

GUIDOLib  
v.1.52

Generated by Doxygen 1.7.2

Fri Dec 6 2013 11:18:07



# Contents

<b>1</b>	<b>The GUIDO Engine Library</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	The GUIDO Music Notation . . . . .	1
1.3	The GUIDO Engine . . . . .	1
1.4	Main library services . . . . .	2
1.5	Graphic devices . . . . .	2
<b>2</b>	<b>Sample code</b>	<b>5</b>
2.1	Displaying a score . . . . .	5
2.2	Building music using the GUIDO Factory . . . . .	6
<b>3</b>	<b>Todo List</b>	<b>9</b>
<b>4</b>	<b>Bug List</b>	<b>11</b>
<b>5</b>	<b>Module Documentation</b>	<b>13</b>
5.1	MIDI Export . . . . .	13
5.1.1	Typedef Documentation . . . . .	13
5.1.1.1	Guido2MidiParams . . . . .	13
5.1.2	Function Documentation . . . . .	14
5.1.2.1	GuidoAR2MIDIFile . . . . .	14
5.2	Error codes . . . . .	14
5.2.1	Enumeration Type Documentation . . . . .	14
5.2.1.1	GuidoErrCode . . . . .	14
5.3	Building abstract and graphic representations . . . . .	15
5.3.1	Detailed Description . . . . .	15
5.3.2	Function Documentation . . . . .	16
5.3.2.1	GuidoInit . . . . .	16
5.3.2.2	GuidoShutdown . . . . .	16
5.3.2.3	__declspec . . . . .	16
5.3.2.4	__declspec . . . . .	17
5.3.2.5	GuidoAR2GR . . . . .	17
5.3.2.6	GuidoUpdateGR . . . . .	17
5.3.2.7	GuidoFreeAR . . . . .	18
5.3.2.8	GuidoFreeGR . . . . .	18
5.3.2.9	GuidoGetErrorString . . . . .	18
5.3.2.10	__declspec . . . . .	18
5.3.2.11	GuidoGetDefaultLayoutSettings . . . . .	18
5.3.3	Variable Documentation . . . . .	19
5.3.3.1	ar . . . . .	19

5.4	Browsing music pages	19
5.4.1	Detailed Description	19
5.4.2	Function Documentation	19
5.4.2.1	GuidoCountVoices	19
5.4.2.2	GuidoGetPageCount	20
5.4.2.3	GuidoGetSystemCount	20
5.4.2.4	GuidoDuration	20
5.4.2.5	GuidoFindEventPage	21
5.4.2.6	GuidoFindPageAt	21
5.4.2.7	GuidoGetPageDate	21
5.5	Score drawing and pages formatting	22
5.5.1	Detailed Description	23
5.5.2	Function Documentation	23
5.5.2.1	GuidoOnDraw	23
5.5.2.2	GuidoSVGExport	23
5.5.2.3	GuidoDrawBoundingBoxes	23
5.5.2.4	GuidoGetDrawBoundingBoxes	23
5.5.2.5	GuidoGetPageFormat	24
5.5.2.6	GuidoSetDefaultPageFormat	24
5.5.2.7	GuidoGetDefaultPageFormat	24
5.5.2.8	GuidoUnit2CM	24
5.5.2.9	GuidoCM2Unit	25
5.5.2.10	GuidoUnit2Inches	25
5.5.2.11	GuidoInches2Unit	25
5.5.2.12	GuidoResizePageToMusic	25
5.6	Miscellaneous	26
5.6.1	Detailed Description	26
5.6.2	Function Documentation	26
5.6.2.1	GuidoGetVersionNums	26
5.6.2.2	GuidoGetVersionStr	27
5.6.2.3	GuidoCheckVersionNums	27
5.6.2.4	GuidoGetLineSpace	27
5.6.2.5	GuidoMarkVoice	28
5.6.2.6	GuidoSetSymbolPath	28
5.6.2.7	GuidoGetSymbolPath	28
5.7	GUIDO Factory	29
5.7.1	Detailed Description	31
5.7.2	Typedef Documentation	31
5.7.2.1	ARFactoryHandler	31
5.7.3	Function Documentation	31
5.7.3.1	GuidoFactoryOpen	31
5.7.3.2	GuidoFactoryClose	32
5.7.3.3	GuidoFactoryOpenMusic	32
5.7.3.4	GuidoFactoryCloseMusic	32
5.7.3.5	GuidoFactoryOpenVoice	32
5.7.3.6	GuidoFactoryCloseVoice	33
5.7.3.7	GuidoFactoryOpenChord	33
5.7.3.8	GuidoFactoryCloseChord	33
5.7.3.9	GuidoFactoryInsertCommata	33
5.7.3.10	GuidoFactoryOpenEvent	34

5.7.3.11	GuidoFactoryCloseEvent . . . . .	34
5.7.3.12	GuidoFactoryAddSharp . . . . .	34
5.7.3.13	GuidoFactoryAddFlat . . . . .	34
5.7.3.14	GuidoFactorySetEventDots . . . . .	34
5.7.3.15	GuidoFactorySetEventAccidentals . . . . .	35
5.7.3.16	GuidoFactorySetOctave . . . . .	35
5.7.3.17	GuidoFactorySetDuration . . . . .	35
5.7.3.18	GuidoFactoryOpenTag . . . . .	36
5.7.3.19	GuidoFactoryOpenRangeTag . . . . .	36
5.7.3.20	GuidoFactoryEndTag . . . . .	36
5.7.3.21	GuidoFactoryCloseTag . . . . .	36
5.7.3.22	GuidoFactoryAddTagParameterString . . . . .	37
5.7.3.23	GuidoFactoryAddTagParameterInt . . . . .	37
5.7.3.24	GuidoFactoryAddTagParameterFloat . . . . .	37
5.7.3.25	GuidoFactorySetParameterName . . . . .	37
5.7.3.26	GuidoFactorySetParameterUnit . . . . .	38
5.8	Parsing GMN files, strings and guido streams . . . . .	38
5.8.1	Function Documentation . . . . .	39
5.8.1.1	GuidoOpenParser . . . . .	39
5.8.1.2	GuidoCloseParser . . . . .	39
5.8.1.3	GuidoGetStream . . . . .	40
5.8.1.4	GuidoFile2AR . . . . .	40
5.8.1.5	GuidoString2AR . . . . .	40
5.8.1.6	GuidoStream2AR . . . . .	40
5.8.1.7	GuidoParserGetErrorCode . . . . .	41
5.8.1.8	GuidoOpenStream . . . . .	41
5.8.1.9	GuidoCloseStream . . . . .	41
5.8.1.10	GuidoWriteStream . . . . .	41
5.8.1.11	GuidoResetStream . . . . .	42
5.9	GUIDO Mapping . . . . .	42
5.9.1	Typedef Documentation . . . . .	44
5.9.1.1	Time2GraphicMap . . . . .	44
5.9.1.2	MapElement . . . . .	44
5.9.2	Enumeration Type Documentation . . . . .	44
5.9.2.1	GuidoeElementSelector . . . . .	44
5.9.2.2	GuidoElementType . . . . .	44
5.9.3	Function Documentation . . . . .	45
5.9.3.1	operator<<< . . . . .	45
5.9.3.2	operator<<< . . . . .	45
5.9.3.3	GuidoGetMap . . . . .	45
5.9.3.4	GuidoGetPageMap . . . . .	45
5.9.3.5	GuidoGetStaffMap . . . . .	46
5.9.3.6	GuidoGetVoiceMap . . . . .	46
5.9.3.7	GuidoGetSystemMap . . . . .	47
5.9.3.8	GuidoGetTime . . . . .	47
5.9.3.9	GuidoGetPoint . . . . .	47
5.9.3.10	GuidoGetSVGMap . . . . .	48
5.9.3.11	GuidoGetTimeMap . . . . .	48
5.10	Virtual Graphic System . . . . .	48
5.10.1	Detailed Description . . . . .	49

5.10.2	Define Documentation	50
5.10.2.1	ALPHA_TRANSPARENT	50
5.10.2.2	ALPHA_OPAQUE	50
5.10.3	Function Documentation	50
5.10.3.1	operator<<	50
5.10.3.2	operator+=	50
<b>6</b>	<b>Class Documentation</b>	<b>51</b>
6.1	GPaintStruct Struct Reference	51
6.1.1	Detailed Description	51
6.1.2	Member Data Documentation	51
6.1.2.1	erase	51
6.1.2.2	left	51
6.1.2.3	top	52
6.1.2.4	right	52
6.1.2.5	bottom	52
6.2	Guido2MidiParams Struct Reference	52
6.2.1	Detailed Description	53
6.2.2	Member Data Documentation	53
6.2.2.1	fTempo	53
6.2.2.2	fTicks	53
6.2.2.3	fChan	53
6.2.2.4	fIntensity	53
6.2.2.5	fAccentFactor	53
6.2.2.6	fMarcatoFactor	53
6.2.2.7	fDFactor	53
6.2.2.8	fStaccatoFactor	54
6.2.2.9	fSlurFactor	54
6.2.2.10	fTenutoFactor	54
6.2.2.11	fFermataFactor	54
6.2.2.12	fVChans	54
6.3	GuidoDate Struct Reference	54
6.3.1	Detailed Description	54
6.3.2	Member Data Documentation	54
6.3.2.1	num	54
6.3.2.2	denom	55
6.4	GuidoElementInfos Struct Reference	55
6.4.1	Member Data Documentation	55
6.4.1.1	type	55
6.4.1.2	staffNum	55
6.4.1.3	voiceNum	55
6.5	GuidoInitDesc Struct Reference	55
6.5.1	Detailed Description	56
6.5.2	Member Data Documentation	56
6.5.2.1	graphicDevice	56
6.5.2.2	reserved	56
6.5.2.3	musicFont	56
6.5.2.4	textFont	56
6.6	GuidoLayoutSettings Struct Reference	56
6.6.1	Detailed Description	57

---

6.6.2	Member Data Documentation . . . . .	57
6.6.2.1	systemsDistance . . . . .	57
6.6.2.2	systemsDistribution . . . . .	57
6.6.2.3	systemsDistribLimit . . . . .	57
6.6.2.4	force . . . . .	57
6.6.2.5	spring . . . . .	57
6.6.2.6	neighborhoodSpacing . . . . .	57
6.6.2.7	optimalPageFill . . . . .	57
6.6.2.8	resizePage2Music . . . . .	57
6.7	GuidoOnDrawDesc Struct Reference . . . . .	58
6.7.1	Detailed Description . . . . .	58
6.7.2	Member Data Documentation . . . . .	58
6.7.2.1	handle . . . . .	58
6.7.2.2	hdc . . . . .	58
6.7.2.3	page . . . . .	58
6.7.2.4	updateRegion . . . . .	59
6.7.2.5	scrollx . . . . .	59
6.7.2.6	scrolly . . . . .	59
6.7.2.7	reserved . . . . .	59
6.7.2.8	sizex . . . . .	59
6.7.2.9	sizey . . . . .	59
6.7.2.10	isprint . . . . .	59
6.8	GuidoPageFormat Struct Reference . . . . .	59
6.8.1	Detailed Description . . . . .	59
6.8.2	Member Data Documentation . . . . .	60
6.8.2.1	width . . . . .	60
6.8.2.2	height . . . . .	60
6.8.2.3	marginleft . . . . .	60
6.8.2.4	marginright . . . . .	60
6.8.2.5	marginright . . . . .	60
6.8.2.6	marginbottom . . . . .	60
6.9	MapCollector Class Reference . . . . .	60
6.9.1	Constructor & Destructor Documentation . . . . .	60
6.9.1.1	~MapCollector . . . . .	60
6.9.2	Member Function Documentation . . . . .	60
6.9.2.1	Graph2TimeMap . . . . .	60
6.10	RectInfos Class Reference . . . . .	61
6.10.1	Constructor & Destructor Documentation . . . . .	61
6.10.1.1	RectInfos . . . . .	61
6.10.1.2	~RectInfos . . . . .	61
6.10.2	Member Function Documentation . . . . .	61
6.10.2.1	time . . . . .	61
6.10.2.2	infos . . . . .	61
6.11	TimeMapCollector Class Reference . . . . .	61
6.11.1	Constructor & Destructor Documentation . . . . .	61
6.11.1.1	~TimeMapCollector . . . . .	61
6.11.2	Member Function Documentation . . . . .	61
6.11.2.1	Time2TimeMap . . . . .	61
6.12	TimeSegment Class Reference . . . . .	62
6.12.1	Detailed Description . . . . .	62

