out	the output stream.

Returns

a Guido error code

4.5.2.4 GuidoErrCode GuidoBinaryExport (const GRHandler *handle,* int *page,* std::ostream & *out*)

Exports an representation of GUIDO draw commands in a data-reduced dsl.

Parameters

out	the output stream.

Returns

a Guido error code

4.5.2.5 void GuidoDrawBoundingBoxes (int *bbMap*)

Control bounding boxes drawing.

Parameters

bbMap	a bits field indicating the set of bounding boxes to draw (default to none).

4.5.2.6 int GuidoGetDrawBoundingBoxes ( )

Gives bounding boxes drawing state.

4.5.2.7 void GuidoGetPageFormat (CGRHandler *inHandleGR,* int *pageNum,* GuidoPageFormat ∗ *format*)

Gives a score page format.

Parameters

inHandleGR	a Guido opaque handle to a GR structure.
pageNum	a page number.
format	on output: the page format

4.5.2.8 void GuidoSetDefaultPageFormat (const GuidoPageFormat ∗ *format*)

Sets the default score page format.

The default page format is used when no `\pageFormat` tag is present. Parameters are Guido internal units. Default values for the default page format are:

- paper size: A4
- left margin: 2cm
- right margin: 2cm
- top margin: 5cm
- bottom margin: 3cm

Parameters

format	the page format

4.5.2.9 void GuidoGetDefaultPageFormat (GuidoPageFormat ∗ *format*)

Gives the default score page format.

Parameters

format	on output: the page format

4.5.2.10 float GuidoUnit2CM (float *val*)

Converts internal Guido units into centimeters.

Parameters

val	the value to be converted

Returns

the converted value

4.5.2.11 float GuidoCM2Unit (float *val*)

Converts centimeters into internal Guido units.

Parameters

val	the value to be converted

Returns

the converted value

4.5.2.12 float GuidoUnit2Inches (float *val*)

Converts internal Guido units into inches.

Parameters

val	the value to be converted

Returns

the converted value

4.5.2.13 float GuidoInches2Unit (float *val*)

Converts inches into internal Guido units.

Parameters

val	the value to be converted

Returns

the converted value

4.5.2.14 GuidoErrCode GuidoResizePageToMusic (GRHandler *inHandleGR*)

Resize the page sizes to the music size.

Parameters

inHandleGR	a Guido opaque handle to a GR structure.

Returns

a Guido error code.

4.6 Miscellaneous

Functions

- void **GuidoGetVersionNums** (int ∗major, int ∗minor, int ∗sub)

  Gives the library version number as three integers.

- const char ∗ **GuidoGetVersionStr** ()

  Gives the library version number as a string.

- **GuidoErrCode GuidoCheckVersionNums** (int major, int minor, int sub)

  Checks a required library version number.

- float **GuidoGetLineSpace** ()

  Gives the distance between two staff lines.

- **GuidoErrCode GuidoMarkVoice** (**ARHandler** inHandleAR, int voicenum, const **GuidoDate** &date, const **GuidoDate** &duration, unsigned char red, unsigned char green, unsigned char blue)

  Gives a color to all notes of a voice between a given time interval.

- **GuidoErrCode GuidoSetSymbolPath** (**ARHandler** inHandleAR, const std::vector< std::string > &inPaths)

Makes the correspondance between an ARMusic and a path.

- **GuidoErrCode GuidoGetSymbolPath** (const **ARHandler** inHandleAR, std::vector< std::string > &inPathVector)

  Returns the path corresponding to an AR.

4.6.1 Detailed Description

Includes various functions for version management and for conversions. The number of version functions is due to historical reasons.

4.6.2 Function Documentation

4.6.2.1 void GuidoGetVersionNums (int ∗ *major,* int ∗ *minor,* int ∗ *sub*)

Gives the library version number as three integers.

Version number format is MAJOR.MINOR.SUB

Parameters

major	on output: the major revision number.
minor	on ouput: the minor revision number.
sub	on ouput: the sub revision number.

Returns

a Guido error code.

4.6.2.2 const char∗ GuidoGetVersionStr ( )

Gives the library version number as a string.

Returns

the version numebr as a string.

4.6.2.3 GuidoErrCode GuidoCheckVersionNums (int *major,* int *minor,* int *sub*)

Checks a required library version number.

Parameters

major	the major revision number.
minor	the minor revision number.
sub	the sub revision number.

Returns

noErr if the library version number is greater or equal to the version number passed as argument.
otherwise guidoErrActionFailed.

4.6.2.4 float GuidoGetLineSpace ( )

Gives the distance between two staff lines.

This value is constant (= 50). It does not depend on the context, it will probably never change in future versions of the library.

Returns

the distance between two lines of staff, in Guido internal units.

4.6.2.5 GuidoErrCode GuidoMarkVoice (ARHandler *inHandleAR,* int *voicenum,* const GuidoDate & *date,* const GuidoDate & *duration,* unsigned char *red,* unsigned char *green,* unsigned char *blue*)

Gives a color to all notes of a voice between a given time interval.

Note

Introduced for GUIDO/MIR; Allows the user to see where a musical theme appears in a voice.

Parameters

inHandleAR	a Guido opaque handle to an AR structure.
voicenum	index of the voice to mark, starting from 1
date	the date where the color-marking must begin (whole note = 1)
duration	the duration that must be covered by the color marking.
red	the red component of the marking color, from 0 to 255.
green	green color component.
blue	blue color component.

Returns

a Guido error code.

4.6.2.6 GuidoErrCode GuidoSetSymbolPath (ARHandler *inHandleAR,* const std::vector< std::string > & *inPaths*)

Makes the correspondance between an ARMusic and a path.

Parameters

inHandleAR	the destination ARHandler.
inPath	the path to associate.

Returns

noErr if the association has been made with success
otherwise guidoErrActionFailed.

4.6.2.7 GuidoErrCode GuidoGetSymbolPath (const ARHandler *inHandleAR,* std::vector< std::string > & *inPathVector*)

Returns the path corresponding to an AR.

Parameters

inHandleAR	the handle given to extract its path.

Returns

the returned path.
noErr if the association has been made with success
otherwise guidoErrActionFailed.

4.7 GUIDO Factory

Typedefs

- typedef void ∗ **ARFactoryHandler**

Functions

- **GuidoErrCode GuidoFactoryOpen** (**ARFactoryHandler** ∗outFactory)

  Opens the Guido Factory.

- void **GuidoFactoryClose** (**ARFactoryHandler** inFactory)

  Closes the Guido Factory.

- **GuidoErrCode GuidoFactoryOpenMusic** (**ARFactoryHandler** inFactory)

  Creates and opens a new music score.

- **ARHandler GuidoFactoryCloseMusic** (**ARFactoryHandler** inFactory)

  Closes the current music score.

- **GuidoErrCode GuidoFactoryOpenVoice** (**ARFactoryHandler** inFactory)

  Creates and opens a new voice.

- **GuidoErrCode GuidoFactoryCloseVoice** (**ARFactoryHandler** inFactory)

  Closes the current voice.

- **GuidoErrCode GuidoFactoryOpenChord** (**ARFactoryHandler** inFactory)

  Creates and open a new chord.

- **GuidoErrCode GuidoFactoryCloseChord** (**ARFactoryHandler** inFactory)

  Closes the current chord.

- **GuidoErrCode GuidoFactoryInsertCommata** (**ARFactoryHandler** inFactory)

  Begins a new chord note commata.

- **GuidoErrCode GuidoFactoryOpenEvent** (**ARFactoryHandler** inFactory, const char ∗inEventName)

  Creates and opens a new event (note or rest).

- **GuidoErrCode GuidoFactoryCloseEvent** (**ARFactoryHandler** inFactory)

  Closes the current event.

- **GuidoErrCode GuidoFactoryAddSharp** (**ARFactoryHandler** inFactory)

  Adds a sharp to the current event.

- **GuidoErrCode GuidoFactoryAddFlat** (**ARFactoryHandler** inFactory)

  Add a flat to the current event.

- **GuidoErrCode GuidoFactorySetEventDots** (**ARFactoryHandler** inFactory, int dots)

  Sets the number of dots the current event.

- **GuidoErrCode GuidoFactorySetEventAccidentals** (**ARFactoryHandler** inFactory, int accident)

  Sets the accidentals of the current event.

- **GuidoErrCode GuidoFactorySetOctave** (**ARFactoryHandler** inFactory, int octave)

  Sets the register (octave) of the current event.

- **GuidoErrCode GuidoFactorySetDuration** (**ARFactoryHandler** inFactory, int numerator, int denominator)

  Sets the duration of the current event.

- **GuidoErrCode GuidoFactoryOpenTag** (**ARFactoryHandler** inFactory, const char ∗name, long tagID)
- **GuidoErrCode GuidoFactoryOpenRangeTag** (**ARFactoryHandler** inFactory, const char ∗name, long tagID)
- **GuidoErrCode GuidoFactoryEndTag** (**ARFactoryHandler** inFactory)

  Indicates the end of a range tag.

- **GuidoErrCode GuidoFactoryCloseTag** (**ARFactoryHandler** inFactory)

  Closes the current tag.

- **GuidoErrCode GuidoFactoryAddTagParameterString** (**ARFactoryHandler** inFactory, const char ∗val)

  Adds a new string parameter to the current tag.

- **GuidoErrCode GuidoFactoryAddTagParameterInt** (**ARFactoryHandler** inFactory, int val)

  Adds a new integer parameter to the current tag.

- **GuidoErrCode GuidoFactoryAddTagParameterFloat** (**ARFactoryHandler** inFactory, double val)

  Adds a new floating-point parameter to the current tag.

- **GuidoErrCode GuidoFactorySetParameterName** (**ARFactoryHandler** inFactory, const char ∗name)

  Defines the name (when applicable) of the last added tag-parameter.

- **GuidoErrCode GuidoFactorySetParameterUnit** (**ARFactoryHandler** inFactory, const char ∗unit)

  Defines the unit of the last added tag-parameter.

4.7.1 Detailed Description

The GUIDO Factory API provides a set of functions to create a GUIDO abstract representation from scratch and to convert it into a graphical representation.

The GUIDO Factory is a state machine that operates on implicit current elements: for example, once you open a voice (`GuidoFactoryOpenVoice()` (p. **??**)), it becomes the current voice and all subsequent created events are implicitly added to this current voice. The elements of the factory state are:

- the current score: modified by `GuidoFactoryOpenMusic()` (p. **??**) and `GuidoFactoryCloseMusic()` (p. **??**)
- the current voice: modified by `GuidoFactoryOpenVoice()` (p. **??**) and `GuidoFactoryCloseVoice()` (p. **??**)
- the current chord: modified by `GuidoFactoryOpenChord()` (p. **??**) and `GuidoFactoryCloseChord()` (p. **??**)
- the current event: modified by `GuidoFactoryOpenEvent()` (p. **??**) and `GuidoFactoryCloseEvent()` (p. **??**)
- the current tag: modified by `GuidoFactoryOpenTag()` (p. **??**) and `GuidoFactoryCloseTag()` (p. **??**)

Some elements of the factory state reflects the GUIDO format specification:

- the current register: modified by `GuidoFactorySetRegister()`
- the current duration: modified by `GuidoFactorySetDuration()` (p. **??**)

4.7.2 Typedef Documentation

4.7.2.1 typedef void∗ ARFactoryHandler

4.7.3 Function Documentation

4.7.3.1 GuidoErrCode GuidoFactoryOpen (ARFactoryHandler ∗ *outFactory*)

Opens the Guido Factory.

Must be called before any other call to the Guido Factory API.

Returns

an integer that is an error code if not null.

4.7.3.2 void GuidoFactoryClose (ARFactoryHandler *inFactory*)

Closes the Guido Factory.

Must be called to release the factory associated resources.

4.7.3.3 GuidoErrCode GuidoFactoryOpenMusic (ARFactoryHandler *inFactory*)

Creates and opens a new music score.

The function modifies the factory state: the new score becomes the current factory score. It fails if a music score is already opened. A music score has to be closed using `GuidoFactoryCloseMusic()` (p. **??**)

Returns

an integer that is an error code if not null.

4.7.3.4 ARHandler GuidoFactoryCloseMusic (ARFactoryHandler *inFactory*)

Closes the current music score.

The function modifies the factory state if a music score is currently opened: the current factory score is set to null. It fails if no music score is opened. You must not have pending events nor pending voice at this point.

The logicical music layout (conversion from abstract to abstract representation) is part of the function operations. Next, the caller is expected to call `GuidoFactoryMakeGR()` with the returned value in order to proceed with the graphical score layout.

Returns

a GUIDO handler to the new AR structure, or 0.

See also

`GuidoFactoryMakeGR()`

4.7.3.5 GuidoErrCode GuidoFactoryOpenVoice (ARFactoryHandler *inFactory*)

Creates and opens a new voice.

The function modifies the factory state: the new voice becomes the current factory voice. It fails if a voice is already opened. A voice has to be closed using `GuidoFactoryCloseVoice()` (p. **??**) Voices are similar to sequence is GMN.

Returns

an error code

4.7.3.6 GuidoErrCode GuidoFactoryCloseVoice (ARFactoryHandler *inFactory*)

Closes the current voice.

The function modifies the factory state if a voice is currently opened: the current factory voice is set to null. It fails if no voice is opened. You must not have pending events at this point. The voice is first converted to its normal form and next added to the current score.

Returns

an error code

4.7.3.7 GuidoErrCode GuidoFactoryOpenChord (ARFactoryHandler *inFactory*)

Creates and open a new chord.

The function modifies the factory state: the new chord becomes the current factory chord. It fails if a chord is already opened. A chord has to be closed using `GuidoFactoryCloseChord()` (p. **??**)

Returns

an error code

4.7.3.8 GuidoErrCode GuidoFactoryCloseChord (ARFactoryHandler *inFactory*)

Closes the current chord.

The function modifies the factory state if a chord is currently opened: the current factory chord is set to null. It fails if no chord is opened. The chord is added to the current voice.

Returns

an error code

4.7.3.9 GuidoErrCode GuidoFactoryInsertCommata (ARFactoryHandler *inFactory*)

Begins a new chord note commata.

