Generic Canvas

Table of Contents

Index a

1 Symbol Reference

1.1 __gnu_cxx Namespace

This is namespace __gnu_cxx.

Structs

Struct	Description
hash<A*> (see page 1)	This is class __gnu_cxx::hash<A*>.

1.1.1 hash<A*> Struct

Class Hierarchy

```
struct hash<A*> {
};
```

File

myx_gc.h (see page 216)

Remarks

This is class __gnu_cxx::hash<A*>.

Members

Operators

Operator	Description
() (see page 1)	This is (), a member of class hash<A*>.

Operators

Operator	Description
() (see page 1)	This is (), a member of class hash<A*>.

Legend

	Method

1.1.1.1 Operators

1.1.1.1.1 hash<A*>::() Operator

```
size_t operator ()(A * __x) const;
```

Remarks

This is (), a member of class hash<A*>.

1.2 Classes

1.2.1 CBoundingBoxComputer Class

Class Hierarchy


```
class CBoundingBoxComputer;
```

File

myx_gc_utilities.h (see page 232)

Remarks

The bounding box computer is a neat little helper class to construct a final bound box out of an arbitrary number of other boxes as well as (lists of) points.

Constructors

Constructor	Description
CBoundingBoxComputer (see page 2)	

Members

Constructors

Constructor	Description
CBoundingBoxComputer (see page 2)	

Methods

Method	Description
boundingBox (see page 3)	Returns the current bounding box.
include (see page 3)	Takes the new box and merges it with the current bounding box. The new data is not required to be ordered.
includeVertex (see page 4)	
transform (see page 4)	Transforms the given coordinate using the provided matrix.

Legend

	Method
	protected

1.2.1.1 Constructors

1.2.1.1.1 CBoundingBoxComputer

1.2.1.1.1.1 CBoundingBoxComputer::CBoundingBoxComputer Constructor (const TBoundingBox&)

```
CBoundingBoxComputer(const TBoundingBox& InitialBox);
```

1.2.1.1.1.2 CBoundingBoxComputer::CBoundingBoxComputer Constructor (void)

```
CBoundingBoxComputer(void);
```

Remarks

CBoundingBoxComputer

1.2.1.2 Methods

1.2.1.2.1 CBoundingBoxComputer::boundingBox Method

```
TBoundingBox boundingBox();
```

Returns

The current bounding box.

Remarks

Returns the current bounding box.

1.2.1.2.2 include

1.2.1.2.2.1 CBoundingBoxComputer::include Method (TMatrix, TBoundingBox*)

```
void include(TMatrix Matrix, TBoundingBox* NewBox);
```

Parameters

Parameters	Description
`TBoundingBox* NewBox`	A new bounding box to merge in.

Remarks

Takes the new box and merges it with the current bounding box. The new data is not required to be ordered.

1.2.1.2.2.2 CBoundingBoxComputer::include Method (TMatrix, const TVertex&)

```
void include(TMatrix Matrix, const TVertex& Vertex);
```

Parameters

Parameters	Description
`const TVertex& Vertex`	The vertex to be included

Remarks

Takes the vertex and merges it with the current bounding box.

1.2.1.2.2.3 CBoundingBoxComputer::include Method (TMatrix, const float&, const float&)

```
void include(TMatrix Matrix, const float& X, const float& Y);
```

Parameters

Parameters	Description
x	The x coordinate of the point to include.
y	The y coordinate of the point to include.

Remarks

Takes the x and y values and merges them with the current bounding box.

1.2.1.2.3 CBoundingBoxComputer::includeVertex Method

```
void includeVertex(const TVertex& Vertex);
```

1.2.1.2.4 CBoundingBoxComputer::transform Method

```
TVertex transform(TMatrix Matrix, const float& X, const float& Y);
```

Parameters

Parameters	Description
TMatrix Matrix	The matrix with which to do the transformation. It is a usual 4x4 matrix, although only the upper-left 2x2 entries are used.
x	The x coordinate.
y	The y coordinate.

Returns

The transformed vertex.

Remarks

Transforms the given coordinate using the provided matrix.

1.2.2 CCanvasListener Class

Class Hierarchy


```
class CCanvasListener : public CGCListener;
```

File

myx_gc_canvas.h (see page 218)

Remarks

This is class CCanvasListener.

Members

Data Members

Data Member	Description
canvas (see page 5)	This is canvas, a member of class CCanvasListener.

Methods

Method	Description
onChange (see page 5)	CCanvasListener
onDestroy (see page 5)	
onError (see page 5)	

CGCListener Class

CGCListener Class	Description
onChange (see page 78)	This is onChange, a member of class CGCListener.
onDestroy (see page 78)	This is onDestroy, a member of class CGCListener.
onError (see page 78)	This is onError, a member of class CGCListener.

Friends

Friend	Description
class CGenericCanvas (see page 5)	This is friend friend class CGenericCanvas.

Legend

	protected
	Data Member
	Method
V	virtual
A	abstract

1.2.2.1 Data Members

1.2.2.1.1 CCanvasListener::canvas Data Member

```
CGenericCanvas* canvas;
```

Remarks

This is canvas, a member of class CCanvasListener.

1.2.2.2 Methods

1.2.2.2.1 CCanvasListener::onChange Method

```
virtual void __cdecl onChange(CGCBase* sender, CGCBase* origin, TGCChangeReason reason);
```

Remarks

CCanvasListener (see page 4)

1.2.2.2.2 CCanvasListener::onDestroy Method

```
virtual void __cdecl onDestroy(CGCBase* object);
```

1.2.2.2.3 CCanvasListener::onError Method

```
virtual void __cdecl onError(CGCBase* sender, CGCBase* origin, const char* message);
```

1.2.2.3 Friends

1.2.2.3.1 friend class CGenericCanvas Friend

```
friend class CGenericCanvas;
```

Remarks

This is friend friend class CGenericCanvas.

1.2.3 CCaptionElement Class

Class Hierarchy


```
class CCaptionElement : public CGCBase;
```

File

myx_gc_figure.h (see page 222)

Remarks

Instance for figure elements and captions.

Constructors

Constructor	Description
CCaptionElement (see page 7)	CCaptionElement

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CCaptionElement (see page 8)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Members

Constructors

Constructor	Description
CCaptionElement (see page 7)	CCaptionElement

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CCaptionElement (see page 8)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Methods

Method	Description
applyAlignment (see page 8)	Recomputes the offsets for the caption to maintain the current alignments.
bounds (see page 8)	This is bounds, a member of class CCaptionElement.
makeDirty (see page 8)	Marks the element as changed so its bounding box it is validated next time it is used.

property (see page 8)	Retrieves the value of the property given by name. The name syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name of a container class (e.g. layers, figures) and property is the name of a simple property of that container.
render (see page 9)	Renders the caption element.
text (see page 9)	Sets a new text string.
validate (see page 9)	Validates the bounding box.
zoomChanged (see page 9)	Called when the current zoom (scale) factor has changed. Recompute font size.

CGCBase Class

CGCBase Class	Description
addListener (see page 75)	Adds a listener to the internal list of listeners, if it is not already there.
beginUpdate (see page 75)	Increases the update count by 1 to stop any recursive update until (@see endUpdate (see page 76)()) was called.
canvas (see page 75)	This is canvas, a member of class CGCBase.
change (see page 75)	Triggers the onCange event of all registered listeners to notfied them about a particular change.
classIs (see page 76)	Determines if this class is of a specific type by comparing its class name to the given name.
className (see page 76)	This is className, a member of class CGCBase.
destroying (see page 76)	This is destroying, a member of class CGCBase.
endUpdate (see page 76)	The counterpart to (@see beginUpdate (see page 75)). It releases one update lock and also the global lock if the count drops to 0.
error (see page 76)	Triggers the onError event of all registered listeners to notfied them about an error.
property (see page 76)	This is property, a member of class CGCBase.
release (see page 77)	This is release, a member of class CGCBase.
removeListener (see page 77)	
setDestroying (see page 77)	Helper to set destroying (see page 76) state explicitely.
updating (see page 77)	This is updating, a member of class CGCBase.

Friends

Friend	Description
class CFigureElement (see page 10)	This is friend friend class CFigureElement.

CGCBase Class

CGCBase Class	Description
class CGenericCanvas (see page 77)	This is friend friend class CGenericCanvas.

Data Members

CGCBase Class

CGCBase Class	Description
_className (see page 74)	Used to determine the actual class.

Legend

	Method
	virtual
	abstract
	protected
	Data Member

1.2.3.1 Constructors

1.2.3.1.1 CCaptionElement::CCaptionElement Constructor

```
CCaptionElement(CGenericCanvas* canvas);
```

Remarks

CCaptionElement

1.2.3.2 Destructors

1.2.3.2.1 CCaptionElement::~CCaptionElement Destructor

```
virtual ~CCaptionElement(void);
```

1.2.3.3 Methods

1.2.3.3.1 CCaptionElement::applyAlignment Method

```
void applyAlignment(void);
```

Remarks

Recomputes the offsets for the caption to maintain the current alignments.

1.2.3.3.2 CCaptionElement::bounds Method

```
TBoundingBox bounds(void);
```

Remarks

This is bounds, a member of class CCaptionElement.

1.2.3.3.3 CCaptionElement::makeDirty Method

```
void makeDirty(void);
```

Remarks

Marks the element as changed so its bounding box it is validated next time it is used.

1.2.3.3.4 property

1.2.3.3.4.1 CCaptionElement::property Method (const char*, unsigned int)

```
virtual TGCVariant __cdecl property(const char* name, unsigned int index);
```

Parameters

Parameters	Description
`const char* name`	The name of the property to return.
`unsigned int index`	If the property is a list then this is the index into that list.

Returns

A description of the property value and, if the property is simple, the actual value.

Remarks

Retrieves the value of the property given by name. The name syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name of a container class (e.g. layers, figures) and property is the name of a simple property of that container.

1.2.3.3.4.2 CCaptionElement::property Method (const char*, unsigned int, const TGCVariant&)

```
virtual void __cdecl property(const char* name, unsigned int index, const TGCVariant& value);
```

Parameters

Parameters	Description
const char* name	The name of the property.
unsigned int index	If the property is a list then this is the index into that list.
Value	The new value of the property. Automatic conversion is performed where possible.

Remarks

Sets the value of the given property, which must be a simple property.

1.2.3.3.5 CCaptionElement::render Method

```
void render(void);
```

Remarks

Renders the caption element.

1.2.3.3.6 CCaptionElement::text Method

```
void text(wstring newText);
```

Parameters

Parameters	Description
wstring newText	The new text to set.

Remarks

Sets a new text string.

1.2.3.3.7 CCaptionElement::validate Method

```
void validate(void);
```

Remarks

Validates the bounding box.

1.2.3.3.8 CCaptionElement::zoomChanged Method

```
bool zoomChanged(float ZoomFactor);
```

Parameters

Parameters	Description
zoomFactor	The new zoom factor.

Returns

True if the zoom change (see page 75) has an effect on this element.

Remarks

Called when the current zoom (scale) factor has changed. Recompute font size.

1.2.3.4 Friends

1.2.3.4.1 friend class CFigureElement Friend

```
friend class CFigureElement;
```

Remarks

This is friend friend class CFigureElement.

1.2.4 CCaptionElementTemplate Class

Class Hierarchy


```
class CCaptionElementTemplate;
```

File

myx_gc_figure.h (see page 222)

Remarks

Special text element class.

Notes

captions are directly bound to their owning parent element and not handled as a separate child element.

Constructors

Constructor	Description
CCaptionElementTemplate (see page 11)	CCaptionElementTemplate

Members

Constructors

Constructor	Description
CCaptionElementTemplate (see page 11)	CCaptionElementTemplate

Methods

Method	Description
initialize (see page 11)	
key (see page 11)	This is key, a member of class CCaptionElementTemplate.

Friends

Friend	Description
class CFigureElement (see page 11)	This is friend friend class CFigureElement.
class CFigureElementTemplate (see page 11)	This is friend friend class CFigureElementTemplate.

Legend

	Method

1.2.4.1 Constructors

1.2.4.1.1 CCaptionElementTemplate::CCaptionElementTemplate Constructor

```
CCaptionElementTemplate(wstring key);
```

Remarks

CCaptionElementTemplate

1.2.4.2 Methods

1.2.4.2.1 CCaptionElementTemplate::initialize Method

```
void initialize(wstring Text, float X, float Y, string FontFamily, int FontSize, int
Weight, string FontStyle, TAlignment HorzontalAlignment, TAlignment VerticalAlignment,
GLubyte* Color, const TConstraints& Constraints);
```

1.2.4.2.2 CCaptionElementTemplate::key Method

```
wstring key(void);
```

Remarks

This is key, a member of class CCaptionElementTemplate.

1.2.4.3 Friends

1.2.4.3.1 friend class CFigureElement Friend

```
friend class CFigureElement;
```

Remarks

This is friend friend class CFigureElement.

1.2.4.3.2 friend class CFigureElementTemplate Friend

```
friend class CFigureElementTemplate;
```

Remarks

This is friend friend class CFigureElementTemplate.

1.2.5 CColumnLayouter Class

Class Hierarchy


```
class CColumnLayouter : public CLayouter;
```

File

myx_gc_layout.h (see page 227)

Remarks

This is class CColumnLayouter.

Constructors

Constructor	Description
CColumnLayouter (see page 12)	This is CColumnLayouter, a member of class CColumnLayouter.

CLayouter Class

CLayouter Class	Description
CLayouter (see page 125)	CLayouter

Members

Constructors

Constructor	Description
CColumnLayouter (see page 12)	This is CColumnLayouter, a member of class CColumnLayouter.

CLayouter Class

CLayouter Class	Description
CLayouter (see page 125)	CLayouter

Methods

Method	Description
V nextBoundingBox (see page 13)	Returns the transformed bounding box of the next element.

CLayouter Class

CLayouter Class	Description
V hasNext (see page 125)	Tells the caller whether there is still a next value available.
nextAction (see page 125)	Executes the doAction function of the current element in the layout order. For this to work the given coordinates must be transformed to local coordinates.
A nextBoundingBox (see page 126)	This is nextBoundingBox, a member of class CLayouter.
V renderNext (see page 126)	Renders the current child element and moves on to the next in the list.
reset (see page 126)	Resets layout computation to start over from origin.

Data Members

CLayouter Class

CLayouter Class	Description
FElement (see page 124)	The element we are layouting.
FIterator (see page 125)	The iterator used to go through the child list of the element to layout.
FX (see page 125)	This is FX, a member of class CLayouter.
FY (see page 125)	This is FY, a member of class CLayouter.

Legend

	Method
V	virtual
A	abstract
	protected
	Data Member

1.2.5.1 Constructors

1.2.5.1.1 CColumnLayouter::CColumnLayouter Constructor

```
CColumnLayouter(CFigureElement* Element);
```

Remarks

This is CColumnLayouter, a member of class CColumnLayouter.

1.2.5.2 Methods

1.2.5.2.1 CColumnLayouter::nextBoundingBox Method

```
virtual void nextBoundingBox(TBoundingBox* BoundingBox);
```

Parameters

Parameters	Description
`TBoundingBox* BoundingBox`	The bounding box to fill with the new values.

Remarks

Returns the transformed bounding box of the next element.

1.2.6 CConnection Class

Class Hierarchy


```
class CConnection : public CGCBase;
```

File

myx_gc_connection.h (see page 219)

Remarks

A class comprising data for a connection.

Constructors

Constructor	Description
CConnection (see page 15)	CConnection

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CConnection (see page 15)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Members

Constructors

Constructor	Description
CConnection (see page 15)	CConnection

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CConnection (see page 15)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Methods

Method	Description
property (see page 15)	Retrieves the value of the property given by path. The path syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name of a container class (e.g. layers, figures) and property is the name of a simple property of that container.

CGCBase Class

CGCBase Class	Description
addListener (see page 75)	Adds a listener to the internal list of listeners, if it is not already there.
beginUpdate (see page 75)	Increases the update count by 1 to stop any recursive update until (@see endUpdate (see page 76)()) was called.
canvas (see page 75)	This is canvas, a member of class CGCBase.
change (see page 75)	Triggers the onCange event of all registered listeners to notfied them about a particular change.
classIs (see page 76)	Determines if this class is of a specific type by comparing its class name to the given name.
className (see page 76)	This is className, a member of class CGCBase.
destroying (see page 76)	This is destroying, a member of class CGCBase.
endUpdate (see page 76)	The counterpart to (@see beginUpdate (see page 75)). It releases one update lock and also the global lock if the count drops to 0.
error (see page 76)	Triggers the onError event of all registered listeners to notfied them about an error.
property (see page 76)	This is property, a member of class CGCBase.
release (see page 77)	This is release, a member of class CGCBase.
removeListener (see page 77)	
setDestroying (see page 77)	Helper to set destroying (see page 76) state explicitely.
updating (see page 77)	This is updating, a member of class CGCBase.

Data Members

CGCBase Class

CGCBase Class	Description
_className (see page 74)	Used to determine the actual class.

Friends

CGCBase Class

CGCBase Class	Description
class CGenericCanvas (see page 77)	This is friend friend class CGenericCanvas.

Legend

	Method
	virtual
	abstract
	protected
	Data Member

1.2.6.1 Constructors

1.2.6.1.1 CConnection::CConnection Constructor

```
CConnection(CGenericCanvas* canvas, CFigure* endPoint1, CFigure* endPoint2);
```

Remarks

CConnection

1.2.6.2 Destructors

1.2.6.2.1 CConnection::~CConnection Destructor

```
virtual ~CConnection(void);
```

1.2.6.3 Methods

1.2.6.3.1 property

1.2.6.3.1.1 CConnection::property Method (const char*, unsigned int)

```
virtual TGCVariant __cdecl property(const char* name, unsigned int index);
```

Parameters

Parameters	Description
`const char* name`	The name of the property to return.
`unsigned int index`	The index of the sub property to return if it is located in a list.

Returns

A description of the property value and, if the property is simple, the actual value.

Remarks

Retrieves the value of the property given by path. The path syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name of a container class (e.g. layers, figures) and property is the name of a simple property of that container.

1.2.6.3.1.2 CConnection::property Method (const char*, unsigned int, const TGCVariant&)

```
virtual void __cdecl property(const char* name, unsigned int index, const TGCVariant&
value);
```

Parameters

Parameters	Description
`const char* name`	The name of the property.
`unsigned int index`	The index of the sub property to return if it is located in a list.
`const TGCVariant& value`	The new value of the property. Automatic conversion is performed where possible.

Remarks

Sets the value of the given property, which must be a simple property.

1.2.7 CConnectionInstance Class

Class Hierarchy


```
class CConnectionInstance : public CGCBase;
```

File

myx_gc_connection.h (see page 219)

Remarks

A concrete instance for a connection.

Constructors

Constructor	Description
CConnectionInstance (see page 17)	Constructor of the class.

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CConnectionInstance (see page 18)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Members

Constructors

Constructor	Description
CConnectionInstance (see page 17)	Constructor of the class.

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CConnectionInstance (see page 18)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Methods

Method	Description
computeCoordinates (see page 18)	Computes the final coordinates of this connection at the given end point (which must be one of the end points of this connection instance).
dirty (see page 18)	This is dirty, a member of class CConnectionInstance.

endPoint1 (see page 18)	This is endPoint1, a member of class CConnectionInstance.
endPoint2 (see page 18)	This is endPoint2, a member of class CConnectionInstance.
getDirection (see page 19)	Determines in which direction this connection leaves the given point. This depends entirely on the positions of the endpoints.
makeDirty (see page 19)	Marks the instance as invalid so its display list is recreated on next render (see page 20).
V property (see page 19)	Retrieves the value of the property given by path. The path syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name of a container class (e.g. layers, figures) and property is the name of a simple property of that container.
render (see page 20)	Renders this instance.
V setLineStyle (see page 20)	Changes the visual style of the connection lines.
validate (see page 20)	Computes all detail elements for this connection depending on computed start and end point as well as other properties.

CGCBase Class

CGCBase Class	Description
V addListener (see page 75)	Adds a listener to the internal list of listeners, if it is not already there.
V beginUpdate (see page 75)	Increases the update count by 1 to stop any recursive update until (@see endUpdate (see page 76)()) was called.
V canvas (see page 75)	This is canvas, a member of class CGCBase.
V change (see page 75)	Triggers the onCange event of all registered listeners to notfied them about a particular change.
V classIs (see page 76)	Determines if this class is of a specific type by comparing its class name to the given name.
V className (see page 76)	This is className, a member of class CGCBase.
V destroying (see page 76)	This is destroying, a member of class CGCBase.
V endUpdate (see page 76)	The counterpart to (@see beginUpdate (see page 75)). It releases one update lock and also the global lock if the count drops to 0.
V error (see page 76)	Triggers the onError event of all registered listeners to notfied them about an error.
A property (see page 76)	This is property, a member of class CGCBase.
V release (see page 77)	This is release, a member of class CGCBase.
V removeListener (see page 77)	
setDestroying (see page 77)	Helper to set destroying (see page 76) state explicitely.
V updating (see page 77)	This is updating, a member of class CGCBase.

Data Members

CGCBase Class

CGCBase Class	Description
_className (see page 74)	Used to determine the actual class.

Friends

CGCBase Class

CGCBase Class	Description
class CGenericCanvas (see page 77)	This is friend friend class CGenericCanvas.

Legend

	Method
V	virtual
A	abstract
	protected
	Data Member

1.2.7.1 Constructors

1.2.7.1.1 CConnectionInstance::CConnectionInstance Constructor

```
CConnectionInstance(CGenericCanvas* canvas, CConnection* connection, CFigureInstance*
```

```
endPoint1, CFigureInstance* endPoint2);
```

Parameters

Parameters	Description
`CGenericCanvas* canvas`	The canvas (see page 75) to which this class belongs.
`CConnection* connection`	The connection which is instantiated in this class.
`CFigureInstance* endPoint1`	The first endpoint.
`CFigureInstance* endPoint2`	The second endpoint.

Remarks

Constructor of the class.

1.2.7.2 Destructors

1.2.7.2.1 CConnectionInstance::~CConnectionInstance Destructor

```
virtual ~CConnectionInstance(void);
```

1.2.7.3 Methods

1.2.7.3.1 CConnectionInstance::computeCoordinates Method

```
void computeCoordinates(CFigureInstance* point, TConnectionDirection direction, float slot);
```

Parameters

Parameters	Description
`CFigureInstance* point`	One of the end points of this connection.
`TConnectionDirection direction`	Indicates the edge where the connection leaves the end point.
`float slot`	A distribution offset in the range (0..1) that controls the relative position along the edge.

Remarks

Computes the final coordinates of this connection at the given end point (which must be one of the end points of this connection instance).

1.2.7.3.2 CConnectionInstance::dirty Method

```
bool dirty(void);
```

Remarks

This is dirty, a member of class CConnectionInstance.

1.2.7.3.3 CConnectionInstance::endPoint1 Method

```
CFigureInstance* endPoint1(void);
```

Remarks

This is endPoint1, a member of class CConnectionInstance.

1.2.7.3.4 CConnectionInstance::endPoint2 Method

```
CFigureInstance* endPoint2(void);
```

Remarks

This is endPoint2, a member of class CConnectionInstance.

1.2.7.3.5 CConnectionInstance::getDirection Method

```
TConnectionDirection getDirection(CFigureInstance* point);
```

Parameters

Parameters	Description
`CFigureInstance* point`	The point to consider, must be one of the end points of this connection.

Returns

One of the direction flags.

Remarks

Determines in which direction this connection leaves the given point. This depends entirely on the positions of the endpoints.

1.2.7.3.6 CConnectionInstance::makeDirty Method

```
void makeDirty(bool sendEvent = true);
```

Parameters

Parameters	Description
`bool sendEvent = true`	If true then a change (see page 75) event is triggered.

Remarks

Marks the instance as invalid so its display list is recreated on next render (see page 20).

1.2.7.3.7 property

1.2.7.3.7.1 CConnectionInstance::property Method (const char*, unsigned int)

```
virtual TGCVariant __cdecl property(const char* name, unsigned int index);
```

Parameters

Parameters	Description
`const char* name`	The name of the property to return.
`unsigned int index`	The index of the sub property to return if it is located in a list.

Returns

A description of the property value and, if the property is simple, the actual value.

Remarks

Retrieves the value of the property given by path. The path syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name of a container class (e.g. layers, figures) and property is the name of a simple property of that container.

1.2.7.3.7.2 CConnectionInstance::property Method (const char*, unsigned int, const TGCVariant&)

```
virtual void __cdecl property(const char* name, unsigned int index, const TGCVariant&
value);
```

Parameters

Parameters	Description
`const char* name`	The name of the property.