6.12.2	Constructor & Destructor Documentation . . . . .	63
6.12.2.1	TimeSegment . . . . .	63
6.12.2.2	TimeSegment . . . . .	63
6.12.2.3	TimeSegment . . . . .	63
6.12.2.4	~TimeSegment . . . . .	63
6.12.3	Member Function Documentation . . . . .	63
6.12.3.1	print . . . . .	63
6.12.3.2	empty . . . . .	63
6.12.3.3	intersect . . . . .	63
6.12.3.4	include . . . . .	63
6.12.3.5	include . . . . .	63
6.12.3.6	operator< . . . . .	63
6.12.3.7	operator== . . . . .	63
6.12.3.8	operator& . . . . .	63
6.13	VGColor Class Reference . . . . .	64
6.13.1	Detailed Description . . . . .	64
6.13.2	Constructor & Destructor Documentation . . . . .	64
6.13.2.1	VGColor . . . . .	64
6.13.2.2	VGColor . . . . .	64
6.13.2.3	VGColor . . . . .	64
6.13.2.4	VGColor . . . . .	64
6.13.3	Member Function Documentation . . . . .	65
6.13.3.1	Set . . . . .	65
6.13.3.2	Set . . . . .	65
6.13.3.3	operator== . . . . .	65
6.13.3.4	operator!= . . . . .	65
6.13.3.5	print . . . . .	65
6.13.4	Member Data Documentation . . . . .	65
6.13.4.1	mRed . . . . .	65
6.13.4.2	mGreen . . . . .	65
6.13.4.3	mBlue . . . . .	65
6.13.4.4	mAlpha . . . . .	65
6.14	VGDevice Class Reference . . . . .	66
6.14.1	Detailed Description . . . . .	69
6.14.2	Member Enumeration Documentation . . . . .	69
6.14.2.1	VRasterOpMode . . . . .	69
6.14.2.2	VTextAlignMode . . . . .	70
6.14.3	Constructor & Destructor Documentation . . . . .	70
6.14.3.1	~VGDevice . . . . .	70
6.14.4	Member Function Documentation . . . . .	70
6.14.4.1	IsValid . . . . .	70
6.14.4.2	BeginDraw . . . . .	70
6.14.4.3	EndDraw . . . . .	70
6.14.4.4	InvalidateRect . . . . .	70
6.14.4.5	MoveTo . . . . .	71
6.14.4.6	LineTo . . . . .	71
6.14.4.7	Line . . . . .	71
6.14.4.8	Frame . . . . .	71
6.14.4.9	Arc . . . . .	71
6.14.4.10	Triangle . . . . .	71



6.14.4.11	Polygon . . . . .	71
6.14.4.12	Rectangle . . . . .	72
6.14.4.13	SetMusicFont . . . . .	72
6.14.4.14	GetMusicFont . . . . .	72
6.14.4.15	SetTextFont . . . . .	72
6.14.4.16	GetTextFont . . . . .	72
6.14.4.17	selectfont . . . . .	72
6.14.4.18	SelectPen . . . . .	72
6.14.4.19	SelectFillColor . . . . .	72
6.14.4.20	PushPen . . . . .	73
6.14.4.21	PopPen . . . . .	73
6.14.4.22	PushFillColor . . . . .	73
6.14.4.23	PopFillColor . . . . .	73
6.14.4.24	SetRasterOpMode . . . . .	73
6.14.4.25	GetRasterOpMode . . . . .	73
6.14.4.26	CopyPixels . . . . .	73
6.14.4.27	CopyPixels . . . . .	74
6.14.4.28	CopyPixels . . . . .	74
6.14.4.29	CopyPixels . . . . .	74
6.14.4.30	SetScale . . . . .	74
6.14.4.31	SetOrigin . . . . .	74
6.14.4.32	OffsetOrigin . . . . .	74
6.14.4.33	LogicalToDevice . . . . .	75
6.14.4.34	DeviceToLogical . . . . .	75
6.14.4.35	GetXScale . . . . .	75
6.14.4.36	GetYScale . . . . .	75
6.14.4.37	GetXOrigin . . . . .	75
6.14.4.38	GetYOrigin . . . . .	75
6.14.4.39	NotifySize . . . . .	75
6.14.4.40	GetWidth . . . . .	75
6.14.4.41	GetHeight . . . . .	75
6.14.4.42	DrawMusicSymbol . . . . .	75
6.14.4.43	DrawString . . . . .	76
6.14.4.44	SetFontColor . . . . .	76
6.14.4.45	GetFontColor . . . . .	76
6.14.4.46	SetFontBackgroundColor . . . . .	76
6.14.4.47	GetFontBackgroundColor . . . . .	76
6.14.4.48	SetFontAlign . . . . .	76
6.14.4.49	GetFontAlign . . . . .	76
6.14.4.50	SetDPITag . . . . .	76
6.14.4.51	GetDPITag . . . . .	76
6.14.4.52	GetBitMapPixels . . . . .	77
6.14.4.53	ReleaseBitMapPixels . . . . .	77
6.14.4.54	GetImageData . . . . .	77
6.14.4.55	ReleaseImageData . . . . .	77
6.14.4.56	getVGSystem . . . . .	77
6.14.4.57	GetNativeContext . . . . .	77
6.14.4.58	SelectPenColor . . . . .	77
6.14.4.59	SelectPenWidth . . . . .	77
6.14.4.60	PushPenColor . . . . .	78

6.14.4.61	PopPenColor	78
6.14.4.62	PushPenWidth	78
6.14.4.63	PopPenWidth	78
6.14.5	Friends And Related Function Documentation	78
6.14.5.1	VGFont	78
6.14.5.2	DecoratorDevice	78
6.15	VGFont Class Reference	78
6.15.1	Detailed Description	79
6.15.2	Member Enumeration Documentation	79
6.15.2.1	"@2	79
6.15.3	Constructor & Destructor Documentation	79
6.15.3.1	~VGFont	79
6.15.4	Member Function Documentation	79
6.15.4.1	GetName	79
6.15.4.2	GetSize	79
6.15.4.3	GetProperties	79
6.15.4.4	GetExtent	79
6.15.4.5	GetExtent	80
6.15.4.6	GetContext	80
6.16	VGPen Class Reference	80
6.16.1	Detailed Description	80
6.16.2	Constructor & Destructor Documentation	80
6.16.2.1	VGPen	80
6.16.3	Member Function Documentation	80
6.16.3.1	Set	80
6.16.3.2	Set	80
6.16.3.3	Set	81
6.16.4	Member Data Documentation	81
6.16.4.1	mColor	81
6.16.4.2	mWidth	81
6.17	VGSystem Class Reference	81
6.17.1	Detailed Description	81
6.17.2	Constructor & Destructor Documentation	82
6.17.2.1	~VGSystem	82
6.17.3	Member Function Documentation	82
6.17.3.1	CreateDisplayDevice	82
6.17.3.2	CreateMemoryDevice	82
6.17.3.3	CreateMemoryDevice	82
6.17.3.4	CreatePrinterDevice	82
6.17.3.5	CreateAntiAliasedMemoryDevice	82
6.17.3.6	CreateVGFont	82
<b>7</b>	<b>File Documentation</b>	<b>83</b>
7.1	globaldoc.doxygen File Reference	83
7.2	GUIDO2Midi.h File Reference	83
7.3	GUIDOEngine.h File Reference	84
7.3.1	Typedef Documentation	87
7.3.1.1	ARHandler	87
7.3.1.2	GRHandler	87
7.3.1.3	CARHandler	87

---

7.3.1.4	CGRHandler . . . . .	87
7.3.1.5	GuidoLayoutSettings . . . . .	87
7.3.2	Enumeration Type Documentation . . . . .	87
7.3.2.1	"@0 . . . . .	87
7.3.2.2	"@1 . . . . .	88
7.3.3	Function Documentation . . . . .	88
7.3.3.1	AddGGSOutput . . . . .	88
7.3.3.2	AddGuidoOutput . . . . .	88
7.4	GUIDOExport.h File Reference . . . . .	88
7.4.1	Define Documentation . . . . .	88
7.4.1.1	class_export . . . . .	88
7.4.1.2	GUIDOAPI . . . . .	88
7.5	GUIDOFactory.h File Reference . . . . .	88
7.6	GUIDOParse.h File Reference . . . . .	90
7.7	GUIDOScoreMap.h File Reference . . . . .	91
7.8	samples.doxygen File Reference . . . . .	93
7.9	VGColor.h File Reference . . . . .	93
7.10	VGDevice.h File Reference . . . . .	93
7.11	VGFont.h File Reference . . . . .	93
7.12	VGPen.h File Reference . . . . .	94
7.13	VGSystem.h File Reference . . . . .	94



# 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 GNU LGPL 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. 66)) to the GGR objects, in place of the previous windows HDC handle.

`VGDevice` (p. 66) 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. 66) 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. 66) 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. 66) objects with a higher abstraction level than a pure graphic layer.

Existing implementations of `VGDevices`:

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

`VGDevice` (p. 66) implementations not maintained:

<code>GDeviceWin2000:</code>	Windows 2000 / XP (gdi+)
<code>GDeviceGL:</code>	OpenGL implementation
<code>GDeviceGTK:</code>	Linux GTK implementation
<code>GDevicePostScript:</code>	EPS files (encapsulated postscript)
<code>GDeviceWx:</code>	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.sizeex = 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

## Todo List

Member `GuidoFactoryInsertCommata` (p. 33)(`ARFactoryHandler inFactory`) check semantic and rename



## Chapter 4

# Bug List

**Member GuidoFindEventPage (p. 21)(CGRHandler inHandleGR, const GuidoDate (p. 54) &date)** returns page + 1 when input date falls on the last system.

**Member GuidoFindPageAt (p. 21)(CGRHandler inHandleGR, const GuidoDate (p. 54) &date)** returns page + 1 when input date falls on the last system.





## Chapter 5

# Module Documentation

### 5.1 MIDI Export

#### Classes

- struct **Guido2MidiParams**

*The **GuidoInitDesc** (p. 55) data structure contains all information required by **GuidoInit()** (p. 16)*

#### Typedefs

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

*The **GuidoInitDesc** (p. 55) data structure contains all information required by **GuidoInit()** (p. 16)*

#### Functions

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

*Export to a MIDI file.*

#### 5.1.1 Typedef Documentation

##### 5.1.1.1 typedef struct Guido2MidiParams Guido2MidiParams

The **GuidoInitDesc** (p. 55) data structure contains all information required by **GuidoInit()** (p. 16)

## 5.1.2 Function Documentation

### 5.1.2.1 `GuidoErrCode GuidoAR2MIDIFile ( const ARHandler ar, const char * filename, const Guido2MidiParams * params )`

Export to a MIDI file.

#### Parameters

<code>ar,:</code>	Guido AR handler
<code>filename,:</code>	the output file name
<code>params,:</code>	conversions parameters.

#### Returns

a Guido error code.