Called to tell the factory that a new chord-voice is beginning. This is important for the ranges that need to be added (dispdur and shareStem)

Returns

an error code

4.7.3.10 GuidoErrCode GuidoFactoryOpenEvent (ARFactoryHandler *inFactory,* const char * *inEventName*)

Creates and opens a new event (note or rest).

The function modifies the factory state: the new event becomes the current factory event. It fails if an event is already opened. An event has to be closed using `GuidoFactoryCloseEvent()` (p. **??**)

Returns

an error code

4.7.3.11 GuidoErrCode GuidoFactoryCloseEvent (ARFactoryHandler *inFactory*)

Closes the current event.

The function modifies the factory state if an event is currently opened: the current factory event is set to null. It fails if no event is opened. The event is added to the current voice.

Returns

an error code

4.7.3.12 GuidoErrCode GuidoFactoryAddSharp (ARFactoryHandler *inFactory*)

Adds a sharp to the current event.

The current event must be a note.

Returns

an error code

4.7.3.13 GuidoErrCode GuidoFactoryAddFlat (ARFactoryHandler *inFactory*)

Add a flat to the current event.

The current event must be a note.

Returns

an error code.

4.7.3.14 GuidoErrCode GuidoFactorySetEventDots (ARFactoryHandler *inFactory,* int *dots*)

Sets the number of dots the current event.

Parameters

inFactory	a handler to a Guido Factory (created with GuidoFactoryOpen)
dots	the number of dots to be carried by the current event.

Returns

an error code.

4.7.3.15 GuidoErrCode GuidoFactorySetEventAccidentals (ARFactoryHandler *inFactory,* int *accident*)

Sets the accidentals of the current event.

Parameters

inFactory	a handler to a Guido Factory (created with GuidoFactoryOpen)
accident,:	positive values are used for sharp and negative values for flats

Returns

an error code.

4.7.3.16 GuidoErrCode GuidoFactorySetOctave (ARFactoryHandler *inFactory,* int *octave*)

Sets the register (octave) of the current event.

The current event must be a note. The register becomes the current register ie next notes will carry this register until otherwise specified.

Parameters

inFactory	a handler to a Guido Factory (created with GuidoFactoryOpen)
octave	is an integer value indicating the octave of the note where *a1* is *A* 440Hz. All octaves start with the pitch class *c*.

Returns

an error code.

4.7.3.17 GuidoErrCode GuidoFactorySetDuration (ARFactoryHandler *inFactory,* int *numerator,* int *denominator*)

Sets the duration of the current event.

Durations are expressed as fractional value of a whole note: for example, a quarter note duration is 1/4. The duration becomes the current duration ie next notes will carry this duration until otherwise specified.

Parameters

inFactory	a handler to a Guido Factory (created with GuidoFactoryOpen)
numerator,:	the rational duration numerator
denominator,:	the rational duration denominator

Returns

an error code.

4.7.3.18 GuidoErrCode GuidoFactoryOpenTag (ARFactoryHandler *inFactory,* const char ∗ *name,* long *tagID*)

4.7.3.19 GuidoErrCode GuidoFactoryOpenRangeTag (ARFactoryHandler *inFactory,* const char ∗ *name,* long *tagID*)

4.7.3.20 GuidoErrCode GuidoFactoryEndTag (ARFactoryHandler *inFactory*)

Indicates the end of a range tag.

The function is applied to the current tag. It must be called when the end of a tag's range has been reached.

Parameters

inFactory	a handler to a Guido Factory (created with GuidoFactoryOpen)

Returns

an error code.

4.7.3.21 GuidoErrCode GuidoFactoryCloseTag (ARFactoryHandler *inFactory*)

Closes the current tag.

The function is applied to the current tag. Must be called after parameter and before the range.

With the following examples:

- tag<1,2,3>(c d e) : call the function before parsing c
- tag<1,2> c d : call the function before parsing c

Parameters

inFactory	a handler to a Guido Factory (created with GuidoFactoryOpen)

Returns

an error code.

4.7.3.22 GuidoErrCode GuidoFactoryAddTagParameterString (ARFactoryHandler *inFactory,* const char $*$ *val*)

Adds a new string parameter to the current tag.

Parameters

inFactory	a handler to a Guido Factory (created with GuidoFactoryOpen)
val,:	the string parameter value

Returns

an error code.

4.7.3.23 GuidoErrCode GuidoFactoryAddTagParameterInt (ARFactoryHandler *inFactory,* int *val*)

Adds a new integer parameter to the current tag.

Parameters

inFactory	a handler to a Guido Factory (created with GuidoFactoryOpen)
val,:	the parameter value

Returns

an error code.

4.7.3.24 GuidoErrCode GuidoFactoryAddTagParameterFloat (ARFactoryHandler *inFactory*, double *val*)

Adds a new floating-point parameter to the current tag.

Parameters

inFactory	a handler to a Guido Factory (created with GuidoFactoryOpen)
val,:	the parameter value

Returns

an error code.

4.7.3.25 GuidoErrCode GuidoFactorySetParameterName (ARFactoryHandler *inFactory*, const char ∗ *name*)

Defines the name (when applicable) of the last added tag-parameter.

Parameters

inFactory	a handler to a Guido Factory (created with GuidoFactoryOpen)
name,:	the tag parameter name

Returns

an error code.

4.7.3.26 GuidoErrCode GuidoFactorySetParameterUnit (ARFactoryHandler *inFactory*, const char ∗ *unit*)

Defines the unit of the last added tag-parameter.

Parameters

inFactory	a handler to a Guido Factory (created with GuidoFactoryOpen)

unit,:	a string defining the unit. The following units are supported: • `m` - meter • `cm` - centimeter • `mm` - millimeter • `in` - inch • `pt` - point (= 1/72.27 inch) • `pc` - pica (= 12pt) • `hs` - halfspace (half of the space between two lines of the current staff) • `rl` - relative measure in percent (used for positioning on score page)

Returns

an error code.

4.8 Parsing GMN files, strings and guido streams

Functions

- GuidoParser ∗ **GuidoOpenParser** ()

  Creates a new parser.

- **GuidoErrCode GuidoCloseParser** (GuidoParser ∗p)

  Close a guido parser and releases all the associated ressources.

- const char ∗ **GuidoGetStream** (GuidoStream ∗gStream)

  returns the string of the GuidoStream

- **ARHandler GuidoFile2AR** (GuidoParser ∗p, const char ∗file)

  Parse a file and create the corresponding AR.

- **ARHandler GuidoString2AR** (GuidoParser ∗p, const char ∗str)

  Parse a string and create the corresponding AR.

- **ARHandler GuidoStream2AR** (GuidoParser ∗p, GuidoStream ∗stream)

  Parse a GuidoStream and create the corresponding AR.

- **GuidoErrCode GuidoParserGetErrorCode** (GuidoParser ∗p, int &line, int &col, const char ∗∗msg)

  Get the error syntax line/column.

- GuidoStream ∗ **GuidoOpenStream** ()

  Open a guido stream.

- **GuidoErrCode GuidoCloseStream** (GuidoStream ∗s)
  Close a guido stream.
- **GuidoErrCode GuidoWriteStream** (GuidoStream ∗s, const char ∗str)
  Write data to the stream.
- **GuidoErrCode GuidoResetStream** (GuidoStream ∗s)
  Erase all stream content in order to reuse it.

4.8.1 Function Documentation

4.8.1.1 GuidoParser∗ GuidoOpenParser ( )

Creates a new parser.

Returns

a guido parser.

4.8.1.2 GuidoErrCode GuidoCloseParser (GuidoParser ∗ *p*)

Close a guido parser and releases all the associated ressources.

Parameters

p	a parser previously opened with GuidoOpenParser

Returns

a Guido error code.

4.8.1.3 const char∗ GuidoGetStream (GuidoStream ∗ *gStream*)

returns the string of the GuidoStream

Parameters

gStream	a GuidoStream

Returns

a std::string.

4.8.1.4 ARHandler GuidoFile2AR (GuidoParser ∗ *p*, const char ∗ *file*)

Parse a file and create the corresponding AR.

Parameters

p	a parser previously opened with GuidoOpenParser
file	the file to parse.

Returns

a ARHandler or 0 in case of error.

4.8.1.5 ARHandler GuidoString2AR (GuidoParser ∗ *p*, const char ∗ *str*)

Parse a string and create the corresponding AR.

Parameters

p	a parser previously opened with GuidoOpenParser
str	the string to parse.

Returns

a ARHandler or 0 in case of error.

4.8.1.6 ARHandler GuidoStream2AR (GuidoParser ∗ *p*, GuidoStream ∗ *stream*)

Parse a GuidoStream and create the corresponding AR.

Parameters

p	a parser previously opened with GuidoOpenParser
stream	the GuidoStream to parse.

Returns

a ARHandler or 0 in case of error.

4.8.1.7 GuidoErrCode GuidoParserGetErrorCode (GuidoParser ∗ *p*, int & *line*, int & *col*, const char ∗∗ *msg*)

Get the error syntax line/column.

Parameters

p	a parser previously opened with GuidoOpenParser
line	a reference that will contain a line number in case of syntax error

col	a reference that will contain a column number in case of syntax error
msg	a string that will contain the error message

Returns

a Guido error code.

4.8.1.8 GuidoStream∗ GuidoOpenStream ( )

Open a guido stream.

Guido streams are intended to implement real-time input to the parser. In particular, streams allow to retrieve an AR in while the stream is still opened.

Returns

a guido stream.

4.8.1.9 GuidoErrCode GuidoCloseStream (GuidoStream ∗ *s*)

Close a guido stream.

Parameters

s	a GuidoStream

Returns

a Guido error code.

4.8.1.10 GuidoErrCode GuidoWriteStream (GuidoStream ∗ *s*, const char ∗ *str*)

Write data to the stream.

Writing data to a stream may be viewed as writing gmn code by portion. Syntax errors concerning music/voice/tag/event/parameter non-closure won't be declared as such (GuidoWriteStream uses an automatic-closure mechanism). When a syntax error (other than a non-closure) occurs when writting data to the stream, the stream becomes invalid and should be closed. Further attempts to write data will always result in a syntax error.

Regarding syntax errors, allowed incomplete constructs are :

- opened music i.e. { without closing }
- opened voice i.e. [without closing]
- opened range tag i.e. (without closing)

- opened range parameter i.e. < without closing > but with at least one parameter
- opened chord i.e. (without closing) but with at least one note

Note

for incomplete chords and range parameters, the ',' separator must always be followed by a note or a parameter. For example, don't write "{a," and then "b}" but "{a" and then ",b}".

Parameters

s	a GuidoStream previoulsy opened with GuidoOpenStream
str	a string containing a portion of gmn code

Returns

a Guido error code.

4.8.1.11 GuidoErrCode GuidoResetStream (GuidoStream ∗ *s*)

Erase all stream content in order to reuse it.

Parameters

s	a GuidoStream previoulsy opened with GuidoOpenStream

Returns

a Guido error code.

4.9 Create a piano roll

Functions

- PianoRoll ∗ **GuidoAR2PianoRoll** (**PianoRollType** type, **ARHandler** arh)

  Creates a new piano roll from AR, corresponding to type : simplePianoRoll -> basic piano roll trajectoryPianoRoll -> every event is graphically linked to the previous one.

- PianoRoll ∗ **GuidoMidi2PianoRoll** (**PianoRollType** type, const char ∗midiFileName)

  Creates a new piano roll from Midi, corresponding to type : simplePianoRoll -> basic piano roll trajectoryPianoRoll -> every event is graphically linked to the previous one.

- **GuidoErrCode GuidoDestroyPianoRoll** (PianoRoll ∗pr)

  Destroys a guido piano roll and releases all the associated ressources.

- **GuidoErrCode GuidoPianoRollSetLimits** (PianoRoll ∗pr, **LimitParams** limitParams)

  Sets limits to a piano roll (start/end date, lower/higher pitch)

- **GuidoErrCode GuidoPianoRollEnableKeyboard** (PianoRoll ∗pr, bool enabled)

Enables keyboard or not (not enabled by default)

- **GuidoErrCode GuidoPianoRollGetKeyboardWidth** (PianoRoll ∗pr, int height, float &keyboardWidth)

  Gets the piano roll keyboard width.

- **GuidoErrCode GuidoPianoRollEnableAutoVoicesColoration** (PianoRoll ∗pr, bool enabled)

  Enables or not the automatic voices coloration (not enabled by default) (not for a midi rendering) // REM: ir If a color is manually set with GuidoPianoRollSetColorToVoice, automatic color will not be applied for this voice.

- **GuidoErrCode GuidoPianoRollSetRGBColorToVoice** (PianoRoll ∗pr, int voiceNum, int r, int g, int b, int a)

  Sets a RGB color to a voice (first voice is number 1) (black by default)

- **GuidoErrCode GuidoPianoRollSetHtmlColorToVoice** (PianoRoll ∗pr, int voiceNum, long color)

  Sets a html color to a voice (first voice is number 1) (black by default)

- **GuidoErrCode GuidoPianoRollEnableMeasureBars** (PianoRoll ∗pr, bool enabled)

  Enables or not measure bars (false by default)

- **GuidoErrCode GuidoPianoRollSetPitchLinesDisplayMode** (PianoRoll ∗pr, int mode)

  Sets the pitch lines display mode (automatic by default). Use Pitch lines display mode constants to pick lines which will be be displayed. Example : "kCLine + kGLine" will displayed C and G line. "kNoLine" doesn't display any line. "kAutoLines" adjust line display according to piano roll pitch range (automatic possibilities : no line, C line, C and G line, chromatic scale, diatonic scale);.

- **GuidoErrCode GuidoPianoRollGetMap** (PianoRoll ∗pr, int width, int height, **Time2GraphicMap** &outmap)

  Gets the piano roll map.