`unsigned int index`	The index of the sub property to return if it is located in a list.
`const TGCVariant& value`	The new value of the property. Automatic conversion is performed where possible.

Remarks

Sets the value of the given property, which must be a simple property.

1.2.7.3.8 CConnectionInstance::render Method

```
void render(void);
```

Remarks

Renders this instance.

1.2.7.3.9 CConnectionInstance::setLineStyle Method

```
virtual void __cdecl setLineStyle(TConnectionLineStyle lineStyle);
```

Parameters

Parameters	Description
`TConnectionLineStyle lineStyle`	The new line style to apply.

Remarks

Changes the visual style of the connection lines.

1.2.7.3.10 CConnectionInstance::validate Method

```
void validate(void);
```

Remarks

Computes all detail elements for this connection depending on computed start and end point as well as other properties.

1.2.8 CConnectionLayer Class

Class Hierarchy


```
class CConnectionLayer : public CLayer;
```

File

myx_gc_layer.h (see page 226)

Remarks

The connection layer is a special layer variant (see page 166) that renders connections between figures.

Constructors

Constructor	Description
CConnectionLayer (see page 23)	CConnectionLayer

CLayer Class

CLayer Class	Description
CLayer (see page 117)	CLayer

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CConnectionLayer (see page 23)	

CLayer Class

CLayer Class	Description
~CLayer (see page 117)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Members

Constructors

Constructor	Description
CConnectionLayer (see page 23)	CConnectionLayer

CLayer Class

CLayer Class	Description
CLayer (see page 117)	CLayer

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CConnectionLayer (see page 23)	

CLayer Class

CLayer Class	Description
~CLayer (see page 117)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Methods

Method	Description
createInstance (see page 23)	Creates a new connection instance.
invalidateEndPoint (see page 24)	Marks all connection instances the are connected to the given endpoint as dirty, if they are not already.
invalidateInstances (see page 24)	Invalidates all connection instances that are connected to the end point of the given connection instance.
removeInstance (see page 24)	Removes the given instance from the internal list. The connection instance is not destroyed, though.
renderLayerContent (see page 24)	Renders all connections, which have been validate (see page 122) before in validateLayerContent (see page 24).
validateEndPoint (see page 24)	Computes the coordinates of all connection instances touching the given point.
validateLayerContent (see page 24)	Computes the layout for all connection instances.

CLayer Class

CLayer Class	Description
addInstance (see page 118)	Adds the given figure instance to the end of the instance list. If instance belongs to another layer currently it is removed from the other's instance list first.

applyTransformations (see page 118)	Applies the layer's transformations for rendering, feedback etc.
bringToFront (see page 118)	If the given figure instance is currently on this layer then it is moved to the last place in the list making it so the top most instance (they are rendered as stored in the instances array).
checkError (see page 118)	Triggers the error (see page 76) checking of the canvas (see page 75).
clear (see page 118)	This is clear, a member of class CLayer.
createInstance (see page 118)	Creates a new instance for the given figure and adds it to this layer.
enabled (see page 119)	Sets the layer's enabled state.
getHitTestInfoAt (see page 119)	Fills the hit results with all figure instances whose bounds contain the given coordinates.
makeDirty (see page 119)	Marks the display list for this layer as invalid, hence it will be recreated next time validate (see page 122) is called. If a list already exists then it is freed.
name (see page 119)	This is name, a member of class CLayer.
property (see page 120)	Retrieves the value of the property given by path. The path syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name (see page 119) of a container class (e.g. layers, figures) and property is the name (see page 119) of a simple property of that container.
removeInstance (see page 120)	Removes the given figure instance from the instance list if it is currently there. No error (see page 76) is raised if the instance does not belong to this layer.
render (see page 120)	Checks the validity of the figure display list and executes it.
renderFeedback (see page 121)	Helper method to determine the transformed vertices of the given figure instance. The layer applies its own transformations and only renders the figure instance.
renderLayerContent (see page 121)	Renders layer content that is not determined by figure instances. This method might be overridden by descendants.
scale (see page 121)	Scales the layer by the amount given in Factor. If Accumulative is true then the new scale factors are multiplied with the existing values. This version of scale uses an array of values in the parameter list.
sendToBack (see page 122)	If the given figure instance is currently on this layer then it is moved to the first place in the list making it so the bottom most instance (they are rendered as stored in the instances array).
translate (see page 122)	Moves the layer by the amount given in Tx, Ty and Tz. If Accumulative is true then the new translation factors are multiplied with the existing values. This version of translate uses an array for the values in the parameter list.
translateV (see page 122)	Moves the layer by the amount given in Factor. If Accumulative is true then the new translation factors are multiplied with the existing values. This version of translate (see page 122) uses an array for the values in the parameter list.
validate (see page 122)	Creates the display list of this figure (and all child figures) if necessary.
validateLayerContent (see page 122)	Prepares layer content that is not determined by figure instances. This method might be overridden by descendants.
visible (see page 123)	Sets the layer's visibility state.

CGCBase Class

CGCBase Class	Description
addListener (see page 75)	Adds a listener to the internal list of listeners, if it is not already there.
beginUpdate (see page 75)	Increases the update count by 1 to stop any recursive update until (@see endUpdate (see page 76)()) was called.
canvas (see page 75)	This is canvas, a member of class CGCBase.
change (see page 75)	Triggers the onCange event of all registered listeners to notfied them about a particular change.
classIs (see page 76)	Determines if this class is of a specific type by comparing its class name to the given name.
className (see page 76)	This is className, a member of class CGCBase.
destroying (see page 76)	This is destroying, a member of class CGCBase.
endUpdate (see page 76)	The counterpart to (@see beginUpdate (see page 75)). It releases one update lock and also the global lock if the count drops to 0.
error (see page 76)	Triggers the onError event of all registered listeners to notfied them about an error.
property (see page 76)	This is property, a member of class CGCBase.
release (see page 77)	This is release, a member of class CGCBase.
removeListener (see page 77)	
setDestroying (see page 77)	Helper to set destroying (see page 76) state explicitely.
updating (see page 77)	This is updating, a member of class CGCBase.

Friends

CLayer Class

CLayer Class	Description
class CFigureInstance (see page 123)	This is friend friend class CFigureInstance.
class CFigureInstanceEnumerator (see page 123)	This is friend friend class CFigureInstanceEnumerator.
class CInstanceListener (see page 123)	This is friend friend class CInstanceListener.

CGCBase Class

CGCBase Class	Description
class CGenericCanvas (see page 77)	This is friend friend class CGenericCanvas.

Data Members

CGCBase Class

CGCBase Class	Description
_className (see page 74)	Used to determine the actual class.

Legend

	Method
V	virtual
	protected
A	abstract
	Data Member

1.2.8.1 Constructors

1.2.8.1.1 CConnectionLayer::CConnectionLayer Constructor

```
CConnectionLayer(string name, CGenericCanvas* canvas);
```

Remarks

CConnectionLayer

1.2.8.2 Destructors

1.2.8.2.1 CConnectionLayer::~CConnectionLayer Destructor

```
virtual ~CConnectionLayer(void);
```

1.2.8.3 Methods

1.2.8.3.1 CConnectionLayer::createInstance Method

```
CConnectionInstance* createInstance(CConnection* connection, CFigureInstance* endPoint1,
CFigureInstance* endPoint2);
```

Parameters

Parameters	Description
`CConnection* connection`	The connection for which the instance is to be created.
`CFigureInstance* endPoint1`	One end point of the instnace.
`CFigureInstance* endPoint2`	The other end point of the instance.

Remarks

Creates a new connection instance.

1.2.8.3.2 CConnectionLayer::invalidateEndPoint Method

```
void invalidateEndPoint(CFigureInstance* point);
```

Parameters

Parameters	Description
`CFigureInstance* point`	The endpoint to iterate.

Remarks

Marks all connection instances the are connected to the given endpoint as dirty, if they are not already.

1.2.8.3.3 CConnectionLayer::invalidateInstances Method

```
void invalidateInstances(CConnectionInstance* instance);
```

Parameters

Parameters	Description
`The`	connection instance that has changed.

Remarks

Invalidates all connection instances that are connected to the end point of the given connection instance.

1.2.8.3.4 CConnectionLayer::removeInstance Method

```
void removeInstance(CConnectionInstance* instance);
```

Parameters

Parameters	Description
`CConnectionInstance* instance`	The connection instance to be removed.

Remarks

Removes the given instance from the internal list. The connection instance is not destroyed, though.

1.2.8.3.5 CConnectionLayer::renderLayerContent Method

```
virtual void renderLayerContent(void);
```

Remarks

Renders all connections, which have been validate (see page 122) before in validateLayerContent (see page 24).

1.2.8.3.6 CConnectionLayer::validateEndPoint Method

```
void validateEndPoint(CFigureInstance* point);
```

Parameters

Parameters	Description
`CFigureInstance* point`	The endpoint to validate (see page 122).

Remarks

Computes the coordinates of all connection instances touching the given point.

1.2.8.3.7 CConnectionLayer::validateLayerContent Method

```
virtual void validateLayerContent(void);
```

Remarks

Computes the layout for all connection instances.

Notes

the invalidation code (@see invalidateInstance) takes care that either all connection instances connected to an end point are dirty or none of them.

1.2.9 CElementListener Class

Class Hierarchy


```
class CElementListener : private CGCListener;
```

File

myx_gc_figure.h (see page 222)

Remarks

This is class CElementListener.

Members

Data Members

Data Member	Description
element (see page 26)	This is element, a member of class CElementListener.

Methods

Method	Description
onChange (see page 26)	CElementListener
onDestroy (see page 26)	
onError (see page 26)	

CGCListener Class

CGCListener Class	Description
onChange (see page 78)	This is onChange, a member of class CGCListener.
onDestroy (see page 78)	This is onDestroy, a member of class CGCListener.
onError (see page 78)	This is onError, a member of class CGCListener.

Friends

Friend	Description
class CFigureElement (see page 26)	This is friend friend class CFigureElement.

Legend

	protected
	Data Member
	Method
	virtual
	abstract

1.2.9.1 Data Members

1.2.9.1.1 CElementListener::element Data Member

```
CFigureElement* element;
```

Remarks

This is element, a member of class CElementListener.

1.2.9.2 Methods

1.2.9.2.1 CElementListener::onChange Method

```
virtual void __cdecl onChange(CGCBase* sender, CGCBase* origin, TGCChangeReason reason);
```

Remarks

CElementListener (see page 25)

1.2.9.2.2 CElementListener::onDestroy Method

```
virtual void __cdecl onDestroy(CGCBase* sender);
```

1.2.9.2.3 CElementListener::onError Method

```
virtual void __cdecl onError(CGCBase* sender, CGCBase* origin, const char* message);
```

1.2.9.3 Friends

1.2.9.3.1 friend class CFigureElement Friend

```
friend class CFigureElement;
```

Remarks

This is friend friend class CFigureElement.

1.2.10 CFeedbackLayer Class

Class Hierarchy


```
class CFeedbackLayer : public CLayer;
```

File

myx_gc_layer.h (see page 226)

Remarks

The selection layer is a special layer variant (see page 166) that renders decorations for selected figures and can be queried for quick hit tests and lists of selected figures.

Constructors

Constructor	Description
CFeedbackLayer (see page 29)	CFeedbackLayer

CLayer Class

CLayer Class	Description
CLayer (see page 117)	CLayer

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CFeedbackLayer (see page 30)	

CLayer Class

CLayer Class	Description
~CLayer (see page 117)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Members

Constructors

Constructor	Description
CFeedbackLayer (see page 29)	CFeedbackLayer

CLayer Class

CLayer Class	Description
CLayer (see page 117)	CLayer

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CFeedbackLayer (see page 30)	

CLayer Class

CLayer Class	Description
~CLayer (see page 117)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Methods

Method	Description
addToSelection (see page 30)	Adds the given figure instance to the current selection.
clearSelection (see page 30)	Removes all figure instances from the selection set, making it empty.
createSelectionDecoration (see page 30)	Creates the display list for the selection decoration, which is shared among all selection entries.
getFeedbackInfo (see page 30)	Determines what feedback action could be executed at the given position. The returned info is usually used to set an indicator (e.g. the mouse pointer) to a certain state to reflect what is possible at that point.
internalAddToSelection (see page 31)	Helper method to add a figure instance to the selection list. No change (see page 75) event is triggered.

internalRemoveFromSelection (see page 31)	Helper method to remove a figure instance from the selection list. No change (see page 75) event is triggered.
invalidateBounds (see page 31)	Invalidates the selection decoration of the given instance (or all instances if instance is NULL) so they are recomputed next time the selection layer draws them.
moveSelectedInstances (see page 32)	Translates all currently selected figure instances by the given amount.
removeFromSelection (see page 32)	Removes the given figure instance from the current selection.
removeInstance (see page 32)	Method from CLayer (see page 115) overriden to also remove the instance from the current selection if necessary.
renderLayerContent (see page 32)	Renders the decorations for all figure instances that are currently selected.
resizeFiguresStart (see page 32)	
resizeFiguresStop (see page 32)	
resizeFiguresTo (see page 32)	
rubberRectResize (see page 33)	When in rubber rectangle mode then this function extends the current rubber rectangle from the start point to the given coordinates and handles selection/deselection of figure instances.
rubberRectStart (see page 33)	Starts the rubber rectangle if none is active currently. Otherwise it does nothing.
rubberRectStop (see page 33)	Stops the rubber rectangle if it is active currently. Does nothing if not.
validateLayerContent (see page 33)	Creates display lists for all invalid decorations.

CLayer Class

CLayer Class	Description
addInstance (see page 118)	Adds the given figure instance to the end of the instance list. If instance belongs to another layer currently it is removed from the other's instance list first.
applyTransformations (see page 118)	Applies the layer's transformations for rendering, feedback etc.
bringToFront (see page 118)	If the given figure instance is currently on this layer then it is moved to the last place in the list making it so the top most instance (they are rendered as stored in the instances array).
checkError (see page 118)	Triggers the error (see page 76) checking of the canvas (see page 75).
clear (see page 118)	This is clear, a member of class CLayer.
createInstance (see page 118)	Creates a new instance for the given figure and adds it to this layer.
enabled (see page 119)	Sets the layer's enabled state.
getHitTestInfoAt (see page 119)	Fills the hit results with all figure instances whose bounds contain the given coordinates.
makeDirty (see page 119)	Marks the display list for this layer as invalid, hence it will be recreated next time validate (see page 122) is called. If a list already exists then it is freed.
name (see page 119)	This is name, a member of class CLayer.
property (see page 120)	Retrieves the value of the property given by path. The path syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name (see page 119) of a container class (e.g. layers, figures) and property is the name (see page 119) of a simple property of that container.
removeInstance (see page 120)	Removes the given figure instance from the instance list if it is currently there. No error (see page 76) is raised if the instance does not belong to this layer.
render (see page 120)	Checks the validity of the figure display list and executes it.
renderFeedback (see page 121)	Helper method to determine the transformed vertices of the given figure instance. The layer applies its own transformations and only renders the figure instance.
renderLayerContent (see page 121)	Renders layer content that is not determined by figure instances. This method might be overridden by descendants.
scale (see page 121)	Scales the layer by the amount given in Factor. If Accumulative is true then the new scale factors are multiplied with the existing values. This version of scale uses an array of values in the parameter list.
sendToBack (see page 122)	If the given figure instance is currently on this layer then it is moved to the first place in the list making it so the bottom most instance (they are rendered as stored in the instances array).
translate (see page 122)	Moves the layer by the amount given in Tx, Ty and Tz. If Accumulative is true then the new translation factors are multiplied with the existing values. This version of translate uses an array for the values in the parameter list.
translateV (see page 122)	Moves the layer by the amount given in Factor. If Accumulative is true then the new translation factors are multiplied with the existing values. This version of translate (see page 122) uses an array for the values in the parameter list.
validate (see page 122)	Creates the display list of this figure (and all child figures) if necessary.
validateLayerContent (see page 122)	Prepares layer content that is not determined by figure instances. This method might be overridden by descendants.
visible (see page 123)	Sets the layer's visibility state.

CGCBase Class

CGCBase Class	Description
addListener (see page 75)	Adds a listener to the internal list of listeners, if it is not already there.
beginUpdate (see page 75)	Increases the update count by 1 to stop any recursive update until (@see endUpdate (see page 76)()) was called.
canvas (see page 75)	This is canvas, a member of class CGCBase.
change (see page 75)	Triggers the onCange event of all registered listeners to notfied them about a particular change.
classIs (see page 76)	Determines if this class is of a specific type by comparing its class name to the given name.
className (see page 76)	This is className, a member of class CGCBase.
destroying (see page 76)	This is destroying, a member of class CGCBase.
endUpdate (see page 76)	The counterpart to (@see beginUpdate (see page 75)). It releases one update lock and also the global lock if the count drops to 0.
error (see page 76)	Triggers the onError event of all registered listeners to notfied them about an error.
property (see page 76)	This is property, a member of class CGCBase.
release (see page 77)	This is release, a member of class CGCBase.
removeListener (see page 77)	
setDestroying (see page 77)	Helper to set destroying (see page 76) state explicitely.
updating (see page 77)	This is updating, a member of class CGCBase.

Friends

CLayer Class

CLayer Class	Description
class CFigureInstance (see page 123)	This is friend friend class CFigureInstance.
class CFigureInstanceEnumerator (see page 123)	This is friend friend class CFigureInstanceEnumerator.
class CInstanceListener (see page 123)	This is friend friend class CInstanceListener.

CGCBase Class

CGCBase Class	Description
class CGenericCanvas (see page 77)	This is friend friend class CGenericCanvas.

Data Members

CGCBase Class

CGCBase Class	Description
_className (see page 74)	Used to determine the actual class.

Legend

	Method
	virtual
	protected
	abstract
	Data Member

1.2.10.1 Constructors

1.2.10.1.1 CFeedbackLayer::CFeedbackLayer Constructor

```
CFeedbackLayer(string name, CGenericCanvas* canvas);
```

Remarks

CFeedbackLayer

1.2.10.2 Destructors

1.2.10.2.1 CFeedbackLayer::~CFeedbackLayer Destructor

```
virtual ~CFeedbackLayer(void);
```

1.2.10.3 Methods

1.2.10.3.1 CFeedbackLayer::addToSelection Method

```
virtual void __cdecl addToSelection(CFigureInstance* instance);
```

Parameters

Parameters	Description
`CFigureInstance* instance`	The instance to be added to the selection. If it is already in the set it won't be added again.

Remarks

Adds the given figure instance to the current selection.

1.2.10.3.2 CFeedbackLayer::clearSelection Method

```
virtual void __cdecl clearSelection(void);
```

Remarks

Removes all figure instances from the selection set, making it empty.

1.2.10.3.3 CFeedbackLayer::createSelectionDecoration Method

```
void createSelectionDecoration(void);
```

Remarks

Creates the display list for the selection decoration, which is shared among all selection entries.

1.2.10.3.4 getFeedbackInfo

1.2.10.3.4.1 CFeedbackLayer::getFeedbackInfo Method (float, float, float, CFigureInstance**)

```
virtual TFeedbackInfo __cdecl getFeedbackInfo(float localX, float localY, float zoom,
CFigureInstance** instance);
```

Parameters

Parameters	Description
`CFigureInstance** instance`	[out] Returns the instance that was hit, if any.
`x`	The horizontal target position in layer space.
`y`	The vertical target position in layer space.

Returns

A flag indicating the possible action state.

Remarks

Determines what feedback action could be executed at the given position. The returned info is usually used to set an

indicator (e.g. the mouse pointer) to a certain state to reflect what is possible at that point.

1.2.10.3.4.2 CFeedbackLayer::getFeedbackInfo Method (int, int, float, CFigureInstance**)

```
virtual TFeedbackInfo __cdecl getFeedbackInfo(int windowX, int windowY, float zoom,
CFigureInstance** instance);
```

Parameters

Parameters	Description
`int windowX`	The horizontal target position in window coordinates.
`int windowY`	The vertical target position in window coordinate.
`CFigureInstance** instance`	[out] Returns the instance that was hit, if any.

Returns

A flag indicating the possible action state.

Remarks

Determines what feedback action could be executed at the given position. The returned info is usually used to set an indicator (e.g. the mouse pointer) to a certain state to reflect what is possible at that point.

1.2.10.3.5 CFeedbackLayer::internalAddToSelection Method

```
bool internalAddToSelection(CFigureInstance* instance);
```

Parameters

Parameters	Description
`CFigureInstance* instance`	The instance to add.

Returns

If the instance was added (because it wasn't already there) then true is returned, otherwise false.

Remarks

Helper method to add a figure instance to the selection list. No change (see page 75) event is triggered.

1.2.10.3.6 CFeedbackLayer::internalRemoveFromSelection Method

```
void internalRemoveFromSelection(CFigureInstance* instance);
```

Parameters

Parameters	Description
`CFigureInstance* instance`	The instance to add.

Remarks

Helper method to remove a figure instance from the selection list. No change (see page 75) event is triggered.

1.2.10.3.7 CFeedbackLayer::invalidateBounds Method

```
virtual void __cdecl invalidateBounds(CFigureInstance* instance);
```

Parameters

Parameters	Description
`CFigureInstance* instance`	The figure instance whose bounds need recomputation. If this parameter is NULL then all bounds are invalidated.

Remarks

Invalidates the selection decoration of the given instance (or all instances if instance is NULL) so they are recomputed next time the selection layer draws them.

1.2.10.3.8 CFeedbackLayer::moveSelectedInstances Method

```
void moveSelectedInstances(float x, float y, float z, bool accumulative);
```

Parameters

Parameters	Description
`float x`	The horizontal amount for moving.
`float y`	The vertical amount for moving.
`float z`	The depth amount for moving.
`bool accumulative`	Relative or absolute move.

Remarks

Translates all currently selected figure instances by the given amount.

1.2.10.3.9 CFeedbackLayer::removeFromSelection Method

```
virtual void __cdecl removeFromSelection(CFigureInstance* instance);
```

Parameters

Parameters	Description
`CFigureInstance* instance`	The instance to be removed. If it isn't actually selected then nothing happens.

Remarks

Removes the given figure instance from the current selection.

1.2.10.3.10 CFeedbackLayer::removeInstance Method

```
virtual void __cdecl removeInstance(CFigureInstance* instance);
```

Parameters

Parameters	Description
`CFigureInstance* instance`	The instance to be removed.

Remarks

Method from CLayer (see page 115) overriden to also remove the instance from the current selection if necessary.

1.2.10.3.11 CFeedbackLayer::renderLayerContent Method

```
virtual void renderLayerContent(void);
```

Remarks

Renders the decorations for all figure instances that are currently selected.

1.2.10.3.12 CFeedbackLayer::resizeFiguresStart Method

```
virtual void __cdecl resizeFiguresStart(int x, int y, TFeedbackInfo direction);
```

1.2.10.3.13 CFeedbackLayer::resizeFiguresStop Method

```
virtual void __cdecl resizeFiguresStop(void);
```

1.2.10.3.14 CFeedbackLayer::resizeFiguresTo Method

```
virtual void __cdecl resizeFiguresTo(int x, int y);
```

1.2.10.3.15 CFeedbackLayer::rubberRectResize Method

```
virtual void __cdecl rubberRectResize(int x, int y, TRRSelectionAction Action);
```

Parameters

Parameters	Description
int x	The x coordinate of the new corner.
int y	The y coordinate of the new corner.
TRRSelectionAction Action	Determines if and how figure instance selection is to be handled. See TRRSelectionAction (see page 190) for a description of the various modes.

Remarks

When in rubber rectangle mode then this function extends the current rubber rectangle from the start point to the given coordinates and handles selection/deselection of figure instances.

1.2.10.3.16 CFeedbackLayer::rubberRectStart Method

```
virtual void __cdecl rubberRectStart(TRubberRectStyle style, int x, int y, bool
removeSelection);
```

Parameters

Parameters	Description
TRubberRectStyle style	Determines the visible (see page 123) style of the rubber rectangle.
int x	The x coordinate of the start point in window coordinates. The y coordinate of the start point in window coordinates.
bool removeSelection	If true then the current selection will be cleared.

Remarks

Starts the rubber rectangle if none is active currently. Otherwise it does nothing.

1.2.10.3.17 CFeedbackLayer::rubberRectStop Method

```
virtual void __cdecl rubberRectStop(void);
```

Remarks

Stops the rubber rectangle if it is active currently. Does nothing if not.

1.2.10.3.18 CFeedbackLayer::validateLayerContent Method

```
virtual void validateLayerContent(void);
```

Remarks

Creates display lists for all invalid decorations.