## 5.2 Error codes

### Enumerations

- 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.*

### 5.2.1 Enumeration Type Documentation

#### 5.2.1.1 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

## 5.3 Building abstract and graphic representations

### Functions

- **GuidoErrCode GuidoInit** (**GuidoInitDesc** \*desc)
- void **GuidoShutdown** ()
- **\_\_declspec** (deprecated("Deprecated function (will be erased soon) : use GUIDOAPI(**ARHandler**) GuidoFile2AR (GuidoParser \*parser, const char \* file) instead.)) GuidoErrCode GuidoParseFile(const char \*filename)
- **\_\_declspec** (deprecated("Deprecated function (will be erased soon) : use GUIDOAPI(**ARHandler**) GuidoString2AR (GuidoParser \*parser, const char \* str) instead.)) GuidoErrCode GuidoParseString(const char \*str)
- **GuidoErrCode GuidoAR2GR** (**ARHandler** ar, const **GuidoLayoutSettings** \*settings, **GRHandler** \*gr)
- **GuidoErrCode GuidoUpdateGR** (**GRHandler** gr, const **GuidoLayoutSettings** \*settings)
- void **GuidoFreeAR** (**ARHandler** ar)
- void **GuidoFreeGR** (**GRHandler** gr)
- const char \* **GuidoGetErrorString** (**GuidoErrCode** errCode)
- **\_\_declspec** (deprecated("Deprecated function (will be erased soon) : use GUIDOAPI(**GuidoErrCode**) GuidoParserGetErrorCode (GuidoParser\* p, int& line, int& col) instead.)) int GuidoGetParseErrorLine()
- void **GuidoGetDefaultLayoutSettings** (**GuidoLayoutSettings** \*settings)

### Variables

- **ARHandler** \* ar

#### 5.3.1 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.

## 5.3.2 Function Documentation

### 5.3.2.1 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

<i>desc</i>	the graphic environment description.
-------------	--------------------------------------

#### Returns

a Guido error code.

WARNING: the caller must ensure desc maintains a constant reference on a valid **VGDevice** (p. 66), because Guido keeps it internally (to calculate fonts, etc.)

### 5.3.2.2 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.

### 5.3.2.3 `__declspec ( deprecated("Deprecated function (will be erased soon) : use GUIDOAPI(ARHandler) GuidoFile2AR (GuidoParser *parser, const char * file) instead.") ) const`

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

#### Parameters

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

#### Returns

a Guido error code.

**5.3.2.4** `__declspec ( deprecated("Deprecated function (will be erased soon) : use GUIDOAPI(ARHandler) GuidoString2AR (GuidoParser *parser, const char * str) instead.") ) const`

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

#### Parameters

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

#### Returns

a Guido error code.

**5.3.2.5** `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

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

#### Returns

a Guido error code.

**5.3.2.6** `GuidoErrCode GuidoUpdateGR ( GRHandler gr, const GuidoLayoutSettings * settings )`

Applies new layout settings to an existing Guido graphic representation.

#### Parameters

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

**Returns**

a Guido error code.

**5.3.2.7 void GuidoFreeAR ( ARHandler *ar* )**

Releases a Guido abstract representation.

**Parameters**

<i>ar</i>	the handler to the abstract representation.
-----------	---

**5.3.2.8 void GuidoFreeGR ( GRHandler *gr* )**

Releases a Guido graphic representation.

**Parameters**

<i>gr</i>	the handler to the graphic representation.
-----------	--

**5.3.2.9 const char\* GuidoGetErrorString ( GuidoErrCode *errCode* )**

Gives a textual description of a Guido error code.

**Parameters**

<i>errCode</i>	a Guido error code.
----------------	---------------------

**Returns**

a string describing the error.

**5.3.2.10 \_\_declspec ( deprecated("Deprecated function (will be erased soon) : use GUIDOAPI(GuidoErrCode) GuidoParserGetErrorCode (GuidoParser\* p, int& line, int& col) instead.") )**

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

**Returns**

a line number.

**5.3.2.11 void GuidoGetDefaultLayoutSettings ( GuidoLayoutSettings \* *settings* )**

Gives the default values of the layout settings.

**Parameters**

<i>settings</i>	on output, a pointer to the settings to be filled with default values.
-----------------	--

**5.3.3 Variable Documentation****5.3.3.1 ARHandler \* ar****5.4 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.*

**5.4.1 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.

**5.4.2 Function Documentation****5.4.2.1 int GuidoCountVoices ( CARHandler inHandleAR )**

Gives the number of score pages of the graphic representation.

**Parameters**

<i>inHandleAR</i>	a Guido opaque handle to a AR structure.
-------------------	--

**Returns**

the number of voices or a guido error code.

**5.4.2.2 int GuidoGetPageCount ( CGRHandler *inHandleGR* )**

Gives the number of score pages of the graphic representation.

**Parameters**

<i>inHandleGR</i>	a Guido opaque handle to a GR structure.
-------------------	--

**Returns**

a number of pages or a guido error code.

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

Gives the number of systems on a given page.

**Parameters**

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

**Returns**

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

**5.4.2.4 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**

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

**Returns**

a Guido error code.



#### 5.4.2.5 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

<i>inHandleGR</i>	a Guido opaque handle to a GR structure.
<i>date</i>	the target date.

##### Returns

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

#### 5.4.2.6 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

<i>inHandleGR</i>	a Guido opaque handle to a GR structure.
<i>date</i>	the target date.

##### Returns

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

#### 5.4.2.7 GuidoErrCode GuidoGetPageDate ( CGRHandler *inHandleGR*, int *pageNum*, GuidoDate \* *date* )

Gives the time location of a Page.

##### Parameters

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

##### Returns

a Guido error code.

## 5.5 Score drawing and pages formatting

### 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.*
- 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.*

### 5.5.1 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.

### 5.5.2 Function Documentation

#### 5.5.2.1 GuidoErrCode GuidoOnDraw ( GuidoOnDrawDesc \* *desc* )

Draws one page of score into a graphic device.

##### Parameters

<i>desc</i>	informations about what to draw and how to draw.
-------------	--

##### Returns

a Guido error code

#### 5.5.2.2 GuidoErrCode GuidoSVGExport ( const GRHandler *handle*, int *page*, std::ostream & *out*, const char \* *fontfile* )

Exports one page of score to SVG.

##### Parameters

<i>page</i>	the page number.
<i>out</i>	the output stream.
<i>fontfile</i>	path of the guido svg font file.

##### Returns

a Guido error code

#### 5.5.2.3 void GuidoDrawBoundingBoxes ( int *bbMap* )

Control bounding boxes drawing.

##### Parameters

<i>bbMap</i>	a bits field indicating the set of bounding boxes to draw (default to none).
--------------	--

#### 5.5.2.4 int GuidoGetDrawBoundingBoxes ( )

Gives bounding boxes drawing state.

### 5.5.2.5 void GuidoGetPageFormat ( CGRHandler *inHandleGR*, int *pageNum*, GuidoPageFormat \* *format* )

Gives a score page format.

#### Parameters

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

### 5.5.2.6 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

<i>format</i>	the page format
---------------	-----------------

### 5.5.2.7 void GuidoGetDefaultPageFormat ( GuidoPageFormat \* *format* )

Gives the default score page format.

#### Parameters

<i>format</i>	on output: the page format
---------------	----------------------------

### 5.5.2.8 float GuidoUnit2CM ( float *val* )

Converts internal Guido units into centimeters.

#### Parameters

<i>val</i>	the value to be converted
------------	---------------------------

**Returns**

the converted value

**5.5.2.9 float GuidoCM2Unit ( float *val* )**

Converts centimeters into internal Guido units.

**Parameters**

<i>val</i>	the value to be converted
------------	---------------------------

**Returns**

the converted value

**5.5.2.10 float GuidoUnit2Inches ( float *val* )**

Converts internal Guido units into inches.

**Parameters**

<i>val</i>	the value to be converted
------------	---------------------------

**Returns**

the converted value

**5.5.2.11 float GuidoInches2Unit ( float *val* )**

Converts inches into internal Guido units.

**Parameters**

<i>val</i>	the value to be converted
------------	---------------------------

**Returns**

the converted value

**5.5.2.12 GuidoErrCode GuidoResizePageToMusic ( GRHandler *inHandleGR* )**

Resize the page sizes to the music size.

**Parameters**

<i>inHandleGR</i>	a Guido opaque handle to a GR structure.
-------------------	--

**Returns**

a Guido error code.

**5.6 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.*

**5.6.1 Detailed Description**

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

**5.6.2 Function Documentation****5.6.2.1 void GuidoGetVersionNums ( int \* major, int \* minor, int \* sub )**

Gives the library version number as three integers.

Version number format is MAJOR.MINOR.SUB

**Parameters**

<i>major</i>	on output: the major revision number.
<i>minor</i>	on output: the minor revision number.
<i>sub</i>	on output: the sub revision number.

**Returns**

a Guido error code.

**5.6.2.2 const char\* GuidoGetVersionStr ( )**

Gives the library version number as a string.

**Returns**

the version number as a string.

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

Checks a required library version number.

**Parameters**

<i>major</i>	the major revision number.
<i>minor</i>	the minor revision number.
<i>sub</i>	the sub revision number.

**Returns**

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

**5.6.2.4 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.

**5.6.2.5** `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

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

#### Returns

a Guido error code.

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

Makes the correspondance between an ARMusic and a path.

#### Parameters

<i>inHandleAR</i>	the destination ARHandler.
<i>inPath</i>	the path to associate.

#### Returns

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

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

Returns the path corresponding to an AR.

#### Parameters

<i>inHandleAR</i>	the handle given to extract its path.
-------------------	---------------------------------------



**Returns**

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

## 5.7 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.*

### 5.7.1 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. 32)), 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. 32) and `GuidoFactoryCloseMusic()` (p. 32)
- the current voice: modified by `GuidoFactoryOpenVoice()` (p. 32) and `GuidoFactoryCloseVoice()` (p. 33)
- the current chord: modified by `GuidoFactoryOpenChord()` (p. 33) and `GuidoFactoryCloseChord()` (p. 33)
- the current event: modified by `GuidoFactoryOpenEvent()` (p. 34) and `GuidoFactoryCloseEvent()` (p. 34)
- the current tag: modified by `GuidoFactoryOpenTag()` (p. 36) and `GuidoFactoryCloseTag()` (p. 36)

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

- the current register: modified by `GuidoFactorySetRegister()`
- the current duration: modified by `GuidoFactorySetDuration()` (p. 35)

### 5.7.2 Typedef Documentation

#### 5.7.2.1 typedef void\* ARFactoryHandler

### 5.7.3 Function Documentation

#### 5.7.3.1 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.

### 5.7.3.2 void GuidoFactoryClose ( ARFactoryHandler *inFactory* )

Closes the Guido Factory.

Must be called to release the factory associated resources.

### 5.7.3.3 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. 32)

#### Returns

an integer that is an error code if not null.

### 5.7.3.4 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 logical 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()`

### 5.7.3.5 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. 33) Voices are similar to sequence is GMN.

#### Returns

an error code

**5.7.3.6 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

**5.7.3.7 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. 33)

**Returns**

an error code

**5.7.3.8 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

**5.7.3.9 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`)

**Todo**

check semantic and rename

**Returns**

an error code

**5.7.3.10 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. 34)

**Returns**

an error code

**5.7.3.11 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

**5.7.3.12 GuidoErrCode GuidoFactoryAddSharp ( ARFactoryHandler *inFactory* )**

Adds a sharp to the current event.

The current event must be a note.

**Returns**

an error code

**5.7.3.13 GuidoErrCode GuidoFactoryAddFlat ( ARFactoryHandler *inFactory* )**

Add a flat to the current event.

The current event must be a note.

**Returns**

an error code.

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

Sets the number of dots the current event.

**Parameters**

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

**Returns**

an error code.

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

Sets the accidentals of the current event.

**Parameters**

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

**Returns**

an error code.

**5.7.3.16 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**

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

**Returns**

an error code.

**5.7.3.17 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**

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

**Returns**

an error code.

**5.7.3.18 GuidoErrCode GuidoFactoryOpenTag ( ARFactoryHandler *inFactory*, const char \* *name*, long *tagID* )**

**5.7.3.19 GuidoErrCode GuidoFactoryOpenRangeTag ( ARFactoryHandler *inFactory*, const char \* *name*, long *tagID* )**

**5.7.3.20 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**

<i>inFactory</i>	a handler to a Guido Factory (created with GuidoFactoryOpen)
------------------	--

**Returns**

an error code.

