JGuido Library: Java API

Technical report n° 2013-2

Fober, D., Pachet, F.

SONY Computer Science Laboratory Paris
6 rue Amyot, 75005 Paris

July 2013

Executive Summary

This Technical Report presents the JGuido Library, the score rendering feature for both cases of rendering the score on-the-fly and a posteriori.
The GUIDO Engine Library is a generic, portable library and C/C++ API for the graphical rendering of musical scores. The JGuido library is based on the GUIDO Music Notation Format as the underlying data format and takes into account the conventional music notation system.
This report gives an overview of the solutions to support the GUIDO Engine in a Java Virtual Machine [Java VM]. Next, it gives an overview of the Guido Java Interface, which is similar to the C/C++ API. The full documentation is given in Appendix.
The JGuido library has been included in MIROR-IMPRO and MIROR-COMPO software developed by Sony Computer Science Laboratory Paris, and released in August 2013. The software itself can be downloaded on request, by contacting the authors here: http://www.csl.sony.fr/contact.php

Acknowledgments

The work described in this report forms part of the European project MIROR **M**usical **I**nteraction **R**elying **O**n **R**eflexion http://www.mirorproject.eu/ , co-funded by the European Community under the Information and Communication Technologies (ICT) theme of the Seventh Framework Programme. (FP7/2007-2013). Grant agreement n° 258338

GUIDO Engine Library
The Java Native Interface

D. Fober
<fober@grame.fr>

1 Introduction

The GUIDO Engine Library[1] is a generic, portable library and C/C++ API for the graphical rendering of musical scores. The library is based on the GUIDO Music Notation Format as the underlying data format. It takes account of the conventional music notation system. The engine provides efficient automatic music score layout [9] although exact score formatting could be specified at notation level.

The GUIDO Music Notation Format [GMN] is a formal language for score level music representation [6]. It is a plain-text, i.e. readable and platform independent format capable of representing all information contained in conventional musical scores. The basic GUIDO Format is very flexible and can be easily extended and adapted to capture a wide variety of musical features beyond conventional musical notation (CMN). The GUIDO design is strongly influenced by the objective to facilitate an adequate representation of musical material, from tiny motives up to complex symphonic scores. GUIDO is a general purpose musical notation format; the intended range of application includes notation software, compositional and analytical systems and tools, performance systems, and large musical databases.

The GUIDO Engine takes place in the landscape of music notation systems where only a few systems provide similar features [1, 5, 7, 8]. Compared to the other systems, the GUIDO Engine is the only framework that can be embedded into stand alone applications. In order to extend the range of potential applications, a Java Native Interface has been designed. This document presents the issues and the solution proposed to support the GUIDO Engine in a Java Virtual Machine [Java VM]. Next, it gives an overview of the Guido Java Interface, which is similar to the C/C++ API. The full documentation is given in appendix.

2 The GUIDO Java Native Interface

The main issue to support the GUIDO Engine in a Java Virtual Machine concerns the graphic environment and the way to draw on a graphic device. Actually and at low level, every software component makes use of native graphic device to draw on screen or to any other graphic device (a printer for example). These native devices depend on the host operating system (CGContext on MacOS X, GDI or GDIPlus on Windows, X Window, GTK, Cairo,... on Linux) and although the global features are generally equivalent, the implementation details makes the graphic source code platform dependent and non-portable.

Since a Java Virtual Machine is platform independent, it requires to build an abstract layer over the native devices. With the Java language, this abstract layer is named AWT (Abstract Window Toolkit) and provides platform independence. However, since the GUIDO Engine is also platform independent, it provides its own graphic device abstraction, named VGDevice (standing for Virtual Graphic Device). Thus the problem with the GUIDO Java Native Interface is to make these two abstract layers living together.

[1] http://guidolib.sourceforge.net

2.1 The AWT native interface

The Java 2 upgrade release introduces the *"AWT Native Interface"*, which is an official way to obtain all the information needed about the peer of a Java Canvas so that one can draw directly to the Canvas from a native code library using the drawing routines provided by the operating system.

The first step in hooking up a native rendering library to a Java Canvas is to define a new class that extends Canvas and overrides the paint method. The Java system routes all drawing operations for a Canvas object through the paint method, as it does for all other GUI objects.

The new paint method, to be implemented in the native rendering library, must be declared as `public native void`, and the native library itself is loaded at runtime by including a call to `System.loadLibrary( "myRenderingLib")` in the static block of the class. *"myRenderingLib"* is the name used for the native shared library.

Here's a simple example of such a class:

```
import java.awt.*;
import java.awt.event.*;

public class MyCanvas extends Canvas {

    static {
        System.loadLibrary("myRenderingLib");
    }
    public native void paint(Graphics g);

    public static void main(String[] args) {
        Frame f = new Frame();
        f.setBounds(0, 0, 500, 110);
        f.add( new MyCanvas() );
        f.addWindowListener( new WindowAdapter() {
            public void windowClosing(WindowEvent ev) { System.exit(0); }
        } );
        f.show();
    }
}
```

The next step is to run the `javah` tool on the `MyCanvas` class file above to generate a C/C++ header file that describes the interface to the native paint method that Java expects to be used. `javah` is a standard tool included with the Java 2 SDK.

The final step is to write the native rendering method, with an interface that conforms to the header file that `javah` generated, and build it as a standard shared library (called myRenderingLib in the above example) by linking it against the jre awt library. This code will call back to the Java VM to get the drawing surface information it needs to access the `MyCanvas` peer. Once this information is available, the code can draw directly to `MyCanvas` using standard drawing routines supplied by the underlying operating system.