1.2.11 CFigure Class

Class Hierarchy


```
class CFigure : public CGCBase;
```

File

myx_gc_figure.h (see page 222)

Remarks

CFigure is the main element in the model (see page 38) and is created from a figure template. It cannot itself appear in a scene but is represented by one or more figure instances.

Constructors

Constructor	Description
CFigure (see page 35)	CFigure

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CFigure (see page 36)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Members

Constructors

Constructor	Description
CFigure (see page 35)	CFigure

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CFigure (see page 36)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Methods

Method	Description
addMapping (see page 36)	Adds a new mapping between the given key and element to the figure, for later lookup.
applyTransformations (see page 36)	Applies the current translation, rotation and scalie factors.
bounds (see page 36)	This is bounds, a member of class CFigure.
buildFromTemplate (see page 36)	Parses the given layoutTemplate and creates its child structure.
content (see page 36)	This is content, a member of class CFigure.
controller (see page 37)	
elementFromId (see page 37)	Determines the child figure element with the given xml id and returns it.
elementFromKey (see page 37)	Determines the child figure element with the given key and returns it.
freeNotification (see page 38)	
makeDirty (see page 38)	Marks the display list for this figure as invalid, hence it will be recreated next time validate (see page 41) is called.
model (see page 38)	This is model, a member of class CFigure.
property (see page 38)	Retrieves the value of the property given by name. The name syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name of a container class (e.g. layers, figures) and property is the name of a simple property of that container.
removeMapping (see page 38)	Removes a previously added mapping (@see addMapping (see page 36)).
render (see page 39)	Checks the validity of the figure display list and executes it.
rotate (see page 39)	Turns the figure around the given axis by the angle Angle (in radians). This version of Rotate uses a vector for the rotation axis in the parameter list.

scale (see page 39)	Scales the figure by the amount given in Factor. If Accumulative is true then the new scale factors are multiplied with the existing values. This version of Scale uses an array of values in the parameter list.
template_ (see page 40)	This is template_, a member of class CFigure.
translate (see page 40)	Moves the figure by the amount given in Factor. If Accumulative is true then the new translation factors are multiplied with the existing values. This version of Translate uses an array for the values in the parameter list.
validate (see page 41)	Creates the display list of this figure (and all child figures) if necessary.

CGCBase Class

CGCBase Class	Description
addListener (see page 75)	Adds a listener to the internal list of listeners, if it is not already there.
beginUpdate (see page 75)	Increases the update count by 1 to stop any recursive update until (@see endUpdate (see page 76)()) was called.
canvas (see page 75)	This is canvas, a member of class CGCBase.
change (see page 75)	Triggers the onCange event of all registered listeners to notfied them about a particular change.
classIs (see page 76)	Determines if this class is of a specific type by comparing its class name to the given name.
className (see page 76)	This is className, a member of class CGCBase.
destroying (see page 76)	This is destroying, a member of class CGCBase.
endUpdate (see page 76)	The counterpart to (@see beginUpdate (see page 75)). It releases one update lock and also the global lock if the count drops to 0.
error (see page 76)	Triggers the onError event of all registered listeners to notfied them about an error.
property (see page 76)	This is property, a member of class CGCBase.
release (see page 77)	This is release, a member of class CGCBase.
removeListener (see page 77)	
setDestroying (see page 77)	Helper to set destroying (see page 76) state explicitely.
updating (see page 77)	This is updating, a member of class CGCBase.

Friends

Friend	Description
class CFigureInstance (see page 41)	This is friend friend class CFigureInstance.
class CGCModel (see page 41)	This is friend friend class CGCModel.

CGCBase Class

CGCBase Class	Description
class CGenericCanvas (see page 77)	This is friend friend class CGenericCanvas.

Data Members

CGCBase Class

CGCBase Class	Description
_className (see page 74)	Used to determine the actual class.

Legend

	Method
	virtual
	protected
	abstract
	Data Member

1.2.11.1 Constructors

1.2.11.1.1 CFigure::CFigure Constructor

```
CFigure(CGCModel* Owner, CFigureTemplate* Template);
```

Remarks

CFigure

1.2.11.2 Destructors

1.2.11.2.1 CFigure::~CFigure Destructor

```
virtual ~CFigure(void);
```

1.2.11.3 Methods

1.2.11.3.1 CFigure::addMapping Method

```
void addMapping(wstring path, CFigureElement* element);
```

Parameters

Parameters	Description
`wstring path`	The key used for the mapping.
`CFigureElement* element`	The element to map to the given key.

Remarks

Adds a new mapping between the given key and element to the figure, for later lookup.

1.2.11.3.2 CFigure::applyTransformations Method

```
void applyTransformations(void);
```

Remarks

Applies the current translation, rotation and scalie factors.

1.2.11.3.3 CFigure::bounds Method

```
virtual TBoundingBox __cdecl bounds(void);
```

Remarks

This is bounds, a member of class CFigure.

1.2.11.3.4 CFigure::buildFromTemplate Method

```
void buildFromTemplate(CFigureTemplate* Template);
```

Remarks

Parses the given layoutTemplate and creates its child structure.

1.2.11.3.5 CFigure::content Method

```
CFigureElement* content(void);
```

Remarks

This is content, a member of class CFigure.

1.2.11.3.6 controller

1.2.11.3.6.1 CFigure::controller Method (CFigureController *)

```
void controller(CFigureController * controller);
```

1.2.11.3.6.2 CFigure::controller Method (void)

```
CFigureController * controller(void);
```

Remarks

This is controller, a member of class CFigure.

1.2.11.3.7 CFigure::elementFromId Method

```
CFigureElement* elementFromId(wstring id);
```

Parameters

Parameters	Description
`wstring id`	The id to search for.

Returns

The figure element that corresponds to the given id or NULL if there is none.

Remarks

Determines the child figure element with the given xml id and returns it.

1.2.11.3.8 elementFromKey

1.2.11.3.8.1 CFigure::elementFromKey Method (const char*)

```
CFigureElement* elementFromKey(const char* key);
```

Parameters

Parameters	Description
`const char* key`	The key to search for (UTF-8 encoded).

Returns

The figure element that corresponds to the given key or NULL if there is none.

Remarks

Determines the child figure element with the given key and returns it.

1.2.11.3.8.2 CFigure::elementFromKey Method (wstring)

```
CFigureElement* elementFromKey(wstring key);
```

Parameters

Parameters	Description
`wstring key`	The key to search for.

Returns

The figure element that corresponds to the given key or NULL if there is none.

Remarks

Determines the child figure element with the given key and returns it. See also description of CFigureElement::elementFromKey.

1.2.11.3.9 CFigure::freeNotification Method

```
void freeNotification(CFigureElement* object);
```

1.2.11.3.10 CFigure::makeDirty Method

```
void makeDirty(void);
```

Remarks

Marks the display list for this figure as invalid, hence it will be recreated next time validate (see page 41) is called.

1.2.11.3.11 CFigure::model Method

```
CGCModel* model(void);
```

Remarks

This is model, a member of class CFigure.

1.2.11.3.12 property

1.2.11.3.12.1 CFigure::property Method (const char*, unsigned int)

```
virtual TGCVariant __cdecl property(const char* name, unsigned int index);
```

Parameters

Parameters	Description
const char* name	The name of the property.
unsigned int index	If the property is a list then this is the index into that list.

Returns

The value of the property, if found.

Remarks

Retrieves the value of the property given by name. The name syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name of a container class (e.g. layers, figures) and property is the name of a simple property of that container.

1.2.11.3.12.2 CFigure::property Method (const char*, unsigned int, const TGCVariant&)

```
virtual void __cdecl property(const char* name, unsigned int index, const TGCVariant&
value);
```

Parameters

Parameters	Description
const char* name	The name of the property.
unsigned int index	If the property is a list then this is the index into that list.
Value	The new value of the property. Automatic conversion is performed where possible.

Remarks

Sets the value of the given property, which must be a simple property.

1.2.11.3.13 CFigure::removeMapping Method

```
void removeMapping(CFigureElement* element);
```

Parameters

Parameters	Description
`CFigureElement* element`	The element to map to the given key.

Remarks

Removes a previously added mapping (@see addMapping (see page 36)).

1.2.11.3.14 CFigure::render Method

```
virtual void __cdecl render(float currentZoom);
```

Parameters

Parameters	Description
`The`	current scale (see page 39) factor.

Remarks

Checks the validity of the figure display list and executes it.

1.2.11.3.15 rotate

1.2.11.3.15.1 CFigure::rotate Method (float, const float Axis[3])

```
virtual void __cdecl rotate(float Angle, const float Axis[3]);
```

Parameters

Parameters	Description
`float Angle`	The angle in radians to turn the figure.
`Axis`	The axis around which the figure is to be rotated. note Currently there is no accumulative version of Rotate available (requires a quaternion lib, which we don't have yet).

Remarks

Turns the figure around the given axis by the angle Angle (in radians). This version of Rotate uses a vector for the rotation axis in the parameter list.

1.2.11.3.15.2 CFigure::rotate Method (float, float, float, float)

```
virtual void __cdecl rotate(float Angle, float Rx, float Ry, float Rz);
```

Parameters

Parameters	Description
`float Angle`	The angle in radians to turn the figure.
`float Rx`	The x part of the rotation axis.
`float Ry`	The y part of the rotation axis.
`float Rz`	The z part of the rotation axis. note Currently there is no accumulative version of Rotate available (requires a quaternion lib, which we don't have yet).

Remarks

Turns the figure around the given axis by the angle Angle (in radians). This version of Rotate uses a single float values in the parameter list.

1.2.11.3.16 scale

1.2.11.3.16.1 CFigure::scale Method (const float Factor[3], bool)

```
virtual void __cdecl scale(const float Factor[3], bool Accumulative = false);
```

Parameters

Parameters	Description
`bool Accumulative = false`	If true then the given values are added to any existing values otherwiese they are used as given.
`Factor`	An array containing the three scale values for x, y and z.

Remarks

Scales the figure by the amount given in Factor. If Accumulative is true then the new scale factors are multiplied with the existing values. This version of Scale uses an array of values in the parameter list.

1.2.11.3.16.2 CFigure::scale Method (float, float, float, bool)

```
virtual void __cdecl scale(float Sx, float Sy, float Sz, bool Accumulative = false);
```

Parameters

Parameters	Description
`float Sx`	The scale factor in x direction.
`float Sy`	The scale factor in y direction.
`float Sz`	The scale factor in z direction.
`bool Accumulative = false`	If true then the given values are added to any existing values otherwiese they are used as given.

Remarks

Scales the figure by the amount given in Factor. If Accumulative is true then the new scale factors are multiplied with the existing values. This version of Scale uses single float values as parameters.

1.2.11.3.17 CFigure::template_ Method

```
CFigureTemplate* template_(void);
```

Remarks

This is template_, a member of class CFigure.

1.2.11.3.18 translate

1.2.11.3.18.1 CFigure::translate Method (const float Factor[3], bool)

```
virtual void __cdecl translate(const float Factor[3], bool Accumulative = false);
```

Parameters

Parameters	Description
`bool Accumulative = false`	If true then the given values are added to any existing values otherwiese they are used as given.
`Factor`	An array of translation values, for each axis one.

Remarks

Moves the figure by the amount given in Factor. If Accumulative is true then the new translation factors are multiplied with the existing values. This version of Translate uses an array for the values in the parameter list.

1.2.11.3.18.2 CFigure::translate Method (float, float, float, bool)

```
virtual void __cdecl translate(float Tx, float Ty, float Tz, bool Accumulative = false);
```

Parameters

Parameters	Description
`float Tx`	Scale factor for the x axis.
`float Ty`	Scale factor for the y axis.

`float Tz`	Scale factor for the z axis.
`bool Accumulative = false`	If true then the given values are added to any existing values otherwise they are used as given.

Remarks

Moves the figure by the amount given in Factor. If Accumulative is true then the new translation factors are multiplied with the existing values. This version of Translate uses an array for the values in the parameter list.

1.2.11.3.19 CFigure::validate Method

```
void validate(void);
```

Remarks

Creates the display list of this figure (and all child figures) if necessary.

1.2.11.4 Friends

1.2.11.4.1 friend class CFigureInstance Friend

```
friend class CFigureInstance;
```

Remarks

This is friend friend class CFigureInstance.

1.2.11.4.2 friend class CGCModel Friend

```
friend class CGCModel;
```

Remarks

This is friend friend class CGCModel.

1.2.12 CFigureController Class

Class Hierarchy


```
class CFigureController;
```

File

myx_gc_figure.h (see page 222)

Constructors

Constructor	Description
CFigureController (see page 42)	

Members

Constructors

Constructor	Description
CFigureController (see page 42)	

Methods

Method	Description
onAddInstance (see page 42)	This is onAddInstance, a member of class CFigureController.
onChange (see page 42)	This is onChange, a member of class CFigureController.
onCreate (see page 42)	This is onCreate, a member of class CFigureController.
update (see page 42)	This is update, a member of class CFigureController.

Legend

	Method
	virtual

1.2.12.1 Constructors

1.2.12.1.1 CFigureController::CFigureController Constructor

```
CFigureController(CFigure * figure);
```

1.2.12.2 Methods

1.2.12.2.1 CFigureController::onAddInstance Method

```
virtual void onAddInstance(CFigureInstance * instance, CLayer * layer);
```

Remarks

This is onAddInstance, a member of class CFigureController.

1.2.12.2.2 CFigureController::onChange Method

```
virtual void onChange();
```

Remarks

This is onChange, a member of class CFigureController.

1.2.12.2.3 CFigureController::onCreate Method

```
virtual void onCreate();
```

Remarks

This is onCreate, a member of class CFigureController.

1.2.12.2.4 CFigureController::update Method

```
virtual void update();
```

Remarks

This is update, a member of class CFigureController.

1.2.13 CFigureElement Class

Class Hierarchy


```
class CFigureElement : public CGCBase;
```

File

myx_gc_figure.h (see page 222)

Remarks

This is class CFigureElement.

Constructors

Constructor	Description
CFigureElement (see page 45)	CFigureElement

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CFigureElement (see page 45)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Members

Constructors

Constructor	Description
CFigureElement (see page 45)	CFigureElement

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CFigureElement (see page 45)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Methods

Method	Description
addSubElement (see page 45)	Helper method to explicitely trigger creation of a child figure (see page 47) element. The point here is that this must only be called for figures that represent lists, that is, figures that only have one child template element, which can appear any number of times.
bounds (see page 45)	Returns the current bounds of this element. Validation is performed if necessary.
children (see page 45)	This is children, a member of class CFigureElement.

computeBoundingBox (see page 46)	(Re) computes the overall bounding box for this element. This includes the bounds (see page 45) of the style (see page 48) as well as all children (see page 45). It is assumed that the caller has already validated the owning figure (see page 47) (and so all contained elements).
createFromTemplate (see page 46)	Creates a figure (see page 47) element from a layoutTemplate and returns it.
doAction (see page 46)	Triggers the default action of the figure (see page 47) element that is located at the given location. The given coordinates are already in figure (see page 47) space, so no further conversion is needed.
elementFromId (see page 46)	Determines the child figure (see page 47) element with the given xml id and returns it.
figure (see page 47)	Returns the owning figure for this element.
layout (see page 47)	This is layout, a member of class CFigureElement.
makeDirty (see page 47)	Called when either a child element or the style (see page 48) changes. The event is propagated to the parent element.
property (see page 47)	Retrieves the value of the property given by name. The name syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name of a container class (e.g. layers, figures) and property is the name of a simple property of that container.
render (see page 48)	This method renders this element and triggers rendering of its (potential) caption as well as its child elements.
resize (see page 48)	Called when by user input a figure (see page 47) must be resized. The current view is handling user input and forwards the appropriate call to the figure (see page 47) instance.
setCaption (see page 48)	Convenience method to set the text of the caption of this element (if there is a caption at all).
style (see page 48)	Sets a new style to be used for this element.
template_ (see page 49)	This is template_, a member of class CFigureElement.
validate (see page 49)	Called before the owner figure (see page 47) creates its display list, so it can be used to create anything necessary that must not be done while a display list is being compiled.
zoomChanged (see page 49)	Called when the current scale factor was changed. Recompute caption if necessary.

CGCBase Class

CGCBase Class	Description
addListener (see page 75)	Adds a listener to the internal list of listeners, if it is not already there.
beginUpdate (see page 75)	Increases the update count by 1 to stop any recursive update until (@see endUpdate (see page 76)()) was called.
canvas (see page 75)	This is canvas, a member of class CGCBase.
change (see page 75)	Triggers the onCange event of all registered listeners to notfied them about a particular change.
classIs (see page 76)	Determines if this class is of a specific type by comparing its class name to the given name.
className (see page 76)	This is className, a member of class CGCBase.
destroying (see page 76)	This is destroying, a member of class CGCBase.
endUpdate (see page 76)	The counterpart to (@see beginUpdate (see page 75)). It releases one update lock and also the global lock if the count drops to 0.
error (see page 76)	Triggers the onError event of all registered listeners to notfied them about an error.
property (see page 76)	This is property, a member of class CGCBase.
release (see page 77)	This is release, a member of class CGCBase.
removeListener (see page 77)	
setDestroying (see page 77)	Helper to set destroying (see page 76) state explicitely.
updating (see page 77)	This is updating, a member of class CGCBase.

Friends

Friend	Description
class CCaptionElement (see page 49)	This is friend friend class CCaptionElement.

CGCBase Class

CGCBase Class	Description
class CGenericCanvas (see page 77)	This is friend friend class CGenericCanvas.

Data Members

CGCBase Class

CGCBase Class	Description
_className (see page 74)	Used to determine the actual class.

Legend

	Method
	virtual
	protected
	abstract
	Data Member

1.2.13.1 Constructors

1.2.13.1.1 CFigureElement::CFigureElement Constructor

```
CFigureElement(CFigureElementTemplate* aTemplate, CGenericCanvas* canvas);
```

Remarks

CFigureElement

1.2.13.2 Destructors

1.2.13.2.1 CFigureElement::~CFigureElement Destructor

```
virtual ~CFigureElement(void);
```

1.2.13.3 Methods

1.2.13.3.1 CFigureElement::addSubElement Method

```
CFigureElement* addSubElement(void);
```

Remarks

Helper method to explicitely trigger creation of a child figure (see page 47) element. The point here is that this must only be called for figures that represent lists, that is, figures that only have one child template element, which can appear any number of times.

1.2.13.3.2 CFigureElement::bounds Method

```
TBoundingBox bounds(void);
```

Remarks

Returns the current bounds of this element. Validation is performed if necessary.

1.2.13.3.3 CFigureElement::children Method

```
CElementList* children(void);
```

Remarks

This is children, a member of class CFigureElement.

1.2.13.3.4 CFigureElement::computeBoundingBox Method

```
void computeBoundingBox(void);
```

Remarks

(Re) computes the overall bounding box for this element. This includes the bounds (see page 45) of the style (see page 48) as well as all children (see page 45). It is assumed that the caller has already validated the owning figure (see page 47) (and so all contained elements).

1.2.13.3.5 CFigureElement::createFromTemplate Method

```
static CFigureElement* createFromTemplate(CGenericCanvas* canvas, CFigure* Owner,
CFigureElementTemplate* Template);
```

Parameters

Parameters	Description
owner	The controller for the new figure (see page 47) element. It is responsible to free the returned instance.
layoutTemplate	The layoutTemplate to be used when creating the figure (see page 47) element.
Model	The model to which the new element belongs.

Returns

The new figure (see page 47) element instance.

Remarks

Creates a figure (see page 47) element from a layoutTemplate and returns it.

1.2.13.3.6 CFigureElement::doAction Method

```
TActionType doAction(CFigureInstance* Instance, const float X, const float Y);
```

Parameters

Parameters	Description
instance	The figure (see page 47) instance for which the action was originally triggered.
x	The horizontal coordinate for the hit test.
y	The vertical coordinate for the hit test.

Returns

The type of the last executed action.

Remarks

Triggers the default action of the figure (see page 47) element that is located at the given location. The given coordinates are already in figure (see page 47) space, so no further conversion is needed.

1.2.13.3.7 CFigureElement::elementFromId Method

```
CFigureElement* elementFromId(wstring id);
```

Parameters

Parameters	Description
wstring id	The id to search for.

Returns

The figure (see page 47) element that corresponds to the given id or NULL if there is none.

Remarks

Determines the child figure (see page 47) element with the given xml id and returns it.

1.2.13.3.8 CFigureElement::figure Method

```
CFigure* figure(void);
```

Returns

The owner of this figure element.

Remarks

Returns the owning figure for this element.

1.2.13.3.9 CFigureElement::layout Method

```
TFigureElementLayout layout(void);
```

Remarks

This is layout, a member of class CFigureElement.

1.2.13.3.10 CFigureElement::makeDirty Method

```
void makeDirty(void);
```

Remarks

Called when either a child element or the style (see page 48) changes. The event is propagated to the parent element.

1.2.13.3.11 property

1.2.13.3.11.1 CFigureElement::property Method (const char*, unsigned int)

```
virtual TGCVariant __cdecl property(const char* name, unsigned int index);
```

Parameters

Parameters	Description
`const char* name`	The name of the property.
`unsigned int index`	If the property is a list then this is the index into that list.

Returns

A description of the property value and, if the property is simple, the actual value.

Remarks

Retrieves the value of the property given by name. The name syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name of a container class (e.g. layers, figures) and property is the name of a simple property of that container.

1.2.13.3.11.2 CFigureElement::property Method (const char*, unsigned int, const TGCVariant&)

```
virtual void __cdecl property(const char* name, unsigned int index, const TGCVariant&
value);
```

Parameters

Parameters	Description
`const char* name`	The name of the property.

`unsigned int index`	If the property is a list then this is the index into that list.
`const TGCVariant& value`	The new value of the property. Automatic conversion is performed where possible.

Remarks

Sets the value of the given property, which must be a simple property.

1.2.13.3.12 CFigureElement::render Method

```
void render(void);
```

Remarks

This method renders this element and triggers rendering of its (potential) caption as well as its child elements.

1.2.13.3.13 CFigureElement::resize Method

```
void resize(float dX, float dY, TFeedbackInfo info);
```

Parameters

Parameters	Description
`float dX`	Horizontal amount to resize.
`float dY`	vertical amount to resize.
`TFeedbackInfo info`	Result of the feedback hit test that tells us how to resize.

Remarks

Called when by user input a figure (see page 47) must be resized. The current view is handling user input and forwards the appropriate call to the figure (see page 47) instance.

1.2.13.3.14 CFigureElement::setCaption Method

```
void setCaption(const char* text);
```

Parameters

Parameters	Description
`const char* text`	The new text to display. Must be UTF-8 encoded.

Remarks

Convenience method to set the text of the caption of this element (if there is a caption at all).

1.2.13.3.15 style

1.2.13.3.15.1 CFigureElement::style Method (CGCStyle*)

```
void style(CGCStyle* NewStyle);
```

Parameters

Parameters	Description
`CGCStyle* NewStyle`	The new style to be used.

Remarks

Sets a new style to be used for this element.

1.2.13.3.15.2 CFigureElement::style Method (void)

```
CGCStyle* style(void);
```

Returns

The current style for this element.

Remarks

Returns the currently used style.

1.2.13.3.16 CFigureElement::template_ Method

```
CFigureElementTemplate* template_(void);
```

Remarks

This is template_, a member of class CFigureElement.

1.2.13.3.17 CFigureElement::validate Method

```
void validate(void);
```

Remarks

Called before the owner figure (see page 47) creates its display list, so it can be used to create anything necessary that must not be done while a display list is being compiled.

1.2.13.3.18 CFigureElement::zoomChanged Method

```
bool zoomChanged(float ZoomFactor);
```

Parameters

Parameters	Description
zoomFactor	The current zoom (scale) factor.

Returns

True if the zoom change (see page 75) has an effect on this element.

Remarks

Called when the current scale factor was changed. Recompute caption if necessary.

1.2.13.4 Friends

1.2.13.4.1 friend class CCaptionElement Friend

```
friend class CCaptionElement;
```

Remarks

This is friend friend class CCaptionElement.

1.2.14 CFigureElementListener Class

Class Hierarchy

```
class CFigureElementListener : private CGCListener;
```

File

myx_gc_figure.h (see page 222)

Remarks

This is class CFigureElementListener.

Members

Data Members

Data Member	Description
figure (see page 50)	This is figure, a member of class CFigureElementListener.

Methods

Method	Description
onChange (see page 50)	CFigureELementListener
onDestroy (see page 51)	
onError (see page 51)	

CGCListener Class

CGCListener Class	Description
onChange (see page 78)	This is onChange, a member of class CGCListener.
onDestroy (see page 78)	This is onDestroy, a member of class CGCListener.
onError (see page 78)	This is onError, a member of class CGCListener.