**5.7.3.21 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**

<i>inFactory</i>	a handler to a Guido Factory (created with GuidoFactoryOpen)
------------------	--

**Returns**

an error code.



### 5.7.3.22 `GuidoErrCode GuidoFactoryAddTagParameterString ( ARFactoryHandler inFactory, const char * val )`

Adds a new string parameter to the current tag.

#### Parameters

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

#### Returns

an error code.

### 5.7.3.23 `GuidoErrCode GuidoFactoryAddTagParameterInt ( ARFactoryHandler inFactory, int val )`

Adds a new integer parameter to the current tag.

#### Parameters

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

#### Returns

an error code.

### 5.7.3.24 `GuidoErrCode GuidoFactoryAddTagParameterFloat ( ARFactoryHandler inFactory, double val )`

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

#### Parameters

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

#### Returns

an error code.

### 5.7.3.25 `GuidoErrCode GuidoFactorySetParameterName ( ARFactoryHandler inFactory, const char * name )`

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

**Parameters**

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

**Returns**

an error code.

### 5.7.3.26 GuidoErrCode GuidoFactorySetParameterUnit ( ARFactoryHandler *inFactory*, const char \* *unit* )

Defines the unit of the last added tag-parameter.

**Parameters**

<i>inFactory</i>	a handler to a Guido Factory (created with GuidoFactoryOpen)
<i>unit,:</i>	a string defining the unit. The following units are supported: <ul style="list-style-type: none"> <li>• m - meter</li> <li>• cm - centimeter</li> <li>• mm - millimeter</li> <li>• in - inch</li> <li>• pt - point (= 1/72.27 inch)</li> <li>• pc - pica (= 12pt)</li> <li>• hs - halfspace (half of the space between two lines of the current staff)</li> <li>• rl - relative measure in percent (used for positioning on score page)</li> </ul>

**Returns**

an error code.

## 5.8 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 resources.*
- 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.*

## 5.8.1 Function Documentation

### 5.8.1.1 GuidoParser\* GuidoOpenParser ( )

Creates a new parser.

#### Returns

a guido parser.

### 5.8.1.2 GuidoErrCode GuidoCloseParser ( GuidoParser \* p )

Close a guido parser and releases all the associated ressources.

#### Parameters

<i>p</i>	a parser previously opened with GuidoOpenParser
----------	---

#### Returns

a Guido error code.

**5.8.1.3 const char\* GuidoGetStream ( GuidoStream \* *gStream* )**

returns the string of the GuidoStream

**Parameters**

<i>gStream</i>	a GuidoStream
----------------	---------------

**Returns**

a std::string.

**5.8.1.4 ARHandler GuidoFile2AR ( GuidoParser \* *p*, const char \* *file* )**

Parse a file and create the corresponding AR.

**Parameters**

<i>p</i>	a parser previously opened with GuidoOpenParser
<i>file</i>	the file to parse.

**Returns**

a ARHandler or 0 in case of error.

**5.8.1.5 ARHandler GuidoString2AR ( GuidoParser \* *p*, const char \* *str* )**

Parse a string and create the corresponding AR.

**Parameters**

<i>p</i>	a parser previously opened with GuidoOpenParser
<i>str</i>	the string to parse.

**Returns**

a ARHandler or 0 in case of error.

**5.8.1.6 ARHandler GuidoStream2AR ( GuidoParser \* *p*, GuidoStream \* *stream* )**

Parse a GuidoStream and create the corresponding AR.

**Parameters**

<i>p</i>	a parser previously opened with GuidoOpenParser
<i>stream</i>	the GuidoStream to parse.

**Returns**

a ARHandler or 0 in case of error.

**5.8.1.7 GuidoErrCode GuidoParserGetErrorCode ( GuidoParser \* *p*, int & *line*, int & *col*, const char \*\* *msg* )**

Get the error syntax line/column.

**Parameters**

<i>p</i>	a parser previously opened with GuidoOpenParser
<i>line</i>	a reference that will contain a line number in case of syntax error
<i>col</i>	a reference that will contain a column number in case of syntax error
<i>msg</i>	a string that will contain the error message

**Returns**

a Guido error code.

**5.8.1.8 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.

**5.8.1.9 GuidoErrCode GuidoCloseStream ( GuidoStream \* *s* )**

Close a guido stream.

**Parameters**

<i>s</i>	a GuidoStream
----------	---------------

**Returns**

a Guido error code.

**5.8.1.10 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 writing 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

<i>s</i>	a GuidoStream previously opened with GuidoOpenStream
<i>str</i>	a string containing a portion of gmn code

#### Returns

a Guido error code.

#### 5.8.1.11 GuidoErrCode GuidoResetStream ( GuidoStream \* s )

Erase all stream content in order to reuse it.

#### Parameters

<i>s</i>	a GuidoStream previously opened with GuidoOpenStream
----------	--

#### Returns

a Guido error code.

## 5.9 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.*

## 5.9.1 Typedef Documentation

5.9.1.1 typedef std::vector<std::pair<**TimeSegment**, **FloatRect**> > **Time2GraphicMap**

5.9.1.2 typedef std::pair<**FloatRect**, **RectInfos**> **MapElement**

## 5.9.2 Enumeration Type Documentation

5.9.2.1 enum **GuidoeElementSelector**

Enumerator:

*kGuidoPage*  
*kGuidoSystem*  
*kGuidoSystemSlice*  
*kGuidoStaff*  
*kGuidoBar*  
*kGuidoEvent*  
*kGuidoScoreElementEnd*

5.9.2.2 enum **GuidoElementType**

Enumerator:

*kNote*  
*kRest*  
*kEmpty*  
*kBar*



*kRepeatBegin**kRepeatEnd**kStaff**kSystemSlice**kSystem**kPage*

### 5.9.3 Function Documentation

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

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

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

References TimeSegment::print().

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

Retrieves the graphic to time mapping.

#### Parameters

<i>gr</i>	a Guido opaque handle to a GR structure.
<i>pagenum</i>	a page index, starting from 1.
<i>width</i>	the page width.
<i>height</i>	the page height.
<i>sel</i>	GuidoeElementSelector to filter undesired objects out.
<i>f</i>	a <b>MapCollector</b> (p. 60) object that will be called for each selected element.

#### Returns

an error code.

5.9.3.4 `GuidoErrCode GuidoGetPageMap ( CGRHandler gr, int pagenum, float w, float h,`  
`Time2GraphicMap & outmap )`

Retrieves a guido page graphic to time mapping.

#### Parameters

<i>gr</i>	a Guido opaque handle to a GR structure.
-----------	--

<i>pagenum</i>	a page index, starting from 1.
<i>w</i>	the page width.
<i>h</i>	the page height.
<i>outmap</i>	contains the mapping on output.

**Returns**

an error code.

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

Retrieves a guido staff graphic to time mapping.

**Parameters**

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

**Returns**

an error code.

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

Retrieves a guido voice graphic to time mapping.

**Parameters**

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

**Returns**

an error code.

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

Retrieves a guido system graphic to time mapping.

#### Parameters

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

#### Returns

an error code.

### 5.9.3.8 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

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

#### Returns

false when there is no segment containing the date.

### 5.9.3.9 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

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

**Returns**

false when there is no segment containing the point.

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

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

**Parameters**

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

**Returns**

an error code.

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

Retrieves the rolled to unrolled time mapping.

**Parameters**

<i>gr</i>	a Guido opaque handle to a GR structure.
<i>f</i>	a <b>TimeMapCollector</b> (p. 61) object that will be called for each time segment.

**Returns**

an error code.

## 5.10 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)

### 5.10.1 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. 81) class: to cover allocation of specific **VGDevice** (p. 66) and **VGFont** (p. 78) objects.
- a **VGDevice** (p. 66) class: specialized on drawing onscreen or offscreen
- a **VGFont** (p. 78) 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.

## 5.10.2 Define Documentation

5.10.2.1 `#define ALPHA_TRANSPARENT 0`

5.10.2.2 `#define ALPHA_OPAQUE 255`

## 5.10.3 Function Documentation

5.10.3.1 `std::ostream& operator<< ( std::ostream & out, const VColor & c )` [inline]

5.10.3.2 `VColor & operator+=( short v )` [inline, inherited]

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

## Chapter 6

# Class Documentation

### 6.1 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**

#### 6.1.1 Detailed Description

A structure to keep information about clipping and redrawing regions.

#### 6.1.2 Member Data Documentation

##### 6.1.2.1 bool erase

a flag to ignore the following rect and to redraw everything

##### 6.1.2.2 int left

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

6.1.2.3 int top

6.1.2.4 int right

6.1.2.5 int bottom

## 6.2 Guido2MidiParams Struct Reference

The `GuidoInitDesc` (p. 55) data structure contains all information required by `GuidoInit()` (p. 16)

### 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)*

### 6.2.1 Detailed Description

The `GuidoInitDesc` (p. 55) data structure contains all information required by `GuidoInit()` (p. 16)

### 6.2.2 Member Data Documentation

#### 6.2.2.1 `int fTempo`

default tempo in quarter per minute - default value: 120

#### 6.2.2.2 `int fTicks`

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

#### 6.2.2.3 `int fChan`

the default Midi channel - default value: 1

#### 6.2.2.4 `float flntensity`

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

#### 6.2.2.5 `float fAccentFactor`

accent intensity factor - default value: 1.1

#### 6.2.2.6 `float fMarcatoFactor`

marcato intensity factor - default value: 1.2

#### 6.2.2.7 `float fDFactor`

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

**6.2.2.8 float fStaccatoFactor**

staccato duration factor - default value: 0.5

**6.2.2.9 float fSlurFactor**

legato duration factor - default value: 1.0

**6.2.2.10 float fTenutoFactor**

tenuto duration factor - default value: 0.90

**6.2.2.11 float fFermataFactor**

fermata duration factor - default value: 2.0

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

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

## 6.3 GuidoDate Struct Reference

### Public Attributes

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

### 6.3.1 Detailed Description

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

### 6.3.2 Member Data Documentation

#### 6.3.2.1 int num

the date numerator

Referenced by operator<<().

### 6.3.2.2 int denom

the date denominator

Referenced by operator<<().

## 6.4 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*

### 6.4.1 Member Data Documentation

#### 6.4.1.1 GuidoElementType type

the element type

#### 6.4.1.2 int staffNum

the element staff number or 0 when na

#### 6.4.1.3 int voiceNum

the element voice number or 0 when na

## 6.5 GuidoInitDesc Struct Reference

The **GuidoInitDesc** (p. 55) data structure contains all information required by **GuidoInit()** (p. 16)

### 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*

### 6.5.1 Detailed Description

The **GuidoInitDesc** (p. 55) data structure contains all information required by **GuidoInit()** (p. 16)

### 6.5.2 Member Data Documentation

#### 6.5.2.1 VGDevice\* graphicDevice

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

#### 6.5.2.2 void\* reserved

#### 6.5.2.3 const char\* musicFont

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

#### 6.5.2.4 const char\* textFont

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

## 6.6 GuidoLayoutSettings Struct Reference

### Public Attributes

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

## 6.6.1 Detailed Description

Settings for the graphic score layout.

## 6.6.2 Member Data Documentation

### 6.6.2.1 float systemsDistance

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

### 6.6.2.2 int systemsDistribution

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

### 6.6.2.3 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)

### 6.6.2.4 float force

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

### 6.6.2.5 float spring

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

### 6.6.2.6 int neighborhoodSpacing

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

### 6.6.2.7 int optimalPageFill

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

### 6.6.2.8 int resizePage2Music

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

## 6.7 GuidoOnDrawDesc Struct Reference

Contains all graphic-related information required by `GuidoOnDraw()` (p. 23)

### 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**

### 6.7.1 Detailed Description

Contains all graphic-related information required by `GuidoOnDraw()` (p. 23) Used to render a page of score into a device context.