- **GuidoErrCode GuidoPianoRollOnDraw** (PianoRoll ∗pr, int width, int height, **VGDevice** ∗dev)

  *Draw the piano roll on a **VGDevice** (*p. **??***).*

4.9.1 Function Documentation

4.9.1.1 PianoRoll∗ GuidoAR2PianoRoll (PianoRollType *type*, ARHandler *arh*)

Creates a new piano roll from AR, corresponding to type : simplePianoRoll -> basic piano roll trajectoryPianoRoll -> every event is graphically linked to the previous one.

Parameters

PianoRoll-Type	the piano roll type
arh	an AR handler

Returns

a guido piano roll.

4.9.1.2 PianoRoll∗ GuidoMidi2PianoRoll (PianoRollType *type,* const char ∗ *midiFileName*)

Creates a new piano roll from Midi, corresponding to type : simplePianoRoll -> basic piano roll trajectoryPianoRoll -> every event is graphically linked to the previous one.

Parameters

PianoRoll-Type	the piano roll type
midiFile-Name	a midi file name

Returns

a guido piano roll.

4.9.1.3 GuidoErrCode GuidoDestroyPianoRoll (PianoRoll ∗ *pr*)

Destroys a guido piano roll and releases all the associated ressources.

Parameters

pr	a pianoroll previously created with GuidoAR2PianoRoll or Guido-Midi2PianoRoll

Returns

a Guido error code

4.9.1.4 GuidoErrCode GuidoPianoRollSetLimits (PianoRoll ∗ *pr,* LimitParams *limitParams*)

Sets limits to a piano roll (start/end date, lower/higher pitch)

Parameters

pr	a pianoroll previously created with GuidoAR2PianoRoll or Guido-Midi2PianoRoll

limitParams	the structure containing limits : start date (**GuidoDate** (p. **??**)) (0/0 to adjust automatically start date to the score's start date) end date (**GuidoDate** (p. **??**)) (0/0 to adjust automatically end date to the score's end date) minimal pitch (midi notation) (-1 to adjust automatically min pitch to the score's minimal pitch) maximal pitch (midi notation) (-1 to adjust automatically max pitch to the score's maximal pitch) Remark : minimal range pitch accepted is 1 octave.

Returns

a Guido error code

4.9.1.5 GuidoErrCode GuidoPianoRollEnableKeyboard (PianoRoll ∗ *pr,* bool *enabled*)

Enables keyboard or not (not enabled by default)

Parameters

pr	a pianoroll previously created with GuidoAR2PianoRoll or Guido-Midi2PianoRoll
enabled	a boolean corresponding to the keyboard draw state

Returns

a Guido error code

4.9.1.6 GuidoErrCode GuidoPianoRollGetKeyboardWidth (PianoRoll ∗ *pr,* int *height,* float & *keyboardWidth*)

Gets the piano roll keyboard width.

Parameters

pr	a pianoroll previously created with GuidoAR2PianoRoll or Guido-Midi2PianoRoll
height	the height of the canvas (-1 to set the default height : 512)
keyboard-Width	the pianoroll keyboard width

Returns

a Guido error code

4.9.1.7 GuidoErrCode GuidoPianoRollEnableAutoVoicesColoration (PianoRoll ∗ *pr,* bool *enabled*)

Enables or not the automatic voices coloration (not enabled by default) (not for a midi rendering) // REM: ir If a color is manually set with GuidoPianoRollSetColorToVoice, automatic color will not be applied for this voice.

Parameters

pr	a pianoroll previously created with GuidoAR2PianoRoll
enabled	a boolean corresponding to the color state

Returns

a Guido error code

4.9.1.8 GuidoErrCode GuidoPianoRollSetRGBColorToVoice (PianoRoll ∗ *pr,* int *voiceNum,* int *r,* int *g,* int *b,* int *a*)

Sets a RGB color to a voice (first voice is number 1) (black by default)

Parameters

pr	a pianoroll previously created with GuidoAR2PianoRoll or Guido-Midi2PianoRoll
voiceNum	the voice number (first voice is number 1)
r	the red param of RGB color
g	the green param of RGB color
b	the blue param of RGB color
a	the alpha param of RGB color

Returns

a Guido error code

4.9.1.9 GuidoErrCode GuidoPianoRollSetHtmlColorToVoice (PianoRoll ∗ *pr,* int *voiceNum,* long *color*)

Sets a html color to a voice (first voice is number 1) (black by default)

Parameters

pr	a pianoroll previously created with GuidoAR2PianoRoll or Guido-Midi2PianoRoll
voiceNum	the voice number (first voice is number 1)
color	the html color (constants are defined in Colors.h)

Returns

a Guido error code

4.9.1.10 GuidoErrCode GuidoPianoRollEnableMeasureBars (PianoRoll ∗ *pr,* bool *enabled*)

Enables or not measure bars (false by default)

Parameters

pr	a pianoroll previously created with GuidoAR2PianoRoll or Guido-Midi2PianoRoll
enabled	a boolean corresponding to the measure bars draw state

Returns

a Guido error code

4.9.1.11 GuidoErrCode GuidoPianoRollSetPitchLinesDisplayMode (PianoRoll ∗ *pr,* int *mode*)

Sets the pitch lines display mode (automatic by default). Use Pitch lines display mode constants to pick lines which will be be displayed. Example : "kCLine + kGLine" will displayed C and G line. "kNoLine" doesn't display any line. "kAutoLines" adjust line display according to piano roll pitch range (automatic possibilities : no line, C line, C and G line, chromatic scale, diatonic scale);.

Parameters

pr	a pianoroll previously created with GuidoAR2PianoRoll or Guido-Midi2PianoRoll
mode	an int corresponding to the pitch lines display mode

Returns

a Guido error code

4.9.1.12 GuidoErrCode GuidoPianoRollGetMap (PianoRoll ∗ *pr,* int *width,* int *height,* Time2GraphicMap & *outmap*)

Gets the piano roll map.

Parameters

pr	a pianoroll previously created with GuidoAR2PianoRoll or Guido-Midi2PianoRoll
width	the width of the piano roll (-1 to set the default width : 1024)
height	the height of the canvas (-1 to set the default height : 512)

Returns

a Guido error code (returns guidoErrBadParameter if keyboard width is higher than width param)

4.9.1.13 GuidoErrCode GuidoPianoRollOnDraw (PianoRoll ∗ *pr,* int *width,* int *height,* VGDevice ∗ *dev*)

Draw the piano roll on a **VGDevice** (p. **??**).

Parameters

pr	a pianoroll previously created with GuidoAR2PianoRoll or Guido-Midi2PianoRoll
width	the width on which piano roll will be drawn (-1 to set the default width : 1024)
height	the height on which piano roll will be drawn (-1 to set the default height : 512)
dev	the device on which piano will be drawn

Returns

a Guido error code (returns guidoErrBadParameter if keyboard width is higher than width param)

4.10 GUIDO Mapping

Classes

- struct **GuidoElementInfos**
- class **TimeSegment**

  a time segment definition and operations
- class **MapCollector**
- class **RectInfos**
- class **TimeMapCollector**

Typedefs

- typedef std::vector< std::pair< **TimeSegment**, FloatRect > > **Time2GraphicMap**
- typedef std::pair< FloatRect, **RectInfos** > **MapElement**

Enumerations

- enum **GuidoeElementSelector** {

  kGuidoPage, **kGuidoSystem**, **kGuidoSystemSlice**, **kGuidoStaff**,

  kGuidoBar, **kGuidoEvent**, **kGuidoScoreElementEnd** }

- enum **GuidoElementType** {

  kNote = 1, **kRest**, **kEmpty**, **kBar**,

  kRepeatBegin, **kRepeatEnd**, **kStaff**, **kSystemSlice**,

  kSystem, **kPage** }

Functions

- std::ostream & **operator**<< (std::ostream &out, const **GuidoDate** &d)
- std::ostream & **operator**<< (std::ostream &out, const **TimeSegment** &s)
- **GuidoErrCode** **GuidoGetMap** (**CGRHandler** gr, int pagenum, float width, float height, **GuidoeElementSelector** sel, **MapCollector** &f)

  Retrieves the graphic to time mapping.

- **GuidoErrCode** **GuidoGetPageMap** (**CGRHandler** gr, int pagenum, float w, float h, **Time2GraphicMap** &outmap)

  Retrieves a guido page graphic to time mapping.

- **GuidoErrCode** **GuidoGetStaffMap** (**CGRHandler** gr, int pagenum, float w, float h, int staff, **Time2GraphicMap** &outmap)

  Retrieves a guido staff graphic to time mapping.

- **GuidoErrCode** **GuidoGetVoiceMap** (**CGRHandler** gr, int pagenum, float w, float h, int voice, **Time2GraphicMap** &outmap)

  Retrieves a guido voice graphic to time mapping.

- **GuidoErrCode** **GuidoGetSystemMap** (**CGRHandler** gr, int pagenum, float w, float h, **Time2GraphicMap** &outmap)

  Retrieves a guido system graphic to time mapping.

- bool **GuidoGetTime** (const **GuidoDate** &date, const **Time2GraphicMap** map, **TimeSegment** &t, FloatRect &r)

  Retrieves a time segment and the associated graphic segment in a mapping.

- bool **GuidoGetPoint** (float x, float y, const **Time2GraphicMap** map, **TimeSegment** &t, FloatRect &r)

  Retrieves a time segment and the associated graphic segment in a mapping.

- **GuidoErrCode** **GuidoGetSVGMap** (**GRHandler** gr, int pagenum, **GuidoeElementSelector** sel, std::vector< **MapElement** > &outMap)

  Retrieves the graphic to time mapping corresponding to the SVG output.

- **GuidoErrCode** **GuidoGetTimeMap** (**CARHandler** gr, **TimeMapCollector** &f)

  Retrieves the rolled to unrolled time mapping.

4.10.1 Typedef Documentation

4.10.1.1 typedef std::vector<std::pair<TimeSegment, FloatRect> > Time2GraphicMap

4.10.1.2 typedef std::pair<FloatRect, RectInfos> MapElement

4.10.2 Enumeration Type Documentation

4.10.2.1 enum GuidoeElementSelector

Enumerator:

kGuidoPage

kGuidoSystem

kGuidoSystemSlice

kGuidoStaff

kGuidoBar

kGuidoEvent

kGuidoScoreElementEnd

4.10.2.2 enum GuidoElementType

Enumerator:

kNote

kRest

kEmpty

kBar

kRepeatBegin

kRepeatEnd

kStaff

kSystemSlice

kSystem

kPage

4.10.3 Function Documentation

4.10.3.1 std::ostream& operator<< (std::ostream & *out,* const GuidoDate & *d*) `[inline]`

References GuidoDate::denom, and GuidoDate::num.

4.10.3.2 std::ostream& operator<< (std::ostream & *out,* const TimeSegment & *s*) [inline]

References TimeSegment::print().

4.10.3.3 GuidoErrCode GuidoGetMap (CGRHandler *gr,* int *pagenum,* float *width,* float *height,* GuidoeElementSelector *sel,* MapCollector & *f*)

Retrieves the graphic to time mapping.

Parameters

gr	a Guido opaque handle to a GR structure.
pagenum	a page index, starting from 1.
width	the page width.
height	the page height.
sel	GuidoeElementSelector to filter undesired objects out.
f	a **MapCollector** (p. **??**) object that will be called for each selected element.

Returns

an error code.

4.10.3.4 GuidoErrCode GuidoGetPageMap (CGRHandler *gr,* int *pagenum,* float *w,* float *h,* Time2GraphicMap & *outmap*)

Retrieves a guido page graphic to time mapping.

Parameters

gr	a Guido opaque handle to a GR structure.
pagenum	a page index, starting from 1.
w	the page width.
h	the page height.
outmap	contains the mapping on output.

Returns

an error code.

4.10.3.5 GuidoErrCode GuidoGetStaffMap (CGRHandler *gr,* int *pagenum,* float *w,* float *h,* int *staff,* Time2GraphicMap & *outmap*)

Retrieves a guido staff graphic to time mapping.

Parameters

gr	a Guido opaque handle to a GR structure.

pagenum	a page index, starting from 1.
w	the page width.
h	the page height.
staff	the staff index (starting from 1).
outmap	contains the mapping on output.

Returns

an error code.

4.10.3.6 GuidoErrCode GuidoGetVoiceMap (CGRHandler *gr,* int *pagenum,* float *w,* float *h,* int *voice,* Time2GraphicMap & *outmap*)

Retrieves a guido voice graphic to time mapping.

Parameters

gr	a Guido opaque handle to a GR structure.
pagenum	a page index, starting from 1.
w	the page width.
h	the page height.
voice	the voice index (starting from 1).
outmap	contains the mapping on output.

Returns

an error code.

4.10.3.7 GuidoErrCode GuidoGetSystemMap (CGRHandler *gr,* int *pagenum,* float *w,* float *h,* Time2GraphicMap & *outmap*)

Retrieves a guido system graphic to time mapping.

Parameters

gr	a Guido opaque handle to a GR structure.
pagenum	a page index, starting from 1.
w	the page width.
h	the page height.
outmap	contains the mapping on output.

Returns

an error code.

4.10.3.8 bool GuidoGetTime (const GuidoDate & *date,* const Time2GraphicMap *map,* TimeSegment & *t,* FloatRect & *r*)

Retrieves a time segment and the associated graphic segment in a mapping.

Parameters

date	a date used to select the container time segment
map	a time to graphic map.
t	on output, the time segment containing the date (note that segments are right opened)
r	on output, the graphic segment associated to the time segment

Returns

false when there is no segment containing the date.

4.10.3.9 bool GuidoGetPoint (float *x,* float *y,* const Time2GraphicMap *map,* TimeSegment & *t,* FloatRect & *r*)

Retrieves a time segment and the associated graphic segment in a mapping.

Parameters

x	a point x coordinate
y	a point y coordinate
map	a time to graphic map.
r	on output, the graphic segment containing the point (note that segments are right and bottom opened)
t	on output, the time segment associated to the graphic segment

Returns

false when there is no segment containing the point.

4.10.3.10 GuidoErrCode GuidoGetSVGMap (GRHandler *gr,* int *pagenum,* GuidoeElementSelector *sel,* std::vector< MapElement > & *outMap*)

Retrieves the graphic to time mapping corresponding to the SVG output.