Friends

Friend	Description
class CFigure (see page 51)	This is friend friend class CFigure.

Legend

	protected
	Data Member
	Method
	virtual
	abstract

1.2.14.1 Data Members

1.2.14.1.1 CFigureElementListener::figure Data Member

```
CFigure* figure;
```

Remarks

This is figure, a member of class CFigureElementListener.

1.2.14.2 Methods

1.2.14.2.1 CFigureElementListener::onChange Method

```
virtual void __cdecl onChange(CGCBase* sender, CGCBase* origin, TGCChangeReason reason);
```

Remarks

CFigureELementListener

1.2.14.2.2 CFigureElementListener::onDestroy Method

```
virtual void __cdecl onDestroy(CGCBase* sender);
```

1.2.14.2.3 CFigureElementListener::onError Method

```
virtual void __cdecl onError(CGCBase* sender, CGCBase* origin, const char* message);
```

1.2.14.3 Friends

1.2.14.3.1 friend class CFigure Friend

```
friend class CFigure;
```

Remarks

This is friend friend class CFigure.

1.2.15 CFigureElementTemplate Class

Class Hierarchy


```
class CFigureElementTemplate;
```

File

myx_gc_figure.h (see page 222)

Remarks

A figure element is one detail in a figure template and so also in a figure. There can be a hierarchy of figure elements to form complex figures.

Constructors

Constructor	Description
CFigureElementTemplate (see page 52)	CFigureElementTemplate

Destructors

Destructor	Description
~CFigureElementTemplate (see page 52)	

Members

Constructors

Constructor	Description
CFigureElementTemplate (see page 52)	CFigureElementTemplate

Destructors

Destructor	Description
~CFigureElementTemplate (see page 52)	

Methods

Method	Description
addAction (see page 52)	

addChild (see page 52)	
computeBoundingBox (see page 52)	This is computeBoundingBox, a member of class CFigureElementTemplate.
freeNotification (see page 53)	Called when a style is about to be destroyed.
getListElement (see page 53)	Returns the list element of this figure element template. There is only a list element if this template is a list that is, has only one child that can appear more than once. This child element is then the list element.
initialize (see page 53)	Helper method used by the figure parser to initialize some members.
isList (see page 53)	Determines if this template is a list. A list is defined as having only one child, which is allowed to appear more than once.
key (see page 53)	This is key, a member of class CFigureElementTemplate.
occurence (see page 54)	This is occurence, a member of class CFigureElementTemplate.
setCaption (see page 54)	Sets the (optional) caption for this element. If there is already one then it is freed.

Friends

Friend	Description
class CFigureElement (see page 54)	This is friend friend class CFigureElement.

Legend

	Method
V	virtual
	protected

1.2.15.1 Constructors

1.2.15.1.1 CFigureElementTemplate::CFigureElementTemplate Constructor

```
CFigureElementTemplate(wstring id, wstring key);
```

Remarks

CFigureElementTemplate

1.2.15.2 Destructors

1.2.15.2.1 CFigureElementTemplate::~CFigureElementTemplate Destructor

```
virtual ~CFigureElementTemplate(void);
```

1.2.15.3 Methods

1.2.15.3.1 CFigureElementTemplate::addAction Method

```
void addAction(const TAction& action);
```

1.2.15.3.2 CFigureElementTemplate::addChild Method

```
void addChild(CFigureElementTemplate* Child);
```

1.2.15.3.3 CFigureElementTemplate::computeBoundingBox Method

```
void computeBoundingBox(void);
```

Remarks

This is computeBoundingBox, a member of class CFigureElementTemplate.

1.2.15.3.4 CFigureElementTemplate::freeNotification Method

```
void freeNotification(CGCBase* object);
```

Parameters

Parameters	Description
`CGCBase* object`	The object, which is about to be freed.

Remarks

Called when a style is about to be destroyed.

1.2.15.3.5 CFigureElementTemplate::getListElement Method

```
CFigureElementTemplate* getListElement(void);
```

Returns

The only child element of this template if it is a list template, otherwise NULL.

Remarks

Returns the list element of this figure element template. There is only a list element if this template is a list that is, has only one child that can appear more than once. This child element is then the list element.

1.2.15.3.6 CFigureElementTemplate::initialize Method

```
void initialize(TFigureElementLayout Layout, TFigureElementResize resizeMode, CGCStyle*
style, const TConstraints& Constraints, TOccurence Occurence);
```

Parameters

Parameters	Description
`TFigureElementResize resizeMode`	Resizable flag.
`layout`	The the layout to be used in the element.
`constraints`	The resize constraints.

Remarks

Helper method used by the figure parser to initialize some members.

1.2.15.3.7 CFigureElementTemplate::isList Method

```
bool isList(void);
```

Returns

True if this template is a list, otherwise false.

Remarks

Determines if this template is a list. A list is defined as having only one child, which is allowed to appear more than once.

1.2.15.3.8 CFigureElementTemplate::key Method

```
wstring key(void);
```

Remarks

This is key, a member of class CFigureElementTemplate.

1.2.15.3.9 CFigureElementTemplate::occurence Method

```
TOccurence occurence(void);
```

Remarks

This is occurence, a member of class CFigureElementTemplate.

1.2.15.3.10 CFigureElementTemplate::setCaption Method

```
void setCaption(CCaptionElementTemplate* Caption);
```

Parameters

Parameters	Description
`CCaptionElementTemplate* Caption`	The caption to be used from now on.

Remarks

Sets the (optional) caption for this element. If there is already one then it is freed.

1.2.15.4 Friends

1.2.15.4.1 friend class CFigureElement Friend

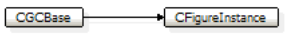
```
friend class CFigureElement;
```

Remarks

This is friend friend class CFigureElement.

1.2.16 CFigureInstance Class

Class Hierarchy




```
class CFigureInstance : public CGCBase;
```

File

myx_gc_figure.h (see page 222)

Remarks

The figure (see page 58) instance class is a proxy for a figure (see page 58) on a particular layer. There can be more than one instance pointing to the same figure (see page 58).

Constructors

Constructor	Description
CFigureInstance (see page 57)	CFigureInstance

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CFigureInstance (see page 57)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Members

Constructors

Constructor	Description
CFigureInstance (see page 57)	CFigureInstance

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CFigureInstance (see page 57)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Methods

Method	Description
applyTransformations (see page 57)	Applies the current translation, rotation and scalie factors.
bounds (see page 57)	This is bounds, a member of class CFigureInstance.
containsPoint (see page 57)	Determines whether the bounds (see page 57) of this instance overlap the given coordinates.
doAction (see page 57)	Triggers the default action of the figure (see page 58) element that is located at the given location. The given coordinates must be converted to figure (see page 58) space first, though. They are given in window coordinates. On call of this function the current view's transformations are already applied. OpenGL modelview and projection matrix are saved and restored.
figure (see page 58)	This is figure, a member of class CFigureInstance.
makeDirty (see page 58)	Marks the display list for this figure (see page 58) instance as invalid, hence it will be recreated next time validate (see page 61) is called.
onDestroy (see page 58)	Called by a class (usually CFigure (see page 33)) with which we registered us as notification sink and which is about to be destroyed.
overlaps (see page 58)	Determines whether this instance lies fully or partially within the given box.
property (see page 58)	Retrieves the value of the property given by name. The name syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name of a container class (e.g. layers, figures) and property is the name of a simple property of that container.
render (see page 59)	Checks the validity of the figure (see page 58) instance display list and executes it.
replaceFigure (see page 59)	Replaces the figure (see page 58) this instance points to.
resize (see page 59)	Called when by user input a figure (see page 58) must be resized. The current view is handling user input and forwards the appropriate call to the figure (see page 58) instance.
rotate (see page 60)	Rotates the figure (see page 58) around the given axis by the angle Angle (in radians). This version of Rotate uses a vector for the rotation axis in the parameter list.
rotateV (see page 60)	Rotates the figure (see page 58) around the given axis by the angle Angle (in radians). This version of Rotate uses a vector for the rotation axis in the parameter list.
scale (see page 60)	Scales the figure (see page 58) by the amount given in Factor. If Accumulative is true then the new scale factors are multiplied with the existing values. This version of Scale uses an array of values in the parameter list.

scaleV (see page 61)	Scales the figure (see page 58) by the amount given in Factor. If Accumulative is true then the new scale (see page 60) factors are multiplied with the existing values. This version of Scale uses an array of values in the parameter list.
selected (see page 61)	Tells the caller whether this instance is currently selected.
translate (see page 61)	Moves the figure (see page 58) by the amount given in Factor. If Accumulative is true then the new translation factors are multiplied with the existing values.
translateV (see page 61)	Moves the figure (see page 58) by the amount given in Factor. If Accumulative is true then the new translation factors are multiplied with the existing values.
validate (see page 61)	Validates the associated figure (see page 58) if it is dirty.

CGCBase Class

CGCBase Class	Description
addListener (see page 75)	Adds a listener to the internal list of listeners, if it is not already there.
beginUpdate (see page 75)	Increases the update count by 1 to stop any recursive update until (@see endUpdate (see page 76)()) was called.
canvas (see page 75)	This is canvas, a member of class CGCBase.
change (see page 75)	Triggers the onCange event of all registered listeners to notfied them about a particular change.
classIs (see page 76)	Determines if this class is of a specific type by comparing its class name to the given name.
className (see page 76)	This is className, a member of class CGCBase.
destroying (see page 76)	This is destroying, a member of class CGCBase.
endUpdate (see page 76)	The counterpart to (@see beginUpdate (see page 75)). It releases one update lock and also the global lock if the count drops to 0.
error (see page 76)	Triggers the onError event of all registered listeners to notfied them about an error.
property (see page 76)	This is property, a member of class CGCBase.
release (see page 77)	This is release, a member of class CGCBase.
removeListener (see page 77)	
setDestroying (see page 77)	Helper to set destroying (see page 76) state explicitely.
updating (see page 77)	This is updating, a member of class CGCBase.

Friends

Friend	Description
class CFeedbackLayer (see page 62)	This is friend friend class CFeedbackLayer.
class CFigure (see page 62)	This is friend friend class CFigure.
class CFigureListener (see page 62)	This is friend friend class CFigureListener.
class CGCView (see page 62)	This is friend friend class CGCView.
class CLayer (see page 62)	This is friend friend class CLayer.

CGCBase Class

CGCBase Class	Description
class CGenericCanvas (see page 77)	This is friend friend class CGenericCanvas.

Data Members

CGCBase Class

CGCBase Class	Description
_className (see page 74)	Used to determine the actual class.

Legend

	Method
	virtual
	protected
	abstract
	Data Member

1.2.16.1 Constructors

1.2.16.1.1 CFigureInstance::CFigureInstance Constructor

```
CFigureInstance(CLayer* Owner, CFigure* Figure);
```

Remarks

CFigureInstance

1.2.16.2 Destructors

1.2.16.2.1 CFigureInstance::~CFigureInstance Destructor

```
virtual ~CFigureInstance(void);
```

1.2.16.3 Methods

1.2.16.3.1 CFigureInstance::applyTransformations Method

```
void applyTransformations(void);
```

Remarks

Applies the current translation, rotation and scalie factors.

1.2.16.3.2 CFigureInstance::bounds Method

```
virtual TBoundingBox __cdecl bounds(void);
```

Remarks

This is bounds, a member of class CFigureInstance.

1.2.16.3.3 CFigureInstance::containsPoint Method

```
virtual bool __cdecl containsPoint(const float X, const float Y);
```

Parameters

Parameters	Description
x	The horizontal hit coordinate.
y	The vertical hit coordinate.

Returns

True if given coordinates are within the instance's bounds (see page 57), otherwise false,

Remarks

Determines whether the bounds (see page 57) of this instance overlap the given coordinates.

1.2.16.3.4 CFigureInstance::doAction Method

```
TActionType doAction(int windowX, int windowY);
```

Parameters

Parameters	Description
int windowX	The horizontal coordinate for the hit test.
int windowY	The vertical coordinate for the hit test.

Returns

The last executed action.

Remarks

Triggers the default action of the figure (see page 58) element that is located at the given location. The given coordinates must be converted to figure (see page 58) space first, though. They are given in window coordinates. On call of this function the current view's transformations are already applied. OpenGL modelview and projection matrix are saved and restored.

1.2.16.3.5 CFigureInstance::figure Method

```
virtual CFigure* __cdecl figure(void);
```

Remarks

This is figure, a member of class CFigureInstance.

1.2.16.3.6 CFigureInstance::makeDirty Method

```
void makeDirty(void);
```

Remarks

Marks the display list for this figure (see page 58) instance as invalid, hence it will be recreated next time validate (see page 61) is called.

1.2.16.3.7 CFigureInstance::onDestroy Method

```
void onDestroy(CGCBase* Figure);
```

Remarks

Called by a class (usually CFigure (see page 33)) with which we registered us as notification sink and which is about to be destroyed.

1.2.16.3.8 CFigureInstance::overlaps Method

```
virtual bool __cdecl overlaps(TBoundingBox& Box);
```

Parameters

Parameters	Description
`TBoundingBox& Box`	The coordinates to check against to learn if this figure (see page 58) instances is in.

Returns

True if the bounds (see page 57) of this instance at least partially overlap the given box bounds (see page 57).

Remarks

Determines whether this instance lies fully or partially within the given box.

1.2.16.3.9 property

1.2.16.3.9.1 CFigureInstance::property Method (const char*, unsigned int)

```
virtual TGCVariant __cdecl property(const char* name, unsigned int index);
```

Parameters

Parameters	Description
`const char* name`	The name of the property.
`unsigned int index`	If the property is a list then this is the index into that list.

Returns

A description of the property value and, if the property is simple, the actual value.

Remarks

Retrieves the value of the property given by name. The name syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name of a container class (e.g. layers, figures) and property is the name of a simple property of that container.

1.2.16.3.9.2 CFigureInstance::property Method (const char*, unsigned int, const TGCVariant&)

```
virtual void __cdecl property(const char* name, unsigned int index, const TGCVariant&
value);
```

Parameters

Parameters	Description
`const char* name`	The name of the property.
`unsigned int index`	If the property is a list then this is the index into that list.
`Value`	The new value of the property. Automatic conversion is performed where possible.

Remarks

Sets the value of the given property, which must be a simple property.

1.2.16.3.10 CFigureInstance::render Method

```
virtual void __cdecl render(float CurrentZoom);
```

Parameters

Parameters	Description
`currentZoom`	The current scale (see page 60) factor.

Remarks

Checks the validity of the figure (see page 58) instance display list and executes it.

1.2.16.3.11 CFigureInstance::replaceFigure Method

```
void replaceFigure(CFigure* figure);
```

Parameters

Parameters	Description
`CFigure* figure`	The new figure (see page 58) this instance should use from now on. It must not be NULL.

Remarks

Replaces the figure (see page 58) this instance points to.

1.2.16.3.12 CFigureInstance::resize Method

```
void resize(float dX, float dY, TFeedbackInfo info);
```

Parameters

Parameters	Description
`float dX`	Horizontal amount to resize.
`float dY`	vertical amount to resize.
`TFeedbackInfo info`	Result of the feedback hit test that tells us how to resize.

Remarks

Called when by user input a figure (see page 58) must be resized. The current view is handling user input and forwards the appropriate call to the figure (see page 58) instance.

1.2.16.3.13 CFigureInstance::rotate Method

```
virtual void __cdecl rotate(float Angle, float Rx, float Ry, float Rz);
```

Parameters

Parameters	Description
`float Angle`	The rotation angle in radians.
`float Rx`	The x part of the axis around which to rotate the figure (see page 58) instance.
`float Ry`	The y part of the axis around which to rotate the figure (see page 58) instance.
`float Rz`	The z part of the axis around which to rotate the figure (see page 58) instance. note: Currently there is no accumulative version of Rotate available (requires a quaternion lib, which we don't have yet).

Remarks

Rotates the figure (see page 58) around the given axis by the angle Angle (in radians). This version of Rotate uses a vector for the rotation axis in the parameter list.

1.2.16.3.14 CFigureInstance::rotateV Method

```
virtual void __cdecl rotateV(float Angle, const float Axis[3]);
```

Parameters

Parameters	Description
`float Angle`	The rotation angle in radians.
`Axis`	The axis around which to rotate (see page 60) the figure (see page 58) instance. note: Currently there is no accumulative version of Rotate available (requires a quaternion lib, which we don't have yet).

Remarks

Rotates the figure (see page 58) around the given axis by the angle Angle (in radians). This version of Rotate uses a vector for the rotation axis in the parameter list.

1.2.16.3.15 CFigureInstance::scale Method

```
virtual void __cdecl scale(float Sx, float Sy, float Sz, bool Accumulative = false);
```

Parameters

Parameters	Description
`float Sx`	The scale value for the x-axis
`float Sy`	The scale value for the y-axis
`float Sz`	The scale value for the z-axis
`bool Accumulative = false`	If true then the new scale values are added to any previously assigned values.

Remarks

Scales the figure (see page 58) by the amount given in Factor. If Accumulative is true then the new scale factors are multiplied with the existing values. This version of Scale uses an array of values in the parameter list.

1.2.16.3.16 CFigureInstance::scaleV Method

```
virtual void __cdecl scaleV(const float Factor[3], bool Accumulative = false);
```

Parameters

Parameters	Description
`bool Accumulative = false`	If true then the new scale (see page 60) values are added to any previously assigned values.
`Factor`	Contains the scaling factors for all three axes. Index 0 contains the value for the x-axis, index 1 that for the y-axis and index 2 for z.

Remarks

Scales the figure (see page 58) by the amount given in Factor. If Accumulative is true then the new scale (see page 60) factors are multiplied with the existing values. This version of Scale uses an array of values in the parameter list.

1.2.16.3.17 CFigureInstance::selected Method

```
virtual bool __cdecl selected(void);
```

Returns

True if this figure (see page 58) instance is currently selected, otherwise false.

Remarks

Tells the caller whether this instance is currently selected.

1.2.16.3.18 CFigureInstance::translate Method

```
virtual void __cdecl translate(float Tx, float Ty, float Tz, bool Accumulative = false);
```

Parameters

Parameters	Description
`float Tx`	The scale (see page 60) factor to apply on the x-axis.
`float Ty`	The scale (see page 60) factor to apply on the y-axis.
`float Tz`	The scale (see page 60) factor to apply on the z-axis.
`bool Accumulative = false`	If true scaling factors are added to the values already set previously.

Remarks

Moves the figure (see page 58) by the amount given in Factor. If Accumulative is true then the new translation factors are multiplied with the existing values.

1.2.16.3.19 CFigureInstance::translateV Method

```
virtual void __cdecl translateV(const float Factor[3], bool Accumulative = false);
```

Parameters

Parameters	Description
`bool Accumulative = false`	If true scaling factors are added to the values already set previously.
`Factor`	The scale (see page 60) factor to apply. Index 0 contains the factor for the x-axis etc.

Remarks

Moves the figure (see page 58) by the amount given in Factor. If Accumulative is true then the new translation factors are multiplied with the existing values.

1.2.16.3.20 CFigureInstance::validate Method

```
void validate(void);
```

Remarks

Validates the associated figure (see page 58) if it is dirty.

1.2.16.4 Friends

1.2.16.4.1 friend class CFeedbackLayer Friend

```
friend class CFeedbackLayer;
```

Remarks

This is friend friend class CFeedbackLayer.

1.2.16.4.2 friend class CFigure Friend

```
friend class CFigure;
```

Remarks

This is friend friend class CFigure.

1.2.16.4.3 friend class CFigureListener Friend

```
friend class CFigureListener;
```

Remarks

This is friend friend class CFigureListener.

1.2.16.4.4 friend class CGCView Friend

```
friend class CGCView;
```

Remarks

This is friend friend class CGCView.

1.2.16.4.5 friend class CLayer Friend

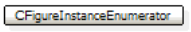
```
friend class CLayer;
```

Remarks

This is friend friend class CLayer.

1.2.17 CFigureInstanceEnumerator Class

Class Hierarchy




```
class CFigureInstanceEnumerator;
```

File

myx_gc_canvas.h (see page 218)

Remarks

The CFigureInstanceEnumerator class is for quick access to all figure instances on all (common) layers. Enumeration happens depth-first. That means for each layer first all instances are enumerated before the next (see page 64) layer is taken.

Constructors

Constructor	Description
CFigureInstanceEnumerator (see page 63)	Constructor of the enumerator class.

Members

Constructors

Constructor	Description
CFigureInstanceEnumerator (see page 63)	Constructor of the enumerator class.

Methods

Method	Description
hasNext (see page 63)	Determines if there is a next (see page 64) figure instance to enumerate.
next (see page 64)	Returns the next figure instance in the sequence.
release (see page 64)	Frees this enumerator instance. Usually called by non-C++ languages as memory is managed by the C++ runtime.
reset (see page 64)	Resets the enumerator to the first figure instance in the canvas.

Legend

	Method
	virtual

1.2.17.1 Constructors

1.2.17.1.1 CFigureInstanceEnumerator::CFigureInstanceEnumerator Constructor

```
CFigureInstanceEnumerator(CGenericCanvas* Canvas);
```

Parameters

Parameters	Description
canvas	The canvas which contains the layers which are to be enumerated.

Remarks

Constructor of the enumerator class.

1.2.17.2 Methods

1.2.17.2.1 CFigureInstanceEnumerator::hasNext Method

```
virtual bool __cdecl hasNext(void);
```

Returns

True if there is still a figure instance otherwise false.

Remarks

Determines if there is a next (see page 64) figure instance to enumerate.

1.2.17.2.2 CFigureInstanceEnumerator::next Method

```
virtual CFigureInstance* __cdecl next(void);
```

Returns

The next figure instance.

Remarks

Returns the next figure instance in the sequence.

1.2.17.2.3 CFigureInstanceEnumerator::release Method

```
virtual void __cdecl release(void);
```

Remarks

Frees this enumerator instance. Usually called by non-C++ languages as memory is managed by the C++ runtime.

1.2.17.2.4 CFigureInstanceEnumerator::reset Method

```
virtual void __cdecl reset(void);
```

Remarks

Resets the enumerator to the first figure instance in the canvas.

1.2.18 CFigureParser Class

Class Hierarchy


```
class CFigureParser : public CGCBase;
```

File

myx_gc_figure_parser.h (see page 223)

Remarks

CFigureParser converts a figure descriptions given in XML to elements in our internal model.

Constructors

Constructor	Description
CFigureParser (see page 66)	CFigureParser

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CFigureParser (see page 66)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Members

Constructors

Constructor	Description
CFigureParser (see page 66)	CFigureParser

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CFigureParser (see page 66)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Methods

Method	Description
checkLookupTables (see page 66)	Checks if the static lookup tables are set up already. If not then it is done.
parseActions (see page 66)	Parses an "action" entry in the layout definition.
parseCaption (see page 66)	Takes the given XML node and interprets it as a caption definition.
parseElement (see page 67)	Parses a single element and returns a new figure element instance. Can be called recursively.
parseLayoutDefinition (see page 67)	Parses a single layout definition and creates a figure template from it.
property (see page 67)	This is property, a member of class CFigureParser.

CGCBase Class

CGCBase Class	Description
addListener (see page 75)	Adds a listener to the internal list of listeners, if it is not already there.
beginUpdate (see page 75)	Increases the update count by 1 to stop any recursive update until (@see endUpdate (see page 76)()) was called.
canvas (see page 75)	This is canvas, a member of class CGCBase.
change (see page 75)	Triggers the onCange event of all registered listeners to notfied them about a particular change.
classIs (see page 76)	Determines if this class is of a specific type by comparing its class name to the given name.
className (see page 76)	This is className, a member of class CGCBase.
destroying (see page 76)	This is destroying, a member of class CGCBase.
endUpdate (see page 76)	The counterpart to (@see beginUpdate (see page 75)). It releases one update lock and also the global lock if the count drops to 0.
error (see page 76)	Triggers the onError event of all registered listeners to notfied them about an error.
property (see page 76)	This is property, a member of class CGCBase.
release (see page 77)	This is release, a member of class CGCBase.
removeListener (see page 77)	
setDestroying (see page 77)	Helper to set destroying (see page 76) state explicitely.
updating (see page 77)	This is updating, a member of class CGCBase.

Data Members

CGCBase Class

CGCBase Class	Description
_className (see page 74)	Used to determine the actual class.

Friends

CGCBase Class

CGCBase Class	Description
class CGenericCanvas (see page 77)	This is friend friend class CGenericCanvas.