### 6.7.2 Member Data Documentation

#### 6.7.2.1 GRHandler handle

A Guido handler to a graphic representation.

#### 6.7.2.2 VGDevice\* hdc

A graphic device context.

#### 6.7.2.3 int page

The page number. Starts from 1.

#### 6.7.2.4 GPaintStruct updateRegion

Indicates what to (re)draw.

#### 6.7.2.5 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.

#### 6.7.2.6 int scrolly

#### 6.7.2.7 float reserved

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

#### 6.7.2.8 int sizex

#### 6.7.2.9 int sizey

#### 6.7.2.10 int isprint

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

## 6.8 GuidoPageFormat Struct Reference

### Public Attributes

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

### 6.8.1 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`

## 6.8.2 Member Data Documentation

6.8.2.1 float width

6.8.2.2 float height

6.8.2.3 float marginleft

6.8.2.4 float margintop

6.8.2.5 float marginright

6.8.2.6 float marginbottom

## 6.9 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*

### 6.9.1 Constructor & Destructor Documentation

6.9.1.1 virtual `~MapCollector ( ) [inline, virtual]`

### 6.9.2 Member Function Documentation

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

a method called by the GuidoGetMap function

#### Parameters

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



## 6.10 RectInfos Class Reference

### Public Member Functions

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

### 6.10.1 Constructor & Destructor Documentation

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

6.10.1.2 virtual **~RectInfos** ( ) [inline, virtual]

### 6.10.2 Member Function Documentation

6.10.2.1 const **TimeSegment**& **time** ( ) const [inline]

6.10.2.2 const **GuidoElementInfos**& **infos** ( ) const [inline]

## 6.11 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*

### 6.11.1 Constructor & Destructor Documentation

6.11.1.1 virtual **~TimeMapCollector** ( ) [inline, virtual]

### 6.11.2 Member Function Documentation

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

a method called by the GuidoGetTimeMap function

#### Parameters

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

## 6.12 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)*

### 6.12.1 Detailed Description

a time segment definition and operations

## 6.12.2 Constructor & Destructor Documentation

6.12.2.1 `TimeSegment ( )` [inline]

6.12.2.2 `TimeSegment ( const TimeSegment & s )` [inline]

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

6.12.2.4 `virtual ~TimeSegment ( )` [inline, virtual]

## 6.12.3 Member Function Documentation

6.12.3.1 `void print ( std::ostream & out ) const`

print the time segment

Referenced by operator<<().

6.12.3.2 `bool empty ( ) const`

check for empty segment

6.12.3.3 `bool intersect ( const TimeSegment & ts ) const`

check for segments intersection

6.12.3.4 `bool include ( const GuidoDate & date ) const`

check for date inclusion

6.12.3.5 `bool include ( const TimeSegment & ts ) const`

check for segment inclusion

6.12.3.6 `bool operator< ( const TimeSegment & ts ) const`

order relationship: the smaller is the smaller first date

6.12.3.7 `bool operator== ( const TimeSegment & ts ) const`

6.12.3.8 `TimeSegment operator& ( const TimeSegment & ts ) const`

intersection operation (may return an arbitrary empty segment)

## 6.13 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**

#### 6.13.1 Detailed Description

Generic class to manipulate device independant colors.

#### 6.13.2 Constructor & Destructor Documentation

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

6.13.2.2 **VGColor** ( const **VGColor** & *in* ) [inline]

References Set().

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

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

References Set().

### 6.13.3 Member Function Documentation

**6.13.3.1** `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().

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

References mAlpha, mBlue, mGreen, and mRed.

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

References mAlpha, mBlue, mGreen, and mRed.

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

References mAlpha, mBlue, mGreen, and mRed.

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

References mAlpha, mBlue, mGreen, and mRed.

### 6.13.4 Member Data Documentation

**6.13.4.1** `unsigned char mRed`

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

**6.13.4.2** `unsigned char mGreen`

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

**6.13.4.3** `unsigned char mBlue`

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

**6.13.4.4** `unsigned char mAlpha`

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

## 6.14 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. 78).*
- virtual void **SetTextFont** (const **VGFont** \*font)=0
- virtual const **VGFont** \* **GetTextFont** () const =0  
*Returns the currently selected text VGFont (p. 78).*

- virtual void **selectfont** (int font)  
*Selects a font (only for SVG device).*
- virtual void **SelectPen** (const **VGColor** &inColor, float width)=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. 66) to the input values.*
- virtual void **SetOrigin** (float x, float y)=0  
*Specifies which **VGDevice** (p. 66) 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. 66).*
- virtual int **GetHeight** () const =0  
*Returns the height (set via **NotifySize**) of the current **VGDevice** (p. 66).*
- 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. 66).*

- virtual **VGColor** **GetFontColor** () const =0  
*Returns the text/music color of the current VGDevice (p. 66).*
- virtual void **SetFontBackgroundColor** (const **VGColor** &inColor)=0  
*Sets the text/music background color for the current VGDevice (p. 66).*
- virtual **VGColor** **GetFontBackgroundColor** () const =0  
*Returns the text/music background color of the current VGDevice (p. 66).*
- 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. 66).*
- virtual float **GetDPITag** () const =0  
*Returns the printing resolution of the current VGDevice (p. 66).*
- 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. 80) object with the specified VGColor (p. 64).*
- virtual void **SelectPenWidth** (float width)=0  
*Creates a new VGPen (p. 80) object with the specified VGColor (p. 64).*
- 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**

### 6.14.1 Detailed Description

Generic platform independant drawing device. **VGDevice** (p. 66) 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. 66) derived classes must then be provided by client applications.

**VGDevice** (p. 66) thus provides standard graphic functions (Lines, Arc, Rectangles, Polygons, Text), coordinate transformation (zoom / scaling), and symbolic music symbols handlers.

A **VGDevice** (p. 66) 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. 66) for double buffering mechanism, it also provides bit-block copy operations from one **VGDevice** (p. 66) to another.

Each **VGDevice** (p. 66) needs to be dynamically associated with external music and text **VGFont** (p. 78) 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.

### 6.14.2 Member Enumeration Documentation

#### 6.14.2.1 enum VRasterOpMode

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

Enumerator:

- kUnknown*
- kOpCopy*
- kOpAnd*
- kOpXOr*

*kOpInvert*

*kOpOr*

#### 6.14.2.2 enum VTextAlignMode

Text alignment modes.

Enumerator:

*kAlignBase*

*kAlignBottom*

*kAlignTop*

*kAlignCenter*

*kAlignLeft*

*kAlignRight*

*kAlignBaseLeft*

### 6.14.3 Constructor & Destructor Documentation

6.14.3.1 virtual `~VGDevice ( )` [`inline`, `virtual`]

### 6.14.4 Member Function Documentation

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

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

6.14.4.2 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.

6.14.4.3 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.

6.14.4.4 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.

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

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

**6.14.4.6** `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).

**6.14.4.7** `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).

**6.14.4.8** `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.

**6.14.4.9** `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 coordinates 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.

**6.14.4.10** `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.

**6.14.4.11** `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.

**6.14.4.12** `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.

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

Selects the specified music **VGFont** (p.78) into the current **VGDevice** (p.66). Warning ! this method doesn't return the previously selected font anymore. Use **GetMusicFont()** (p.72) instead.

**6.14.4.14** `virtual const VGFont* GetMusicFont ( ) const` [pure virtual]

Returns the currently selected music **VGFont** (p.78).

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

Selects the specified text **VGFont** (p.78) into the current **VGDevice** (p.66). Warning ! this method doesn't return the previously selected font anymore. Use **GetMusicFont()** (p.72) instead.

**6.14.4.16** `virtual const VGFont* GetTextFont ( ) const` [pure virtual]

Returns the currently selected text **VGFont** (p.78).

**6.14.4.17** `virtual void selectfont ( int font )` [inline, virtual]

Selects a font (only for SVG device).

**6.14.4.18** `virtual void SelectPen ( const VGColor & inColor, float width )` [pure virtual]

Creates a new **VGPen** (p.80) object with the specified **VGColor** (p.64) and width and selects it into the current **VGDevice** (p.66).

**6.14.4.19** `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.64) and selects it into the current **VGDevice** (p.66). This method was previously called **SelectBrush()**.

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

Saves the current **VGPen** (p. 80) and selects the new one using the specified **VGColor** (p. 64) and width into the current **VGDevice** (p. 66).

**6.14.4.21** `virtual void PopPen ( )` [pure virtual]

Restores the previous **VGPen** (p. 80) object from the stack into the current **VGDevice** (p. 66).

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

Saves the current color brush and selects the new one using the specified **VGColor** (p. 64) into the current **VGDevice** (p. 66). This method was previously called `PushBrush()`.

**6.14.4.23** `virtual void PopFillColor ( )` [pure virtual]

Restores the previous color brush from the stack into the current **VGDevice** (p. 66). This method was previously called `PopBrush()`.

**6.14.4.24** `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.

**6.14.4.25** `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.

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

Copies the entire content (pixmap) of the `pSrcDC` source **VGDevice** (p. 66) to the current device. The raster operation mode should be specified using `SetRasterOpMode()` (p. 73). 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.

**6.14.4.27** `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.66) to the specified destination of the current device. The raster operation mode should be specified using **SetRasterOpMode()** (p.73). 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.

**6.14.4.28** `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.

**6.14.4.29** `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.

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

Sets the scale factors of the current **VGDevice** (p.66) to the input values.

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

Specifies which **VGDevice** (p.66) point (x,y) maps to the window origin (0,0).

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

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

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

Computes input coordinates to get their values in the current **VGDevice** (p. 66) coordinate system.

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

Computes input **VGDevice** (p. 66) coordinates to get their values outside its **VGDevice** (p. 66) coordinate system.

**6.14.4.35** `virtual float GetXScale ( ) const` [pure virtual]

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

**6.14.4.36** `virtual float GetYScale ( ) const` [pure virtual]

**6.14.4.37** `virtual float GetXOrigin ( ) const` [pure virtual]

**6.14.4.38** `virtual float GetYOrigin ( ) const` [pure virtual]

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

Sets the size of the current **VGDevice** (p. 66). Use this method to update the derived device's attributes and actual size.

**6.14.4.40** `virtual int GetWidth ( ) const` [pure virtual]

Returns the width (set via NotifySize) of the current **VGDevice** (p. 66).

**6.14.4.41** `virtual int GetHeight ( ) const` [pure virtual]

Returns the height (set via NotifySize) of the current **VGDevice** (p. 66).

**6.14.4.42** `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.

**6.14.4.43** `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.

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

Sets the text/music color for the current `VGDevice` (p. 66).

**6.14.4.45** `virtual VGColor GetFontColor ( ) const` [pure virtual]

Returns the text/music color of the current `VGDevice` (p. 66).

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

Sets the text/music background color for the current `VGDevice` (p. 66).

**6.14.4.47** `virtual VGColor GetFontBackgroundColor ( ) const` [pure virtual]

Returns the text/music background color of the current `VGDevice` (p. 66).

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

Sets the text/music alignment mode of the current `VGDevice` (p. 66). See enum `VTextAlignMode` above.

**6.14.4.49** `virtual unsigned int GetFontAlign ( ) const` [pure virtual]

Returns the text/music alignment mode of the current `VGDevice` (p. 66). See enum `VTextAlignMode` above.

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

Sets the printing resolution of the current `VGDevice` (p. 66).

**6.14.4.51** `virtual float GetDPITag ( ) const` [pure virtual]

Returns the printing resolution of the current `VGDevice` (p. 66).



**6.14.4.52** `virtual void* GetBitMapPixels ( ) [pure virtual]`

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

**6.14.4.53** `virtual void ReleaseBitMapPixels ( ) [pure virtual]`

Update bitmap pixels and ends pixels operations.

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

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

**6.14.4.55** `virtual void ReleaseImageData ( const char * ) const [pure virtual]`

Release the pointer returned by GetImageData.