Here is sample source code for a native paint method designed for use in a Solaris X11-based drawing environment and a Java VM where the AWT Native Interface is present:

```
#include "MyCanvas.h"
#include "jawt_md.h"

/*
 * Class:     MyCanvas
```

```
 * Method:    paint
 * Signature: (Ljava/awt/Graphics;)V
 */
JNIEXPORT void JNICALL Java_MyCanvas_paint (JNIEnv* env, jobject canvas, jobject graphics)
{
    JAWT awt;
    JAWT_DrawingSurface* ds;
    JAWT_DrawingSurfaceInfo* dsi;
    JAWT_X11DrawingSurfaceInfo* dsi_x11;
    jboolean result;
    jint lock;
    GC gc;

    const char * testString = " rendered from native code ";

    /* Get the AWT */
    awt.version = JAWT_VERSION_1_3;
    if (JAWT_GetAWT(env, &awt) == JNI_FALSE) {
        printf("AWT Not found\n");
        return;
    }

    /* Get the drawing surface */
    ds = awt.GetDrawingSurface(env, canvas);
    if (ds == NULL) {
        printf("NULL drawing surface\n");
        return;
    }

    /* Lock the drawing surface */
    lock = ds->Lock(ds);
    if((lock & JAWT_LOCK_ERROR) != 0) {
        printf("Error locking surface\n");
        awt.FreeDrawingSurface(ds);
        return;
    }

    /* Get the drawing surface info */
    dsi = ds->GetDrawingSurfaceInfo(ds);
    if (dsi == NULL) {
        printf("Error getting surface info\n");
        ds->Unlock(ds);
        awt.FreeDrawingSurface(ds);
        return;
    }

    /* Get the platform-specific drawing info */
    dsi_x11 = (JAWT_X11DrawingSurfaceInfo*)dsi->platformInfo;

    /* Now paint */
    gc = XCreateGC(dsi_x11->display, dsi_x11->drawable, 0, 0);
    XDrawImageString (dsi_x11->display, dsi_x11->drawable, gc,
                      100, 110, testString, strlen(testString));
    XFreeGC(dsi_x11->display, gc);

    /* Free the drawing surface info */
```

```
    ds->FreeDrawingSurfaceInfo(dsi);

    /* Unlock the drawing surface */
    ds->Unlock(ds);

    /* Free the drawing surface */
    awt.FreeDrawingSurface(ds);
}
```

The key data structure here is JAWT: it provides access to all the information the native code needs to get the job done. The first part of the native method is boilerplate: it populates the JAWT structure, gets a `JAWT_DrawingSurface` structure, locks the surface (only one drawing engine at a time), then gets a `JAWT_DrawingSurfaceInfo` structure that contains a pointer (in the platformInfo field) to the necessary platform-specific drawing information. It also includes the bounding rectangle of the drawing surface and the current clipping region.

The structure of the information pointed to by platformInfo is defined in a machine-dependent header file called jawt_md.h. For Solaris/X11 drawing, it includes information about the X11 display and X11 drawable associated with `MyCanvas`. After the drawing operations are completed, there is more boilerplate code as `JAWT_DrawingSurfaceInfo` is freed and `JAWT_DrawingSurface` is unlocked and freed.

The corresponding code for the Windows platform would be structured similarly, but would include the version of jawt_md.h for Windows 32 and the structure located in the platformInfo field of drawing surface info would be cast as a `JAWT_Win32DrawingSurfaceInfo*`. And, of course, the actual drawing operations would need to be changed to those appropriate for the Windows platform.

The problem with the above code (provided by Sun Microsystems, Inc.), is that the `jobject graphics` parameter, corresponding to the actual `java/awt/Graphics` context is never used. The native graphic context is statically attached to the `Canvas` and always addresses the screen device. With this method, a printer graphic context is ignored and thus you cannot send your drawing to a printer.

2.2 The GUIDO Engine native interface

In order to avoid the above limitation, the GUIDO Engine native interface takes a different approach: it draws using a native device but offscreen, i.e. to a memory bitmap, and next copy this bitmap to a `java/awt/Graphics` context, making a real use of the Java `paint` method `Graphics` parameter.

First, a Java `guidoscore` provides a native method that can be called draw the score and to retrieve the offscreen bitmap data:

```
/** Draws the score into a bitmap.

  Draws the score to an offscreen that is next copied to the destination bitmap.
  @param dst the destination bitmap ARGB array
  @param w the bitmap width
  @param h the bitmap height
  @param desc the score drawing descriptor
  @param area clipping description
  @param color the color used to draw the score
*/
public native final synchronized int  GetBitmap (int[] dst, int w, int h,
                      guidodrawdesc desc, guidopaint area, Color color);
```

Next, the `Draw` method makes use of any valid `Graphics` context to build an image from the bitmap data and to draw this image in the current graphic context:

```
public synchronized int  Draw (Graphics g, int w, int h,
                     guidodrawdesc desc, guidopaint area, Color color) {
  class foo implements ImageObserver {
    public boolean  imageUpdate(Image img, int flags, int x, int y, int width, int height)
                              { return true; }
  }
  BufferedImage img = new BufferedImage (w, h, BufferedImage.TYPE_4BYTE_ABGR_PRE);
  int[] outPixels = new int[w*h];
  int result = GetBitmap (outPixels, w, h, desc, new guidopaint(), color);
  img.setRGB( 0, 0, w, h, outPixels, 0, w );
  g.drawImage (img, 0, 0, new foo());
  return result;
}
```

The native implementation is completely disconnected from the Java AWT native interface. It uses the GUIDO Engine abstract layer to draw the score offscreen and finally copy the offscreen bitmap data to a Java array.

```
int getBitmap (jint* dstBitmap, int w, int h, GuidoOnDrawDesc& desc, const VGColor& color)
{
  /* first uses the VGDevice GUIDO abstraction to draw the score
     in a platform independent way
  */
  VGDevice * dev = gSystem->CreateMemoryDevice (w, h);
  desc.hdc = dev;
  dev->SelectFillColor(color);
  dev->SelectPenColor(color);
  dev->SetFontColor (color);

  GuidoErrCode err = GuidoOnDraw (&desc);
  if (err == guidoNoErr)
    /* the only platform dependent part: the copy of the bitmap
       that is implemented according to the native VGDevice support */
    bimap_copy (dev, dstBitmap, w, h);
  delete dev;
  return err;
}
```

With this method, any valid Java `Graphics` can be used and thus it can draw to a screen device and a printer device as well. The drawback of the method is that it looses the vectorial information, which is not critical for screen devices since no scaling is required, but it may produce low quality output on printers, depending on the printer driver.

3 The GUIDO Java API

The GUIDO Java API is similar to the GUIDO C/C++ API [4] with an object oriented design.

It provides java classes to cover the GUIDO data structures:

- guidopageformat: a data structure used to control the default page format strategy of the score layout engine.
- guidodate: a guido date expressed as a rational value.