Legend

	Method
V	virtual
	protected
A	abstract
	Data Member

1.2.18.1 Constructors

1.2.18.1.1 CFigureParser::CFigureParser Constructor

```
CFigureParser(CGenericCanvas* canvas);
```

Remarks

CFigureParser

1.2.18.2 Destructors

1.2.18.2.1 CFigureParser::~CFigureParser Destructor

```
virtual ~CFigureParser(void);
```

1.2.18.3 Methods

1.2.18.3.1 CFigureParser::checkLookupTables Method

```
void checkLookupTables(void);
```

Remarks

Checks if the static lookup tables are set up already. If not then it is done.

1.2.18.3.2 CFigureParser::parseActions Method

```
void parseActions(string source, CFigureElementTemplate* template_, unsigned short
lineNumber);
```

Parameters

Parameters	Description
`string source`	The source string to parse.
`CFigureElementTemplate* template_`	The target template that gets the actions.

Remarks

Parses an "action" entry in the layout definition.

1.2.18.3.3 CFigureParser::parseCaption Method

```
CCaptionElementTemplate* parseCaption(xmlNodePtr Node);
```

Parameters

Parameters	Description
node	The XML node to parse.

Returns

A new caption element.

Remarks

Takes the given XML node and interprets it as a caption definition.

1.2.18.3.4 CFigureParser::parseElement Method

```
CFigureElementTemplate* parseElement(xmlNodePtr Node, CGCModel* Model);
```

Parameters

Parameters	Description
node	The XML node to parse.

Returns

The new figure element instance created out of the element description.

Remarks

Parses a single element and returns a new figure element instance. Can be called recursively.

1.2.18.3.5 CFigureParser::parseLayoutDefinition Method

```
void parseLayoutDefinition(xmlNodePtr Definition, CGCModel* Model);
```

Parameters

Parameters	Description
definition	The definition to parse.
The	model class that gets the new template.

Remarks

Parses a single layout definition and creates a figure template from it.

1.2.18.3.6 property

1.2.18.3.6.1 CFigureParser::property Method (const char*, unsigned int)

```
virtual TGCVariant __cdecl property(const char* name, unsigned int index);
```

Remarks

This is property, a member of class CFigureParser.

1.2.18.3.6.2 CFigureParser::property Method (const char*, unsigned int, const TGCVariant&)

```
virtual void __cdecl property(const char* name, unsigned int index, const TGCVariant&
value);
```

Remarks

This is property, a member of class CFigureParser.

1.2.19 CFigureTemplate Class

Class Hierarchy


```
class CFigureTemplate : public CGCBase;
```

File

myx_gc_figure.h (see page 222)

Remarks

CFigureTemplate is a description of how a concrete figure has to look and act. It is loaded from a description file and created by the figure parser.

Constructors

Constructor	Description
CFigureTemplate (see page 69)	CFigureTemplate

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CFigureTemplate (see page 70)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Members

Constructors

Constructor	Description
CFigureTemplate (see page 69)	CFigureTemplate

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CFigureTemplate (see page 70)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Methods

Method	Description
content (see page 70)	This is content, a member of class CFigureTemplate.
layoutClass (see page 70)	This is layoutClass, a member of class CFigureTemplate.
property (see page 70)	Retrieves the value of the property given by path. The path syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name of a container class (e.g. layers, figures) and property is the name of a simple property of that container.

type (see page 71)	This is type, a member of class CFigureTemplate.

CGCBase Class

CGCBase Class	Description
addListener (see page 75)	Adds a listener to the internal list of listeners, if it is not already there.
beginUpdate (see page 75)	Increases the update count by 1 to stop any recursive update until (@see endUpdate (see page 76)()) was called.
canvas (see page 75)	This is canvas, a member of class CGCBase.
change (see page 75)	Triggers the onCange event of all registered listeners to notfied them about a particular change.
classIs (see page 76)	Determines if this class is of a specific type by comparing its class name to the given name.
className (see page 76)	This is className, a member of class CGCBase.
destroying (see page 76)	This is destroying, a member of class CGCBase.
endUpdate (see page 76)	The counterpart to (@see beginUpdate (see page 75)). It releases one update lock and also the global lock if the count drops to 0.
error (see page 76)	Triggers the onError event of all registered listeners to notfied them about an error.
property (see page 76)	This is property, a member of class CGCBase.
release (see page 77)	This is release, a member of class CGCBase.
removeListener (see page 77)	
setDestroying (see page 77)	Helper to set destroying (see page 76) state explicitely.
updating (see page 77)	This is updating, a member of class CGCBase.

Friends

Friend	Description
class CFigureParser (see page 71)	This is friend friend class CFigureParser.

CGCBase Class

CGCBase Class	Description
class CGenericCanvas (see page 77)	This is friend friend class CGenericCanvas.

Data Members

CGCBase Class

CGCBase Class	Description
_className (see page 74)	Used to determine the actual class.

Legend

	Method
	virtual
	abstract
	protected
	Data Member

1.2.19.1 Constructors

1.2.19.1.1 CFigureTemplate::CFigureTemplate Constructor

```
CFigureTemplate(CGCModel* model, wstring type, wstring layoutClass);
```

Remarks

CFigureTemplate

1.2.19.2 Destructors

1.2.19.2.1 CFigureTemplate::~CFigureTemplate Destructor

```
virtual ~CFigureTemplate(void);
```

1.2.19.3 Methods

1.2.19.3.1 CFigureTemplate::content Method

```
CFigureElementTemplate* content(void);
```

Remarks

This is content, a member of class CFigureTemplate.

1.2.19.3.2 CFigureTemplate::layoutClass Method

```
wstring layoutClass(void);
```

Remarks

This is layoutClass, a member of class CFigureTemplate.

1.2.19.3.3 property

1.2.19.3.3.1 CFigureTemplate::property Method (const char*, unsigned int)

```
virtual TGCVariant __cdecl property(const char* name, unsigned int index);
```

Parameters

Parameters	Description
`const char* name`	The name of the property to return.
`unsigned int index`	The index of the sub property to return if it is located in a list.

Returns

A description of the property value and, if the property is simple, the actual value.

Remarks

Retrieves the value of the property given by path. The path syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name of a container class (e.g. layers, figures) and property is the name of a simple property of that container.

1.2.19.3.3.2 CFigureTemplate::property Method (const char*, unsigned int, const TGCVariant&)

```
virtual void __cdecl property(const char* name, unsigned int index, const TGCVariant&
value);
```

Parameters

Parameters	Description
`const char* name`	The name of the property.
`unsigned int index`	The index of the sub property to return if it is located in a list.
`const TGCVariant& value`	The new value of the property. Automatic conversion is performed where possible.

Remarks

Sets the value of the given property, which must be a simple property.

1.2.19.3.4 CFigureTemplate::type Method

```
wstring type(void);
```

Remarks

This is type, a member of class CFigureTemplate.

1.2.19.4 Friends

1.2.19.4.1 friend class CFigureParser Friend

```
friend class CFigureParser;
```

Remarks

This is friend friend class CFigureParser.

1.2.20 CFontManager Class

Class Hierarchy


```
class CFontManager;
```

File

myx_gc_font_manager.h (see page 224)

Remarks

CFontManager is a helper class for text output in the generic canvas. It maps a description string for a font with attributes to a display list. If there is no display for a given font then one is created. The font manager is basically a singleton class. We only need one instance of it.

Constructors

Constructor	Description
CFontManager (see page 72)	Constructor of the class.

Destructors

Destructor	Description
~CFontManager (see page 72)	Destructor of the class. Does some clean up.

Members

Constructors

Constructor	Description
CFontManager (see page 72)	Constructor of the class.

Destructors

Destructor	Description
~CFontManager (see page 72)	Destructor of the class. Does some clean up.

Methods

Method	Description
boundingBox (see page 72)	
clear (see page 72)	Clears the font file list.
createLookupKey (see page 72)	Creates a string that can be used for lookup in the font list. Either parameter can be NULL (or 0 for Weight) causing the manager to use the default values for each missing parameter. See header file for the list of default values.
getFontFile (see page 73)	Determines platform dependantly the full path of a font file depending on some characteristics.
textOut (see page 73)	

Legend

	Method
V	virtual
	protected

1.2.20.1 Constructors

1.2.20.1.1 CFontManager::CFontManager Constructor

```
CFontManager(void);
```

Remarks

Constructor of the class.

1.2.20.2 Destructors

1.2.20.2.1 CFontManager::~CFontManager Destructor

```
virtual ~CFontManager(void);
```

Remarks

Destructor of the class. Does some clean up.

1.2.20.3 Methods

1.2.20.3.1 CFontManager::boundingBox Method

```
void boundingBox(const wstring& Output, string FontFamily, string FontStyle, int Weight,
int FontSize, TBoundingBox* Box);
```

1.2.20.3.2 CFontManager::clear Method

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

Remarks

Clears the font file list.

1.2.20.3.3 CFontManager::createLookupKey Method

```
string createLookupKey(const string& Family, const string& Style, int Weight, int FontSize);
```

Parameters

Parameters	Description
`const string& Family`	The font family (Arial, Courier etc.)
`const string& Style`	The font style.
`int Weight`	The boldness of the font (400 for normal).
`int FontSize`	The font size.

Remarks

Creates a string that can be used for lookup in the font list. Either parameter can be NULL (or 0 for Weight) causing the manager to use the default values for each missing parameter. See header file for the list of default values.

1.2.20.3.4 CFontManager::getFontFile Method

```
string getFontFile(string Family, string Style, int Weight);
```

Parameters

Parameters	Description
`string Family`	The font family (e.g. Arial, Tahoma, Verdana).
`string Style`	The font style (normal, italic).
`int Weight`	The "boldness" of the font in the range of [100..900]. Currently values <= 400 are considered als normal font, everything else as bold.

Returns

The full path and file name of a font that represents the given characteristics. A kind of intelligent replacement is done here, though. If there is no file with the given characteristics (or cannot be found) then the one from the same family, but normal styles, is used instead. If there is no entry for the family or even the standard style for a family cannot be found then Arial Standard is returned as default. If this files is also not present on the system then the FTGL lib will throw an exception. note The returned file name is ANSI encoded as FTGL expects it so.

Remarks

Determines platform dependantly the full path of a font file depending on some characteristics.

1.2.20.3.5 CFontManager::textOut Method

```
void textOut(const wstring& Output, string FontFamily, string FontStyle, int Weight, int
FontSize);
```

1.2.21 CGCBase Class

Class Hierarchy


```
class CGCBase;
```

File

myx_gc_base.h (see page 217)

Remarks

CGCBase serves as general base class for all generic canvas (see page 75) classes.

Constructors

Constructor	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CGCBase (see page 75)	

Members

Data Members

Data Member	Description
_className (see page 74)	Used to determine the actual class.

Constructors

Constructor	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CGCBase (see page 75)	

Methods

Method	Description
addListener (see page 75)	Adds a listener to the internal list of listeners, if it is not already there.
beginUpdate (see page 75)	Increases the update count by 1 to stop any recursive update until (@see endUpdate (see page 76)()) was called.
canvas (see page 75)	This is canvas, a member of class CGCBase.
change (see page 75)	Triggers the onCange event of all registered listeners to notfied them about a particular change.
classIs (see page 76)	Determines if this class is of a specific type by comparing its class name to the given name.
className (see page 76)	This is className, a member of class CGCBase.
destroying (see page 76)	This is destroying, a member of class CGCBase.
endUpdate (see page 76)	The counterpart to (@see beginUpdate (see page 75)). It releases one update lock and also the global lock if the count drops to 0.
error (see page 76)	Triggers the onError event of all registered listeners to notfied them about an error.
property (see page 76)	This is property, a member of class CGCBase.
release (see page 77)	This is release, a member of class CGCBase.
removeListener (see page 77)	
setDestroying (see page 77)	Helper to set destroying (see page 76) state explicitely.
updating (see page 77)	This is updating, a member of class CGCBase.

Friends

Friend	Description
class CGenericCanvas (see page 77)	This is friend friend class CGenericCanvas.

Legend

	protected
	Data Member
	Method
	virtual
	abstract

1.2.21.1 Data Members

1.2.21.1.1 CGCBase::_className Data Member

```
string _className;
```

Remarks

Used to determine the actual class.

1.2.21.2 Constructors

1.2.21.2.1 CGCBase::CGCBase Constructor

```
CGCBase(CGenericCanvas* canvas);
```

Remarks

CGCBase

1.2.21.3 Destructors

1.2.21.3.1 CGCBase::~CGCBase Destructor

```
virtual ~CGCBase(void);
```

1.2.21.4 Methods

1.2.21.4.1 CGCBase::addListener Method

```
virtual void __cdecl addListener(CGCListener* listener);
```

Parameters

Parameters	Description
`CGCListener* listener`	The new listener to add to the internal list. A listener is only added once.

Remarks

Adds a listener to the internal list of listeners, if it is not already there.

1.2.21.4.2 CGCBase::beginUpdate Method

```
virtual void __cdecl beginUpdate(void);
```

Remarks

Increases the update count by 1 to stop any recursive update until (@see endUpdate (see page 76)()) was called.

1.2.21.4.3 CGCBase::canvas Method

```
virtual CGenericCanvas* __cdecl canvas(void);
```

Remarks

This is canvas, a member of class CGCBase.

1.2.21.4.4 CGCBase::change Method

```
virtual void __cdecl change(CGCBase* origin, TGCChangeReason reason);
```

Parameters

Parameters	Description
`CGCBase* origin`	The object where the change happend.
`TGCChangeReason reason`	What was the reason of that change?

Remarks

Triggers the onCange event of all registered listeners to notfied them about a particular change.

1.2.21.4.5 CGCBase::classIs Method

```
virtual bool __cdecl classIs(const char* className);
```

Parameters

Parameters	Description
const char* className	Name to compare.

Remarks

Determines if this class is of a specific type by comparing its class name to the given name.

1.2.21.4.6 CGCBase::className Method

```
virtual const char* __cdecl className(void);
```

Remarks

This is className, a member of class CGCBase.

1.2.21.4.7 CGCBase::destroying Method

```
virtual bool __cdecl destroying(void);
```

Remarks

This is destroying, a member of class CGCBase.

1.2.21.4.8 CGCBase::endUpdate Method

```
virtual void __cdecl endUpdate(void);
```

Remarks

The counterpart to (@see beginUpdate (see page 75)). It releases one update lock and also the global lock if the count drops to 0.

1.2.21.4.9 CGCBase::error Method

```
virtual void __cdecl error(CGCBase* origin, const char* message);
```

Parameters

Parameters	Description
CGCBase* origin	The object where the error happend.
const char* message	A message describing the error.

Remarks

Triggers the onError event of all registered listeners to notfied them about an error.

1.2.21.4.10 property

1.2.21.4.10.1 CGCBase::property Method (const char*, unsigned int)

```
virtual TGCVariant __cdecl property(const char* name, unsigned int index) = 0;
```

Remarks

This is property, a member of class CGCBase.

1.2.21.4.10.2 CGCBase::property Method (const char*, unsigned int, const TGCVariant&)

```
virtual void __cdecl property(const char* name, unsigned int index, const TGCVariant&
value) = 0;
```

Remarks

This is property, a member of class CGCBase.

1.2.21.4.11 CGCBase::release Method

```
virtual void __cdecl release(void);
```

Remarks

This is release, a member of class CGCBase.

1.2.21.4.12 CGCBase::removeListener Method

```
virtual void __cdecl removeListener(CGCListener* listener);
```

1.2.21.4.13 CGCBase::setDestroying Method

```
void setDestroying(void);
```

Remarks

Helper to set destroying (see page 76) state explicitely.

1.2.21.4.14 CGCBase::updating Method

```
virtual bool __cdecl updating(void);
```

Remarks

This is updating, a member of class CGCBase.

1.2.21.5 Friends

1.2.21.5.1 friend class CGenericCanvas Friend

```
friend class CGenericCanvas;
```

Remarks

This is friend friend class CGenericCanvas.

1.2.22 CGCListener Class

Class Hierarchy

```
class CGCListener;
```

File

myx_gc_base.h (see page 217)

Remarks

The general listener class is used to notify users of the canvas about general events like repaints and errors. This class is only an abstract class and must get a concrete implemention in the application. All Listener classes are meant to be a means for calling back the application. They are implemented and instantiated in the application and must be freed there. Don't forget to remove the listener class before you free it!

Members

Methods

Method	Description
onChange (see page 78)	This is onChange, a member of class CGCListener.
onDestroy (see page 78)	This is onDestroy, a member of class CGCListener.
onError (see page 78)	This is onError, a member of class CGCListener.

Legend

	Method
	abstract

1.2.22.1 Methods

1.2.22.1.1 CGCListener::onChange Method

```
virtual void __cdecl onChange(CGCBase* sender, CGCBase* origin, TGCChangeReason reason) = 0;
```

Remarks

This is onChange, a member of class CGCListener.

1.2.22.1.2 CGCListener::onDestroy Method

```
virtual void __cdecl onDestroy(CGCBase* sender) = 0;
```

Remarks

This is onDestroy, a member of class CGCListener.

1.2.22.1.3 CGCListener::onError Method

```
virtual void __cdecl onError(CGCBase* sender, CGCBase* origin, const char* message) = 0;
```

Remarks

This is onError, a member of class CGCListener.

1.2.23 CGCModel Class

Class Hierarchy


```
class CGCModel : public CGCBase;
```

File

myx_gc_model.h (see page 228)

Remarks

This is class CGCModel.

Constructors

Constructor	Description
CGCModel (see page 81)	CGCModel

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CGCModel (see page 81)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Members

Constructors

Constructor	Description
CGCModel (see page 81)	CGCModel

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CGCModel (see page 81)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Methods

Method	Description
addFigure (see page 81)	Adds the given figure to the internal list and registers a listener with it.
clearConnections (see page 81)	Removes all connections.
clearFigures (see page 81)	Clears the model, that is, the figures defined in this model.
clearLayouts (see page 81)	Clears the figure template list (layouts). note: Existing figures are not concerned by deleting the template list.
clearStyles (see page 81)	Clears all defined styles. note: Deleting all styles means to leave all figures without visual representation.
createConnection (see page 82)	Creates a connection object for a logical connection between both end points. This connection, like a figure, cannot be displayed on its own. It needs a connection instance, which is placed on a connection layer in a view. Connections aren't directed so there is no start point. They just have two end points.
createFigure (see page 82)	Creates a new figure and puts in into the internal list.
createLayout (see page 82)	Creates and returns the layout (see page 82) entry for the given type and class. If the layout (see page 82) entry already exists then no new instance is created, but the existing one is returned.
layout (see page 82)	Same as the getLayout(wstring, wstring) but for UTF-8 encoded strings.

property (see page 83)	Retrieves the value of the property given by path. The path syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name of a container class (e.g. layers, figures) and property is the name of a simple property of that container.
removeConnection (see page 83)	Removes the given connection from the model. The connection itself is not destroyed.
removeFigure (see page 84)	Removes the given figure from the internal figures list.
style (see page 84)	Returns the style entry for the given identifier. If there is no style with that name one is created.

CGCBase Class

CGCBase Class	Description
addListener (see page 75)	Adds a listener to the internal list of listeners, if it is not already there.
beginUpdate (see page 75)	Increases the update count by 1 to stop any recursive update until (@see endUpdate (see page 76)()) was called.
canvas (see page 75)	This is canvas, a member of class CGCBase.
change (see page 75)	Triggers the onCange event of all registered listeners to notfied them about a particular change.
classIs (see page 76)	Determines if this class is of a specific type by comparing its class name to the given name.
className (see page 76)	This is className, a member of class CGCBase.
destroying (see page 76)	This is destroying, a member of class CGCBase.
endUpdate (see page 76)	The counterpart to (@see beginUpdate (see page 75)). It releases one update lock and also the global lock if the count drops to 0.
error (see page 76)	Triggers the onError event of all registered listeners to notfied them about an error.
property (see page 76)	This is property, a member of class CGCBase.
release (see page 77)	This is release, a member of class CGCBase.
removeListener (see page 77)	
setDestroying (see page 77)	Helper to set destroying (see page 76) state explicitely.
updating (see page 77)	This is updating, a member of class CGCBase.

Friends

Friend	Description
class CFigure (see page 84)	This is friend friend class CFigure.
class CFigureParser (see page 84)	This is friend friend class CFigureParser.
class CSVGParser (see page 84)	This is friend friend class CSVGParser.

CGCBase Class

CGCBase Class	Description
class CGenericCanvas (see page 77)	This is friend friend class CGenericCanvas.

Data Members

CGCBase Class

CGCBase Class	Description
_className (see page 74)	Used to determine the actual class.

Legend

	Method
	virtual
	protected
	abstract
	Data Member

1.2.23.1 Constructors

1.2.23.1.1 CGCModel::CGCModel Constructor

```
CGCModel(CGenericCanvas* canvas);
```

Remarks

CGCModel

1.2.23.2 Destructors

1.2.23.2.1 CGCModel::~CGCModel Destructor

```
virtual ~CGCModel(void);
```

1.2.23.3 Methods

1.2.23.3.1 CGCModel::addFigure Method

```
void addFigure(CFigure* figure);
```

Remarks

Adds the given figure to the internal list and registers a listener with it.

1.2.23.3.2 CGCModel::clearConnections Method

```
void clearConnections(void);
```

Remarks

Removes all connections.

1.2.23.3.3 CGCModel::clearFigures Method

```
void clearFigures(void);
```

Remarks

Clears the model, that is, the figures defined in this model.

1.2.23.3.4 CGCModel::clearLayouts Method

```
void clearLayouts(void);
```

Remarks

Clears the figure template list (layouts).

note: Existing figures are not concerned by deleting the template list.

1.2.23.3.5 CGCModel::clearStyles Method

```
void clearStyles(void);
```

Remarks

Clears all defined styles.

note: Deleting all styles means to leave all figures without visual representation.

1.2.23.3.6 CGCModel::createConnection Method

```
CConnection* createConnection(CFigure* endPoint1, CFigure* endPoint2);
```

Parameters

Parameters	Description
`CFigure* endPoint1`	One endpoint of the connection. Another endpoint of the connection

Remarks

Creates a connection object for a logical connection between both end points. This connection, like a figure, cannot be displayed on its own. It needs a connection instance, which is placed on a connection layer in a view. Connections aren't directed so there is no start point. They just have two end points.

1.2.23.3.7 CGCModel::createFigure Method

```
CFigure* createFigure(wstring type, wstring layoutClass);
```

Parameters

Parameters	Description
`wstring type`	The name of the application defined type for which to create the figure.
`wstring layoutClass`	The layout (see page 82) class to be used (e.g. icon, detail). Also application defined.

Remarks

Creates a new figure and puts in into the internal list.

1.2.23.3.8 CGCModel::createLayout Method

```
CFigureTemplate* createLayout(wstring type, wstring layoutClass);
```

Parameters

Parameters	Description
`wstring type`	The type to which the layout (see page 82) is associated.
`wstring layoutClass`	An additional criterion specifying a certain layout (see page 82) arrangement. This value is app. specific.

Returns

The corresponding layout (see page 82) entry.

Remarks

Creates and returns the layout (see page 82) entry for the given type and class. If the layout (see page 82) entry already exists then no new instance is created, but the existing one is returned.

1.2.23.3.9 layout

1.2.23.3.9.1 CGCModel::layout Method (const char*, const char*)

```
CFigureTemplate* layout(const char* type, const char* layoutClass);
```

Remarks

Same as the getLayout(wstring, wstring) but for UTF-8 encoded strings.

1.2.23.3.9.2 CGCModel::layout Method (wstring, wstring)

```
CFigureTemplate* layout(wstring type, wstring layoutClass);
```

Parameters

Parameters	Description
`wstring type`	The type to which the style (see page 84) is associated.
`wstring layoutClass`	An additional criterion specifying a certain layout arrangement. This value is app. specific.

Returns

The corresponding layout entry if it exists or NULL if not.

Remarks

Returns the layout entry for the given type if it exists.

1.2.23.3.10 property

1.2.23.3.10.1 CGCModel::property Method (const char*, unsigned int)

```
virtual TGCVariant __cdecl property(const char* name, unsigned int index);
```

Parameters

Parameters	Description
`const char* name`	The name of the property to return.
`unsigned int index`	If the property is a list then this is the index into that list.

Returns

A description of the property value and, if the property is simple, the actual value.

Remarks

Retrieves the value of the property given by path. The path syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name of a container class (e.g. layers, figures) and property is the name of a simple property of that container.

1.2.23.3.10.2 CGCModel::property Method (const char*, unsigned int, const TGCVariant&)

```
virtual void __cdecl property(const char* name, unsigned int index, const TGCVariant&
value);
```

Parameters

Parameters	Description
`const char* name`	The name of the property.
`unsigned int index`	If the property is a list then this is the index into that list.
`Value`	The new value of the property. Automatic conversion is performed where possible.