**6.14.4.56** `virtual VGSystem* getVGSystem ( ) const [pure virtual]`

temporary hack - must be removed asap

**6.14.4.57** `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().

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

Creates a new **VGPen** (p. 80) object with the specified **VGColor** (p. 64).

**6.14.4.59** `virtual void SelectPenWidth ( float width ) [pure virtual]`

Creates a new **VGPen** (p. 80) object with the specified **VGColor** (p. 64).

6.14.4.60 `virtual void PushPenColor ( const VGColor & inColor )` [pure virtual]

6.14.4.61 `virtual void PopPenColor ( )` [pure virtual]

6.14.4.62 `virtual void PushPenWidth ( float width )` [pure virtual]

6.14.4.63 `virtual void PopPenWidth ( )` [pure virtual]

## 6.14.5 Friends And Related Function Documentation

6.14.5.1 `friend class VGFont` [friend]

6.14.5.2 `friend class DecoratorDevice` [friend]

## 6.15 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`

### 6.15.1 Detailed Description

Generic pure virtual & device-independant font class. This class replaces the previously defined GFontInfos class that was attached to a **VGDevice** (p. 66). 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. 81) and then associated when necessary to the **VGDevice** (p. 66).

### 6.15.2 Member Enumeration Documentation

#### 6.15.2.1 anonymous enum

Font properties.

Enumerator:

*kFontNone*

*kFontBold*

*kFontItalic*

*kFontUnderline*

### 6.15.3 Constructor & Destructor Documentation

6.15.3.1 `virtual ~VGFont( ) [inline, virtual]`

### 6.15.4 Member Function Documentation

6.15.4.1 `virtual const char* GetName( ) const [pure virtual]`

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

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

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

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

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

6.15.4.4 `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. 66)

**6.15.4.5** `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. 66)

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

References `VGDevice::GetNativeContext()`.

## 6.16 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`

#### 6.16.1 Detailed Description

Generic class to manipulate device independant pens.

#### 6.16.2 Constructor & Destructor Documentation

**6.16.2.1** `VGPen ( )` [inline]

#### 6.16.3 Member Function Documentation

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

References `mColor`, and `mWidth`.

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

References `mColor`.

### 6.16.3.3 void Set ( float *inWidth* ) [inline]

References `mWidth`.

## 6.16.4 Member Data Documentation

### 6.16.4.1 VGColor `mColor`

Referenced by `Set()`.

### 6.16.4.2 float `mWidth`

Referenced by `Set()`.

## 6.17 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. 66).*
- 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. 78) using the specified parameters.*

### 6.17.1 Detailed Description

Generic pure virtual class for manipulating platform independant drawing devices and fonts. A `VGSystem` (p. 81) 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 instantiate an object of the appropriate platform dependant `VGSystem` (p. 81) derived class using the correct platform dependant parameters.

A `VGSystem` (p. 81) object deals mainly with two kinds of graphical objects:

- **VGDevice** (p. 66) objects, which can be display devices, memory devices or printer devices;
- **VGFont** (p. 78) objects, used to describe device independant fonts (for music or text).

## 6.17.2 Constructor & Destructor Documentation

6.17.2.1 `virtual ~VGSystem ( ) [inline, virtual]`

## 6.17.3 Member Function Documentation

6.17.3.1 `virtual VGDevice* CreateDisplayDevice ( ) [pure virtual]`

Creates and returns a pointer to a new display **VGDevice** (p.66) which can be used to draw directly on the screen.

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

Creates and returns a pointer to a new memory **VGDevice** (p.66) compatible with the application's current screen. This device can be used to draw into a bitmap.

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

Creates and returns a pointer to a new memory **VGDevice** (p.66) compatible with the file (pixmap) located at the specified path.

6.17.3.4 `virtual VGDevice* CreatePrinterDevice ( ) [pure virtual]`

Creates and returns a pointer to a new printer **VGDevice** (p.66).

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

Creates and returns a pointer to a new memory **VGDevice** (p.66) compatible with the application's current screen and using anti-aliasing capabilities. This device can be used to draw into a bitmap.

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

Creates and returns a pointer to a new **VGFont** (p.78) using the specified parameters.

## Chapter 7

# File Documentation

### 7.1 globaldoc.doxygen File Reference

### 7.2 GUIDO2Midi.h File Reference

#### Classes

- struct **Guido2MidiParams**

*The **GuidoInitDesc** (p. 55) data structure contains all information required by **GuidoInit()** (p. 16)*

#### Typedefs

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

*The **GuidoInitDesc** (p. 55) data structure contains all information required by **GuidoInit()** (p. 16)*

#### Functions

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

*Export to a MIDI file.*

## 7.3 GUIDOEngine.h File Reference

### Classes

- struct **GuidoInitDesc**  
*The **GuidoInitDesc** (p. 55) data structure contains all information required by **GuidoInit()** (p. 16)*
- struct **GPaintStruct**  
*A structure to keep information about clipping and redrawing regions.*
- struct **GuidoOnDrawDesc**  
*Contains all graphic-related information required by **GuidoOnDraw()** (p. 23)*
- 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** ()
- **\_\_declspec** (deprecated("Deprecated function (will be erased soon) : use GUIDOAPI(**ARHandler**) GuidoFile2AR (GuidoParser \*parser, const char \* file) instead.)) GuidoErrCode GuidoParseFile(const char \*filename)
- **\_\_declspec** (deprecated("Deprecated function (will be erased soon) : use GUIDOAPI(**ARHandler**) GuidoString2AR (GuidoParser \*parser, const char \* str) instead.)) GuidoErrCode GuidoParseString(const char \*str)
- **GuidoErrCode GuidoAR2GR** (**ARHandler** ar, const **GuidoLayoutSettings** \*settings, **GRHandler** \*gr)
- **GuidoErrCode GuidoUpdateGR** (**GRHandler** gr, const **GuidoLayoutSettings** \*settings)
- void **GuidoFreeAR** (**ARHandler** ar)
- void **GuidoFreeGR** (**GRHandler** gr)
- const char \* **GuidoGetErrorString** (**GuidoErrCode** errCode)
- **\_\_declspec** (deprecated("Deprecated function (will be erased soon) : use GUIDOAPI(**GuidoErrCode**) GuidoParserGetErrorCodes (GuidoParser\* p, int& line, int& col) instead.)) 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.*

- 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)

## Variables

- **ARHandler \* ar**

### 7.3.1 Typedef Documentation

7.3.1.1 typedef struct NodeAR\* **ARHandler**

7.3.1.2 typedef struct NodeGR\* **GRHandler**

7.3.1.3 typedef struct NodeAR\* **CARHandler**

7.3.1.4 typedef struct NodeGR\* **CGRHandler**

7.3.1.5 typedef struct GuidoLayoutSettings **GuidoLayoutSettings**

Settings for the graphic score layout.

### 7.3.2 Enumeration Type Documentation

7.3.2.1 anonymous enum

Bounding boxes drawing control constants.

Enumerator:

*kNoBB*

*kPageBB*

*kSystemsBB*

*kSystemsSliceBB*

*kStavesBB*

*kMeasureBB*

*kEventsBB*

#### 7.3.2.2 anonymous enum

Enumerator:

*kAutoDistrib*

*kAlwaysDistrib*

*kNeverDistrib*

### 7.3.3 Function Documentation

7.3.3.1 void AddGGSOutput ( const char \* s )

7.3.3.2 void AddGuidoOutput ( const char \* s )

## 7.4 GUIDOExport.h File Reference

### Defines

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

#### 7.4.1 Define Documentation

7.4.1.1 #define `class_export` class

7.4.1.2 #define `GUIDOAPI( type )` type

## 7.5 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.*

## 7.6 GUIDOParse.h File Reference

### Functions

- GuidoParser \* **GuidoOpenParser** ()  
*Creates a new parser.*
- **GuidoErrCode GuidoCloseParser** (GuidoParser \*p)  
*Close a guido parser and releases all the associated resources.*

- `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.*

## 7.7 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.*

## 7.8 samples.doxygen File Reference

## 7.9 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`)

## 7.10 VGDevice.h File Reference

### Classes

- class **VGDevice**

*Generic platform independant drawing device.*

## 7.11 VGFont.h File Reference

### Classes

- class **VGFont**

*Generic pure virtual & device-independant font class.*

## 7.12 VGPen.h File Reference

### Classes

- class **VGPen**

*Generic class to manipulate device independant pens.*

## 7.13 VGSystem.h File Reference

### Classes

- class **VGSystem**

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

# Index

- ~MapCollector
  - MapCollector, 60
- ~RectInfos
  - RectInfos, 61
- ~TimeMapCollector
  - TimeMapCollector, 61
- ~TimeSegment
  - TimeSegment, 63
- ~VGDevice
  - VGDevice, 70
- ~VGFont
  - VGFont, 79
- ~VGSystem
  - VGSystem, 82
- \_\_declspec
  - Engine, 16, 18
- AddGGSOutput
  - GUIDOEngine.h, 88
- AddGuidoOutput
  - GUIDOEngine.h, 88
- ALPHA\_OPAQUE
  - VGSys, 50
- ALPHA\_TRANSPARENT
  - VGSys, 50
- ar
  - Engine, 19
- Arc
  - VGDevice, 71
- ARFactoryHandler
  - Factory, 31
- ARHandler
  - GUIDOEngine.h, 87
- BeginDraw
  - VGDevice, 70
- bottom
  - GPaintStruct, 52
- Browsing music pages, 19
- Building abstract and graphic representations, 15
- CARHandler
  - GUIDOEngine.h, 87
- CGRHandler
  - GUIDOEngine.h, 87
- class\_export
  - GUIDOExport.h, 88
- CopyPixels
  - VGDevice, 73, 74
- CreateAntiAliasedMemoryDevice
  - VGSystem, 82
- CreateDisplayDevice
  - VGSystem, 82
- CreateMemoryDevice
  - VGSystem, 82
- CreatePrinterDevice
  - VGSystem, 82
- CreateVGFont
  - VGSystem, 82
- DecoratorDevice
  - VGDevice, 78
- denom
  - GuidoDate, 54
- DeviceToLogical
  - VGDevice, 75
- DrawMusicSymbol
  - VGDevice, 75
- DrawString
  - VGDevice, 75
- empty
  - TimeSegment, 63
- EndDraw
  - VGDevice, 70
- Engine
  - \_\_declspec, 16, 18
  - ar, 19
  - GuidoAR2GR, 17
  - GuidoFreeAR, 18
  - GuidoFreeGR, 18
  - GuidoGetDefaultLayoutSettings, 18