- guidopaint: a data structure used for clipping of the drawing.
- guidodrawdesc: a data structure used to indicate how to draw a score.
- guidolayout: a data structure for setting the engine layout parameters.
- guidoelementinfo: a data structure used by the score map API.
- guidorect: a rectangle data structure, used by the score map API.
- guidosegment: a time segment data structure, used by the score map API.

The main classes to cover the GUIDO API are:

- guido: provides basic information about the GUIDO engine.
- guidoscore: the main score API. A guido score has an internal abstract representation (AR) that is converted into a graphic representation (GR). The guidoscore class reflects this architecture and provices the methods to build an AR representation from textual music notation, to convert an AR to a GR representation, to control the layout, to draw the score.
- guidofactory: provides a set of methods to dynamically create a GUIDO abstract representation.

Additionally, the engine provides information about the relation between the graphic space and the time space. It defines a score map API to collect this information using an an abstract class for graphic map collection: the mapcollector class. The guidoscoremap class defines constants to select various mapping information.

And finally, the engine provides information about how the score time relates to the performance time (i.e. with repetitions, jumps to coda, etc) using an abstract class for time map collection: the timemapcollector.

The GUIDO JNI implements also a transparent support of the MusicXML format [3][2] via a *weak link* to the MusicXML library[2] that provides a converter to the Guido Music Notation format.

[2] http://libmusicxml.sourceforge.net

References

[1] S. B., "Common music notation," in *Beyond MIDI, The handbook of Musical Codes.*, S.-F. E., Ed. MIT Press, 1997, pp. 217–222.

[2] D. Fober, S.Letz, and Y.Orlarey, "Open source tools for music representation and notation," in *Proceedings of the first Sound and Music Computing conference - SMC'04*. IRCAM, 2004, pp. 91–95.

[3] M. Good, "MusicXML for Notation and Analysis." in *The Virtual Score*, W. B. Hewlett and E. Selfridge-Field, Eds. MIT Press, 2001, pp. 113–124.

[4] *GUIDOLib 1.39 - Developer documentation*, Grame.

[5] K. Hamel, "NoteAbility, a comprehensive music notation system." in *Proceedings of the International Computer Music Conference.*, 1998, pp. 506–509.

[6] H. Hoos, K. Hamel, K. Renz, and J. Kilian, "The GUIDO Music Notation Format - a Novel Approach for Adequately Representing Score-level Music." in *Proceedings of the International Computer Music Conference*. ICMA, 1998, pp. 451–454.

[7] M. Kuuskankare and M. Laurson, "Expressive notation package," *Computer Music Journal*, vol. 30, no. 4, pp. 67–79, 2006.

[8] H.-W. Nienhuys and J. Nieuwenhuizen, "LilyPond, a system for automated music engraving," in *Proceedings of the XIV Colloquium on Musical Informatics*, 2003.

[9] K. Renz, "Algorithms and data structures for a music notation system based on GUIDO music notation," Ph.D. dissertation, Technischen Universität Darmstadt, 2002.

Appendix. GUIDO JNI 1.0

Contents

Chapter 1

Class Documentation

1.1 guido Class Reference

Static Public Member Functions

- static native final synchronized int Init (String guidoFont, String textFont)
- static native final boolean xml2gmn ()
- static native final String musicxmlversion ()
- static native final String musicxml2guidoversion ()
- static native final String xml2gmn (String filename)
- static native final String GetErrorString (int errCode)
- static native final int GetParseErrorLine ()
- static native final float Unit2CM (float val)
- static native final float CM2Unit (float val)
- static native final float Unit2Inches (float val)
- static native final float Inches2Unit (float val)
- static native final String GetVersion ()
- static native final String GetJNIVersion ()
- static native final int CheckVersionNums (int major, int minor, int sub)
- static native final float GetLineSpace ()

Static Public Attributes

- static final int kNoBB = 0
- static final int kPageBB = 1
- static final int kSystemsBB = 2
- static final int kSystemsSliceBB = 4
- static final int kStavesBB = 8
- static final int kMeasureBB = 0x10
- static final int kEventsBB = 0x20
- static final int guidoNoErr = 0

- static final int guidoErrParse = -1
- static final int guidoErrMemory = -2
- static final int guidoErrFileAccess = -3
- static final int guidoErrUserCancel = -4
- static final int guidoErrNoMusicFont = -5
- static final int guidoErrNoTextFont = -6
- static final int guidoErrBadParameter = -7
- static final int guidoErrInvalidHandle = -8
- static final int guidoErrNotInitialized = -9
- static final int guidoErrActionFailed = -10

Static Package Functions

- [static initializer]

1.1.1 Detailed Description

The main Guido Engine class.

Provides basic information about the engine (version, units) Defines error codes.

1.1.2 Member Function Documentation

1.1.2.1 [static initializer] ( ) [static, package]

1.1.2.2 static native final synchronized int Init (String *guidoFont,* String *textFont*) [static]

Initialises the Guido Engine.

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

Parameters

guidoFont	a music font name (only "Guido2" supported).
textFont	a text font name.

Returns

a Guido error code.

1.1.2.3 static native final boolean xml2gmn ( ) [static]

Check the libMusicXML library availability

Returns

true when available

1.1.2.4 static native final String musicxmlversion ( ) [static]

Gives the libMusicXML library version when available

Returns

a version string (empty when the library is not available)

1.1.2.5 static native final String musicxml2guidoversion ( ) [static]

Gives the MusicXML to guido converter version

Returns

a version string (empty when the library is not available)

1.1.2.6 static native final String xml2gmn (String *filename*) [static]

Converts a MusicXML file to a GMN string

Parameters

filename	the file name

Returns

a string

1.1.2.7 static native final String GetErrorString (int *errCode*) [static]

Gives a textual description of a Guido error code.

Parameters

errCode	a Guido error code.

Returns

a string describing the error.

1.1.2.8 static native final int GetParseErrorLine ( ) [static]

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

Returns

a line number.

1.1.2.9 static native final float Unit2CM (float *val*) [static]

Converts internal Guido units into centimeters.

Parameters

val	the value to be converted

Returns

the converted value

1.1.2.10 static native final float CM2Unit (float *val*) [static]

Converts centimeters into internal Guido units.