Parameters

gr	a Guido opaque handle to a GR structure.
pagenum	a page index, starting from 1.
sel	GuidoeElementSelector to filter undesired objects out.
outMap	on output: a vector containing the map elements

Returns

an error code.

4.10.3.11 GuidoErrCode GuidoGetTimeMap (CARHandler *gr*, TimeMapCollector & *f*)

Retrieves the rolled to unrolled time mapping.

Parameters

gr	a Guido opaque handle to a GR structure.
f	a **TimeMapCollector** (p. **??**) object that will be called for each time segment.

Returns

an error code.

4.11 Virtual Graphic System

Classes

- class **VGColor**

  Generic class to manipulate device independant colors.

- class **VGDevice**

  Generic platform independant drawing device.

- class **VGFont**

  Generic pure virtual & device-independant font class.

- class **VGPen**

  Generic class to manipulate device independant pens.

- class **VGSystem**

  Generic pure virtual class for manipulating platform independant drawing devices and fonts.

Defines

- #define **ALPHA_TRANSPARENT** 0
- #define **ALPHA_OPAQUE** 255

Functions

- std::ostream & **operator**<< (std::ostream &out, const **VGColor** &c)
- **VGColor** & **operator+=** (short v)

4.11.1 Detailed Description

The virtual graphic system is intended as an abstract layer covering platform dependencies at graphic level. It represents a set of abstract classes covering the basic needs of an application: printing text, drawing to the screen or to an offscreen, etc... The set of abstract classes includes:

- a **VGSystem** (p. **??**) class: to cover allocation of specific **VGDevice** (p. **??**) and **VGFont** (p. **??**) objects.
- a **VGDevice** (p. **??**) class: specialized on drawing onscreen of offscreen
- a **VGFont** (p. **??**) class: to cover fonts management

This set of classes is implemented for different target platforms: implementations are provided for Windows GDI and Mac OSX Quartz, implementations for Windows GDI+, OpenGL and Linux GTK are in progress.

4.11.2 Define Documentation

4.11.2.1 #define ALPHA_TRANSPARENT 0

4.11.2.2 #define ALPHA_OPAQUE 255

4.11.3 Function Documentation

4.11.3.1 std::ostream& operator<< (std::ostream & *out*, const VGColor & *c*) `[inline]`

4.11.3.2 VGColor & operator+= (short *v*) `[inline, inherited]`

References VGColor::mBlue, VGColor::mGreen, and VGColor::mRed.

Chapter 5

Class Documentation

5.1 GPaintStruct Struct Reference

A structure to keep information about clipping and redrawing regions.

Public Attributes

- bool **erase**

  a flag to ignore the following rect and to redraw everything

- int **left**
- int **top**
- int **right**
- int **bottom**

5.1.1 Detailed Description

A structure to keep information about clipping and redrawing regions.

5.1.2 Member Data Documentation

5.1.2.1 bool erase

a flag to ignore the following rect and to redraw everything

5.1.2.2 int left

Absolute Guido virtual coordinates of the clipping rectangle. Only systems that intersect with this rectangle will be drawn.

5.1.2.3 int top

5.1.2.4 int right

5.1.2.5 int bottom

5.2 Guido2MidiParams Struct Reference

The **GuidoInitDesc** (p. **??**) data structure contains all information required by **GuidoInit()** (p. **??**)

Public Attributes

- int **fTempo**

  default tempo in quarter per minute - default value: 120

- int **fTicks**

  ticks per quarternote - default value: 960 (64∗3∗5)

- int **fChan**

  the default Midi channel - default value: 1

- float **fIntensity**

  default intensity [0.0 ... 1.0] - default value: 0.8

- float **fAccentFactor**

  accent intensity factor - default value: 1.1

- float **fMarcatoFactor**

  marcato intensity factor - default value: 1.2

- float **fDFactor**

  default duration factor [0.0 ... 1.0] - default value: 0.8

- float **fStaccatoFactor**

  staccato duration factor - default value: 0.5

- float **fSlurFactor**

  legato duration factor - default value: 1.0

- float **fTenutoFactor**

  tenuto duration factor - default value: 0.90

- float **fFermataFactor**

  fermata duration factor - default value: 2.0

- std::map< int, int > **fVChans**

  a map between voice numbers and MIDI channels (all indexed from 1)

5.2.1 Detailed Description

The **GuidoInitDesc** (p. **??**) data structure contains all information required by **GuidoInit()** (p. **??**)

5.2.2 Member Data Documentation

5.2.2.1 int fTempo

default tempo in quarter per minute - default value: 120

5.2.2.2 int fTicks

ticks per quarternote - default value: 960 ($64*3*5$)

5.2.2.3 int fChan

the default Midi channel - default value: 1

5.2.2.4 float fIntensity

default intensity [0.0 ... 1.0] - default value: 0.8

5.2.2.5 float fAccentFactor

accent intensity factor - default value: 1.1

5.2.2.6 float fMarcatoFactor

marcato intensity factor - default value: 1.2

5.2.2.7 float fDFactor

default duration factor [0.0 ... 1.0] - default value: 0.8

5.2.2.8 float fStaccatoFactor

staccato duration factor - default value: 0.5

5.2.2.9 float fSlurFactor

legato duration factor - default value: 1.0

5.2.2.10 float fTenutoFactor

tenuto duration factor - default value: 0.90

5.2.2.11 float fFermataFactor

fermata duration factor - default value: 2.0

5.2.2.12 std::map<int, int> fVChans

a map between voice numbers and MIDI channels (all indexed from 1)

5.3 GuidoDate Struct Reference

Public Attributes

- int **num**
  the date numerator
- int **denom**
  the date denominator

5.3.1 Detailed Description

A Guido date is expressed as a fractional value where 1/1 represents the whole note.

5.3.2 Member Data Documentation

5.3.2.1 int num

the date numerator

Referenced by operator<<().

5.3.2.2 int denom

the date denominator

Referenced by operator<<().

5.4 GuidoElementInfos Struct Reference

Public Attributes

- **GuidoElementType type**

  the element type

- int **staffNum**

  the element staff number or 0 when na

- int **voiceNum**

  the element voice number or 0 when na

5.4.1 Member Data Documentation

5.4.1.1 GuidoElementType type

the element type

5.4.1.2 int staffNum

the element staff number or 0 when na

5.4.1.3 int voiceNum

the element voice number or 0 when na

5.5 GuidoInitDesc Struct Reference

The **GuidoInitDesc** (p. **??**) data structure contains all information required by **GuidoInit()** (p. **??**)

Public Attributes

- **VGDevice** $*$ **graphicDevice**

a graphic device pointer, if null a default device is used

- void ∗ **reserved**
- const char ∗ **musicFont**

  the music font name, defaults to "guido" font when null

- const char ∗ **textFont**

  a text font name, defaults to "times" font when null

5.5.1 Detailed Description

The **GuidoInitDesc** (p. **??**) data structure contains all information required by **GuidoInit()** (p. **??**)

5.5.2 Member Data Documentation

5.5.2.1 VGDevice∗ graphicDevice

a graphic device pointer, if null a default device is used

5.5.2.2 void∗ reserved

5.5.2.3 const char∗ musicFont

the music font name, defaults to "guido" font when null

5.5.2.4 const char∗ textFont

a text font name, defaults to "times" font when null

5.6 GuidoLayoutSettings Struct Reference

Public Attributes

- float **systemsDistance**
- int **systemsDistribution**
- float **systemsDistribLimit**
- float **force**
- float **spring**
- int **neighborhoodSpacing**
- int **optimalPageFill**
- int **resizePage2Music**

5.6.1 Detailed Description

Settings for the graphic score layout.

5.6.2 Member Data Documentation

5.6.2.1 float systemsDistance

Control distance between systems, distance is in internal units (default value: 75)

5.6.2.2 int systemsDistribution

control systems distribution. Possible values: kAutoDistrib (default), kAlwaysDistrib, kNeverDistrib

5.6.2.3 float systemsDistribLimit

Maximum distance allowed between two systems, for automatic distribution mode. Distance is relative to the height of the inner page. Default value: 0.25 (that is: 1/4 of the page height)

5.6.2.4 float force

force value of the Space-Force function typical values range from 400 to 1500. Default value: 750

5.6.2.5 float spring

the spring parameter typical values range from 1 to 5. Default value: 1.1

5.6.2.6 int neighborhoodSpacing

boolean value to tell the engine to use the Neighborhood spacing algorithm or not (default value: 0)

5.6.2.7 int optimalPageFill

boolean value to tell the engine to use the optimal page fill algorithm or not (default value: 1)

5.6.2.8 int resizePage2Music

boolean value to tell the engine to resize page to music (default value: 1)

5.7 GuidoOnDrawDesc Struct Reference

Contains all graphic-related information required by **GuidoOnDraw()** (p. **??**)

Public Attributes

- **GRHandler handle**

  A Guido handler to a graphic representation.

- **VGDevice** ∗ **hdc**

  A graphic device context.

- int **page**

  The page number. Starts from 1.

- **GPaintStruct updateRegion**

  Indicates what to (re)draw.

- int **scrollx**
- int **scrolly**
- float **reserved**
- int **sizex**
- int **sizey**
- int **isprint**

5.7.1 Detailed Description

Contains all graphic-related information required by **GuidoOnDraw()** (p. **??**) Used to render a page of score into a device context.

5.7.2 Member Data Documentation

5.7.2.1 GRHandler handle

A Guido handler to a graphic representation.

5.7.2.2 VGDevice∗ hdc

A graphic device context.

5.7.2.3 int page

The page number. Starts from 1.

5.7.2.4 GPaintStruct updateRegion

Indicates what to (re)draw.

5.7.2.5 int scrollx

Indicates the coordinates of the score point that will appear at the graphic origin. Typical values are 0. Non null values have the effect of moving a window over the score page, like scroll bars that move a page view. Units are internal units.

5.7.2.6 int scrolly

5.7.2.7 float reserved

Indicates the size of the drawing zone. The size is expressed in graphic device units (pixels for a screen for example)

5.7.2.8 int sizex

5.7.2.9 int sizey

5.7.2.10 int isprint

If true, the engine ignores scroll, zoom and sizes parameters. If false, the engine draws a white background in the graphic device.

5.8 GuidoPageFormat Struct Reference

Public Attributes

- float **width**
- float **height**
- float **marginleft**
- float **margintop**
- float **marginright**
- float **marginbottom**

5.8.1 Detailed Description

The page format parameters

Page format should be given in internal units. To convert from cm or inches you should use `GuidoCM2Unit` or `GuidoInches2Unit`

5.8.2 Member Data Documentation

5.8.2.1 float width

5.8.2.2 float height

5.8.2.3 float marginleft

5.8.2.4 float margintop

5.8.2.5 float marginright

5.8.2.6 float marginbottom

5.9 LimitParams Struct Reference

Public Attributes

- **GuidoDate startDate**
- **GuidoDate endDate**
- int **lowPitch**
- int **highPitch**

5.9.1 Member Data Documentation

5.9.1.1 GuidoDate startDate

5.9.1.2 GuidoDate endDate

5.9.1.3 int lowPitch

5.9.1.4 int highPitch

5.10 MapCollector Class Reference

Public Member Functions

- virtual ∼**MapCollector** ()
- virtual void **Graph2TimeMap** (const FloatRect &box, const **TimeSegment** &dates, const **GuidoElementInfos** &infos)=0

  a method called by the GuidoGetMap function

5.10.1 Constructor & Destructor Documentation

5.10.1.1 virtual ∼MapCollector ( ) `[inline, virtual]`

5.10.2 Member Function Documentation

5.10.2.1 virtual void Graph2TimeMap (const FloatRect & *box,* const TimeSegment & *dates,* const GuidoElementInfos & *infos*) `[pure virtual]`

a method called by the GuidoGetMap function

Parameters

box	a graphic rectangle expressed with no scaling and no coordinates offset.
dates	a time segment containing the corresponding start and end dates
infos	information about the corresponding element.

5.11 RectInfos Class Reference

Public Member Functions

- **RectInfos** (const **TimeSegment** &ts, const **GuidoElementInfos** &infos)
- virtual ∼**RectInfos** ()
- const **TimeSegment** & **time** () const
- const **GuidoElementInfos** & **infos** () const

5.11.1 Constructor & Destructor Documentation

5.11.1.1 RectInfos (const TimeSegment & *ts,* const GuidoElementInfos & *infos*) `[inline]`

5.11.1.2 virtual ∼RectInfos ( ) `[inline, virtual]`

5.11.2 Member Function Documentation

5.11.2.1 const TimeSegment& time ( ) const `[inline]`

5.11.2.2 const GuidoElementInfos& infos ( ) const `[inline]`

5.12 TimeMapCollector Class Reference

Public Member Functions

- virtual ∼**TimeMapCollector** ()
- virtual void **Time2TimeMap** (const **TimeSegment** &from, const **TimeSegment** &to)=0

a method called by the GuidoGetTimeMap function

5.12.1 Constructor & Destructor Documentation

5.12.1.1 virtual ∼TimeMapCollector ( ) `[inline, virtual]`

5.12.2 Member Function Documentation

5.12.2.1 virtual void Time2TimeMap (const TimeSegment & *from,* const TimeSegment & *to*) `[pure virtual]`

a method called by the GuidoGetTimeMap function

Parameters

from	a time segment expressed in score time i.e. rolled time.
to	a time segment expressed in unrolled time

5.13 TimeSegment Class Reference

a time segment definition and operations

Public Member Functions

- **TimeSegment** ()
- **TimeSegment** (const **TimeSegment** &s)
- **TimeSegment** (const **GuidoDate** &a, const **GuidoDate** &b)
- virtual ∼**TimeSegment** ()
- void **print** (std::ostream &out) const

  print the time segment

- bool **empty** () const

  check for empty segment

- bool **intersect** (const **TimeSegment** &ts) const

  check for segments intersection

- bool **include** (const **GuidoDate** &date) const

  check for date inclusion

- bool **include** (const **TimeSegment** &ts) const

  check for segment inclusion

- bool **operator**< (const **TimeSegment** &ts) const *order relationship: the smaller is the smaller first date*

- bool **operator==** (const **TimeSegment** &ts) const
- **TimeSegment operator&** (const **TimeSegment** &ts) const

  intersection operation (may return an arbitrary empty segment)

5.13.1 Detailed Description

a time segment definition and operations

5.13.2 Constructor & Destructor Documentation

5.13.2.1 TimeSegment ( ) `[inline]`

5.13.2.2 TimeSegment (const TimeSegment & *s*) `[inline]`

5.13.2.3 TimeSegment (const GuidoDate & *a*, const GuidoDate & *b*) `[inline]`

5.13.2.4 virtual ∼TimeSegment ( ) `[inline, virtual]`

5.13.3 Member Function Documentation

5.13.3.1 void print (std::ostream & *out*) const

print the time segment

Referenced by operator<<().