- GuidoGetErrorString, 18
- GuidoInit, 16
- GuidoShutdown, 16
- GuidoUpdateGR, 17
- erase
  - GPaintStruct, 51
- Error codes, 14
- Errors
  - guidoErrActionFailed, 15
  - guidoErrBadParameter, 15
  - GuidoErrCode, 14
  - guidoErrFileAccess, 14
  - guidoErrInvalidHandle, 15
  - guidoErrMemory, 14
  - guidoErrNoMusicFont, 14
  - guidoErrNoTextFont, 14
  - guidoErrNotInitialized, 15
  - guidoErrParse, 14
  - guidoErrUserCancel, 14
  - guidoNoErr, 14
- fAccentFactor
  - Guido2MidiParams, 53
- Factory
  - ARFactoryHandler, 31
  - GuidoFactoryAddFlat, 34
  - GuidoFactoryAddSharp, 34
  - GuidoFactoryAddTagParameterFloat, 37
  - GuidoFactoryAddTagParameterInt, 37
  - GuidoFactoryAddTagParameterString, 37
  - GuidoFactoryClose, 31
  - GuidoFactoryCloseChord, 33
  - GuidoFactoryCloseEvent, 34
  - GuidoFactoryCloseMusic, 32
  - GuidoFactoryCloseTag, 36
  - GuidoFactoryCloseVoice, 32
  - GuidoFactoryEndTag, 36
  - GuidoFactoryInsertCommata, 33
  - GuidoFactoryOpen, 31
  - GuidoFactoryOpenChord, 33
  - GuidoFactoryOpenEvent, 33
  - GuidoFactoryOpenMusic, 32
  - GuidoFactoryOpenRangeTag, 36
  - GuidoFactoryOpenTag, 36
  - GuidoFactoryOpenVoice, 32
  - GuidoFactorySetDuration, 35
  - GuidoFactorySetEventAccidentals, 35
  - GuidoFactorySetEventDots, 34
- GuidoFactorySetOctave, 35
- GuidoFactorySetParameterName, 37
- GuidoFactorySetParameterUnit, 38
- fChan
  - Guido2MidiParams, 53
- fDFactor
  - Guido2MidiParams, 53
- fFermataFactor
  - Guido2MidiParams, 54
- fIntensity
  - Guido2MidiParams, 53
- fMarcatoFactor
  - Guido2MidiParams, 53
- force
  - GuidoLayoutSettings, 57
- Format
  - GuidoCM2Unit, 25
  - GuidoDrawBoundingBoxes, 23
  - GuidoGetDefaultPageFormat, 24
  - GuidoGetDrawBoundingBoxes, 23
  - GuidoGetPageFormat, 23
  - GuidoInches2Unit, 25
  - GuidoOnDraw, 23
  - GuidoResizePageToMusic, 25
  - GuidoSetDefaultPageFormat, 24
  - GuidoSVGExport, 23
  - GuidoUnit2CM, 24
  - GuidoUnit2Inches, 25
- Frame
  - VGDevice, 71
- fSlurFactor
  - Guido2MidiParams, 54
- fStaccatoFactor
  - Guido2MidiParams, 53
- fTempo
  - Guido2MidiParams, 53
- fTenutoFactor
  - Guido2MidiParams, 54
- fTicks
  - Guido2MidiParams, 53
- fVChans
  - Guido2MidiParams, 54
- GetBitMapPixels
  - VGDevice, 76
- GetContext
  - VGFont, 80
- GetDPITag
  - VGDevice, 76
- GetExtent

- VGFont, 79
- GetFontAlign
  - VGDevice, 76
- GetFontBackgroundColor
  - VGDevice, 76
- GetFontColor
  - VGDevice, 76
- GetHeight
  - VGDevice, 75
- GetImageData
  - VGDevice, 77
- GetMusicFont
  - VGDevice, 72
- GetName
  - VGFont, 79
- GetNativeContext
  - VGDevice, 77
- GetProperties
  - VGFont, 79
- GetRasterOpMode
  - VGDevice, 73
- GetSize
  - VGFont, 79
- GetTextFont
  - VGDevice, 72
- getVGSystem
  - VGDevice, 77
- GetWidth
  - VGDevice, 75
- GetXOrigin
  - VGDevice, 75
- GetXScale
  - VGDevice, 75
- GetYOrigin
  - VGDevice, 75
- GetYScale
  - VGDevice, 75
- globaldoc.doxygen, 83
- GPaintStruct, 51
  - bottom, 52
  - erase, 51
  - left, 51
  - right, 52
  - top, 51
- Graph2TimeMap
  - MapCollector, 60
- graphicDevice
  - GuidoInitDesc, 56
- GRHandler
  - GUIDOEngine.h, 87
- GUIDO Factory, 29
- GUIDO Mapping, 42
- GUIDO2Midi.h, 83
- Guido2MidiParams, 52
  - fAccentFactor, 53
  - fChan, 53
  - fDFactor, 53
  - fFermataFactor, 54
  - fIntensity, 53
  - fMarcatoFactor, 53
  - fSlurFactor, 54
  - fStaccatoFactor, 53
  - fTempo, 53
  - fTenutoFactor, 54
  - fTicks, 53
  - fVChans, 54
  - midi, 13
- GUIDOAPI
  - GUIDOExport.h, 88
- GuidoAR2GR
  - Engine, 17
- GuidoAR2MIDIFile
  - midi, 14
- GuidoCheckVersionNums
  - Misc, 27
- GuidoCloseParser
  - Parser, 39
- GuidoCloseStream
  - Parser, 41
- GuidoCM2Unit
  - Format, 25
- GuidoCountVoices
  - Pages, 19
- GuidoDate, 54
  - denom, 54
  - num, 54
- GuidoDrawBoundingBoxes
  - Format, 23
- GuidoDuration
  - Pages, 20
- GuidoeElementSelector
  - Mapping, 44
- GuidoElementInfos, 55
  - staffNum, 55
  - type, 55
  - voiceNum, 55
- GuidoElementType
  - Mapping, 44
- GUIDOEngine.h, 84
  - AddGGSOutput, 88

- AddGuidoOutput, 88
- ARHandler, 87
- CARHandler, 87
- CGRHandler, 87
- GRHandler, 87
- GuidoLayoutSettings, 87
- kAlwaysDistrib, 88
- kAutoDistrib, 88
- kEventsBB, 88
- kMeasureBB, 88
- kNeverDistrib, 88
- kNoBB, 87
- kPageBB, 87
- kStavesBB, 88
- kSystemsBB, 87
- kSystemsSliceBB, 87
- guidoErrActionFailed
  - Errors, 15
- guidoErrBadParameter
  - Errors, 15
- GuidoErrCode
  - Errors, 14
- guidoErrFileAccess
  - Errors, 14
- guidoErrInvalidHandle
  - Errors, 15
- guidoErrMemory
  - Errors, 14
- guidoErrNoMusicFont
  - Errors, 14
- guidoErrNoTextFont
  - Errors, 14
- guidoErrNotInitialized
  - Errors, 15
- guidoErrParse
  - Errors, 14
- guidoErrUserCancel
  - Errors, 14
- GUIDOExport.h, 88
  - class\_export, 88
  - GUIDOAPI, 88
- GUIDOFactory.h, 88
- GuidoFactoryAddFlat
  - Factory, 34
- GuidoFactoryAddSharp
  - Factory, 34
- GuidoFactoryAddTagParameterFloat
  - Factory, 37
- GuidoFactoryAddTagParameterInt
  - Factory, 37
- GuidoFactoryAddTagParameterString
  - Factory, 37
- GuidoFactoryClose
  - Factory, 31
- GuidoFactoryCloseChord
  - Factory, 33
- GuidoFactoryCloseEvent
  - Factory, 34
- GuidoFactoryCloseMusic
  - Factory, 32
- GuidoFactoryCloseTag
  - Factory, 36
- GuidoFactoryCloseVoice
  - Factory, 32
- GuidoFactoryEndTag
  - Factory, 36
- GuidoFactoryInsertCommata
  - Factory, 33
- GuidoFactoryOpen
  - Factory, 31
- GuidoFactoryOpenChord
  - Factory, 33
- GuidoFactoryOpenEvent
  - Factory, 33
- GuidoFactoryOpenMusic
  - Factory, 32
- GuidoFactoryOpenRangeTag
  - Factory, 36
- GuidoFactoryOpenTag
  - Factory, 36
- GuidoFactoryOpenVoice
  - Factory, 32
- GuidoFactorySetDuration
  - Factory, 35
- GuidoFactorySetEventAccidentals
  - Factory, 35
- GuidoFactorySetEventDots
  - Factory, 34
- GuidoFactorySetOctave
  - Factory, 35
- GuidoFactorySetParameterName
  - Factory, 37
- GuidoFactorySetParameterUnit
  - Factory, 38
- GuidoFile2AR
  - Parser, 40
- GuidoFindEventPage
  - Pages, 20
- GuidoFindPageAt
  - Pages, 21

- GuidoFreeAR
  - Engine, 18
- GuidoFreeGR
  - Engine, 18
- GuidoGetDefaultLayoutSettings
  - Engine, 18
- GuidoGetDefaultPageFormat
  - Format, 24
- GuidoGetDrawBoundingBoxes
  - Format, 23
- GuidoGetErrorString
  - Engine, 18
- GuidoGetLineSpace
  - Misc, 27
- GuidoGetMap
  - Mapping, 45
- GuidoGetPageCount
  - Pages, 20
- GuidoGetPageDate
  - Pages, 21
- GuidoGetPageFormat
  - Format, 23
- GuidoGetPageMap
  - Mapping, 45
- GuidoGetPoint
  - Mapping, 47
- GuidoGetStaffMap
  - Mapping, 46
- GuidoGetStream
  - Parser, 39
- GuidoGetSVGMap
  - Mapping, 48
- GuidoGetSymbolPath
  - Misc, 28
- GuidoGetSystemCount
  - Pages, 20
- GuidoGetSystemMap
  - Mapping, 46
- GuidoGetTime
  - Mapping, 47
- GuidoGetTimeMap
  - Mapping, 48
- GuidoGetVersionNums
  - Misc, 26
- GuidoGetVersionStr
  - Misc, 27
- GuidoGetVoiceMap
  - Mapping, 46
- GuidoInches2Unit
  - Format, 25
- GuidoInit
  - Engine, 16
- GuidoInitDesc, 55
  - graphicDevice, 56
  - musicFont, 56
  - reserved, 56
  - textFont, 56
- GuidoLayoutSettings, 56
  - force, 57
  - GUIDOEngine.h, 87
  - neighborhoodSpacing, 57
  - optimalPageFill, 57
  - resizePage2Music, 57
  - spring, 57
  - systemsDistance, 57
  - systemsDistribLimit, 57
  - systemsDistribution, 57
- GuidoMarkVoice
  - Misc, 27
- guidoNoErr
  - Errors, 14
- GuidoOnDraw
  - Format, 23
- GuidoOnDrawDesc, 58
  - handle, 58
  - hdc, 58
  - isprint, 59
  - page, 58
  - reserved, 59
  - scrollx, 59
  - scrolly, 59
  - sizeX, 59
  - sizeY, 59
  - updateRegion, 58
- GuidoOpenParser
  - Parser, 39
- GuidoOpenStream
  - Parser, 41
- GuidoPageFormat, 59
  - height, 60
  - marginbottom, 60
  - marginleft, 60
  - marginright, 60
  - margintop, 60
  - width, 60
- GUIDOParse.h, 90
- GuidoParserGetErrorCode
  - Parser, 41
- GuidoResetStream
  - Parser, 42

- GuidoResizePageToMusic
  - Format, 25
- GUIDOScoreMap.h, 91
- GuidoSetDefaultPageFormat
  - Format, 24
- GuidoSetSymbolPath
  - Misc, 28
- GuidoShutdown
  - Engine, 16
- GuidoStream2AR
  - Parser, 40
- GuidoString2AR
  - Parser, 40
- GuidoSVGExport
  - Format, 23
- GuidoUnit2CM
  - Format, 24
- GuidoUnit2Inches
  - Format, 25
- GuidoUpdateGR
  - Engine, 17
- GuidoWriteStream
  - Parser, 41
  
- handle
  - GuidoOnDrawDesc, 58
- hdc
  - GuidoOnDrawDesc, 58
- height
  - GuidoPageFormat, 60
  
- include
  - TimeSegment, 63
- infos
  - RectInfos, 61
- intersect
  - TimeSegment, 63
- InvalidateRect
  - VGDevice, 70
- isprint
  - GuidoOnDrawDesc, 59
- IsValid
  - VGDevice, 70
  
- kAlignBase
  - VGDevice, 70
- kAlignBaseLeft
  - VGDevice, 70
- kAlignBottom
  - VGDevice, 70
- kAlignCenter
  - VGDevice, 70
- kAlignLeft
  - VGDevice, 70
- kAlignRight
  - VGDevice, 70
- kAlignTop
  - VGDevice, 70
- kAlwaysDistrib
  - GUIDOEngine.h, 88
- kAutoDistrib
  - GUIDOEngine.h, 88
- kBar
  - Mapping, 44
- kEmpty
  - Mapping, 44
- kEventsBB
  - GUIDOEngine.h, 88
- kFontBold
  - VGFont, 79
- kFontItalic
  - VGFont, 79
- kFontNone
  - VGFont, 79
- kFontUnderline
  - VGFont, 79
- kGuidoBar
  - Mapping, 44
- kGuidoEvent
  - Mapping, 44
- kGuidoPage
  - Mapping, 44
- kGuidoScoreElementEnd
  - Mapping, 44
- kGuidoStaff
  - Mapping, 44
- kGuidoSystem
  - Mapping, 44
- kGuidoSystemSlice
  - Mapping, 44
- kMeasureBB
  - GUIDOEngine.h, 88
- kNeverDistrib
  - GUIDOEngine.h, 88
- kNoBB
  - GUIDOEngine.h, 87
- kNote
  - Mapping, 44
- kOpAnd
  - VGDevice, 69