Parameters

val	the value to be converted

Returns

the converted value

1.1.2.11 static native final float Unit2Inches (float *val*) [static]

Converts internal Guido units into inches.

Parameters

val	the value to be converted

Returns

the converted value

1.1.2.12 static native final float Inches2Unit (float *val*) [static]

Converts inches into internal Guido units.

Parameters

val	the value to be converted

Returns

the converted value

1.1.2.13 static native final String GetVersion ( ) [static]

Gives the library version number as a string

Version number format is MAJOR.MINOR.SUB

1.1.2.14 static native final String GetJNIVersion ( ) [static]

Gives the JNI library version number as a string

Version number format is MAJOR.MINOR.SUB

1.1.2.15 static native final int CheckVersionNums (int *major,* int *minor,* int *sub*) [static]

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.

1.1.2.16 static native final float GetLineSpace ( ) [static]

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.

1.1.3 Member Data Documentation

1.1.3.1 final int kNoBB = 0 [static]

1.1.3.2 final int kPageBB = 1 [static]

1.1.3.3 final int kSystemsBB = 2 [static]

1.1.3.4 final int kSystemsSliceBB = 4 [static]

1.1.3.5 final int kStavesBB = 8 [static]

1.1.3.6 final int kMeasureBB = 0x10 [static]

1.1.3.7 final int kEventsBB = 0x20 [static]

1.1.3.8 final int guidoNoErr = 0 [static]

1.1.3.9 final int guidoErrParse = -1 [static]

1.1.3.10 final int guidoErrMemory = -2 [static]

1.1.3.11 final int guidoErrFileAccess = -3 [static]

1.1.3.12 final int guidoErrUserCancel = -4 [static]

1.1.3.13 final int guidoErrNoMusicFont = -5 [static]

1.1.3.14 final int guidoErrNoTextFont = -6 [static]

1.1.3.15 final int guidoErrBadParameter = -7 [static]

1.1.3.16 final int guidoErrInvalidHandle = -8 [static]

1.1.3.17 final int guidoErrNotInitialized = -9 [static]

1.1.3.18 final int guidoErrActionFailed = -10 [static]

1.2 guidodate Class Reference

Public Member Functions

- guidodate ()
- guidodate (int num, int denum)

Static Public Member Functions

- static native void Init ()

Public Attributes

- int fNum
- int fDenum

1.2.1 Detailed Description

Guido dates

Guido dates are rational values indicating fractions of a whole note. They are used for dates as well as for durations.

1.2.2 Constructor & Destructor Documentation

1.2.2.1 guidodate ( )

1.2.2.2 guidodate (int *num,* int *denum*)

1.2.3 Member Function Documentation

1.2.3.1 static native void Init ( ) [static]

Internal jni initialization method. Automatically called at package init.

1.2.4 Member Data Documentation

1.2.4.1 int fNum

1.2.4.2 int fDenum

1.3 guidodrawdesc Class Reference

Public Member Functions

- guidodrawdesc ()
- guidodrawdesc (int w, int h)
- void print ()

Static Public Member Functions

- static native void Init ()

Public Attributes

- int fPage
- int fScrollx
- int fWidth
- boolean fIsprint

Package Attributes

- int fScrolly
- int fHeight

1.3.1 Detailed Description

Guido score drawing descriptor

guidodrawdesc is basically a data structure used to indicate how to draw a score to the guido engine.

See also

guidoscore.Draw

1.3.2 Constructor & Destructor Documentation

1.3.2.1 guidodrawdesc ( )

1.3.2.2 guidodrawdesc (int *w*, int *h*)

1.3.3 Member Function Documentation

1.3.3.1 void print ( )

Print utility.

1.3.3.2 static native void Init ( ) [static]

Internal jni initialization method. Automatically called at package init.

1.3.4 Member Data Documentation

1.3.4.1 int fPage

The page number. Starts from 1.

1.3.4.2 int fScrollx

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 guido internal units.

1.3.4.3 int fScrolly [package]

1.3.4.4 int fWidth

Indicates the size of the drawing area.

Units are device units (typically pixels).

1.3.4.5 int fHeight [package]

1.3.4.6 boolean fIsprint

Flag for printing.

If true, the engine ignores scroll and sizes parameters. (probably obsolete now!)

1.4 guidoelementinfo Class Reference

Public Member Functions

- guidoelementinfo ()
- guidoelementinfo (int t, int sn, int vn)

Static Public Member Functions

- static native void Init ()

Public Attributes

- int type
- int staffNum
- int voiceNum

Static Public Attributes

- static final int kNote = 1
- static final int kRest = 2
- static final int kEmpty = 3

- static final int kBar = 4
- static final int kRepeatBegin = 5
- static final int kRepeatEnd = 6
- static final int kStaff = 7
- static final int kSystemSlice = 8
- static final int kSystem = 9
- static final int kPage = 10

1.4.1 Detailed Description

Guido score element basic description

guidoelementinfo is basically a data structure used by the score map API.

See also

guidoscore.GetMap
mapcollector.Graph2TimeMap

1.4.2 Constructor & Destructor Documentation

1.4.2.1 guidoelementinfo ( )

1.4.2.2 guidoelementinfo (int *t*, int *sn*, int *vn*)

1.4.3 Member Function Documentation

1.4.3.1 static native void Init ( ) [static]

Internal jni initialization method. Automatically called at package init.

1.4.4 Member Data Documentation

1.4.4.1 final int kNote = 1 [static]

1.4.4.2 final int kRest = 2 [static]

1.4.4.3 final int kEmpty = 3 [static]

1.4.4.4 final int kBar = 4 [static]

1.4.4.5 final int kRepeatBegin = 5 [static]

1.4.4.6 final int kRepeatEnd = 6 [static]

1.4.4.7 final int kStaff = 7 [static]

1.4.4.8 final int kSystemSlice = 8 [static]

1.4.4.9 final int kSystem = 9 [static]

1.4.4.10 final int kPage = 10 [static]

1.4.4.11 int type

the event type (

```
kNote, kBar,...
```

)

1.4.4.12 int staffNum

the current staff number or 0 when na

1.4.4.13 int voiceNum

the current voice number or 0 when na

1.5 guidofactory Class Reference

Public Member Functions

- guidofactory ()
- native final synchronized int Open ()
- native final synchronized void Close ()
- native final synchronized int OpenMusic ()