Remarks

Sets the value of the given property, which must be a simple property.

1.2.23.3.11 CGCModel::removeConnection Method

```
void removeConnection(CConnection* connection);
```

Parameters

Parameters	Description
`CConnection* connection`	The connection to be removed from this model.

Remarks

Removes the given connection from the model. The connection itself is not destroyed.

1.2.23.3.12 CGCModel::removeFigure Method

```
void removeFigure(CFigure* figure);
```

Parameters

Parameters	Description
`CFigure* figure`	The figure to be removed.

Remarks

Removes the given figure from the internal figures list.

1.2.23.3.13 CGCModel::style Method

```
CGCStyle* style(wstring ID);
```

Parameters

Parameters	Description
`wstring ID`	The identification of the style.

Returns

The corresponding style entry, can be NULL if ID is empty.

Remarks

Returns the style entry for the given identifier. If there is no style with that name one is created.

1.2.23.4 Friends

1.2.23.4.1 friend class CFigure Friend

```
friend class CFigure;
```

Remarks

This is friend friend class CFigure.

1.2.23.4.2 friend class CFigureParser Friend

```
friend class CFigureParser;
```

Remarks

This is friend friend class CFigureParser.

1.2.23.4.3 friend class CSVGParser Friend

```
friend class CSVGParser;
```

Remarks

This is friend friend class CSVGParser.

1.2.24 CGCStyle Class

Class Hierarchy


```
class CGCStyle : public CGCBase;
```

File

myx_gc_style.h (see page 229)

Remarks

A compiled style with its associated bounding box.

Constructors

Constructor	Description
CGCStyle (see page 86)	CGCStyle

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CGCStyle (see page 87)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Members

Constructors

Constructor	Description
CGCStyle (see page 86)	CGCStyle

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CGCStyle (see page 87)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Methods

Method	Description
boundingBox (see page 87)	This is boundingBox, a member of class CGCStyle.
displayList (see page 87)	This is displayList, a member of class CGCStyle.
property (see page 87)	Retrieves the value of the property given by path. The path syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name of a container class (e.g. layers, figures) and property is the name of a simple property of that container.

CGCBase Class

CGCBase Class	Description
addListener (see page 75)	Adds a listener to the internal list of listeners, if it is not already there.
beginUpdate (see page 75)	Increases the update count by 1 to stop any recursive update until (@see endUpdate (see page 76)()) was called.
canvas (see page 75)	This is canvas, a member of class CGCBase.
change (see page 75)	Triggers the onCange event of all registered listeners to notfied them about a particular change.
classIs (see page 76)	Determines if this class is of a specific type by comparing its class name to the given name.
className (see page 76)	This is className, a member of class CGCBase.
destroying (see page 76)	This is destroying, a member of class CGCBase.
endUpdate (see page 76)	The counterpart to (@see beginUpdate (see page 75)). It releases one update lock and also the global lock if the count drops to 0.
error (see page 76)	Triggers the onError event of all registered listeners to notfied them about an error.
property (see page 76)	This is property, a member of class CGCBase.
release (see page 77)	This is release, a member of class CGCBase.
removeListener (see page 77)	
setDestroying (see page 77)	Helper to set destroying (see page 76) state explicitely.
updating (see page 77)	This is updating, a member of class CGCBase.

Friends

Friend	Description
class CSVGParser (see page 88)	This is friend friend class CSVGParser.

CGCBase Class

CGCBase Class	Description
class CGenericCanvas (see page 77)	This is friend friend class CGenericCanvas.

Data Members

CGCBase Class

CGCBase Class	Description
_className (see page 74)	Used to determine the actual class.

Legend

	Method
	virtual
	abstract
	protected
	Data Member

1.2.24.1 Constructors

1.2.24.1.1 CGCStyle::CGCStyle Constructor

```
CGCStyle(CGCModel* model, wstring name);
```

Remarks

CGCStyle

1.2.24.2 Destructors

1.2.24.2.1 CGCStyle::~CGCStyle Destructor

```
virtual ~CGCStyle(void);
```

1.2.24.3 Methods

1.2.24.3.1 CGCStyle::boundingBox Method

```
TBoundingBox* boundingBox(void);
```

Remarks

This is boundingBox, a member of class CGCStyle.

1.2.24.3.2 CGCStyle::displayList Method

```
GLuint displayList(void);
```

Remarks

This is displayList, a member of class CGCStyle.

1.2.24.3.3 property

1.2.24.3.3.1 CGCStyle::property Method (const char*, unsigned int)

```
virtual TGCVariant __cdecl property(const char* name, unsigned int index);
```

Parameters

Parameters	Description
`const char* name`	The name of the property to return.
`unsigned int index`	The index of the sub property to return if it is located in a list.

Returns

A description of the property value and, if the property is simple, the actual value.

Remarks

Retrieves the value of the property given by path. The path syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name of a container class (e.g. layers, figures) and property is the name of a simple property of that container.

1.2.24.3.3.2 CGCStyle::property Method (const char*, unsigned int, const TGCVariant&)

```
virtual void __cdecl property(const char* name, unsigned int index, const TGCVariant&
value);
```

Parameters

Parameters	Description
`const char* name`	The name of the property.
`unsigned int index`	The index of the sub property to return if it is located in a list.
`const TGCVariant& value`	The new value of the property. Automatic conversion is performed where possible.

Remarks

Sets the value of the given property, which must be a simple property.

1.2.24.4 Friends

1.2.24.4.1 friend class CSVGParser Friend

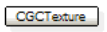
```
friend class CSVGParser;
```

Remarks

This is friend friend class CSVGParser.

1.2.25 CGCTexture Class

Class Hierarchy




```
class CGCTexture;
```

File

myx_gc_texture.h (see page 230)

Remarks

CGCTexture encapsulates a png image used to texture a figure in Generic Canvas. It loads the image data and manages it as well as the OpenGL properties for it.

Notes

A texture can be shared amongst several figures including its properties (e.g. minification filter).

Constructors

Constructor	Description
CGCTexture (see page 89)	CGCTexture

Destructors

Destructor	Description
~CGCTexture (see page 89)	

Members

Constructors

Constructor	Description
CGCTexture (see page 89)	CGCTexture

Destructors

Destructor	Description
~CGCTexture (see page 89)	

Methods

Method	Description
ActivateTexture (see page 89)	Activates this texture in OpenGL so all following vertex definitions are textured using this texture. If the texture has not been loaded yet it will be done now. Additionally, texture mode is enabled in OpenGL.
DeactivateTexture (see page 89)	Deactivates this texture and the texture mode in OpenGL.
LoadTexture (see page 89)	Delay loads texture data. Called from ActivateTexture (see page 89), that is, when the texture is used the first time. All level-of-detail texture data is loaded here and uploaded to OpenGL depending on the number of images given and what is set for the minification filter.

LoadTextureImage (see page 90)	Loads the image data referenced by name and returns it. If the either size of the image is not a power of 2 then the image is scaled up so that it becomes this size.

Legend

	Method
	protected

1.2.25.1 Constructors

1.2.25.1.1 CGCTexture::CGCTexture Constructor

```
CGCTexture(CTextureManager* Controller, const TLODList& LODData, const string& ID, GLenum WrapModeS, GLenum WrapModeT, GLenum MinFilter, GLenum MagFilter, int Dimensions, GLenum TextureMode);
```

Remarks

CGCTexture

1.2.25.2 Destructors

1.2.25.2.1 CGCTexture::~CGCTexture Destructor

```
~CGCTexture(void);
```

1.2.25.3 Methods

1.2.25.3.1 CGCTexture::ActivateTexture Method

```
void ActivateTexture(void);
```

Remarks

Activates this texture in OpenGL so all following vertex definitions are textured using this texture. If the texture has not been loaded yet it will be done now. Additionally, texture mode is enabled in OpenGL.

1.2.25.3.2 CGCTexture::DeactivateTexture Method

```
void DeactivateTexture(void);
```

Remarks

Deactivates this texture and the texture mode in OpenGL.

1.2.25.3.3 CGCTexture::LoadTexture Method

```
void LoadTexture(void);
```

Remarks

Delay loads texture data. Called from ActivateTexture (see page 89), that is, when the texture is used the first time. All level-of-detail texture data is loaded here and uploaded to OpenGL depending on the number of images given and what is set for the minification filter.

1.2.25.3.4 CGCTexture::LoadTextureImage Method

```
TImage* LoadTextureImage(const string& name, unsigned char*& Buffer);
```

Parameters

Parameters	Description
`const string& name`	The name of the file to load.
`unsigned char*& Buffer`	A variable that gets either the the address of the actual image data (TImage (see page 188)->Data) or a new memory reference if the image must be scaled. The caller is responsible to free this buffer if it differs from TImage (see page 188)->Data.

Returns

An image structure containing the actual image data. The caller is responsible to free this stucture via freeImage (see page 156) if it is non null. If the image could not be loaded then the result is NULL and the content of Buffer is not touched.

Remarks

Loads the image data referenced by name and returns it. If the either size of the image is not a power of 2 then the image is scaled up so that it becomes this size.

1.2.26 CGCView Class

Class Hierarchy


```
class CGCView : public CGCBase;
```

File

myx_gc_view.h (see page 234)

Remarks

A view implements an association between a set of layers and their visual representation on screen. Views can have individual zoom and offset values, viewports and other properties. There can always be only one active view. Views are managed by the canvas (see page 75) class.

Constructors

Constructor	Description
CGCView (see page 92)	CGCView

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CGCView (see page 93)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Members

Constructors

Constructor	Description
CGCView (see page 92)	CGCView

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CGCView (see page 93)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Methods

Method	Description
activate (see page 93)	Used to set up things that need only to be done once if a view becomes active.
addLayer (see page 93)	Adds a layer to the internal list of layers that belong to this view (only if not yet there).
applyTransformations (see page 93)	Sets up the current projection and modelview matrices.
clearContent (see page 93)	Removes the current content of this view.
clearSelection (see page 93)	Removes all currently selected figure instances from the selection set.
color (see page 94)	Sets the new background color of this view
contains (see page 94)	Tells the caller whether this view contains a reference to the given layer.
createConnectionInstance (see page 94)	Proxy function for the connection layer. Read there for a description.
getFeedbackInfo (see page 94)	Determines what action could be executed at the given position. Considered are any feedback state (selection, resize etc.) from the feedback layer as well as actions defined in a figure element. The returned info is usually used to set an indicator (e.g. the mouse pointer) to a certain state to reflect what is possible at that point.
getHitTestInfoAt (see page 95)	Takes the given coordinates and tries to find figure instances that were rendered at this position. Positions are given in view space.
grid (see page 95)	This is grid, a member of class CGCView.
handleMouseDown (see page 95)	Main handler routine for mouse button down handling.
handleMouseInput (see page 95)	Called by the viewer to let the current view handle user input with the mouse.
handleMouseMove (see page 96)	Main handler routine for mouse button move handling.
handleMouseUp (see page 96)	Main handler routine for mouse button up handling.
jitter (see page 96)	Jittering the viewport (see page 99) a little bit sometimes improves display quality (e.g. for thin lines). This function sets this value for this view.
offsetX (see page 97)	Sets the horizontal offset of the view. The offset is value about which the content of the view is moved.
offsetY (see page 97)	Sets the new vertical offset.
property (see page 98)	Retrieves the value of the property given by path. The path syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name of a container class (e.g. layers, figures) and property is the name of a simple property of that container.
removeLayer (see page 98)	Removes the given layer from the list of layers that comprise this view.
render (see page 99)	This is the main paint routine. It is called by the canvas (see page 75) if this view is the current view.
setupPaper (see page 99)	Used to initialize the paper layer of this view.
showSelection (see page 99)	Hides or shows the current selection.
viewport (see page 99)	Sets the new viewport for this view.
windowToView (see page 100)	Converts the given window (viewer) coordinates into local (view) space. note This function resets the current projection and view matrices.
zoomX (see page 100)	Sets a new horizontal zoom factor.
zoomY (see page 100)	Sets the new vertical zoom factor.

CGCBase Class

CGCBase Class	Description
addListener (see page 75)	Adds a listener to the internal list of listeners, if it is not already there.
beginUpdate (see page 75)	Increases the update count by 1 to stop any recursive update until (@see endUpdate (see page 76)()) was called.
canvas (see page 75)	This is canvas, a member of class CGCBase.
change (see page 75)	Triggers the onCange event of all registered listeners to notfied them about a particular change.
classIs (see page 76)	Determines if this class is of a specific type by comparing its class name to the given name.
className (see page 76)	This is className, a member of class CGCBase.
destroying (see page 76)	This is destroying, a member of class CGCBase.
endUpdate (see page 76)	The counterpart to (@see beginUpdate (see page 75)). It releases one update lock and also the global lock if the count drops to 0.
error (see page 76)	Triggers the onError event of all registered listeners to notfied them about an error.
property (see page 76)	This is property, a member of class CGCBase.
release (see page 77)	This is release, a member of class CGCBase.
removeListener (see page 77)	
setDestroying (see page 77)	Helper to set destroying (see page 76) state explicitely.
updating (see page 77)	This is updating, a member of class CGCBase.

Friends

Friend	Description
class CGenericCanvas (see page 101)	This is friend friend class CGenericCanvas.

CGCBase Class

CGCBase Class	Description
class CGenericCanvas (see page 77)	This is friend friend class CGenericCanvas.

Data Members

CGCBase Class

CGCBase Class	Description
_className (see page 74)	Used to determine the actual class.

Legend

	Method
	virtual
	protected
	abstract
	Data Member

1.2.26.1 Constructors

1.2.26.1.1 CGCView::CGCView Constructor

```
CGCView(CGenericCanvas* canvas, string name);
```

Remarks

CGCView

1.2.26.2 Destructors

1.2.26.2.1 CGCView::~CGCView Destructor

```
virtual ~CGCView(void);
```

1.2.26.3 Methods

1.2.26.3.1 CGCView::activate Method

```
void activate(void);
```

Remarks

Used to set up things that need only to be done once if a view becomes active.

1.2.26.3.2 CGCView::addLayer Method

```
virtual void __cdecl addLayer(CLayer* layer);
```

Parameters

Parameters	Description
`CLayer* layer`	The layer to add.

Remarks

Adds a layer to the internal list of layers that belong to this view (only if not yet there).

1.2.26.3.3 CGCView::applyTransformations Method

```
void applyTransformations(bool DoProjection);
```

Parameters

Parameters	Description
`doProjection`	A flag telling if also the project matrix should be set. Useful to avoid unnecessary applications.

Remarks

Sets up the current projection and modelview matrices.

1.2.26.3.4 CGCView::clearContent Method

```
virtual void __cdecl clearContent(bool removeLayers);
```

Parameters

Parameters	Description
`bool removeLayers`	If true then all curently defined (normal) layers are removed from this view (but not the canvas (see page 75)) in addition to being cleared.

Remarks

Removes the current content of this view.

1.2.26.3.5 CGCView::clearSelection Method

```
virtual void __cdecl clearSelection(void);
```

Remarks

Removes all currently selected figure instances from the selection set.

1.2.26.3.6 color

1.2.26.3.6.1 CGCView::color Method (GLfloat*)

```
virtual void __cdecl color(GLfloat* newColor);
```

Parameters

Parameters	Description
GLfloat* newColor	The new color to use.

Remarks

Sets the new background color of this view

1.2.26.3.6.2 CGCView::color Method (float, float, float, float)

```
virtual void __cdecl color(float red, float green, float blue, float alpha);
```

Parameters

Parameters	Description
float red	The red color component.
float green	The green color component.
float blue	The blue color component.
float alpha	The transparency component.

Remarks

Sets the new background color of this view.

1.2.26.3.7 CGCView::contains Method

```
virtual bool __cdecl contains(CLayer* layer);
```

Parameters

Parameters	Description
CLayer* layer	The layer to look for.
True	if the layer is referenced in this view otherwise false.

Remarks

Tells the caller whether this view contains a reference to the given layer.

1.2.26.3.8 CGCView::createConnectionInstance Method

```
virtual CConnectionInstance* __cdecl createConnectionInstance(CConnection* connection,
CFigureInstance* endPoint1, CFigureInstance* endPoint2);
```

Remarks

Proxy function for the connection layer. Read there for a description.

1.2.26.3.9 CGCView::getFeedbackInfo Method

```
virtual TFeedbackInfo __cdecl getFeedbackInfo(int windowX, int windowY);
```

Parameters

Parameters	Description
int windowX	The horizontal target position in window coordinates.
int windowY	The vertical target position in window coordinate.

Returns

A flag indicating the possible action state.

Remarks

Determines what action could be executed at the given position. Considered are any feedback state (selection, resize etc.) from the feedback layer as well as actions defined in a figure element. The returned info is usually used to set an indicator (e.g. the mouse pointer) to a certain state to reflect what is possible at that point.

1.2.26.3.10 CGCView::getHitTestInfoAt Method

```
virtual CHitResults* __cdecl getHitTestInfoAt(TVertex point, bool singleHit);
```

Parameters

Parameters	Description
`TVertex point`	The point to check given in view space. If necessary convert window coordinates first by using windowToView (see page 100).
`bool singleHit`	If true then search for hits is stopped after the first one was found.

Returns

A hit result class is returned regardless of the actual number of hits. It must be freed by the caller.

Remarks

Takes the given coordinates and tries to find figure instances that were rendered at this position. Positions are given in view space.

1.2.26.3.11 CGCView::grid Method

```
virtual CGridLayer* __cdecl grid(void);
```

Remarks

This is grid, a member of class CGCView.

1.2.26.3.12 CGCView::handleMouseDown Method

```
bool handleMouseDown(TMouseButton button, int modifiers, int windowX, int windowY, TVertex&
viewCoords);
```

Parameters

Parameters	Description
`TMouseButton button`	Which button has been pressed (left, middle, right).
`int modifiers`	Special flags that control the processing.
`int windowX`	Horizontal mouse coordinate in window space.
`int windowY`	Vertical mouse coordinate in window space.
`TVertex& viewCoords`	Mouse coordinates converted to view space.

Returns

True if the input was handled, otherwise false.

Remarks

Main handler routine for mouse button down handling.

1.2.26.3.13 CGCView::handleMouseInput Method

```
virtual bool __cdecl handleMouseInput(TMouseEvent event, TMouseButton button, int
modifiers, int x, int y);
```

Parameters

Parameters	Description
`TMouseEvent event`	The actual event (e.g. mouse down or up).
`TMouseButton button`	Specifies the mouse button for which this event was triggered.
`int modifiers`	Any combination of TModifier. Specifies further how to handle the input.
`int x`	The horizontal window coordinate of the mouse pointer.
`int y`	The vertical window coordinate of the mouse pointer.

Returns

True if the mouse input was handled in some way, otherwise false.

Remarks

Called by the viewer to let the current view handle user input with the mouse.

1.2.26.3.14 CGCView::handleMouseMove Method

```
bool handleMouseMove(int modifiers, int windowX, int windowY, TVertex& viewCoords);
```

Parameters

Parameters	Description
`int modifiers`	Special flags that control the processing.
`int windowX`	Horizontal mouse coordinate in window space.
`int windowY`	Vertical mouse coordinate in window space.
`TVertex& viewCoords`	Mouse coordinates converted to view space.

Returns

True if the input was handled, otherwise false.

Remarks

Main handler routine for mouse button move handling.

1.2.26.3.15 CGCView::handleMouseUp Method

```
bool handleMouseUp(TMouseButton button, int modifiers, int windowX, int windowY, TVertex&
viewCoords);
```

Parameters

Parameters	Description
`TMouseButton button`	Which button has been released (left, middle, right).
`int modifiers`	Special flags that control the processing.
`int windowX`	Horizontal mouse coordinate in window space.
`int windowY`	Vertical mouse coordinate in window space.
`TVertex& viewCoords`	Mouse coordinates converted to view space.

Returns

True if the input was handled, otherwise false.

Remarks

Main handler routine for mouse button up handling.

1.2.26.3.16 jitter

1.2.26.3.16.1 CGCView::jitter Method (float)

```
virtual void __cdecl jitter(float value);
```

Parameters

Parameters	Description
`float value`	The new jitter value.

Remarks

Jittering the viewport (see page 99) a little bit sometimes improves display quality (e.g. for thin lines). This function sets this value for this view.

1.2.26.3.16.2 CGCView::jitter Method (void)

```
virtual float __cdecl jitter(void);
```

Returns

The current jitter value.

Remarks

Returns the current jitter value.

1.2.26.3.17 offsetX

1.2.26.3.17.1 CGCView::offsetX Method (float)

```
virtual void __cdecl offsetX(float value);
```

Parameters

Parameters	Description
`float value`	The new horizontal offset.

Remarks

Sets the horizontal offset of the view. The offset is value about which the content of the view is moved.

1.2.26.3.17.2 CGCView::offsetX Method (void)

```
virtual float __cdecl offsetX(void);
```

Returns

The current horizontal offset.

Remarks

Returns the current horizontal offset.

1.2.26.3.18 offsetY

1.2.26.3.18.1 CGCView::offsetY Method (float)

```
virtual void __cdecl offsetY(float value);
```

Parameters

Parameters	Description
`float value`	The new vertical offset.

Remarks

Sets the new vertical offset.

1.2.26.3.18.2 CGCView::offsetY Method (void)

```
virtual float __cdecl offsetY(void);
```

Returns

The curretn vertical offset.

Remarks

Returns the current vertical offset.

1.2.26.3.19 property

1.2.26.3.19.1 CGCView::property Method (const char*, unsigned int)

```
virtual TGCVariant __cdecl property(const char* name, unsigned int index);
```

Parameters

Parameters	Description
`const char* name`	The name of the property.
`unsigned int index`	If the property is a list then this parameter gives the index into that list.

Returns

A description of the property value and, if the property is simple, the actual value.

Remarks

Retrieves the value of the property given by path. The path syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name of a container class (e.g. layers, figures) and property is the name of a simple property of that container.

1.2.26.3.19.2 CGCView::property Method (const char*, unsigned int, const TGCVariant&)

```
virtual void __cdecl property(const char* name, unsigned int index, const TGCVariant&
value);
```

Parameters

Parameters	Description
`const char* name`	The name of the property.
`unsigned int index`	If the property is a list then this parameter gives the index into that list.
`const TGCVariant& value`	The new value of the property. Automatic conversion is performed where possible.

Remarks

Sets the value of the given property, which must be a simple property.

1.2.26.3.20 CGCView::removeLayer Method

```
virtual void __cdecl removeLayer(CLayer* layer);
```

Parameters

Parameters	Description
`CLayer* layer`	The layer to remove.

Remarks

Removes the given layer from the list of layers that comprise this view.

1.2.26.3.21 CGCView::render Method

```
void render(void);
```

Remarks

This is the main paint routine. It is called by the canvas (see page 75) if this view is the current view.

1.2.26.3.22 CGCView::setupPaper Method

```
virtual void __cdecl setupPaper(const char* style, float width, float height, const
TBoundingBox& usableBounds);
```

Parameters

Parameters	Description
`const char* style`	The name of the style to use for the paper figure.
`float width`	The virtual paper width.
`float height`	The virtual paper height.
`const TBoundingBox& usableBounds`	The coordinates within the given width and height rectangle that should be used for content.

Remarks

Used to initialize the paper layer of this view.

1.2.26.3.23 CGCView::showSelection Method

```
virtual void __cdecl showSelection(bool Visible);
```

Parameters

Parameters	Description
`visible`	If true the selection is shown, otherwise not.

Remarks

Hides or shows the current selection.

1.2.26.3.24 viewport

1.2.26.3.24.1 CGCView::viewport Method (const TGCViewport&)

```
virtual void __cdecl viewport(const TGCViewport& newViewport);
```

Parameters

Parameters	Description
`NewViewport`	The new viewport to be used.

Remarks

Sets the new viewport for this view.

1.2.26.3.24.2 CGCView::viewport Method (void)

```
virtual TGCViewport __cdecl viewport(void);
```

Remarks

This is viewport, a member of class CGCView.

1.2.26.3.25 CGCView::windowToView Method

```
virtual TVertex __cdecl windowToView(int x, int y);
```

Parameters

Parameters	Description
int x	Horizontal window coordinate in pixels.
int y	Vertical window coordinate in pixels.

Returns

A vertex containing the local coordiantes.

Remarks

Converts the given window (viewer) coordinates into local (view) space. note This function resets the current projection and view matrices.

1.2.26.3.26 zoomX

1.2.26.3.26.1 CGCView::zoomX Method (float)

```
virtual void __cdecl zoomX(float value);
```

Parameters

Parameters	Description
float value	The new zoom factor.

Remarks

Sets a new horizontal zoom factor.