5.13.3.2 bool empty ( ) const

check for empty segment

5.13.3.3 bool intersect (const TimeSegment & *ts*) const

check for segments intersection

5.13.3.4 bool include (const GuidoDate & *date*) const

check for date inclusion

5.13.3.5 bool include (const TimeSegment & *ts*) const

check for segment inclusion

5.13.3.6 bool operator< (const TimeSegment & *ts*) const

order relationship: the smaller is the smaller first date

5.13.3.7 bool operator== (const TimeSegment & *ts*) const

5.13.3.8 TimeSegment operator& (const TimeSegment & *ts*) const

intersection operation (may return an arbitrary empty segment)

5.14 VGColor Class Reference

Generic class to manipulate device independant colors.

Public Member Functions

- **VGColor** (unsigned char gray=0)
- **VGColor** (const **VGColor** &in)
- **VGColor** (unsigned char r, unsigned char g, unsigned char b, unsigned char a=255)
- **VGColor** (const unsigned char col[4])
- void **Set** (unsigned char r, unsigned char g, unsigned char b, unsigned char a=255)
- void **Set** (const **VGColor** &in)
- bool **operator==** (const **VGColor** &col) const
- bool **operator!=** (const **VGColor** &col) const
- **VGColor** & **operator+=** (short v)
- std::ostream & **print** (std::ostream &out) const

Public Attributes

- unsigned char **mRed**
- unsigned char **mGreen**
- unsigned char **mBlue**
- unsigned char **mAlpha**

5.14.1 Detailed Description

Generic class to manipulate device independant colors.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 VGColor (unsigned char *gray* = `0`) `[inline]`

5.14.2.2 VGColor (const VGColor & *in*) `[inline]`

References Set().

5.14.2.3 VGColor (unsigned char *r*, unsigned char *g*, unsigned char *b*, unsigned char *a* = `255`) `[inline]`

5.14.2.4 VGColor (const unsigned char *col[4]*) `[inline, explicit]`

References Set().

5.14.3 Member Function Documentation

5.14.3.1 void Set (unsigned char *r*, unsigned char *g*, unsigned char *b*, unsigned char *a* = `255`) `[inline]`

References mAlpha, mBlue, mGreen, and mRed.

Referenced by VGColor().

5.14.3.2 void Set (const VGColor & *in*) `[inline]`

References mAlpha, mBlue, mGreen, and mRed.

5.14.3.3 bool operator== (const VGColor & *col*) const `[inline]`

References mAlpha, mBlue, mGreen, and mRed.

5.14.3.4 bool operator!= (const VGColor & *col*) const `[inline]`

References mAlpha, mBlue, mGreen, and mRed.

5.14.3.5 std::ostream& print (std::ostream & *out*) const `[inline]`

References mAlpha, mBlue, mGreen, and mRed.

5.14.4 Member Data Documentation

5.14.4.1 unsigned char mRed

Referenced by operator!=(), operator+=(), operator==(), print(), and Set().

5.14.4.2 unsigned char mGreen

Referenced by operator!=(), operator+=(), operator==(), print(), and Set().

5.14.4.3 unsigned char mBlue

Referenced by operator!=(), operator+=(), operator==(), print(), and Set().

5.14.4.4 unsigned char mAlpha

Referenced by operator!=(), operator==(), print(), and Set().

5.15 VGDevice Class Reference

Generic platform independant drawing device.

Public Types

- enum **VRasterOpMode** {
  kUnknown = 0, **kOpCopy** = 1, **kOpAnd** = 2, **kOpXOr** = 4,
  kOpInvert = 8, **kOpOr** = 16 }

  Raster operation modes (color fill, bit copy, etc.)
- enum **VTextAlignMode** {
  kAlignBase = 1, **kAlignBottom** = 2, **kAlignTop** = 4, **kAlignCenter** = 8,
  kAlignLeft = 16, **kAlignRight** = 32, **kAlignBaseLeft** = kAlignLeft | kAlignBase }

  Text alignment modes.

Public Member Functions

- virtual ~**VGDevice** ()
- virtual bool **IsValid** () const =0

  Returns the ability of the current VGdevice to be drawn into.
- virtual bool **BeginDraw** ()=0

- virtual void **EndDraw** ()=0
- virtual void **InvalidateRect** (float left, float top, float right, float bottom)=0
- virtual void **MoveTo** (float x, float y)=0

  Moves the current position to the point specified by (x,y).

- virtual void **LineTo** (float x, float y)=0
- virtual void **Line** (float x1, float y1, float x2, float y2)=0
- virtual void **Frame** (float left, float top, float right, float bottom)=0
- virtual void **Arc** (float left, float top, float right, float bottom, float startX, float startY, float endX, float endY)=0
- virtual void **Triangle** (float x1, float y1, float x2, float y2, float x3, float y3)=0
- virtual void **Polygon** (const float ∗xCoords, const float ∗yCoords, int count)=0
- virtual void **Rectangle** (float left, float top, float right, float bottom)=0
- virtual void **SetMusicFont** (const **VGFont** ∗font)=0
- virtual const **VGFont** ∗ **GetMusicFont** () const =0

  *Returns the currently selected music **VGFont** (*p. **??***).*

- virtual void **SetTextFont** (const **VGFont** ∗font)=0
- virtual const **VGFont** ∗ **GetTextFont** () const =0

  *Returns the currently selected text **VGFont** (*p. **??***).*

- virtual void **selectfont** (int)

  Selects a font (only for SVG device).

- virtual void **SelectPen** (const **VGColor** &inColor, float witdh)=0
- virtual void **SelectFillColor** (const **VGColor** &c)=0
- virtual void **PushPen** (const **VGColor** &inColor, float inWidth)=0
- virtual void **PopPen** ()=0
- virtual void **PushFillColor** (const **VGColor** &inColor)=0
- virtual void **PopFillColor** ()=0
- virtual void **SetRasterOpMode** (**VRasterOpMode** ROpMode)=0
- virtual **VRasterOpMode** **GetRasterOpMode** () const =0
- virtual bool **CopyPixels** (**VGDevice** ∗pSrcDC, float alpha=-1.0)=0
- virtual bool **CopyPixels** (int xDest, int yDest, **VGDevice** ∗pSrcDC, int xSrc, int ySrc, int nSrcWidth, int nSrcHeight, float alpha=-1.0)=0
- virtual bool **CopyPixels** (int xDest, int yDest, int dstWidth, int dstHeight, **VGDevice** ∗pSrcDC, float alpha=-1.0)=0
- virtual bool **CopyPixels** (int xDest, int yDest, int dstWidth, int dstHeight, **VGDevice** ∗pSrcDC, int xSrc, int ySrc, int nSrcWidth, int nSrcHeight, float alpha=-1.0)=0
- virtual void **SetScale** (float x, float y)=0

  *Sets the scale factors of the current **VGDevice** (*p. **??***) to the input values.*

- virtual void **SetOrigin** (float x, float y)=0

  *Specifies which **VGDevice** (*p. **??***) point (x,y) maps to the window origin (0,0).*

- virtual void **OffsetOrigin** (float x, float y)=0

Offsets the current VGDevice's origin (see above).

- virtual void **LogicalToDevice** (float ∗x, float ∗y) const =0
- virtual void **DeviceToLogical** (float ∗x, float ∗y) const =0
- virtual float **GetXScale** () const =0
- virtual float **GetYScale** () const =0
- virtual float **GetXOrigin** () const =0
- virtual float **GetYOrigin** () const =0
- virtual void **NotifySize** (int inWidth, int inHeight)=0
- virtual int **GetWidth** () const =0

  Returns the width (set via NotifySize) of the current ***VGDevice*** *(*p. **??***).*

- virtual int **GetHeight** () const =0

  Returns the height (set via NotifySize) of the current ***VGDevice*** *(*p. **??***).*

- virtual void **DrawMusicSymbol** (float x, float y, unsigned int inSymbolID)=0
- virtual void **DrawString** (float x, float y, const char ∗s, int inCharCount)=0
- virtual void **SetFontColor** (const **VGColor** &inColor)=0

  Sets the text/music color for the current ***VGDevice*** *(*p. **??***).*

- virtual **VGColor GetFontColor** () const =0

  Returns the text/music color of the current ***VGDevice*** *(*p. **??***).*

- virtual void **SetFontBackgroundColor** (const **VGColor** &inColor)=0

  Sets the text/music background color for the current ***VGDevice*** *(*p. **??***).*

- virtual **VGColor GetFontBackgroundColor** () const =0

  Returns the text/music background color of the current ***VGDevice*** *(*p. **??***).*

- virtual void **SetFontAlign** (unsigned int inAlign)=0
- virtual unsigned int **GetFontAlign** () const =0
- virtual void **SetDPITag** (float inDPI)=0

  Sets the printing resolution of the current ***VGDevice*** *(*p. **??***).*

- virtual float **GetDPITag** () const =0

  Returns the printing resolution of the current ***VGDevice*** *(*p. **??***).*

- virtual void ∗ **GetBitMapPixels** ()=0

  Allows pixels operations and returns a pointer to the bitmap pixels.

- virtual void **ReleaseBitMapPixels** ()=0

  Update bitmap pixels and ends pixels operations.

- virtual const char ∗ **GetImageData** (const char ∗&outDataPtr, int &outLength)=0

  Gives the current device data and returns the data associated mime type.

- virtual void **ReleaseImageData** (const char $*$) const =0

  Release the pointer returned by GetImageData.

- virtual **VGSystem** $*$ **getVGSystem** () const =0

  temporary hack - must be removed asap

- virtual void $*$ **GetNativeContext** () const =0

  Exports all graphical data to an image file.

- virtual void **SelectPenColor** (const **VGColor** &inColor)=0

  Creates a new ***VGPen*** *(p.* **??***) object with the specified* ***VGColor*** *(p.* **??***).*

- virtual void **SelectPenWidth** (float witdh)=0

  Creates a new ***VGPen*** *(p.* **??***) object with the specified* ***VGColor*** *(p.* **??***).*

- virtual void **PushPenColor** (const **VGColor** &inColor)=0
- virtual void **PopPenColor** ()=0
- virtual void **PushPenWidth** (float width)=0
- virtual void **PopPenWidth** ()=0

Friends

- class **VGFont**
- class **DecoratorDevice**

5.15.1 Detailed Description

Generic platform independant drawing device. **VGDevice** (p. **??**) is a pure virtual class that declares the minimal set of methods required by the GGR (Guido Graphic Representation) objects to communicate their graphical operations. Implementations of **VGDevice** (p. **??**) derived classes must then be provided by client applications.

VGDevice (p. **??**) thus provides standard graphic functions (Lines, Arc, Rectangles, Polygons, Text), coordinate transformation (zoom / scaling), and symbolic music symbols handlers.

A **VGDevice** (p. **??**) can be seen as the association of:

- a drawing context, describing how these operations are performed, including pen and color state;
- a graphic 'device' ('port' or 'output'), describing 'where' the graphical operations are to be performed (these various outputs can be a screen, an offscreen pixmap, a printer, a file, a network stream, etc.);
- a 'link-to-guido' class allowing the client app to collect informations about the score and give them to the guido engine (see Set/GetSymbolMap).

To allow using a **VGDevice** (p. **??**) for double buffering mechanism, it also provides bit-block copy operations from one **VGDevice** (p. **??**) to another.

Each **VGDevice** (p. **??**) needs to be dynamicaly associated with external music and text **VGFont** (p. **??**) objects, using the appropriate SetFont() method. Here we have to repeat that Guido, for higher abstraction, makes a clear distinction between text characters and music symbols, although music symbols are generally glyphs in a music font.

5.15.2 Member Enumeration Documentation

5.15.2.1 enum VRasterOpMode

Raster operation modes (color fill, bit copy, etc.)

Enumerator:

kUnknown

kOpCopy

kOpAnd

kOpXOr

kOpInvert

kOpOr

5.15.2.2 enum VTextAlignMode

Text alignment modes.

Enumerator:

kAlignBase

kAlignBottom

kAlignTop

kAlignCenter

kAlignLeft

kAlignRight

kAlignBaseLeft

5.15.3 Constructor & Destructor Documentation

5.15.3.1 virtual ∼VGDevice ( ) `[inline, virtual]`

5.15.4 Member Function Documentation

5.15.4.1 virtual bool IsValid ( ) const `[pure virtual]`

Returns the ability of the current VGdevice to be drawn into.

5.15.4.2 virtual bool BeginDraw ( ) `[pure virtual]`

Prepares the device's context before a set of drawing operations and saves the current one (like the previous SaveDC() method). This method should be used before every set of drawing operation.

5.15.4.3 virtual void EndDraw ( ) `[pure virtual]`

Restores the device's context after a set of drawing operations. and restore the previous one (like the previous RestoreDC() method). This method should be used after every set of drawing operation.

5.15.4.4 virtual void InvalidateRect (float *left*, float *top*, float *right*, float *bottom*) `[pure virtual]`

Invalidate a rectangle i.e. indicates the native graphic device that the corresponding rectangle needs to be refreshed.

5.15.4.5 virtual void MoveTo (float *x*, float *y*) `[pure virtual]`

Moves the current position to the point specified by (x,y).

5.15.4.6 virtual void LineTo (float *x*, float *y*) `[pure virtual]`

Draws a line from the current position up to, but not including, the point specified by (x,y).