- native final synchronized long CloseMusic ()
- native final synchronized int OpenVoice ()
- native final synchronized int CloseVoice ()
- native final synchronized int OpenChord ()
- native final synchronized int CloseChord ()
- native final synchronized int InsertCommata ()
- native final synchronized int OpenEvent (String eventName)
- native final synchronized int CloseEvent ()
- native final synchronized int AddSharp ()
- native final synchronized int AddFlat ()
- native final synchronized int SetEventDots (int dots)
- native final synchronized int SetEventAccidentals (int accident)
- native final synchronized int SetOctave (int octave)
- native final synchronized int SetDuration (int numerator, int denominator)
- native final synchronized int OpenTag (String tagName, long tagID)
- native final synchronized int IsRangeTag ()
- native final synchronized int EndTag ()
- native final synchronized int CloseTag ()
- native final synchronized int AddTagParameterString (String val)
- native final synchronized int AddTagParameterInt (int val)
- native final synchronized int AddTagParameterFloat (double val)
- native final synchronized int SetParameterName (String name)
- native final synchronized int SetParameterUnit (String unit)

Static Public Member Functions

- static native void Init ()

Public Attributes

- final long fFactoryHandler

1.5.1 Detailed Description

The GUIDO Factory provides a set of methods to dynamically create a GUIDO abstract representation.

The GUIDO Factory is a state machine that operates on implicit current elements: for example, once you open a voice (

```
OpenVoice()
```

), 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

`OpenMusic()`

and

`CloseMusic()`

- the current voice: modified by

`OpenVoice()`

and

`CloseVoice()`

- the current chord: modified by

`OpenChord()`

and

`CloseChord()`

- the current event: modified by

`OpenEvent()`

and

`CloseEvent()`

- the current tag: modified by

`OpenTag()`

and

`CloseTag()`

1.5.2 Constructor & Destructor Documentation

1.5.2.1 guidofactory ( )

1.5.3 Member Function Documentation

1.5.3.1 native final synchronized int Open ( )

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.

1.5.3.2 native final synchronized void Close ( )

Closes the Guido Factory.

Must be called to release the factory associated resources.

1.5.3.3 native final synchronized int OpenMusic ( )

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

```
CloseMusic()
```

Returns

an integer that is an error code if not null.

See also

guidofactory.CloseMusic

1.5.3.4 native final synchronized long CloseMusic ( )

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.

Returns

a GUIDO handler to the new AR structure, or 0. This handler may be used to build a new guidoscore.

See also

guidofactory.OpenMusic

1.5.3.5 native final synchronized int OpenVoice ( )

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

```
CloseVoice()
```

Voices are similar to sequence is GMN.

Returns

an error code

See also

guidofactory.CloseVoice

1.5.3.6 native final synchronized int CloseVoice ( )

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

See also

guidofactory.OpenVoice

1.5.3.7 native final synchronized int OpenChord ( )

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

```
CloseChord()
```

Returns

an error code

See also

guidofactory.CloseChord

1.5.3.8 native final synchronized int CloseChord ( )

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

See also

guidofactory.OpenChord

1.5.3.9 native final synchronized int InsertCommata ( )

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

1.5.3.10 native final synchronized int OpenEvent (String *eventName*)

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

```
CloseEvent()
```

Parameters

eventName	a note, rest or empty name confroming to the GMN format

Returns

an error code

See also

guidofactory.CloseEvent

1.5.3.11 native final synchronized int CloseEvent ( )

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

See also

guidofactory.OpenEvent

1.5.3.12 native final synchronized int AddSharp ( )

Adds a sharp to the current event.

The current event must be a note.

Returns

an error code

1.5.3.13 native final synchronized int AddFlat ( )

Add a flat to the current event.

The current event must be a note.

Returns

an error code.

1.5.3.14 native final synchronized int SetEventDots (int *dots*)

Sets the number of dots of the current event.

Parameters

dots	the number of dots to be carried by the current event.

Returns

an error code.

1.5.3.15 native final synchronized int SetEventAccidentals (int *accident*)

Sets the accidentals of the current event.

Parameters

accident	positive values are used for sharp and negative values for flats

Returns

an error code.

1.5.3.16 native final synchronized int SetOctave (int *octave*)

Sets the octave of the current event.

The current event must be a note. The octave number becomes the current octave i.e. next notes will carry this octave number until otherwise specified.

Parameters

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.

1.5.3.17 native final synchronized int SetDuration (int *numerator,* int *denominator*)

Sets the duration of the current event.

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

Parameters

numerator	the rational duration numerator
denominator	the rational duration denominator

Returns

an error code.

1.5.3.18 native final synchronized int OpenTag (String *tagName,* long *tagID*)

Add a tag to the current voice.

Parameters

tagName	the tag name
tagID	is the number that the parser generates for advanced GUIDO ?????

Returns

an error code.

1.5.3.19 native final synchronized int IsRangeTag ( )

Indicates that the current tag is a range tag.

Returns

an error code.

1.5.3.20 native final synchronized int EndTag ( )

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. If the tag has no range, it must be called directly after

```
CloseTag()
```

.

With the following examples:

- `staff<1> c d`
  : call

  ```
  EndTag()
  ```

  after

  ```
  CloseTag()
  ```

  and before creating the

  ```
  c
  ```

  note
- `slur(c d e) f`
  : call

  ```
  EndTag()
  ```

  before creating the

  ```
  f
  ```

  note

Returns

an error code.

1.5.3.21 native final synchronized int CloseTag ( )

Closes the current tag.

The function is applied to the current tag. Must be called after adding parameter and before the range. With the following examples:

- `tag<1,2,3>(c d e )`
  : call

  ```
  CloseTag()
  ```

  , next

  ```
  IsRangeTag()
  ```

  creating the

  ```
  c d e
  ```

  notes and call

  ```
  EndTag()
  ```

- `tag<1,2> c d`
  : call

  ```
  CloseTag()
  ```

  before creating the

  ```
  c
  ```

  note

Returns

an error code.

1.5.3.22 native final synchronized int AddTagParameterString (String *val*)

Adds a new string parameter to the current tag.

Parameters

val	the string parameter value

Returns

an error code.

1.5.3.23 native final synchronized int AddTagParameterInt (int *val*)

Adds a new integer parameter to the current tag.