1.2.26.3.26.2 CGCView::zoomX Method (void)

```
virtual float __cdecl zoomX(void);
```

Returns

The current horizontal zoom factor.

Remarks

Returns the current horizontal zoom factor.

1.2.26.3.27 zoomY

1.2.26.3.27.1 CGCView::zoomY Method (float)

```
virtual void __cdecl zoomY(float value);
```

Parameters

Parameters	Description
float value	The new zoom factor.

Remarks

Sets the new vertical zoom factor.

1.2.26.3.27.2 CGCView::zoomY Method (void)

```
virtual float __cdecl zoomY(void);
```

Returns

The current vertical zoom factor.

Remarks

Return the current vertical zomm factor.

1.2.26.4 Friends

1.2.26.4.1 friend class CGenericCanvas Friend

```
friend class CGenericCanvas;
```

Remarks

This is friend friend class CGenericCanvas.

1.2.27 CGenericCanvas Class

Class Hierarchy


```
class CGenericCanvas : public CGCBase;
```

File

myx_gc_canvas.h (see page 218)

Remarks

CGenericCanvas is the main class of the library and is the base for all further functionality (e.g. it creates and maintains the model). Instances are created via the exported CreateGenericCanvas (see page 153) function (if called from non C++ languages). CGenericCanvas serves as the controller in the model-view-controller pattern, which is used here and communicates with the viewer via callbacks. The viewer is platform specific and must be implemented individually. It is responsible to create a canvas (see page 75) controller class.

Related Topics

CreateGenericCanvas (see page 153)

Constructors

Constructor	Description
CGenericCanvas (see page 103)	CGenericCanvas

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CGenericCanvas (see page 104)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Members

Constructors

Constructor	Description
CGenericCanvas (see page 103)	CGenericCanvas

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CGenericCanvas (see page 104)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Methods

Method	Description
addLayer (see page 104)	
addLayoutsFromFile (see page 104)	Reads the layout info stored in the given (XML) file and creates figure templates. Existing templates remain in where they are but are replaced if a new definition with an existing name is found.
addStylesFromFile (see page 104)	Reads the style template info stored in the given (XML) file and creates style templates (OpenGL display lists). Existing templates remain where they are but are replaced if a new definition with an existing name is found.
checkError (see page 104)	Checks if there is an OpenGL error (see page 76) registered and triggers the error (see page 76) method if so.
clearBuffers (see page 105)	This is clearBuffers, a member of class CGenericCanvas.
clearContent (see page 105)	Removes all GC content.
clearLayouts (see page 105)	Removes all layout info.
clearStyles (see page 105)	Causes the model to clear its style definitions.
createConnection (see page 105)	Interface function for the model's createConnection function. Read there for details.
createFigure (see page 105)	
createLayer (see page 105)	Creates a new layer with the given name and returns it to the caller. The new layer is added to this canvas (see page 75).
createView (see page 106)	Creates a new view in this canvas (see page 75), adds it to the internal list and returns it to the caller. If there is currently no current view set then the new view becomes the current one.
currentView (see page 106)	Sets the currently active view.
getFigureInstanceEnumerator (see page 106)	Creates and returns a new figure instance enumerator instance.
getModel (see page 106)	For special use only. Speaking MVC pattern, the viewer should not access the model!
layerByName (see page 107)	Returns the first layer with the given name from the layers collection.
lock (see page 107)	Aquires the canvas (see page 75) lock used to synchronize threads.
property (see page 107)	Retrieves the value of the property given by path. The path syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name of a container class (e.g. layers, figures) and property is the name of a simple property of that container.
refresh (see page 108)	Invalidates the viewer, so it does a repaint of the canvas (see page 75).
removeLayer (see page 108)	Removes the given layer from the internal layer list. The layer itself will not be destroyed, just removed.
removeView (see page 108)	Removes the given view from the internal list.
render (see page 108)	This is the main paint routine. It must be called by the viewer holding the reference to this canvas (see page 75) (e.g. when a window must be redrawn).
unlock (see page 108)	
viewByName (see page 108)	Returns the first view with the given name from the views collection.

CGCBase Class

CGCBase Class	Description
addListener (see page 75)	Adds a listener to the internal list of listeners, if it is not already there.
beginUpdate (see page 75)	Increases the update count by 1 to stop any recursive update until (@see endUpdate (see page 76)()) was called.
canvas (see page 75)	This is canvas, a member of class CGCBase.
change (see page 75)	Triggers the onCange event of all registered listeners to notfied them about a particular change.
classIs (see page 76)	Determines if this class is of a specific type by comparing its class name to the given name.
className (see page 76)	This is className, a member of class CGCBase.
destroying (see page 76)	This is destroying, a member of class CGCBase.
endUpdate (see page 76)	The counterpart to (@see beginUpdate (see page 75)). It releases one update lock and also the global lock if the count drops to 0.
error (see page 76)	Triggers the onError event of all registered listeners to notfied them about an error.
property (see page 76)	This is property, a member of class CGCBase.
release (see page 77)	This is release, a member of class CGCBase.
removeListener (see page 77)	
setDestroying (see page 77)	Helper to set destroying (see page 76) state explicitely.
updating (see page 77)	This is updating, a member of class CGCBase.

Friends

Friend	Description
class CFeedbackLayer (see page 109)	This is friend friend class CFeedbackLayer.
class CFigureInstanceEnumerator (see page 109)	This is friend friend class CFigureInstanceEnumerator.
class CGCBase (see page 109)	This is friend friend class CGCBase.
class CGCModel (see page 109)	This is friend friend class CGCModel.

CGCBase Class

CGCBase Class	Description
class CGenericCanvas (see page 77)	This is friend friend class CGenericCanvas.

Data Members

CGCBase Class

CGCBase Class	Description
_className (see page 74)	Used to determine the actual class.

Legend

	Method
	virtual
	protected
	abstract
	Data Member

1.2.27.1 Constructors

1.2.27.1.1 CGenericCanvas::CGenericCanvas Constructor

```
CGenericCanvas(GCContext Context, wstring name);
```

Remarks

CGenericCanvas

1.2.27.2 Destructors

1.2.27.2.1 CGenericCanvas::~CGenericCanvas Destructor

```
virtual ~CGenericCanvas(void);
```

1.2.27.3 Methods

1.2.27.3.1 CGenericCanvas::addLayer Method

```
virtual void __cdecl addLayer(CLayer* Layer);
```

Parameters

Parameters	Description
layer	The layer to add.

Javadoc Summary

Adds the given layer at the end of the layer list. The new layer becomes the top layer in the view then.

1.2.27.3.2 CGenericCanvas::addLayoutsFromFile Method

```
virtual TGCError __cdecl addLayoutsFromFile(const char* FileName);
```

Parameters

Parameters	Description
filename	The name of the file to load.

Returns

Returns GC_NO_ERROR if everything was ok, otherwise an error (see page 76) code.

Remarks

Reads the layout info stored in the given (XML) file and creates figure templates. Existing templates remain in where they are but are replaced if a new definition with an existing name is found.

1.2.27.3.3 CGenericCanvas::addStylesFromFile Method

```
virtual TGCError __cdecl addStylesFromFile(const char* FileName);
```

Parameters

Parameters	Description
filename	The name of the file to load.

Returns

Returns GC_NO_ERROR if everything was ok, otherwise an error (see page 76) code.

Remarks

Reads the style template info stored in the given (XML) file and creates style templates (OpenGL display lists). Existing templates remain where they are but are replaced if a new definition with an existing name is found.

1.2.27.3.4 CGenericCanvas::checkError Method

```
virtual void __cdecl checkError(void);
```

Remarks

Checks if there is an OpenGL error (see page 76) registered and triggers the error (see page 76) method if so.

1.2.27.3.5 CGenericCanvas::clearBuffers Method

```
void clearBuffers();
```

Remarks

This is clearBuffers, a member of class CGenericCanvas.

1.2.27.3.6 CGenericCanvas::clearContent Method

```
virtual void __cdecl clearContent(void);
```

Remarks

Removes all GC content.

1.2.27.3.7 CGenericCanvas::clearLayouts Method

```
virtual void __cdecl clearLayouts(void);
```

Remarks

Removes all layout info.

1.2.27.3.8 CGenericCanvas::clearStyles Method

```
virtual void __cdecl clearStyles(void);
```

Remarks

Causes the model to clear its style definitions.

1.2.27.3.9 CGenericCanvas::createConnection Method

```
virtual CConnection* __cdecl createConnection(CFigure* endPoint1, CFigure* endPoint2);
```

Remarks

Interface function for the model's createConnection function. Read there for details.

1.2.27.3.10 CGenericCanvas::createFigure Method

```
virtual CFigure* __cdecl createFigure(const char* type, const char* class_);
```

1.2.27.3.11 CGenericCanvas::createLayer Method

```
virtual CLayer* __cdecl createLayer(const char* name, bool AddToCurrentView);
```

Parameters

Parameters	Description
`const char* name`	The layer identification, encoded in UTF-8.
`bool AddToCurrentView`	If true then the new layer will be added to the current view (if there is any).

Returns

The new layer.

Remarks

Creates a new layer with the given name and returns it to the caller. The new layer is added to this canvas (see page 75).

1.2.27.3.12 CGenericCanvas::createView Method

```
virtual CGCView* __cdecl createView(const char* name);
```

Parameters

Parameters	Description
`const char* name`	The identification of the view (encoded in UTF-8). Should be unique.

Returns

The new view.

Remarks

Creates a new view in this canvas (see page 75), adds it to the internal list and returns it to the caller. If there is currently no current view set then the new view becomes the current one.

1.2.27.3.13 currentView

1.2.27.3.13.1 CGenericCanvas::currentView Method (CGCView*)

```
virtual void __cdecl currentView(CGCView* View);
```

Parameters

Parameters	Description
`CGCView* View`	The new view to activate.

Remarks

Sets the currently active view.

1.2.27.3.13.2 CGenericCanvas::currentView Method (void)

```
virtual CGCView* __cdecl currentView(void);
```

Returns

The currently active view.

Remarks

Returns the currently active view.

1.2.27.3.14 CGenericCanvas::getFigureInstanceEnumerator Method

```
virtual CFigureInstanceEnumerator* __cdecl getFigureInstanceEnumerator(void);
```

Returns

The new enumerator.

Remarks

Creates and returns a new figure instance enumerator instance.

1.2.27.3.15 CGenericCanvas::getModel Method

```
CGCModel* getModel(void);
```

Remarks

For special use only. Speaking MVC pattern, the viewer should not access the model!

1.2.27.3.16 CGenericCanvas::layerByName Method

```
virtual CLayer* __cdecl layerByName(const char* name);
```

Parameters

Parameters	Description
const char* name	The name of the layer to return.

Returns

The first found layer with the given name or NULL if no layer could be found.

Remarks

Returns the first layer with the given name from the layers collection.

1.2.27.3.17 CGenericCanvas::lock Method

```
void lock(void);
```

Remarks

Aquires the canvas (see page 75) lock used to synchronize threads.

1.2.27.3.18 property

1.2.27.3.18.1 CGenericCanvas::property Method (const char*, unsigned int)

```
virtual TGCVariant __cdecl property(const char* name, unsigned int index);
```

Parameters

Parameters	Description
const char* name	The name of the property.
unsigned int index	If the property is a list then this is the index into that list.

Returns

A description of the property value and, if the property is simple, the actual value.

Remarks

Retrieves the value of the property given by path. The path syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name of a container class (e.g. layers, figures) and property is the name of a simple property of that container.

1.2.27.3.18.2 CGenericCanvas::property Method (const char*, unsigned int, const TGCVariant&)

```
virtual void __cdecl property(const char* name, unsigned int index, const TGCVariant&
value);
```

Parameters

Parameters	Description
const char* name	The name of the property.
unsigned int index	If the property is a list then this is the index into that list.
const TGCVariant& value	The new value of the property. Automatic conversion is performed where possible.

Remarks

Set the value of the given property, which must be a simple property.

1.2.27.3.19 CGenericCanvas::refresh Method

```
virtual void __cdecl refresh(void);
```

Remarks

Invalidates the viewer, so it does a repaint of the canvas (see page 75).

1.2.27.3.20 CGenericCanvas::removeLayer Method

```
virtual void __cdecl removeLayer(CLayer* Layer);
```

Parameters

Parameters	Description
layer	The layer to be removed.

Remarks

Removes the given layer from the internal layer list. The layer itself will not be destroyed, just removed.

1.2.27.3.21 CGenericCanvas::removeView Method

```
virtual void __cdecl removeView(CGCView* View);
```

Parameters

Parameters	Description
CGCView* View	The view to be removed.

Remarks

Removes the given view from the internal list.

1.2.27.3.22 CGenericCanvas::render Method

```
virtual void __cdecl render(void);
```

Remarks

This is the main paint routine. It must be called by the viewer holding the reference to this canvas (see page 75) (e.g. when a window must be redrawn).

1.2.27.3.23 CGenericCanvas::unlock Method

```
void unlock(void);
```

1.2.27.3.24 CGenericCanvas::viewByName Method

```
virtual CGCView* __cdecl viewByName(const char* name);
```

Parameters

Parameters	Description
const char* name	The name of the view to return.

Returns

The first found view with the given name or NULL if no view could be found.

Remarks

Returns the first view with the given name from the views collection.

1.2.27.4 Friends

1.2.27.4.1 friend class CFeedbackLayer Friend

```
friend class CFeedbackLayer;
```

Remarks

This is friend friend class CFeedbackLayer.

1.2.27.4.2 friend class CFigureInstanceEnumerator Friend

```
friend class CFigureInstanceEnumerator;
```

Remarks

This is friend friend class CFigureInstanceEnumerator.

1.2.27.4.3 friend class CGCBase Friend

```
friend class CGCBase;
```

Remarks

This is friend friend class CGCBase.

1.2.27.4.4 friend class CGCModel Friend

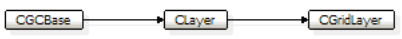
```
friend class CGCModel;
```

Remarks

This is friend friend class CGCModel.

1.2.28 CGridLayer Class

Class Hierarchy




```
class CGridLayer : public CLayer;
```

File

myx_gc_layer.h (see page 226)

Remarks

The grid layer is a special layer variant (see page 166) that renders itself as grid.

Constructors

Constructor	Description
CGridLayer (see page 112)	CGridLayer

CLayer Class

CLayer Class	Description
CLayer (see page 117)	CLayer

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CGridLayer (see page 112)	

CLayer Class

CLayer Class	Description
~CLayer (see page 117)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Members

Constructors

Constructor	Description
CGridLayer (see page 112)	CGridLayer

CLayer Class

CLayer Class	Description
CLayer (see page 117)	CLayer

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CGridLayer (see page 112)	

CLayer Class

CLayer Class	Description
~CLayer (see page 117)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Methods

Method	Description
bounds (see page 112)	Sets the usable area for the grid. The grid is only render (see page 120) within the given coordinates.
renderLayerContent (see page 113)	
validateLayerContent (see page 113)	Validates the grid display list.

CLayer Class

CLayer Class	Description
addInstance (see page 118)	Adds the given figure instance to the end of the instance list. If instance belongs to another layer currently it is removed from the other's instance list first.
applyTransformations (see page 118)	Applies the layer's transformations for rendering, feedback etc.
bringToFront (see page 118)	If the given figure instance is currently on this layer then it is moved to the last place in the list making it so the top most instance (they are rendered as stored in the instances array).
checkError (see page 118)	Triggers the error (see page 76) checking of the canvas (see page 75).
clear (see page 118)	This is clear, a member of class CLayer.
createInstance (see page 118)	Creates a new instance for the given figure and adds it to this layer.

enabled (see page 119)	Sets the layer's enabled state.
getHitTestInfoAt (see page 119)	Fills the hit results with all figure instances whose bounds contain the given coordinates.
makeDirty (see page 119)	Marks the display list for this layer as invalid, hence it will be recreated next time validate (see page 122) is called. If a list already exists then it is freed.
name (see page 119)	This is name, a member of class CLayer.
property (see page 120)	Retrieves the value of the property given by path. The path syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name (see page 119) of a container class (e.g. layers, figures) and property is the name (see page 119) of a simple property of that container.
removeInstance (see page 120)	Removes the given figure instance from the instance list if it is currently there. No error (see page 76) is raised if the instance does not belong to this layer.
render (see page 120)	Checks the validity of the figure display list and executes it.
renderFeedback (see page 121)	Helper method to determine the transformed vertices of the given figure instance. The layer applies its own transformations and only renders the figure instance.
renderLayerContent (see page 121)	Renders layer content that is not determined by figure instances. This method might be overridden by descendants.
scale (see page 121)	Scales the layer by the amount given in Factor. If Accumulative is true then the new scale factors are multiplied with the existing values. This version of scale uses an array of values in the parameter list.
sendToBack (see page 122)	If the given figure instance is currently on this layer then it is moved to the first place in the list making it so the bottom most instance (they are rendered as stored in the instances array).
translate (see page 122)	Moves the layer by the amount given in Tx, Ty and Tz. If Accumulative is true then the new translation factors are multiplied with the existing values. This version of translate uses an array for the values in the parameter list.
translateV (see page 122)	Moves the layer by the amount given in Factor. If Accumulative is true then the new translation factors are multiplied with the existing values. This version of translate (see page 122) uses an array for the values in the parameter list.
validate (see page 122)	Creates the display list of this figure (and all child figures) if necessary.
validateLayerContent (see page 122)	Prepares layer content that is not determined by figure instances. This method might be overridden by descendants.
visible (see page 123)	Sets the layer's visibility state.

CGCBase Class

CGCBase Class	Description
addListener (see page 75)	Adds a listener to the internal list of listeners, if it is not already there.
beginUpdate (see page 75)	Increases the update count by 1 to stop any recursive update until (@see endUpdate (see page 76)()) was called.
canvas (see page 75)	This is canvas, a member of class CGCBase.
change (see page 75)	Triggers the onCange event of all registered listeners to notfied them about a particular change.
classIs (see page 76)	Determines if this class is of a specific type by comparing its class name to the given name.
className (see page 76)	This is className, a member of class CGCBase.
destroying (see page 76)	This is destroying, a member of class CGCBase.
endUpdate (see page 76)	The counterpart to (@see beginUpdate (see page 75)). It releases one update lock and also the global lock if the count drops to 0.
error (see page 76)	Triggers the onError event of all registered listeners to notfied them about an error.
property (see page 76)	This is property, a member of class CGCBase.
release (see page 77)	This is release, a member of class CGCBase.
removeListener (see page 77)	
setDestroying (see page 77)	Helper to set destroying (see page 76) state explicitely.
updating (see page 77)	This is updating, a member of class CGCBase.

Friends

CLayer Class

CLayer Class	Description
class CFigureInstance (see page 123)	This is friend friend class CFigureInstance.
class CFigureInstanceEnumerator (see page 123)	This is friend friend class CFigureInstanceEnumerator.
class CInstanceListener (see page 123)	This is friend friend class CInstanceListener.

CGCBase Class

CGCBase Class	Description
class CGenericCanvas (see page 77)	This is friend friend class CGenericCanvas.

Data Members

CGCBase Class

CGCBase Class	Description
_className (see page 74)	Used to determine the actual class.

Legend

	Method
	virtual
	protected
	abstract
	Data Member

1.2.28.1 Constructors

1.2.28.1.1 CGridLayer::CGridLayer Constructor

```
CGridLayer(CGCView* View, CGenericCanvas* canvas);
```

Remarks

CGridLayer

1.2.28.2 Destructors

1.2.28.2.1 CGridLayer::~CGridLayer Destructor

```
virtual ~CGridLayer(void);
```

1.2.28.3 Methods

1.2.28.3.1 bounds

1.2.28.3.1.1 CGridLayer::bounds Method (TBoundingBox)

```
void bounds(TBoundingBox bounds);
```

Parameters

Parameters	Description
`TBoundingBox bounds`	The usable area, that is, the area the is used for the grid.

Remarks

Sets the usable area for the grid. The grid is only render (see page 120) within the given coordinates.

1.2.28.3.1.2 CGridLayer::bounds Method (void)

```
TBoundingBox bounds(void);
```

Remarks

This is bounds, a member of class CGridLayer.

1.2.28.3.2 CGridLayer::renderLayerContent Method

```
virtual void renderLayerContent(void);
```

1.2.28.3.3 CGridLayer::validateLayerContent Method

```
virtual void validateLayerContent(void);
```

Remarks

Validates the grid display list.

1.2.29 CHitResults Class

Class Hierarchy


```
class CHitResults;
```

File

myx_gc_view.h (see page 234)

Remarks

The CHitResult class is used to collect a number of figures that are located at a given point in the canvas.

note Never hold the given hit results record for a long time. The referenced figure instances may disappear at any time.

Constructors

Constructor	Description
CHitResults (see page 114)	CHitResults

Destructors

Destructor	Description
~CHitResults (see page 114)	

Members

Constructors

Constructor	Description
CHitResults (see page 114)	CHitResults

Destructors

Destructor	Description
~CHitResults (see page 114)	

Methods

Method	Description
addHit (see page 114)	
count (see page 114)	
hasNext (see page 114)	
next (see page 114)	
release (see page 114)	

reset (see page 114)	

Friends

Friend	Description
class CGCView (see page 115)	This is friend friend class CGCView.
class CLayer (see page 115)	This is friend friend class CLayer.

Legend

	Method
	virtual
	protected

1.2.29.1 Constructors

1.2.29.1.1 CHitResults::CHitResults Constructor

```
CHitResults(void);
```

Remarks

CHitResults

1.2.29.2 Destructors

1.2.29.2.1 CHitResults::~CHitResults Destructor

```
virtual ~CHitResults(void);
```

1.2.29.3 Methods

1.2.29.3.1 CHitResults::addHit Method

```
void addHit(CFigureInstance* Instance);
```

1.2.29.3.2 CHitResults::count Method

```
virtual int __cdecl count(void);
```

1.2.29.3.3 CHitResults::hasNext Method

```
virtual bool __cdecl hasNext(void);
```

1.2.29.3.4 CHitResults::next Method

```
virtual CFigureInstance* __cdecl next(void);
```

1.2.29.3.5 CHitResults::release Method

```
virtual void __cdecl release(void);
```

1.2.29.3.6 CHitResults::reset Method

```
virtual void __cdecl reset(void);
```

1.2.29.4 Friends

1.2.29.4.1 friend class CGCView Friend

```
friend class CGCView;
```

Remarks

This is friend friend class CGCView.

1.2.29.4.2 friend class CLayer Friend

```
friend class CLayer;
```

Remarks

This is friend friend class CLayer.

1.2.30 CLayer Class

Class Hierarchy


```
class CLayer : public CGCBase;
```

File

myx_gc_layer.h (see page 226)

Remarks

This is the base layer class, which is used by views to display their content. There are descendants for special things like feedback, grids and so on.

Constructors

Constructor	Description
CLayer (see page 117)	CLayer

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CLayer (see page 117)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Members

Constructors

Constructor	Description
CLayer (see page 117)	CLayer

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

Destructor	Description
~CLayer (see page 117)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Methods

Method	Description
addInstance (see page 118)	Adds the given figure instance to the end of the instance list. If instance belongs to another layer currently it is removed from the other's instance list first.
applyTransformations (see page 118)	Applies the layer's transformations for rendering, feedback etc.
bringToFront (see page 118)	If the given figure instance is currently on this layer then it is moved to the last place in the list making it so the top most instance (they are rendered as stored in the instances array).
checkError (see page 118)	Triggers the error (see page 76) checking of the canvas (see page 75).
clear (see page 118)	This is clear, a member of class CLayer.
createInstance (see page 118)	Creates a new instance for the given figure and adds it to this layer.
enabled (see page 119)	Sets the layer's enabled state.
getHitTestInfoAt (see page 119)	Fills the hit results with all figure instances whose bounds contain the given coordinates.
makeDirty (see page 119)	Marks the display list for this layer as invalid, hence it will be recreated next time validate (see page 122) is called. If a list already exists then it is freed.
name (see page 119)	This is name, a member of class CLayer.
property (see page 120)	Retrieves the value of the property given by path. The path syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name (see page 119) of a container class (e.g. layers, figures) and property is the name (see page 119) of a simple property of that container.
removeInstance (see page 120)	Removes the given figure instance from the instance list if it is currently there. No error (see page 76) is raised if the instance does not belong to this layer.
render (see page 120)	Checks the validity of the figure display list and executes it.
renderFeedback (see page 121)	Helper method to determine the transformed vertices of the given figure instance. The layer applies its own transformations and only renders the figure instance.
renderLayerContent (see page 121)	Renders layer content that is not determined by figure instances. This method might be overridden by descendants.
scale (see page 121)	Scales the layer by the amount given in Factor. If Accumulative is true then the new scale factors are multiplied with the existing values. This version of scale uses an array of values in the parameter list.
sendToBack (see page 122)	If the given figure instance is currently on this layer then it is moved to the first place in the list making it so the bottom most instance (they are rendered as stored in the instances array).
translate (see page 122)	Moves the layer by the amount given in Tx, Ty and Tz. If Accumulative is true then the new translation factors are multiplied with the existing values. This version of translate uses an array for the values in the parameter list.
translateV (see page 122)	Moves the layer by the amount given in Factor. If Accumulative is true then the new translation factors are multiplied with the existing values. This version of translate (see page 122) uses an array for the values in the parameter list.
validate (see page 122)	Creates the display list of this figure (and all child figures) if necessary.
validateLayerContent (see page 122)	Prepares layer content that is not determined by figure instances. This method might be overridden by descendants.
visible (see page 123)	Sets the layer's visibility state.