5.15.4.7 virtual void Line (float *x1*, float *y1*, float *x2*, float *y2*) `[pure virtual]`

Draws a line from the position specified by (x1,y1) up to, but not including, the point specified by (x2,y2).

5.15.4.8 virtual void Frame (float *left*, float *top*, float *right*, float *bottom*) `[pure virtual]`

Draws a frame using the specified coordinates. The frame is outlined by using the current pen, but not filled.

5.15.4.9 virtual void Arc (float *left*, float *top*, float *right*, float *bottom*, float *startX*, float *startY*, float *endX*, float *endY*) `[pure virtual]`

Draws a counter-clockwise elliptical arc between the startX, startY, endX & endY co-ordinates and inside the [left, top, right, bottom] elliptical bounding box. The distance from these points to the middle of the ellipse is not important, only the angle is taken in account.

5.15.4.10 virtual void Triangle (float *x1,* float *y1,* float *x2,* float *y2,* float *x3,* float *y3*) `[pure virtual]`

Draws a triangle consisting of three points connected by straight lines. The triangle is NOT outlined but simply filled using the current fill color.

5.15.4.11 virtual void Polygon (const float ∗ *xCoords,* const float ∗ *yCoords,* int *count*) `[pure virtual]`

Draws a polygon consisting of count vertices connected by straight lines. The polygon is NOT outlined but simply filled using the current fill color.

5.15.4.12 virtual void Rectangle (float *left,* float *top,* float *right,* float *bottom*) `[pure virtual]`

Draws a rectangle. The rectangle is NOT outlined but simply filled using the current fill color.

5.15.4.13 virtual void SetMusicFont (const VGFont ∗ *font*) `[pure virtual]`

Selects the specified music **VGFont** (p. **??**) into the current **VGDevice** (p. **??**). Warning ! this method doesn't return the previously selected font anymore. Use **GetMusicFont()** (p. **??**) instead.

5.15.4.14 virtual const VGFont∗ GetMusicFont ( ) const `[pure virtual]`

Returns the currently selected music **VGFont** (p. **??**).

5.15.4.15 virtual void SetTextFont (const VGFont ∗ *font*) `[pure virtual]`

Selects the specified text **VGFont** (p. **??**) into the current **VGDevice** (p. **??**). Warning ! this method doesn't return the previously selected font anymore. Use **GetMusicFont()** (p. **??**) instead.

5.15.4.16 virtual const VGFont∗ GetTextFont ( ) const `[pure virtual]`

Returns the currently selected text **VGFont** (p. **??**).

5.15.4.17 virtual void selectfont (int) `[inline, virtual]`

Selects a font (only for SVG device).

5.15.4.18 virtual void SelectPen (const VGColor & *inColor,* float *witdh*) [pure virtual]

Creates a new **VGPen** (p. **??**) object with the specified **VGColor** (p. **??**) and width and selects it into the current **VGDevice** (p. **??**).

5.15.4.19 virtual void SelectFillColor (const VGColor & *c*) [pure virtual]

Creates a new solid brush (to paint the interiors of filled shapes) with the specified **VGColor** (p. **??**) and selects it into the current **VGDevice** (p. **??**). This method was previously called SelectBrush().

5.15.4.20 virtual void PushPen (const VGColor & *inColor,* float *inWidth*) [pure virtual]

Saves the current **VGPen** (p. **??**) and selects the new one using the specified **VGColor** (p. **??**) and width into the current **VGDevice** (p. **??**).

5.15.4.21 virtual void PopPen ( ) [pure virtual]

Restores the previous **VGPen** (p. **??**) object from the stack into the current **VGDevice** (p. **??**).

5.15.4.22 virtual void PushFillColor (const VGColor & *inColor*) [pure virtual]

Saves the current color brush and selects the new one using the specified **VGColor** (p. **??**) into the current **VGDevice** (p. **??**). This method was previously called PushBrush().

5.15.4.23 virtual void PopFillColor ( ) [pure virtual]

Restores the previous color brush from the stack into the current **VGDevice** (p. **??**). This method was previously called PopBrush().

5.15.4.24 virtual void SetRasterOpMode (VRasterOpMode *ROpMode*) [pure virtual]

Sets the current foreground mix mode. We use the foreground mix mode to combine pens and interiors of filled objects with the colors already on the device. The foreground mix mode defines how colors from the brush or pen and the colors in the existing image are to be combined. See enum VRasterOpMode above.

5.15.4.25 virtual VRasterOpMode GetRasterOpMode ( ) const [pure virtual]

Retrieves the foreground mix mode of the specified device. The mix mode specifies how the pen or interior color and the color already on the device are combined to yield a new color.

5.15.4.26 virtual bool CopyPixels (VGDevice ∗ *pSrcDC,* float *alpha* = -1.0) [pure virtual]

Copies the entire content (pixmap) of the pSrcDC source **VGDevice** (p. **??**) to the current device. The raster operation mode should be specified using **SetRasterOpMode()** (p. **??**). Default alpha (-1.0) means copying bits using their own transparency values, if any; if no alpha channel value is available, opaque copy (alpha = 1.0) is performed.

5.15.4.27 virtual bool CopyPixels (int *xDest,* int *yDest,* VGDevice ∗ *pSrcDC,* int *xSrc,* int *ySrc,* int *nSrcWidth,* int *nSrcHeight,* float *alpha* = -1.0) [pure virtual]

Makes the exact copy of the content (pixmap) of the specified rectangle from the pSrcDC source **VGDevice** (p. **??**) to the specified destination of the current device. The raster operation mode should be specified using **SetRasterOpMode()** (p. **??**). Default alpha (-1.0) means copying bits using their own transparency values, if any; if no alpha channel value is available, opaque copy (alpha = 1.0) is performed.

5.15.4.28 virtual bool CopyPixels (int *xDest,* int *yDest,* int *dstWidth,* int *dstHeight,* VGDevice ∗ *pSrcDC,* float *alpha* = -1.0) [pure virtual]

Copies a pixmap from a source rectangle into a destination rectangle, stretching or compressing the pixmap to fit the dimensions of the destination rectangle, if necessary. The method stretches or compresses the pixmap using the raster mode currently set in the destination VGdevice. Default alpha (-1.0) means copying bits using their own transparency values, if any; if no alpha channel value is available, opaque copy (alpha = 1.0) is performed.

5.15.4.29 virtual bool CopyPixels (int *xDest,* int *yDest,* int *dstWidth,* int *dstHeight,* VGDevice ∗ *pSrcDC,* int *xSrc,* int *ySrc,* int *nSrcWidth,* int *nSrcHeight,* float *alpha* = -1.0) [pure virtual]

Copies a pixmap from a source rectangle into a destination rectangle, stretching or compressing the pixmap to fit the dimensions of the destination rectangle, if necessary. The method stretches or compresses the pixmap using the raster mode currently set in the destination VGdevice. Default alpha (-1.0) means copying bits using their own transparency values, if any; if no alpha channel value is available, opaque copy (alpha = 1.0) is performed.

5.15.4.30 virtual void SetScale (float *x*, float *y*) `[pure virtual]`

Sets the scale factors of the current **VGDevice** (p. **??**) to the input values.

5.15.4.31 virtual void SetOrigin (float *x*, float *y*) `[pure virtual]`

Specifies which **VGDevice** (p. **??**) point (x,y) maps to the window origin (0,0).

5.15.4.32 virtual void OffsetOrigin (float *x*, float *y*) `[pure virtual]`

Offsets the current VGDevice's origin (see above).

5.15.4.33 virtual void LogicalToDevice (float ∗ *x*, float ∗ *y*) const `[pure virtual]`

Computes input coordinates to get their values in the current **VGDevice** (p. **??**) coordinate system.

5.15.4.34 virtual void DeviceToLogical (float ∗ *x*, float ∗ *y*) const `[pure virtual]`

Computes input **VGDevice** (p. **??**) coordinates to get their values outside its **VGDevice** (p. **??**) coordinate system.

5.15.4.35 virtual float GetXScale ( ) const `[pure virtual]`

GetXScale, GetYScale, GetXOrigin, GetYOrigin : get VGDevice's scaling/coordinate atributes.

5.15.4.36 virtual float GetYScale ( ) const `[pure virtual]`

5.15.4.37 virtual float GetXOrigin ( ) const `[pure virtual]`

5.15.4.38 virtual float GetYOrigin ( ) const `[pure virtual]`

5.15.4.39 virtual void NotifySize (int *inWidth*, int *inHeight*) `[pure virtual]`

Sets the size of the current **VGDevice** (p. **??**). Use this method to update the derived device's attributes and actual size.

5.15.4.40 virtual int GetWidth ( ) const `[pure virtual]`

Returns the width (set via NotifySize) of the current **VGDevice** (p. **??**).

5.15.4.41 virtual int GetHeight ( ) const [pure virtual]

Returns the height (set via NotifySize) of the current **VGDevice** (p. **??**).

5.15.4.42 virtual void DrawMusicSymbol (float *x,* float *y,* unsigned int *inSymbolID*) [pure virtual]

Writes the music symbol specified by the input inSymbolID at the specified location, using the currently selected music font, background color, and text color.

5.15.4.43 virtual void DrawString (float *x,* float *y,* const char ∗ *s,* int *inCharCount*) [pure virtual]

Writes the specified inCharCount number of text characters at the specified location, using the currently selected text font, background color, and text color.

5.15.4.44 virtual void SetFontColor (const VGColor & *inColor*) [pure virtual]

Sets the text/music color for the current **VGDevice** (p. **??**).

5.15.4.45 virtual VGColor GetFontColor ( ) const [pure virtual]

Returns the text/music color of the current **VGDevice** (p. **??**).

5.15.4.46 virtual void SetFontBackgroundColor (const VGColor & *inColor*) [pure virtual]

Sets the text/music background color for the current **VGDevice** (p. **??**).

5.15.4.47 virtual VGColor GetFontBackgroundColor ( ) const [pure virtual]

Returns the text/music background color of the current **VGDevice** (p. **??**).

5.15.4.48 virtual void SetFontAlign (unsigned int *inAlign*) [pure virtual]

Sets the text/music alignment mode of the current **VGDevice** (p. **??**). See enum VTextAlignMode above.

5.15.4.49 virtual unsigned int GetFontAlign ( ) const [pure virtual]

Returns the text/music alignment mode of the current **VGDevice** (p. **??**). See enum VTextAlignMode above.

5.15.4.50 virtual void SetDPITag (float *inDPI*) `[pure virtual]`

Sets the printing resolution of the current **VGDevice** (p. **??**).

5.15.4.51 virtual float GetDPITag ( ) const `[pure virtual]`

Returns the printing resolution of the current **VGDevice** (p. **??**).

5.15.4.52 virtual void∗ GetBitMapPixels ( ) `[pure virtual]`

Allows pixels operations and returns a pointer to the bitmap pixels.

5.15.4.53 virtual void ReleaseBitMapPixels ( ) `[pure virtual]`

Update bitmap pixels and ends pixels operations.

5.15.4.54 virtual const char∗ GetImageData (const char ∗& *outDataPtr,* int & *outLength*) `[pure virtual]`

Gives the current device data and returns the data associated mime type.

5.15.4.55 virtual void ReleaseImageData (const char ∗) const `[pure virtual]`

Release the pointer returned by GetImageData.

5.15.4.56 virtual VGSystem∗ getVGSystem ( ) const `[pure virtual]`

temporary hack - must be removed asap

5.15.4.57 virtual void∗ GetNativeContext ( ) const `[pure virtual]`

Exports all graphical data to an image file.

Returns the platform-specific device context object.

Referenced by VGFont::GetContext().

5.15.4.58 virtual void SelectPenColor (const VGColor & *inColor*) `[pure virtual]`

Creates a new **VGPen** (p. **??**) object with the specified **VGColor** (p. **??**).

5.15.4.59 virtual void SelectPenWidth (float *witdh*) `[pure virtual]`

Creates a new **VGPen** (p. **??**) object with the specified **VGColor** (p. **??**).

5.15.4.60 virtual void PushPenColor (const VGColor & *inColor*) [pure virtual]

5.15.4.61 virtual void PopPenColor ( ) [pure virtual]

5.15.4.62 virtual void PushPenWidth (float *width*) [pure virtual]

5.15.4.63 virtual void PopPenWidth ( ) [pure virtual]

5.15.5 Friends And Related Function Documentation

5.15.5.1 friend class VGFont [friend]

5.15.5.2 friend class DecoratorDevice [friend]

5.16 VGFont Class Reference

Generic pure virtual & device-independant font class.

Public Types

- enum { **kFontNone** = 0, **kFontBold** = 1, **kFontItalic** = 2, **kFontUnderline** = 4 }

  Font properties.

Public Member Functions

- virtual ∼**VGFont** ()
- virtual const char ∗ **GetName** () const =0

  Returns the current object's name (as a string)
- virtual int **GetSize** () const =0

  Returns the current object's size (as an int)
- virtual int **GetProperties** () const =0

  Returns the current object's property value(s) (see enum above)
- virtual void **GetExtent** (const char ∗s, int inCharCount, float ∗outWidth, float ∗outHeight, **VGDevice** ∗context) const =0
- virtual void **GetExtent** (unsigned char c, float ∗outWidth, float ∗outHeight, **VGDevice** ∗context) const =0

Protected Member Functions

- void ∗ **GetContext** (**VGDevice** ∗context) const

5.16.1 Detailed Description

Generic pure virtual & device-independant font class. This class replaces the previously defined GFontInfos class that was attached to a **VGDevice** (p. **??**). It declares the minimal set of necessary methods/attributes to use a font. Fonts can now be seen as independant objects to be created by the **VGSystem** (p. **??**) and then associated when necessary to the **VGDevice** (p. **??**).

5.16.2 Member Enumeration Documentation

5.16.2.1 anonymous enum

Font properties.