Parameters

val	the parameter value

Returns

an error code.

1.5.3.24 native final synchronized int AddTagParameterFloat (double *val*)

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

Parameters

val	the parameter value

Returns

an error code.

1.5.3.25 native final synchronized int SetParameterName (String *name*)

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

Parameters

name	the tag parameter name

Returns

an error code.

1.5.3.26 native final synchronized int SetParameterUnit (String *unit*)

Defines the unit of the last added tag-parameter

Parameters

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.

1.5.3.27 static native void Init ( ) [static]

Internal jni initialization method. Automatically called at package init.

1.5.4 Member Data Documentation

1.5.4.1 final long fFactoryHandler

1.6 guidolayout Class Reference

Public Member Functions

- native final void GetDefault ()
- guidolayout ()
- void print ()

Static Public Member Functions

- static native void Init ()

Public Attributes

- float fSystemsDistance
- int fSystemsDistribution
- float fSystemsDistribLimit
- float fForce
- float fSpring
- int fNeighborhoodSpacing
- boolean fOptimalPageFill

Static Public Attributes

- static final int kAutoDistrib = 1
- static final int kAlwaysDistrib = 2
- static final int kNeverDistrib = 3

1.6.1 Detailed Description

Global settings of the Guido Engine for the graphic score layout.

1.6.2 Constructor & Destructor Documentation

1.6.2.1 guidolayout ( )

1.6.3 Member Function Documentation

1.6.3.1 native final void GetDefault ( )

Retrieves the engine settings values.

On output, the guidolayout structure contains the engine settings values.

1.6.3.2 void print ( )

Print utility.

1.6.3.3 static native void Init ( ) [static]

Internal jni initialization method. Automatically called at package init.

1.6.4 Member Data Documentation

1.6.4.1 final int kAutoDistrib = 1 [static]

1.6.4.2 final int kAlwaysDistrib = 2 [static]

1.6.4.3 final int kNeverDistrib = 3 [static]

1.6.4.4 float fSystemsDistance

Distance between systems

Distance is in internal units (default value: 75)

1.6.4.5 int fSystemsDistribution

Systems distribution.

Possible values: kAutoDistrib (default), kAlwaysDistrib, kNeverDistrib

1.6.4.6 float fSystemsDistribLimit

Maximum distance allowed between two systems.

Used in 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)

1.6.4.7 float fForce

Force value of the Space-Force function.

Typical values range from 400 to 1500. Default value: 750

1.6.4.8 float fSpring

Spring parameter

Typical values range from 1 to 5. Default value: 1.1

1.6.4.9 int fNeighborhoodSpacing

Spacing algorithm control

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

1.6.4.10 boolean fOptimalPageFill

Optimal page fill algorithm control

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

1.7 guidopageformat Class Reference

Public Member Functions

- native final void GetDefault ()
- native final void SetDefault ()
- guidopageformat ()
- guidopageformat (float w, float h, float ml, float mt, float mr, float mb)
- void print ()

Static Public Member Functions

- static native void Init ()

Public Attributes

- float fWidth
- float fHeight
- float fMarginleft
- float fMargintop
- float fMarginright
- float fMarginbottom

1.7.1 Detailed Description

Guido page format

The Guido language includes a

```
\\pageFormat
```

tag to specify the page layout within the score descritpion. When a guido score description doesn't include this

```
\\pageFormat
```

tag, the guido engine applies a default page format. The guidopageformat is basically a data structure used to control the default page format strategy of the score layout engine.

1.7.2 Constructor & Destructor Documentation

1.7.2.1 guidopageformat ( )

1.7.2.2 guidopageformat (float *w,* float *h,* float *ml,* float *mt,* float *mr,* float *mb*)

1.7.3 Member Function Documentation

1.7.3.1 native final void GetDefault ( )

Retrieve the engine default page format.

1.7.3.2 native final void SetDefault ( )

Sets the engine 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

1.7.3.3 void print ( )

Print utility.

1.7.3.4 static native void Init ( ) [static]

Internal jni initialization method. Automatically called at package init.

1.7.4 Member Data Documentation

1.7.4.1 float fWidth

1.7.4.2 float fHeight

1.7.4.3 float fMarginleft

1.7.4.4 float fMargintop

1.7.4.5 float fMarginright

1.7.4.6 float fMarginbottom

1.8 guidopaint Class Reference

Public Member Functions

- guidopaint ()
- guidopaint (int left, int top, int right, int bottom)
- void print ()

Static Public Member Functions

- static native void Init ()

Public Attributes

- boolean fErase
- int fLeft
- int fTop
- int fRight
- int fBottom

1.8.1 Detailed Description

Guido score drawing descriptor

guidopaint is basically a data structure used for clipping. Only systems that intersect with this rectangle will be drawn. Coordinates should be given in internal units.

See also

guidoscore.Draw

1.8.2 Constructor & Destructor Documentation

1.8.2.1 guidopaint ( )

1.8.2.2 guidopaint (int *left,* int *top,* int *right,* int *bottom*)

1.8.3 Member Function Documentation

1.8.3.1 void print ( )

Print utility.

1.8.3.2 static native void Init ( ) [static]

Internal jni initialization method. Automatically called at package init.

1.8.4 Member Data Documentation

1.8.4.1 boolean fErase

a flag to ignore the following rect and to redraw everything

1.8.4.2 int fLeft

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

1.8.4.3 int fTop

1.8.4.4 int fRight

1.8.4.5 int fBottom

1.9 guidorect Class Reference

Public Member Functions

- int height ()
- int width ()
- guidorect ()
- guidorect (int l, int t, int r, int b)

Static Public Member Functions

- static native void Init ()

Public Attributes

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

1.9.1 Detailed Description

Guido rectangle descriptor

guidorect is basically a data structure used by the score map API.

See also

guidoscore.GetMap
mapcollector.Graph2TimeMap

1.9.2 Constructor & Destructor Documentation

1.9.2.1 guidorect ( )

1.9.2.2 guidorect (int *l*, int *t*, int *r*, int *b*)

1.9.3 Member Function Documentation

1.9.3.1 int height ( )

1.9.3.2 int width ( )

1.9.3.3 static native void Init ( ) [static]

Internal jni initialization method. Automatically called at package init.