- kOpCopy
  - VGDevice, 69
- kOpInvert
  - VGDevice, 69
- kOpOr
  - VGDevice, 70
- kOpXOr
  - VGDevice, 69
- kPage
  - Mapping, 45
- kPageBB
  - GUIDOEngine.h, 87
- kRepeatBegin
  - Mapping, 44
- kRepeatEnd
  - Mapping, 45
- kRest
  - Mapping, 44
- kStaff
  - Mapping, 45
- kStavesBB
  - GUIDOEngine.h, 88
- kSystem
  - Mapping, 45
- kSystemsBB
  - GUIDOEngine.h, 87
- kSystemSlice
  - Mapping, 45
- kSystemsSliceBB
  - GUIDOEngine.h, 87
- kUnknown
  - VGDevice, 69
- left
  - GPaintStruct, 51
- Line
  - VGDevice, 71
- LineTo
  - VGDevice, 71
- LogicalToDevice
  - VGDevice, 74
- mAlpha
  - VGColor, 65
- MapCollector, 60
  - ~MapCollector, 60
  - Graph2TimeMap, 60
- MapElement
  - Mapping, 44
- Mapping
  - GuidoeElementSelector, 44
  - GuidoElementType, 44
  - GuidoGetMap, 45
  - GuidoGetPageMap, 45
  - GuidoGetPoint, 47
  - GuidoGetStaffMap, 46
  - GuidoGetSVGMap, 48
  - GuidoGetSystemMap, 46
  - GuidoGetTime, 47
  - GuidoGetTimeMap, 48
  - GuidoGetVoiceMap, 46
  - kBar, 44
  - kEmpty, 44
  - kGuidoBar, 44
  - kGuidoEvent, 44
  - kGuidoPage, 44
  - kGuidoScoreElementEnd, 44
  - kGuidoStaff, 44
  - kGuidoSystem, 44
  - kGuidoSystemSlice, 44
  - kNote, 44
  - kPage, 45
  - kRepeatBegin, 44
  - kRepeatEnd, 45
  - kRest, 44
  - kStaff, 45
  - kSystem, 45
  - kSystemSlice, 45
  - MapElement, 44
  - operator<<, 45
  - Time2GraphicMap, 44
- marginbottom
  - GuidoPageFormat, 60
- marginleft
  - GuidoPageFormat, 60
- marginright
  - GuidoPageFormat, 60
- margintop
  - GuidoPageFormat, 60
- mBlue
  - VGColor, 65
- mColor
  - VGPen, 81
- mGreen
  - VGColor, 65
- midi
  - Guido2MidiParams, 13
  - GuidoAR2MIDIFile, 14
- MIDI Export, 13
- Misc

- GuidoCheckVersionNums, 27
- GuidoGetLineSpace, 27
- GuidoGetSymbolPath, 28
- GuidoGetVersionNums, 26
- GuidoGetVersionStr, 27
- GuidoMarkVoice, 27
- GuidoSetSymbolPath, 28
- Miscellaneous, 26
- MoveTo
  - VGDevice, 70
- mRed
  - VGColor, 65
- musicFont
  - GuidoInitDesc, 56
- mWidth
  - VGPen, 81
- neighborhoodSpacing
  - GuidoLayoutSettings, 57
- NotifySize
  - VGDevice, 75
- num
  - GuidoDate, 54
- OffsetOrigin
  - VGDevice, 74
- operator<
  - TimeSegment, 63
- operator<<
  - Mapping, 45
  - VGSys, 50
- operator+=
  - VGSys, 50
- operator==
  - TimeSegment, 63
  - VGColor, 65
- operator&
  - TimeSegment, 63
- optimalPageFill
  - GuidoLayoutSettings, 57
- page
  - GuidoOnDrawDesc, 58
- Pages
  - GuidoCountVoices, 19
  - GuidoDuration, 20
  - GuidoFindEventPage, 20
  - GuidoFindPageAt, 21
  - GuidoGetPageCount, 20
  - GuidoGetPageDate, 21
  - GuidoGetSystemCount, 20
- Parser
  - GuidoCloseParser, 39
  - GuidoCloseStream, 41
  - GuidoFile2AR, 40
  - GuidoGetStream, 39
  - GuidoOpenParser, 39
  - GuidoOpenStream, 41
  - GuidoParserGetErrorCode, 41
  - GuidoResetStream, 42
  - GuidoStream2AR, 40
  - GuidoString2AR, 40
  - GuidoWriteStream, 41
- Parsing GMN files, strings and guido streams, 38
- Polygon
  - VGDevice, 71
- PopFillColor
  - VGDevice, 73
- PopPen
  - VGDevice, 73
- PopPenColor
  - VGDevice, 78
- PopPenWidth
  - VGDevice, 78
- print
  - TimeSegment, 63
  - VGColor, 65
- PushFillColor
  - VGDevice, 73
- PushPen
  - VGDevice, 72
- PushPenColor
  - VGDevice, 77
- PushPenWidth
  - VGDevice, 78
- Rectangle
  - VGDevice, 71
- RectInfos, 61
  - ~RectInfos, 61
  - infos, 61
  - RectInfos, 61
  - time, 61
- ReleaseBitMapPixels
  - VGDevice, 77
- ReleaseImageData
  - VGDevice, 77
- reserved
  - GuidoInitDesc, 56

- GuidoOnDrawDesc, 59
- resizePage2Music
  - GuidoLayoutSettings, 57
- right
  - GPaintStruct, 52
- samples.doxygen, 93
- Score drawing and pages formating, 22
- scrollx
  - GuidoOnDrawDesc, 59
- scrolly
  - GuidoOnDrawDesc, 59
- SelectFillColor
  - VGDevice, 72
- selectfont
  - VGDevice, 72
- SelectPen
  - VGDevice, 72
- SelectPenColor
  - VGDevice, 77
- SelectPenWidth
  - VGDevice, 77
- Set
  - VGColor, 65
  - VGPen, 80
- SetDPITag
  - VGDevice, 76
- SetFontAlign
  - VGDevice, 76
- SetFontBackgroundColor
  - VGDevice, 76
- SetFontColor
  - VGDevice, 76
- SetMusicFont
  - VGDevice, 72
- SetOrigin
  - VGDevice, 74
- SetRasterOpMode
  - VGDevice, 73
- SetScale
  - VGDevice, 74
- SetTextFont
  - VGDevice, 72
- sizeX
  - GuidoOnDrawDesc, 59
- sizeY
  - GuidoOnDrawDesc, 59
- spring
  - GuidoLayoutSettings, 57
- staffNum
  - GuidoElementInfos, 55
- systemsDistance
  - GuidoLayoutSettings, 57
- systemsDistribLimit
  - GuidoLayoutSettings, 57
- systemsDistribution
  - GuidoLayoutSettings, 57
- textFont
  - GuidoInitDesc, 56
- time
  - RectInfos, 61
- Time2GraphicMap
  - Mapping, 44
- Time2TimeMap
  - TimeMapCollector, 61
- TimeMapCollector, 61
  - ~TimeMapCollector, 61
  - Time2TimeMap, 61
- TimeSegment, 62
  - ~TimeSegment, 63
  - empty, 63
  - include, 63
  - intersect, 63
  - operator<, 63
  - operator==, 63
  - operator&, 63
  - print, 63
  - TimeSegment, 63
- top
  - GPaintStruct, 51
- Triangle
  - VGDevice, 71
- type
  - GuidoElementInfos, 55
- updateRegion
  - GuidoOnDrawDesc, 58
- VGColor, 64
  - mAlpha, 65
  - mBlue, 65
  - mGreen, 65
  - mRed, 65
  - operator==, 65
  - print, 65
  - Set, 65
  - VGColor, 64
- VGColor.h, 93
- VGDevice, 66

~VGDevice, 70  
 Arc, 71  
 BeginDraw, 70  
 CopyPixels, 73, 74  
 DecoratorDevice, 78  
 DeviceToLogical, 75  
 DrawMusicSymbol, 75  
 DrawString, 75  
 EndDraw, 70  
 Frame, 71  
 GetBitMapPixels, 76  
 GetDPITag, 76  
 GetFontAlign, 76  
 GetFontBackgroundColor, 76  
 GetFontColor, 76  
 GetHeight, 75  
 GetImageData, 77  
 GetMusicFont, 72  
 GetNativeContext, 77  
 GetRasterOpMode, 73  
 GetTextFont, 72  
 getVGSystem, 77  
 GetWidth, 75  
 GetXOrigin, 75  
 GetXScale, 75  
 GetYOrigin, 75  
 GetYScale, 75  
 InvalidateRect, 70  
 IsValid, 70  
 kAlignBase, 70  
 kAlignBaseLeft, 70  
 kAlignBottom, 70  
 kAlignCenter, 70  
 kAlignLeft, 70  
 kAlignRight, 70  
 kAlignTop, 70  
 kOpAnd, 69  
 kOpCopy, 69  
 kOpInvert, 69  
 kOpOr, 70  
 kOpXOr, 69  
 kUnknown, 69  
 Line, 71  
 LineTo, 71  
 LogicalToDevice, 74  
 MoveTo, 70  
 NotifySize, 75  
 OffsetOrigin, 74  
 Polygon, 71  
 PopFillColor, 73  
 PopPen, 73  
 PopPenColor, 78  
 PopPenWidth, 78  
 PushFillColor, 73  
 PushPen, 72  
 PushPenColor, 77  
 PushPenWidth, 78  
 Rectangle, 71  
 ReleaseBitMapPixels, 77  
 ReleaseImageData, 77  
 SelectFillColor, 72  
 selectfont, 72  
 SelectPen, 72  
 SelectPenColor, 77  
 SelectPenWidth, 77  
 SetDPITag, 76  
 SetFontAlign, 76  
 SetFontBackgroundColor, 76  
 SetFontColor, 76  
 SetMusicFont, 72  
 SetOrigin, 74  
 SetRasterOpMode, 73  
 SetScale, 74  
 SetTextFont, 72  
 Triangle, 71  
 VGFont, 78  
 VRasterOpMode, 69  
 VTextAlignMode, 70  
 VGDevice.h, 93  
 VGFont, 78  
   ~VGFont, 79  
   GetContext, 80  
   GetExtent, 79  
   GetName, 79  
   GetProperties, 79  
   GetSize, 79  
   kFontBold, 79  
   kFontItalic, 79  
   kFontNone, 79  
   kFontUnderline, 79  
   VGDevice, 78  
 VGFont.h, 93  
 VGPen, 80  
   mColor, 81  
   mWidth, 81  
   Set, 80  
   VGPen, 80  
 VGPen.h, 94  
 VGSys  
   ALPHA\_OPAQUE, 50

---

- ALPHA\_TRANSPARENT, 50
- operator<<, 50
- operator+=", 50
- VGSystem, 81
  - ~VGSystem, 82
  - CreateAntiAliasedMemoryDevice, 82
  - CreateDisplayDevice, 82
  - CreateMemoryDevice, 82
  - CreatePrinterDevice, 82
  - CreateVGFont, 82
- VGSystem.h, 94
- Virtual Graphic System, 48
- voiceNum
  - GuidoElementInfos, 55
- VRasterOpMode
  - VGDevice, 69
- VTextAlignMode
  - VGDevice, 70
- width
  - GuidoPageFormat, 60