Enumerator:

kFontNone

kFontBold

kFontItalic

kFontUnderline

5.16.3 Constructor & Destructor Documentation

5.16.3.1 virtual ∼VGFont ( ) `[inline, virtual]`

5.16.4 Member Function Documentation

5.16.4.1 virtual const char∗ GetName ( ) const `[pure virtual]`

Returns the current object's name (as a string)

5.16.4.2 virtual int GetSize ( ) const `[pure virtual]`

Returns the current object's size (as an int)

5.16.4.3 virtual int GetProperties ( ) const `[pure virtual]`

Returns the current object's property value(s) (see enum above)

5.16.4.4 virtual void GetExtent (const char ∗ *s,* int *inCharCount,* float ∗ *outWidth,* float ∗ *outHeight,* VGDevice ∗ *context*) const `[pure virtual]`

Computes the width and height of the input string using the current font capabilities in the input **VGDevice** (p. **??**)

5.16.4.5 virtual void GetExtent (unsigned char *c,* float ∗ *outWidth,* float ∗ *outHeight,* VGDevice ∗ *context*) const `[pure virtual]`

Computes the width and height of the input character using the current font capabilities in the input **VGDevice** (p. **??**)

5.16.4.6 void∗ GetContext (VGDevice ∗ *context*) const `[inline, protected]`

References VGDevice::GetNativeContext().

5.17 VGPen Class Reference

Generic class to manipulate device independant pens.

Public Member Functions

- **VGPen** ()
- void **Set** (const **VGColor** &inColor, float inWidth)
- void **Set** (const **VGColor** &inColor)
- void **Set** (float inWidth)

Public Attributes

- **VGColor mColor**
- float **mWidth**

5.17.1 Detailed Description

Generic class to manipulate device independant pens.

5.17.2 Constructor & Destructor Documentation

5.17.2.1 VGPen ( ) `[inline]`

5.17.3 Member Function Documentation

5.17.3.1 void Set (const VGColor & *inColor,* float *inWidth*) `[inline]`

References mColor, and mWidth.

5.17.3.2 void Set (const VGColor & *inColor*) `[inline]`

References mColor.

5.17.3.3 void Set (float *inWidth*) [inline]

References mWidth.

5.17.4 Member Data Documentation

5.17.4.1 VGColor mColor

Referenced by Set().

5.17.4.2 float mWidth

Referenced by Set().

5.18 VGSystem Class Reference

Generic pure virtual class for manipulating platform independant drawing devices and fonts.

Public Member Functions

- virtual ∼**VGSystem** ()
- virtual **VGDevice** ∗ **CreateDisplayDevice** ()=0
- virtual **VGDevice** ∗ **CreateMemoryDevice** (int inWidth, int inHeight)=0
- virtual **VGDevice** ∗ **CreateMemoryDevice** (const char ∗inPath)=0
- virtual **VGDevice** ∗ **CreatePrinterDevice** ()=0

  *Creates and returns a pointer to a new printer **VGDevice** (*p. **??***).*
- virtual **VGDevice** ∗ **CreateAntiAliasedMemoryDevice** (int inWidth, int inHeight)=0
- virtual const **VGFont** ∗ **CreateVGFont** (const char ∗faceName, int size, int properties) const =0

  *Creates and returns a pointer to a new **VGFont** (*p. **??***) using the specified parameters.*

5.18.1 Detailed Description

Generic pure virtual class for manipulating platform independant drawing devices and fonts. A **VGSystem** (p. **??**) object (for Virtual Guido drawing System) is the highest graphical abstraction whose child-object can be used by a client app to obtain a platform dependant graphical system able to perform guido drawing or printing operations (like double-buffered display on the current screen or printing). To do so, the client app simply needs to intantiate an object of the appropriate platform dependant **VGSystem** (p. **??**) derived class using the correct platform dependant parameters.

A **VGSystem** (p. **??**) object deals mainly with two kinds of graphical objects:

- **VGDevice** (p. **??**) objects, which can be display devices, memory devices or printer devices;
- **VGFont** (p. **??**) objects, used to describe device independant fonts (for music or text).

5.18.2 Constructor & Destructor Documentation

5.18.2.1 virtual ∼VGSystem ( ) `[inline, virtual]`

5.18.3 Member Function Documentation

5.18.3.1 virtual VGDevice∗ CreateDisplayDevice ( ) `[pure virtual]`

Creates and returns a pointer to a new display **VGDevice** (p. **??**) which can be used to draw directly on the screen.

5.18.3.2 virtual VGDevice∗ CreateMemoryDevice (int *inWidth,* int *inHeight*) `[pure virtual]`

Creates and returns a pointer to a new memory **VGDevice** (p. **??**) compatible with the application's current screen. This device can be used to draw into a bitmap.

5.18.3.3 virtual VGDevice∗ CreateMemoryDevice (const char ∗ *inPath*) `[pure virtual]`

Creates and returns a pointer to a new memory **VGDevice** (p. **??**) compatible with the file (pixmap) located at the specified path.

5.18.3.4 virtual VGDevice∗ CreatePrinterDevice ( ) `[pure virtual]`

Creates and returns a pointer to a new printer **VGDevice** (p. **??**).

5.18.3.5 virtual VGDevice∗ CreateAntiAliasedMemoryDevice (int *inWidth,* int *inHeight*) `[pure virtual]`

Creates and returns a pointer to a new memory **VGDevice** (p. **??**) compatible with the application's current screen and using anti-aliasing capabilities. This device can be used to draw into a bitmap.

5.18.3.6 virtual const VGFont∗ CreateVGFont (const char ∗ *faceName,* int *size,* int *properties*) const `[pure virtual]`

Creates and returns a pointer to a new **VGFont** (p. **??**) using the specified parameters.

Chapter 6

File Documentation

6.1 globaldoc.doxygen File Reference

6.2 GUIDO2Midi.h File Reference

Classes

- struct **Guido2MidiParams**

  *The **GuidoInitDesc** (p. **??**) data structure contains all information required by **GuidoInit()** (p. **??**)*

Typedefs

- typedef struct **Guido2MidiParams Guido2MidiParams**

  *The **GuidoInitDesc** (p. **??**) data structure contains all information required by **GuidoInit()** (p. **??**)*

Functions

- **GuidoErrCode GuidoAR2MIDIFile** (const **ARHandler** ar, const char ∗filename, const **Guido2MidiParams** ∗params)

  Export to a MIDI file.

6.3 GUIDOEngine.h File Reference

Classes

- struct **GuidoInitDesc**

  *The **GuidoInitDesc** (p. **??**) data structure contains all information required by **GuidoInit()** (p. **??**)*

- struct **GPaintStruct**

  A structure to keep information about clipping and redrawing regions.

- struct **GuidoOnDrawDesc**

  *Contains all graphic-related information required by **GuidoOnDraw()** (p. **??**)*

- struct **GuidoDate**
- struct **GuidoLayoutSettings**
- struct **GuidoPageFormat**

Typedefs

- typedef struct NodeAR ∗ **ARHandler**
- typedef struct NodeGR ∗ **GRHandler**
- typedef struct NodeAR ∗ **CARHandler**
- typedef struct NodeGR ∗ **CGRHandler**
- typedef struct **GuidoLayoutSettings GuidoLayoutSettings**

Enumerations

- enum {

  kNoBB, **kPageBB**, **kSystemsBB** = 2, **kSystemsSliceBB** = 4,

  kStavesBB = 8, **kMeasureBB** = 0x10, **kEventsBB** = 0x20 }

  Bounding boxes drawing control constants.

- enum **GuidoErrCode** {

  guidoNoErr = 0, **guidoErrParse** = -1, **guidoErrMemory** = -2, **guidoErrFileAccess** = -3,

  guidoErrUserCancel = -4, **guidoErrNoMusicFont** = -5, **guidoErrNoTextFont** = -6, **guidoErrBadParameter** = -7,

  guidoErrInvalidHandle = -8, **guidoErrNotInitialized** = -9, **guidoErrActionFailed** = -10 }

  The guido error codes list.

- enum { **kAutoDistrib** = 1, **kAlwaysDistrib** = 2, **kNeverDistrib** = 3 }

Functions

- **GuidoErrCode GuidoInit** (**GuidoInitDesc** ∗desc)
- void **GuidoShutdown** ()
- **GuidoErrCode GuidoParseFile** (const char ∗filename, **ARHandler** ∗ar)
- **GuidoErrCode GuidoParseString** (const char ∗str, **ARHandler** ∗ar)
- **GuidoErrCode GuidoAR2GR** (**ARHandler** ar, const **GuidoLayoutSettings** ∗settings, **GRHandler** ∗gr)
- **GuidoErrCode GuidoAR2RProportional** (**ARHandler** ar, int width, int height, const **GuidoDate** &start, const **GuidoDate** &end, bool drawdur, **VGDevice** ∗dev)
- **GuidoErrCode GuidoUpdateGR** (**GRHandler** gr, const **GuidoLayoutSettings** ∗settings)
- void **GuidoFreeAR** (**ARHandler** ar)
- void **GuidoFreeGR** (**GRHandler** gr)
- const char ∗ **GuidoGetErrorString** (**GuidoErrCode** errCode)
- int **GuidoGetParseErrorLine** ()
- void **GuidoGetDefaultLayoutSettings** (**GuidoLayoutSettings** ∗settings)
- int **GuidoCountVoices** (**CARHandler** inHandleAR)

  Gives the number of score pages of the graphic representation.

- int **GuidoGetPageCount** (**CGRHandler** inHandleGR)

  Gives the number of score pages of the graphic representation.

- int **GuidoGetSystemCount** (**CGRHandler** inHandleGR, int page)

  Gives the number of systems on a given page.

- **GuidoErrCode GuidoDuration** (**CGRHandler** inHandleGR, **GuidoDate** ∗date)

  Returns the music duration of a score.

- int **GuidoFindEventPage** (**CGRHandler** inHandleGR, const **GuidoDate** &date)

  Finds the page which has an event (note or rest) at a given date.

- int **GuidoFindPageAt** (**CGRHandler** inHandleGR, const **GuidoDate** &date)

  Finds the page which contain a given date.

- **GuidoErrCode GuidoGetPageDate** (**CGRHandler** inHandleGR, int pageNum, **GuidoDate** ∗date)

  Gives the time location of a Page.

- **GuidoErrCode GuidoOnDraw** (**GuidoOnDrawDesc** ∗desc)

  Draws one page of score into a graphic device.

- **GuidoErrCode GuidoSVGExport** (const **GRHandler** handle, int page, std::ostream &out, const char ∗fontfile)

  Exports one page of score to SVG.

- **GuidoErrCode GuidoAbstractExport** (const **GRHandler** handle, int page, std::ostream &out)

Exports an abstract representation of GUIDO draw commands.

- **GuidoErrCode GuidoBinaryExport** (const **GRHandler** handle, int page, std::ostream &out)

  Exports an representation of GUIDO draw commands in a data-reduced dsl.

- void **GuidoDrawBoundingBoxes** (int bbMap)

  Control bounding boxes drawing.

- int **GuidoGetDrawBoundingBoxes** ()

  Gives bounding boxes drawing state.

- void **GuidoGetPageFormat** (**CGRHandler** inHandleGR, int pageNum, **GuidoPageFormat** ∗format)

  Gives a score page format.

- void **GuidoSetDefaultPageFormat** (const **GuidoPageFormat** ∗format)

  Sets the default score page format.

- void **GuidoGetDefaultPageFormat** (**GuidoPageFormat** ∗format)

  Gives the default score page format.

- float **GuidoUnit2CM** (float val)

  Converts internal Guido units into centimeters.

- float **GuidoCM2Unit** (float val)

  Converts centimeters into internal Guido units.

- float **GuidoUnit2Inches** (float val)

  Converts internal Guido units into inches.

- float **GuidoInches2Unit** (float val)

  Converts inches into internal Guido units.

- **GuidoErrCode GuidoResizePageToMusic** (**GRHandler** inHandleGR)

  Resize the page sizes to the music size.

- void **GuidoGetVersionNums** (int ∗major, int ∗minor, int ∗sub)

  Gives the library version number as three integers.

- const char ∗ **GuidoGetVersionStr** ()

  Gives the library version number as a string.

- **GuidoErrCode GuidoCheckVersionNums** (int major, int minor, int sub)

  Checks a required library version number.

- float **GuidoGetLineSpace** ()

  Gives the distance between two staff lines.

- **GuidoErrCode GuidoMarkVoice** (**ARHandler** inHandleAR, int voicenum, const **GuidoDate** &date, const **GuidoDate** &duration, unsigned char red, unsigned char green, unsigned char blue)

  Gives a color to all notes of a voice between a given time interval.

- **GuidoErrCode GuidoSetSymbolPath** (**ARHandler** inHandleAR, const std::vector< std::string > &inPaths)

  Makes the correspondance between an ARMusic and a path.

- **GuidoErrCode GuidoGetSymbolPath** (const **ARHandler** inHandleAR, std::vector< std::string > &inPathVector)

  Returns the path corresponding to an AR.

- void **AddGGSOutput** (const char ∗s)
- void **AddGuidoOutput** (const char ∗s)

6.3.1 Typedef Documentation

6.3.1.1 typedef struct NodeAR∗ ARHandler

6.3.1.2 typedef struct NodeGR∗ GRHandler

6.3.1.3 typedef struct NodeAR∗ CARHandler

6.3.1.4 typedef struct NodeGR∗ CGRHandler

6.3.1.5 typedef struct GuidoLayoutSettings GuidoLayoutSettings

Settings for the graphic score layout.

6.3.2 Enumeration Type Documentation

6.3.2.1 anonymous enum

Bounding boxes drawing control constants.

Enumerator:

kNoBB

kPageBB

kSystemsBB

kSystemsSliceBB

kStavesBB

kMeasureBB

kEventsBB

6.3.2.2 anonymous enum

Enumerator:

kAutoDistrib

kAlwaysDistrib

kNeverDistrib

6.3.3 Function Documentation

6.3.3.1 void AddGGSOutput (const char ∗ *s*)

6.3.3.2 void AddGuidoOutput (const char ∗ *s*)

6.4 GUIDOExport.h File Reference

Defines

- #define **class_export** class
- #define **GUIDOAPI**(type) type

6.4.1 Define Documentation

6.4.1.1 #define class_export class

6.4.1.2 #define GUIDOAPI(*type*) type

6.5 GUIDOFactory.h File Reference

Typedefs

- typedef void ∗ **ARFactoryHandler**

Functions

- **GuidoErrCode GuidoFactoryOpen** (**ARFactoryHandler** ∗outFactory)
  Opens the Guido Factory.
- void **GuidoFactoryClose** (**ARFactoryHandler** inFactory)
  Closes the Guido Factory.