1.9.4 Member Data Documentation

1.9.4.1 int top

1.9.4.2 int left

1.9.4.3 int right

1.9.4.4 int bottom

1.10 guidoscore Class Reference

Public Member Functions

- native final synchronized int ParseFile (String filename)
- native final synchronized int ParseString (String gmn)
- native final synchronized int AR2GR ()
- native final synchronized int AR2GR (guidolayout layout)
- native final synchronized int UpdateGR ()
- native final synchronized int UpdateGR (guidolayout layout)
- native final synchronized int ResizePageToMusic ()
- native final synchronized void FreeAR ()
- native final synchronized void FreeGR ()
- native final synchronized int GetBitmap (int[] dst, int w, int h, guidodrawdesc desc, guidopaint area, Color color)
- synchronized int Draw (Graphics g, int w, int h, guidodrawdesc desc, guidopaint area)
- synchronized int Draw (Graphics g, int w, int h, guidodrawdesc desc, guidopaint area, Color color)
- native final synchronized void GetPageFormat (int pagenum, guidopageformat pf)
- native final synchronized int MarkVoice (int voicenum, guidodate date, guidodate duration, int red, int green, int blue)
- native final synchronized int GetPageCount ()
- native final synchronized int GetDuration (guidodate date)
- native final synchronized int GetPageDate (int pagenum, guidodate date)
- native final synchronized int FindEventPage (guidodate date)
- native final synchronized int FindPageAt (guidodate date)
- native final synchronized int GetMap (int page, float width, float height, int selector, mapcollector f)
- native final synchronized int GetTimeMap (timemapcollector f)
- void close ()
- guidoscore ()
- guidoscore (long ar)
- native final synchronized void DrawBoundingBoxes (int bbMap)
- native final synchronized int GetDrawBoundingBoxes ()

Static Public Member Functions

- static native void Init ()

Public Attributes

- final long fARHandler
- final long fGRHandler

Static Public Attributes

- static final int kNoBB = 0
- static final int kPageBB = 1
- static final int kSystemsBB = 2
- static final int kSystemsSliceBB = 4
- static final int kStavesBB = 8
- static final int kMeasureBB = 0x10
- static final int kEventsBB = 0x20

1.10.1 Detailed Description

The main score API.

A guido score has an internal abstract representation (AR) that is converted into a graphic representation (GR). The guidoscore reflects this architecture and provices the method to convert an AR representation to GR representation.

1.10.2 Constructor & Destructor Documentation

1.10.2.1 guidoscore ( )

1.10.2.2 guidoscore (long *ar*)

1.10.3 Member Function Documentation

1.10.3.1 native final synchronized int ParseFile (String *filename*)

Parse a guido file

On output,

```
fARHandler
```

contains a handler to the Guido AR representation.

Parameters

filename	the file name

Returns

an error code.

1.10.3.2 native final synchronized int ParseString (String *gmn*)

Parse a guido string
On output,

```
fARHandler
```

contains a handler to the Guido AR representation.

Parameters

gmn	a string containing GMN code

Returns

an error code.

See also

"The GUIDO Music Notation Format"

1.10.3.3 native final synchronized int AR2GR ( )

Converts an AR representation into a GR representation
On output,

```
fGRHandler
```

contains a handler to the Guido GR representation.

Returns

an error code.

1.10.3.4 native final synchronized int AR2GR (guidolayout *layout*)

Converts an AR representation into a GR representation
Makes use of the Guido Engine settings given as argument. On output,

```
fGRHandler
```

contains a handler to the Guido GR representation.

Parameters

layout	layout settings

Returns

an error code.

See also

guidolayout

1.10.3.5 native final synchronized int UpdateGR ( )

Updates a GR representation

Should be called for example after changing the default page format.

Returns

an error code.

1.10.3.6 native final synchronized int UpdateGR (guidolayout *layout*)

Updates a GR representation

Makes use of the Guido Engine settings given as argument.

Parameters

layout	layout settings

Returns

an error code.

See also

guidolayout

1.10.3.7 native final synchronized int ResizePageToMusic ( )

Resize the page sizes to the music size.

Returns

an error code.

1.10.3.8 native final synchronized void FreeAR ( )

Tells the engine to release the AR representation handler.

1.10.3.9 native final synchronized void FreeGR ( )

Tells the engine to release the GR representation handler.

1.10.3.10 native final synchronized int GetBitmap (int[] *dst,* int *w,* int *h,* guidodrawdesc *desc,* guidopaint *area,* Color *color*)

Draws the score into a bitmap.

Actually, draws the score to an offscreen that is next copied to the destination bitmap.

Parameters

dst	the destination bitmap ARGB array
w	the bitmap width
h	the bitmap height
desc	the score drawing descriptor
area	clipping description
color	the color used to draw the score

See also

guidodrawdesc
guidopaint

1.10.3.11 synchronized int Draw (Graphics *g,* int *w,* int *h,* guidodrawdesc *desc,* guidopaint *area*)

Draws the score.

Drawing the score should be typically called from the paint method of a Canvas.

Parameters

g	a Graphics
w	the desired drawing width
h	the desired drawing heigth
desc	the score drawing descriptor
area	clipping description

See also

guidodrawdesc
guidopaint

1.10.3.12 synchronized int Draw (Graphics *g,* int *w,* int *h,* guidodrawdesc *desc,* guidopaint *area,* Color *color*)

Draws the score.

Drawing the score should be typically called from the paint method of a Canvas.

Parameters

g	a Graphics
w	the desired drawing width
h	the desired drawing heigth
desc	the score drawing descriptor
area	clipping description
color	the color used to draw the score

See also

guidodrawdesc
guidopaint

1.10.3.13 native final synchronized void GetPageFormat (int *pagenum,* guidopageformat *pf*)

Retrieve the format of a given page.

Parameters

pagenum	a page number, starting from 1
pf	on output, the corresponding page format

1.10.3.14 native final synchronized int MarkVoice (int *voicenum,* guidodate *date,* guidodate *duration,* int *red,* int *green,* int *blue*)

Force the color of all notes of a voice in a given time interval.

Parameters

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.

1.10.3.15 native final synchronized int GetPageCount ( )

Give the score pages count.

Returns

the score pages count or an error code when < 0

1.10.3.16 native final synchronized int GetDuration (guidodate *date*)

Give the score duration.

Parameters

date	on output, the score duration.

Returns

an error code.

See also

guidodate

1.10.3.17 native final synchronized int GetPageDate (int *pagenum,* guidodate *date*)

Give a page date.

Parameters

pagenum	a guido page number (starting from 1)
date	on output, the page date when the page number is found.

Returns

an error code.

See also

guidodate

1.10.3.18 native final synchronized int FindEventPage (guidodate *date*)

1.10.3.19 native final synchronized int FindPageAt (guidodate *date*)

Find a page at a given date.

Parameters

date	a guido date

Returns

a page number (starting from 1) or 0 when no page is found.

See also

guidodate

1.10.3.20 native final synchronized int GetMap (int *page,* float *width,* float *height,* int *selector,* mapcollector *f*)

Retrieves the graphic to time mapping

Parameters

page	a page index, starting from 1.
width	an area width (typically the current drawing zone width).
height	an area height (typically the current drawing zone height).
selector	a filter to focus on specific elements.
f	a mapcollector object that will be called for each selected element.

Returns

an error code.

See also

mapcollector

1.10.3.21 native final synchronized int GetTimeMap (timemapcollector *f*)

Retrieves the wrapped to unwrapped time mapping

Parameters

f	a TimeMapCollector object that will be called for each time segment.

Returns

an error code.

See also

timemapcollector

1.10.3.22 void close ( )

close a score

The close method must be called to notify the Guido Engine that the associated ressources can be released.

1.10.3.23 native final synchronized void DrawBoundingBoxes (int *bbMap*)

Control bounding boxes drawing.

Bounding boxes are internal to the layout engine. This API is for the layout engine debugging purpose.

Parameters

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

1.10.3.24 native final synchronized int GetDrawBoundingBoxes ( )

Gives the drawn bounding boxes set.

This API is for the layout engine debugging purpose.

Returns

a bits field indicating the set of bounding boxes.

1.10.3.25 static native void Init ( ) [static]

Internal jni initialization method. Automatically called at package init.

1.10.4 Member Data Documentation

1.10.4.1 final int kNoBB = 0 [static]

1.10.4.2 final int kPageBB = 1 [static]

1.10.4.3 final int kSystemsBB = 2 [static]

1.10.4.4 final int kSystemsSliceBB = 4 [static]

1.10.4.5 final int kStavesBB = 8 [static]

1.10.4.6 final int kMeasureBB = 0x10 [static]

1.10.4.7 final int kEventsBB = 0x20 [static]

1.10.4.8 final long fARHandler

1.10.4.9 final long fGRHandler

1.11 guidoscoremap Class Reference

Static Public Attributes

- static final int kGuidoPage = 0
- static final int kGuidoSystem = 1
- static final int kGuidoSystemSlice = 2
- static final int kGuidoStaff = 3
- static final int kGuidoBar = 4
- static final int kGuidoEvent = 5

1.11.1 Member Data Documentation

1.11.1.1 final int kGuidoPage = 0 [static]

1.11.1.2 final int kGuidoSystem = 1 [static]

1.11.1.3 final int kGuidoSystemSlice = 2 [static]

1.11.1.4 final int kGuidoStaff = 3 [static]

1.11.1.5 final int kGuidoBar = 4 [static]

1.11.1.6 final int kGuidoEvent = 5 [static]

1.12 guidosegment Class Reference

Public Member Functions

- guidosegment ()
- guidosegment (guidodate s, guidodate e)

Static Public Member Functions

- static native void Init ()

Public Attributes

- guidodate start
- guidodate end

1.12.1 Detailed Description

Guido time segments descriptor

guidosegment is basically a data structure used by the score map API.

See also

guidoscore.GetMap
mapcollector.Graph2TimeMap

1.12.2 Constructor & Destructor Documentation

1.12.2.1 guidosegment ( )

1.12.2.2 guidosegment (guidodate *s,* guidodate *e*)

1.12.3 Member Function Documentation

1.12.3.1 static native void Init ( ) [static]

Internal jni initialization method. Automatically called at package init.

1.12.4 Member Data Documentation

1.12.4.1 guidodate start

1.12.4.2 guidodate end

1.13 mapcollector Interface Reference

Package Functions

- void Graph2TimeMap (guidorect box, guidosegment time, guidoelementinfo infos)

1.13.1 Detailed Description

an abstract class for graphic map collection

A graphic map describes the relation between the graphic space and the time space.

See also

guidoscore.GetMap

1.13.2 Member Function Documentation

1.13.2.1 void Graph2TimeMap (guidorect *box,* guidosegment *time,* guidoelementinfo *infos*) [package]

callback called by

```
guidoscore.GetMap
```

Parameters

box	a graphic rectangle expressed in device coordinates
time	the corresponding time segment in score wrapped time
infos	additional information about the current graphic element

See also

guidoscore.GetMap
guidoelementinfo

1.14 timemapcollector Interface Reference

Package Functions

- void Time2TimeMap (guidosegment from, guidosegment to)

1.14.1 Detailed Description

an abstract class for time map collection

A time map describes the relation between the score wrapped time and unwrapped time i.e. with all repetitions and jumps played.

See also

guidoscore.GetTimeMap

1.14.2 Member Function Documentation

1.14.2.1 void Time2TimeMap (guidosegment *from,* guidosegment *to*) [package]

callback called by

```
guidoscore.GetTimeMap
```

Parameters

from	a segment in score wrapped time
to	a segment in score unwrapped time

See also

guidoscore.GetTimeMap

Executive Summary

This Technical Report is a working paper paper presenting the JGuido Library, a generic, portable library and C/C++ API for the graphical rendering of musical scores. The report introduces the library and the context of its implementation. The library is included into MIROR-IMPRO and MIROR-COMPO software developed by Sony Computer Science Laboratory Paris, and released in August 2013. The software itself can be downloaded on request, by contacting the authors here: http://www.csl.sony.fr/contact.php

Acknowledgments

The work described in this report forms part of the European project MIROR **M**usical **I**nteraction **R**elying **O**n **R**eflexion http://www.mirorproject.eu/ , co-funded by the European Community under the Information and Communication Technologies (ICT) theme of the Seventh Framework Programme. (FP7/2007-2013). Grant agreement n° 258338