- **GuidoErrCode GuidoFactoryOpenMusic** (**ARFactoryHandler** inFactory)

  Creates and opens a new music score.
- **ARHandler GuidoFactoryCloseMusic** (**ARFactoryHandler** inFactory)

  Closes the current music score.
- **GuidoErrCode GuidoFactoryOpenVoice** (**ARFactoryHandler** inFactory)

  Creates and opens a new voice.
- **GuidoErrCode GuidoFactoryCloseVoice** (**ARFactoryHandler** inFactory)

  Closes the current voice.
- **GuidoErrCode GuidoFactoryOpenChord** (**ARFactoryHandler** inFactory)

  Creates and open a new chord.
- **GuidoErrCode GuidoFactoryCloseChord** (**ARFactoryHandler** inFactory)

  Closes the current chord.
- **GuidoErrCode GuidoFactoryInsertCommata** (**ARFactoryHandler** inFactory)

  Begins a new chord note commata.
- **GuidoErrCode GuidoFactoryOpenEvent** (**ARFactoryHandler** inFactory, const char ∗inEventName)

  Creates and opens a new event (note or rest).
- **GuidoErrCode GuidoFactoryCloseEvent** (**ARFactoryHandler** inFactory)

  Closes the current event.
- **GuidoErrCode GuidoFactoryAddSharp** (**ARFactoryHandler** inFactory)

  Adds a sharp to the current event.
- **GuidoErrCode GuidoFactoryAddFlat** (**ARFactoryHandler** inFactory)

  Add a flat to the current event.
- **GuidoErrCode GuidoFactorySetEventDots** (**ARFactoryHandler** inFactory, int dots)

  Sets the number of dots the current event.
- **GuidoErrCode GuidoFactorySetEventAccidentals** (**ARFactoryHandler** inFactory, int accident)

  Sets the accidentals of the current event.
- **GuidoErrCode GuidoFactorySetOctave** (**ARFactoryHandler** inFactory, int octave)

  Sets the register (octave) of the current event.
- **GuidoErrCode GuidoFactorySetDuration** (**ARFactoryHandler** inFactory, int numerator, int denominator)

Sets the duration of the current event.

- **GuidoErrCode GuidoFactoryOpenTag** (**ARFactoryHandler** inFactory, const char ∗name, long tagID)
- **GuidoErrCode GuidoFactoryOpenRangeTag** (**ARFactoryHandler** inFactory, const char ∗name, long tagID)
- **GuidoErrCode GuidoFactoryEndTag** (**ARFactoryHandler** inFactory)

  Indicates the end of a range tag.
- **GuidoErrCode GuidoFactoryCloseTag** (**ARFactoryHandler** inFactory)

  Closes the current tag.
- **GuidoErrCode GuidoFactoryAddTagParameterString** (**ARFactoryHandler** inFactory, const char ∗val)

  Adds a new string parameter to the current tag.
- **GuidoErrCode GuidoFactoryAddTagParameterInt** (**ARFactoryHandler** inFactory, int val)

  Adds a new integer parameter to the current tag.
- **GuidoErrCode GuidoFactoryAddTagParameterFloat** (**ARFactoryHandler** inFactory, double val)

  Adds a new floating-point parameter to the current tag.
- **GuidoErrCode GuidoFactorySetParameterName** (**ARFactoryHandler** inFactory, const char ∗name)

  Defines the name (when applicable) of the last added tag-parameter.
- **GuidoErrCode GuidoFactorySetParameterUnit** (**ARFactoryHandler** inFactory, const char ∗unit)

  Defines the unit of the last added tag-parameter.

6.6 GUIDOParse.h File Reference

Functions

- GuidoParser ∗ **GuidoOpenParser** ()

  Creates a new parser.
- **GuidoErrCode GuidoCloseParser** (GuidoParser ∗p)

  Close a guido parser and releases all the associated ressources.
- const char ∗ **GuidoGetStream** (GuidoStream ∗gStream)

  returns the string of the GuidoStream

- **ARHandler GuidoFile2AR** (GuidoParser ∗p, const char ∗file)

  Parse a file and create the corresponding AR.

- **ARHandler GuidoString2AR** (GuidoParser ∗p, const char ∗str)

  Parse a string and create the corresponding AR.

- **ARHandler GuidoStream2AR** (GuidoParser ∗p, GuidoStream ∗stream)

  Parse a GuidoStream and create the corresponding AR.

- **GuidoErrCode GuidoParserGetErrorCode** (GuidoParser ∗p, int &line, int &col, const char ∗∗msg)

  Get the error syntax line/column.

- GuidoStream ∗ **GuidoOpenStream** ()

  Open a guido stream.

- **GuidoErrCode GuidoCloseStream** (GuidoStream ∗s)

  Close a guido stream.

- **GuidoErrCode GuidoWriteStream** (GuidoStream ∗s, const char ∗str)

  Write data to the stream.

- **GuidoErrCode GuidoResetStream** (GuidoStream ∗s)

  Erase all stream content in order to reuse it.

6.7 GUIDOPianoRoll.h File Reference

Classes

- struct **LimitParams**

Typedefs

- typedef struct **LimitParams LimitParams**

Enumerations

- enum **PianoRollType** { **kSimplePianoRoll**, **kTrajectoryPianoRoll** }

  PianoRollType.

Functions

- PianoRoll ∗ **GuidoAR2PianoRoll** (**PianoRollType** type, **ARHandler** arh)

  Creates a new piano roll from AR, corresponding to type : simplePianoRoll -> basic piano roll trajectoryPianoRoll -> every event is graphically linked to the previous one.

- PianoRoll ∗ **GuidoMidi2PianoRoll** (**PianoRollType** type, const char ∗midiFileName)

  Creates a new piano roll from Midi, corresponding to type : simplePianoRoll -> basic piano roll trajectoryPianoRoll -> every event is graphically linked to the previous one.

- **GuidoErrCode GuidoDestroyPianoRoll** (PianoRoll ∗pr)

  Destroys a guido piano roll and releases all the associated ressources.

- **GuidoErrCode GuidoPianoRollSetLimits** (PianoRoll ∗pr, **LimitParams** limitParams)

  Sets limits to a piano roll (start/end date, lower/higher pitch)

- **GuidoErrCode GuidoPianoRollEnableKeyboard** (PianoRoll ∗pr, bool enabled)

  Enables keyboard or not (not enabled by default)

- **GuidoErrCode GuidoPianoRollGetKeyboardWidth** (PianoRoll ∗pr, int height, float &keyboardWidth)

  Gets the piano roll keyboard width.

- **GuidoErrCode GuidoPianoRollEnableAutoVoicesColoration** (PianoRoll ∗pr, bool enabled)

  Enables or not the automatic voices coloration (not enabled by default) (not for a midi rendering) // REM: ir If a color is manually set with GuidoPianoRollSetColorToVoice, automatic color will not be applied for this voice.

- **GuidoErrCode GuidoPianoRollSetRGBColorToVoice** (PianoRoll ∗pr, int voiceNum, int r, int g, int b, int a)

  Sets a RGB color to a voice (first voice is number 1) (black by default)

- **GuidoErrCode GuidoPianoRollSetHtmlColorToVoice** (PianoRoll ∗pr, int voiceNum, long color)

  Sets a html color to a voice (first voice is number 1) (black by default)

- **GuidoErrCode GuidoPianoRollEnableMeasureBars** (PianoRoll ∗pr, bool enabled)

  Enables or not measure bars (false by default)

- **GuidoErrCode GuidoPianoRollSetPitchLinesDisplayMode** (PianoRoll ∗pr, int mode)

  Sets the pitch lines display mode (automatic by default). Use Pitch lines display mode constants to pick lines which will be be displayed. Example : "kCLine + kGLine" will displayed C and G line. "kNoLine" doesn't display any line. "kAutoLines" adjust line display according to piano roll pitch range (automatic possibilities : no line, C line, C and G line, chromatic scale, diatonic scale);.

- **GuidoErrCode GuidoPianoRollGetMap** (PianoRoll ∗pr, int width, int height, **Time2GraphicMap** &outmap)

  Gets the piano roll map.

- **GuidoErrCode GuidoPianoRollOnDraw** (PianoRoll ∗pr, int width, int height, **VGDevice** ∗dev)

  *Draw the piano roll on a **VGDevice** (*p. **??***).*

Variables

- const int **kCLine** = 1

  Pitch lines display mode.

- const int **kCSharpLine** = 1<<1
- const int **kDLine** = 1<<2
- const int **kDSharpLine** = 1<<3
- const int **kELine** = 1<<4
- const int **kFLine** = 1<<5
- const int **kFSharpLine** = 1<<6
- const int **kGLine** = 1<<7
- const int **kGSharpLine** = 1<<8
- const int **kALine** = 1<<9
- const int **kASharpLine** = 1<<10
- const int **kBLine** = 1<<11
- const int **kAutoLines** = 0
- const int **kNoLine** = -1

6.7.1 Typedef Documentation

6.7.1.1 typedef struct LimitParams LimitParams

6.7.2 Enumeration Type Documentation

6.7.2.1 enum PianoRollType

PianoRollType.

Enumerator:

kSimplePianoRoll

kTrajectoryPianoRoll

6.7.3 Variable Documentation

6.7.3.1 const int kCLine = 1

Pitch lines display mode.

6.7.3.2 const int kCSharpLine = 1<<1

6.7.3.3 const int kDLine = 1<<2

6.7.3.4 const int kDSharpLine = 1<<3

6.7.3.5 const int kELine = 1<<4

6.7.3.6 const int kFLine = 1<<5

6.7.3.7 const int kFSharpLine = 1<<6

6.7.3.8 const int kGLine = 1<<7

6.7.3.9 const int kGSharpLine = 1<<8

6.7.3.10 const int kALine = 1<<9

6.7.3.11 const int kASharpLine = 1<<10

6.7.3.12 const int kBLine = 1<<11

6.7.3.13 const int kAutoLines = 0

6.7.3.14 const int kNoLine = -1

6.8 GUIDOScoreMap.h File Reference

Classes

- struct **GuidoElementInfos**
- class **TimeSegment**

  a time segment definition and operations

- class **MapCollector**
- class **RectInfos**
- class **TimeMapCollector**

Typedefs

- typedef std::vector< std::pair< **TimeSegment**, FloatRect > > **Time2GraphicMap**
- typedef std::pair< FloatRect, **RectInfos** > **MapElement**

Enumerations

- enum **GuidoeElementSelector** {
  kGuidoPage, **kGuidoSystem**, **kGuidoSystemSlice**, **kGuidoStaff**,
  kGuidoBar, **kGuidoEvent**, **kGuidoScoreElementEnd** }
- enum **GuidoElementType** {
  kNote = 1, **kRest**, **kEmpty**, **kBar**,
  kRepeatBegin, **kRepeatEnd**, **kStaff**, **kSystemSlice**,
  kSystem, **kPage** }

Functions

- std::ostream & **operator**<< (std::ostream &out, const **GuidoDate** &d)
- std::ostream & **operator**<< (std::ostream &out, const **TimeSegment** &s)
- **GuidoErrCode GuidoGetMap** (**CGRHandler** gr, int pagenum, float width, float height, **GuidoeElementSelector** sel, **MapCollector** &f)

  Retrieves the graphic to time mapping.

- **GuidoErrCode GuidoGetPageMap** (**CGRHandler** gr, int pagenum, float w, float h, **Time2GraphicMap** &outmap)

  Retrieves a guido page graphic to time mapping.

- **GuidoErrCode GuidoGetStaffMap** (**CGRHandler** gr, int pagenum, float w, float h, int staff, **Time2GraphicMap** &outmap)

  Retrieves a guido staff graphic to time mapping.

- **GuidoErrCode GuidoGetVoiceMap** (**CGRHandler** gr, int pagenum, float w, float h, int voice, **Time2GraphicMap** &outmap)

  Retrieves a guido voice graphic to time mapping.

- **GuidoErrCode GuidoGetSystemMap** (**CGRHandler** gr, int pagenum, float w, float h, **Time2GraphicMap** &outmap)

  Retrieves a guido system graphic to time mapping.

- bool **GuidoGetTime** (const **GuidoDate** &date, const **Time2GraphicMap** map, **TimeSegment** &t, FloatRect &r)

  Retrieves a time segment and the associated graphic segment in a mapping.

- bool **GuidoGetPoint** (float x, float y, const **Time2GraphicMap** map, **TimeSegment** &t, FloatRect &r)

Retrieves a time segment and the associated graphic segment in a mapping.

- **GuidoErrCode GuidoGetSVGMap** (**GRHandler** gr, int pagenum, **GuidoeElementSelector** sel, std::vector< **MapElement** > &outMap)

  Retrieves the graphic to time mapping corresponding to the SVG output.

- **GuidoErrCode GuidoGetTimeMap** (**CARHandler** gr, **TimeMapCollector** &f)

  Retrieves the rolled to unrolled time mapping.

6.9 samples.doxygen File Reference

6.10 VGColor.h File Reference

Classes

- class **VGColor**

  Generic class to manipulate device independant colors.

Defines

- #define **ALPHA_TRANSPARENT** 0
- #define **ALPHA_OPAQUE** 255

Functions

- std::ostream & **operator**<< (std::ostream &out, const **VGColor** &c)

6.11 VGDevice.h File Reference

Classes

- class **VGDevice**

  Generic platform independant drawing device.

6.12 VGFont.h File Reference

Classes

- class **VGFont**

Generic pure virtual & device-independant font class.

6.13 VGPen.h File Reference

Classes

- class **VGPen**

  Generic class to manipulate device independant pens.

6.14 VGSystem.h File Reference

Classes

- class **VGSystem**

  Generic pure virtual class for manipulating platform independant drawing devices and fonts.