CGCBase Class

CGCBase Class	Description
addListener (see page 75)	Adds a listener to the internal list of listeners, if it is not already there.

beginUpdate (see page 75)	Increases the update count by 1 to stop any recursive update until (@see endUpdate (see page 76)()) was called.
canvas (see page 75)	This is canvas, a member of class CGCBase.
change (see page 75)	Triggers the onCange event of all registered listeners to notfied them about a particular change.
classIs (see page 76)	Determines if this class is of a specific type by comparing its class name to the given name.
className (see page 76)	This is className, a member of class CGCBase.
destroying (see page 76)	This is destroying, a member of class CGCBase.
endUpdate (see page 76)	The counterpart to (@see beginUpdate (see page 75)). It releases one update lock and also the global lock if the count drops to 0.
error (see page 76)	Triggers the onError event of all registered listeners to notfied them about an error.
property (see page 76)	This is property, a member of class CGCBase.
release (see page 77)	This is release, a member of class CGCBase.
removeListener (see page 77)	
setDestroying (see page 77)	Helper to set destroying (see page 76) state explicitely.
updating (see page 77)	This is updating, a member of class CGCBase.

Friends

Friend	Description
class CFigureInstance (see page 123)	This is friend friend class CFigureInstance.
class CFigureInstanceEnumerator (see page 123)	This is friend friend class CFigureInstanceEnumerator.
class CInstanceListener (see page 123)	This is friend friend class CInstanceListener.

CGCBase Class

CGCBase Class	Description
class CGenericCanvas (see page 77)	This is friend friend class CGenericCanvas.

Data Members

CGCBase Class

CGCBase Class	Description
_className (see page 74)	Used to determine the actual class.

Legend

	Method
	virtual
	protected
	abstract
	Data Member

1.2.30.1 Constructors

1.2.30.1.1 CLayer::CLayer Constructor

```
CLayer(string name, CGenericCanvas* canvas);
```

Remarks

CLayer

1.2.30.2 Destructors

1.2.30.2.1 CLayer::~CLayer Destructor

```
virtual ~CLayer(void);
```

1.2.30.3 Methods

1.2.30.3.1 CLayer::addInstance Method

```
virtual void __cdecl addInstance(CFigureInstance* instance);
```

Parameters

Parameters	Description
`CFigureInstance* instance`	The figure instance to add.

Remarks

Adds the given figure instance to the end of the instance list. If instance belongs to another layer currently it is removed from the other's instance list first.

1.2.30.3.2 CLayer::applyTransformations Method

```
void applyTransformations();
```

Remarks

Applies the layer's transformations for rendering, feedback etc.

1.2.30.3.3 CLayer::bringToFront Method

```
virtual void __cdecl bringToFront(CFigureInstance* instance);
```

Parameters

Parameters	Description
`CFigureInstance* instance`	The instance to bring to front.

Remarks

If the given figure instance is currently on this layer then it is moved to the last place in the list making it so the top most instance (they are rendered as stored in the instances array).

1.2.30.3.4 CLayer::checkError Method

```
void checkError(void);
```

Remarks

Triggers the error (see page 76) checking of the canvas (see page 75).

1.2.30.3.5 CLayer::clear Method

```
virtual void __cdecl clear();
```

Remarks

This is clear, a member of class CLayer.

1.2.30.3.6 CLayer::createInstance Method

```
virtual CFigureInstance* __cdecl createInstance(CFigure* figure);
```

Parameters

Parameters	Description
`Figure`	The figure for which the instance is to be created.

Returns

A new figure instance.

Remarks

Creates a new instance for the given figure and adds it to this layer.

1.2.30.3.7 enabled

1.2.30.3.7.1 CLayer::enabled Method (bool)

```
virtual void __cdecl enabled(bool isEnabled);
```

Parameters

Parameters	Description
IsEnabled	Set it to true if you want the layer to be visible (see page 123).

Remarks

Sets the layer's enabled state.

1.2.30.3.7.2 CLayer::enabled Method (void)

```
virtual bool __cdecl enabled(void);
```

Returns

The current enabled state.

Remarks

Returns the current enabled state.

1.2.30.3.8 CLayer::getHitTestInfoAt Method

```
void getHitTestInfoAt(CHitResults* hits, const float x, const float y, bool singleHit);
```

Parameters

Parameters	Description
const float x	The horizontal hit point coordinated given in view space.
const float y	The vertical coordinate.
bool singleHit	If true only one hit is returned.
Hits	[out] The hit collection that is updated.

Remarks

Fills the hit results with all figure instances whose bounds contain the given coordinates.

1.2.30.3.9 CLayer::makeDirty Method

```
void makeDirty(void);
```

Remarks

Marks the display list for this layer as invalid, hence it will be recreated next time validate (see page 122) is called. If a list already exists then it is freed.

1.2.30.3.10 CLayer::name Method

```
wstring name(void);
```

Remarks

This is name, a member of class CLayer.

1.2.30.3.11 property

1.2.30.3.11.1 CLayer::property Method (const char*, unsigned int)

```
virtual TGCVariant __cdecl property(const char* name, unsigned int index);
```

Parameters

Parameters	Description
const char* name	The name (see page 119) of the property to return.
unsigned int index	If the property is a list then this is the index into that list.

Returns

A description of the property value and, if the property is simple, the actual value.

Remarks

Retrieves the value of the property given by path. The path syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name (see page 119) of a container class (e.g. layers, figures) and property is the name (see page 119) of a simple property of that container.

1.2.30.3.11.2 CLayer::property Method (const char*, unsigned int, const TGCVariant&)

```
virtual void __cdecl property(const char* name, unsigned int index, const TGCVariant&
value);
```

Parameters

Parameters	Description
const char* name	The name (see page 119) of the property.
unsigned int index	If the property is a list then this is the index into that list.
Value	The new value of the property. Automatic conversion is performed where possible.

Remarks

Sets the value of the given property, which must be a simple property.

1.2.30.3.12 CLayer::removeInstance Method

```
virtual void __cdecl removeInstance(CFigureInstance* instance);
```

Parameters

Parameters	Description
CFigureInstance* instance	The instance to be removed.

Remarks

Removes the given figure instance from the instance list if it is currently there. No error (see page 76) is raised if the instance does not belong to this layer.

1.2.30.3.13 CLayer::render Method

```
virtual void __cdecl render(float currentZoom, TBoundingBox bounds);
```

Parameters

Parameters	Description
TBoundingBox bounds	The area currently visible (see page 123). No need to render anything outside that area.

CurrentZoom	The current zoom factor of the scene.

Remarks

Checks the validity of the figure display list and executes it.

1.2.30.3.14 CLayer::renderFeedback Method

```
void renderFeedback(CFigureInstance* instance);
```

Parameters

Parameters	Description
CFigureInstance* instance	The figure instance for which feedback data is requested.

Remarks

Helper method to determine the transformed vertices of the given figure instance. The layer applies its own transformations and only renders the figure instance.

1.2.30.3.15 CLayer::renderLayerContent Method

```
virtual void renderLayerContent(void);
```

Remarks

Renders layer content that is not determined by figure instances. This method might be overridden by descendants.

1.2.30.3.16 scale

1.2.30.3.16.1 CLayer::scale Method (const float Factor[3], bool)

```
virtual void __cdecl scale(const float Factor[3], bool accumulative = false);
```

Parameters

Parameters	Description
Factor	An array of 3 scale values, one for each axis.
Accumulative	If true then the given values are added to any existing values otherwiese they are used as given.

Remarks

Scales the layer by the amount given in Factor. If Accumulative is true then the new scale factors are multiplied with the existing values. This version of scale uses an array of values in the parameter list.

1.2.30.3.16.2 CLayer::scale Method (float, float, float, bool)

```
virtual void __cdecl scale(float Sx, float Sy, float Sz, bool accumulative = false);
```

Parameters

Parameters	Description
float Sx	scale factor for the x axis.
float Sy	scale factor for the y axis.
float Sz	scale factor for the z axis.
Accumulative	If true then the given values are added to any existing values otherwiese they are used as given.

Remarks

Scales the layer by the amount given in Factor. If Accumulative is true then the new scale factors are multiplied with the existing values. This version of scale uses single float values as parameters.

1.2.30.3.17 CLayer::sendToBack Method

```
virtual void __cdecl sendToBack(CFigureInstance* instance);
```

Parameters

Parameters	Description
CFigureInstance* instance	The instance to send to back.

Remarks

If the given figure instance is currently on this layer then it is moved to the first place in the list making it so the bottom most instance (they are rendered as stored in the instances array).

1.2.30.3.18 CLayer::translate Method

```
virtual void __cdecl translate(float Tx, float Ty, float Tz, bool accumulative = false);
```

Parameters

Parameters	Description
float Tx	scale (see page 121) factor for the x axis.
float Ty	scale (see page 121) factor for the y axis.
float Tz	scale (see page 121) factor for the z axis.
Accumulative	If true then the given values are added to any existing values otherwiese they are used as given.

Remarks

Moves the layer by the amount given in Tx, Ty and Tz. If Accumulative is true then the new translation factors are multiplied with the existing values. This version of translate uses an array for the values in the parameter list.

1.2.30.3.19 CLayer::translateV Method

```
virtual void __cdecl translateV(const float Factor[3], bool accumulative = false);
```

Parameters

Parameters	Description
Factor	An array of translation values, for each axis one.
Accumulative	If true then the given values are added to any existing values otherwiese they are used as given.

Remarks

Moves the layer by the amount given in Factor. If Accumulative is true then the new translation factors are multiplied with the existing values. This version of translate (see page 122) uses an array for the values in the parameter list.

1.2.30.3.20 CLayer::validate Method

```
void validate(void);
```

Remarks

Creates the display list of this figure (and all child figures) if necessary.

1.2.30.3.21 CLayer::validateLayerContent Method

```
virtual void validateLayerContent(void);
```

Remarks

Prepares layer content that is not determined by figure instances. This method might be overridden by descendants.

1.2.30.3.22 visible

1.2.30.3.22.1 CLayer::visible Method (bool)

```
virtual void __cdecl visible(bool isVisible);
```

Parameters

Parameters	Description
IsVisible	Set it to true if you want the layer to be visible.

Remarks

Sets the layer's visibility state.

1.2.30.3.22.2 CLayer::visible Method (void)

```
virtual bool __cdecl visible(void);
```

Returns

The current visibility state.

Remarks

Returns the visibility state.

1.2.30.4 Friends

1.2.30.4.1 friend class CFigureInstance Friend

```
friend class CFigureInstance;
```

Remarks

This is friend friend class CFigureInstance.

1.2.30.4.2 friend class CFigureInstanceEnumerator Friend

```
friend class CFigureInstanceEnumerator;
```

Remarks

This is friend friend class CFigureInstanceEnumerator.

1.2.30.4.3 friend class CInstanceListener Friend

```
friend class CInstanceListener;
```

Remarks

This is friend friend class CInstanceListener.

1.2.31 CLayouter Class

Class Hierarchy


```
class CLayouter;
```

File

myx_gc_layout.h (see page 227)

Remarks

Abstract base class for all layouter classes.

Constructors

Constructor	Description
CLayouter (see page 125)	CLayouter

Members

Data Members

Data Member	Description
FElement (see page 124)	The element we are layouting.
FIterator (see page 125)	The iterator used to go through the child list of the element to layout.
FX (see page 125)	This is FX, a member of class CLayouter.
FY (see page 125)	This is FY, a member of class CLayouter.

Constructors

Constructor	Description
CLayouter (see page 125)	CLayouter

Methods

Method	Description
V hasNext (see page 125)	Tells the caller whether there is still a next value available.
nextAction (see page 125)	Executes the doAction function of the current element in the layout order. For this to work the given coordinates must be transformed to local coordinates.
A nextBoundingBox (see page 126)	This is nextBoundingBox, a member of class CLayouter.
V renderNext (see page 126)	Renders the current child element and moves on to the next in the list.
reset (see page 126)	Resets layout computation to start over from origin.

Legend

	protected
	Data Member
	Method
V	virtual
A	abstract

1.2.31.1 Data Members

1.2.31.1.1 CLayouter::FElement Data Member

```
CFigureElement* FElement;
```

Remarks

The element we are layouting.

1.2.31.1.2 CLayouter::FIterator Data Member

```
CElementList::iterator FIterator;
```

Remarks

The iterator used to go through the child list of the element to layout.

1.2.31.1.3 CLayouter::FX Data Member

```
float FX;
```

Remarks

This is FX, a member of class CLayouter.

1.2.31.1.4 CLayouter::FY Data Member

```
float FY;
```

Remarks

This is FY, a member of class CLayouter.

1.2.31.2 Constructors

1.2.31.2.1 CLayouter::CLayouter Constructor

```
CLayouter(CFigureElement* Element);
```

Remarks

CLayouter

1.2.31.3 Methods

1.2.31.3.1 CLayouter::hasNext Method

```
virtual bool hasNext(void);
```

Returns

True, if there is a next value, otherwise false.

Remarks

Tells the caller whether there is still a next value available.

1.2.31.3.2 CLayouter::nextAction Method

```
TActionType nextAction(CFigureInstance* instance, const float x, const float y);
```

Parameters

Parameters	Description
`CFigureInstance* instance`	The figure instance owning the figure element, which is the owner of this layouter.
`const float x`	The x coordinate for the hit test expressed in the coordinate system of the parent element.
`const float y`	The y coordinate for the hit test expressed in the coordinate system of the parent element.

Returns

The last action executed in the current figure element.

Remarks

Executes the doAction function of the current element in the layout order. For this to work the given coordinates must be transformed to local coordinates.

1.2.31.3.3 CLayouter::nextBoundingBox Method

```
virtual void nextBoundingBox(TBoundingBox* BoundingBox) = 0;
```

Remarks

This is nextBoundingBox, a member of class CLayouter.

1.2.31.3.4 CLayouter::renderNext Method

```
virtual void renderNext(void);
```

Remarks

Renders the current child element and moves on to the next in the list.

1.2.31.3.5 CLayouter::reset Method

```
void reset(void);
```

Remarks

Resets layout computation to start over from origin.

1.2.32 CPaperLayer Class

Class Hierarchy


```
class CPaperLayer : public CLayer;
```

File

myx_gc_layer.h (see page 226)

Remarks

The paper layer is a normal layer but shows only one figure instace it creates implicitely given a certain figure. This layer does not take part in the usual input handling and is displayed as a ground layer on which all other layers render (see page 120) their stuff. This class is not exclusively managed by the view it belongs to.

Constructors

Constructor	Description
CPaperLayer (see page 129)	CPaperLayer

CLayer Class

CLayer Class	Description
CLayer (see page 117)	CLayer

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Destructors

CLayer Class

CLayer Class	Description
~CLayer (see page 117)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Members

Constructors

Constructor	Description
CPaperLayer (see page 129)	CPaperLayer

CLayer Class

CLayer Class	Description
CLayer (see page 117)	CLayer

CGCBase Class

CGCBase Class	Description
CGCBase (see page 75)	CGCBase

Methods

Method	Description
contentBounds (see page 129)	This is contentBounds, a member of class CPaperLayer.
paperDestroyed (see page 129)	
renderLayerContent (see page 129)	Simply renders the paper figure.
setup (see page 130)	Called by the owning view to set the paper layer up.

CLayer Class

CLayer Class	Description
addInstance (see page 118)	Adds the given figure instance to the end of the instance list. If instance belongs to another layer currently it is removed from the other's instance list first.
applyTransformations (see page 118)	Applies the layer's transformations for rendering, feedback etc.
bringToFront (see page 118)	If the given figure instance is currently on this layer then it is moved to the last place in the list making it so the top most instance (they are rendered as stored in the instances array).
checkError (see page 118)	Triggers the error (see page 76) checking of the canvas (see page 75).
clear (see page 118)	This is clear, a member of class CLayer.
createInstance (see page 118)	Creates a new instance for the given figure and adds it to this layer.
enabled (see page 119)	Sets the layer's enabled state.
getHitTestInfoAt (see page 119)	Fills the hit results with all figure instances whose bounds contain the given coordinates.
makeDirty (see page 119)	Marks the display list for this layer as invalid, hence it will be recreated next time validate (see page 122) is called. If a list already exists then it is freed.

name (see page 119)	This is name, a member of class CLayer.
property (see page 120)	Retrieves the value of the property given by path. The path syntax is must be something like (here expressed as regex) (container)*(property), where container is a slash and the name (see page 119) of a container class (e.g. layers, figures) and property is the name (see page 119) of a simple property of that container.
removeInstance (see page 120)	Removes the given figure instance from the instance list if it is currently there. No error (see page 76) is raised if the instance does not belong to this layer.
render (see page 120)	Checks the validity of the figure display list and executes it.
renderFeedback (see page 121)	Helper method to determine the transformed vertices of the given figure instance. The layer applies its own transformations and only renders the figure instance.
renderLayerContent (see page 121)	Renders layer content that is not determined by figure instances. This method might be overridden by descendants.
scale (see page 121)	Scales the layer by the amount given in Factor. If Accumulative is true then the new scale factors are multiplied with the existing values. This version of scale uses an array of values in the parameter list.
sendToBack (see page 122)	If the given figure instance is currently on this layer then it is moved to the first place in the list making it so the bottom most instance (they are rendered as stored in the instances array).
translate (see page 122)	Moves the layer by the amount given in Tx, Ty and Tz. If Accumulative is true then the new translation factors are multiplied with the existing values. This version of translate uses an array for the values in the parameter list.
translateV (see page 122)	Moves the layer by the amount given in Factor. If Accumulative is true then the new translation factors are multiplied with the existing values. This version of translate (see page 122) uses an array for the values in the parameter list.
validate (see page 122)	Creates the display list of this figure (and all child figures) if necessary.
validateLayerContent (see page 122)	Prepares layer content that is not determined by figure instances. This method might be overridden by descendants.
visible (see page 123)	Sets the layer's visibility state.

CGCBase Class

CGCBase Class	Description
addListener (see page 75)	Adds a listener to the internal list of listeners, if it is not already there.
beginUpdate (see page 75)	Increases the update count by 1 to stop any recursive update until (@see endUpdate (see page 76)()) was called.
canvas (see page 75)	This is canvas, a member of class CGCBase.
change (see page 75)	Triggers the onCange event of all registered listeners to notfied them about a particular change.
classIs (see page 76)	Determines if this class is of a specific type by comparing its class name to the given name.
className (see page 76)	This is className, a member of class CGCBase.
destroying (see page 76)	This is destroying, a member of class CGCBase.
endUpdate (see page 76)	The counterpart to (@see beginUpdate (see page 75)). It releases one update lock and also the global lock if the count drops to 0.
error (see page 76)	Triggers the onError event of all registered listeners to notfied them about an error.
property (see page 76)	This is property, a member of class CGCBase.
release (see page 77)	This is release, a member of class CGCBase.
removeListener (see page 77)	
setDestroying (see page 77)	Helper to set destroying (see page 76) state explicitely.
updating (see page 77)	This is updating, a member of class CGCBase.

Destructors

CLayer Class

CLayer Class	Description
~CLayer (see page 117)	

CGCBase Class

CGCBase Class	Description
~CGCBase (see page 75)	

Friends

CLayer Class

CLayer Class	Description
class CFigureInstance (see page 123)	This is friend friend class CFigureInstance.
class CFigureInstanceEnumerator (see page 123)	This is friend friend class CFigureInstanceEnumerator.
class CInstanceListener (see page 123)	This is friend friend class CInstanceListener.

CGCBase Class

CGCBase Class	Description
class CGenericCanvas (see page 77)	This is friend friend class CGenericCanvas.

Data Members

CGCBase Class

CGCBase Class	Description
_className (see page 74)	Used to determine the actual class.

Legend

	Method
	protected
V	virtual
A	abstract
	Data Member

1.2.32.1 Constructors

1.2.32.1.1 CPaperLayer::CPaperLayer Constructor

```
CPaperLayer(CGenericCanvas* canvas);
```

Remarks

CPaperLayer

1.2.32.2 Methods

1.2.32.2.1 CPaperLayer::contentBounds Method

```
TBoundingBox contentBounds(void);
```

Remarks

This is contentBounds, a member of class CPaperLayer.

1.2.32.2.2 CPaperLayer::paperDestroyed Method

```
void paperDestroyed(CFigureInstance* instance);
```

1.2.32.2.3 CPaperLayer::renderLayerContent Method

```
virtual void renderLayerContent(void);
```

Remarks

Simply renders the paper figure.

1.2.32.2.4 CPaperLayer::setup Method

```
void setup(const char* template_, float width, float height, const TBoundingBox&
usableBounds);
```

Parameters

Parameters	Description
`const char* template_`	The figure template to use as the paper. The layer implicitely creates a figure and a figure instance from that value. If there was already a figure (instance) used then its reference is removed but the objects are not destroyed. This parameter must be a UTF-8 encoded name (see page 119).
`float width`	The virtual paper width.
`float height`	The virtual paper height.
`const TBoundingBox& usableBounds`	The coordinates within the given width and height rectangle that should be used for content.

Remarks

Called by the owning view to set the paper layer up.

1.2.33 CRowLayouter Class

Class Hierarchy


```
class CRowLayouter : public CLayouter;
```

File

myx_gc_layout.h (see page 227)

Remarks

This is class CRowLayouter.

Constructors

Constructor	Description
CRowLayouter (see page 131)	This is CRowLayouter, a member of class CRowLayouter.

CLayouter Class

CLayouter Class	Description
CLayouter (see page 125)	CLayouter

Members

Constructors

Constructor	Description
CRowLayouter (see page 131)	This is CRowLayouter, a member of class CRowLayouter.

CLayouter Class

CLayouter Class	Description
CLayouter (see page 125)	CLayouter

Methods

Method	Description
nextBoundingBox (see page 131)	Returns the transformed bounding box of the next element.

CLayouter Class

CLayouter Class	Description
hasNext (see page 125)	Tells the caller whether there is still a next value available.
nextAction (see page 125)	Executes the doAction function of the current element in the layout order. For this to work the given coordinates must be transformed to local coordinates.
nextBoundingBox (see page 126)	This is nextBoundingBox, a member of class CLayouter.
renderNext (see page 126)	Renders the current child element and moves on to the next in the list.
reset (see page 126)	Resets layout computation to start over from origin.

Data Members

CLayouter Class

CLayouter Class	Description
FElement (see page 124)	The element we are layouting.
FIterator (see page 125)	The iterator used to go through the child list of the element to layout.
FX (see page 125)	This is FX, a member of class CLayouter.
FY (see page 125)	This is FY, a member of class CLayouter.

Legend

	Method
	virtual
	abstract
	protected
	Data Member

1.2.33.1 Constructors

1.2.33.1.1 CRowLayouter::CRowLayouter Constructor

```
CRowLayouter(CFigureElement* Element);
```

Remarks

This is CRowLayouter, a member of class CRowLayouter.

1.2.33.2 Methods

1.2.33.2.1 CRowLayouter::nextBoundingBox Method

```
virtual void nextBoundingBox(TBoundingBox* BoundingBox);
```

Parameters

Parameters	Description
`TBoundingBox* BoundingBox`	The bounding box to fill with the new values.

Remarks

Returns the transformed bounding box of the next element.

1.2.34 CStyleListener Class

Class Hierarchy


```
class CStyleListener : private CGCListener;
```

File

myx_gc_figure.h (see page 222)

Remarks

This is class CStyleListener.

Members

Data Members

Data Member	Description
template_ (see page 132)	This is template_, a member of class CStyleListener.

Methods

Method	Description
onChange (see page 133)	
onDestroy (see page 133)	
onError (see page 133)	

CGCListener Class

CGCListener Class	Description
onChange (see page 78)	This is onChange, a member of class CGCListener.
onDestroy (see page 78)	This is onDestroy, a member of class CGCListener.
onError (see page 78)	This is onError, a member of class CGCListener.

Friends

Friend	Description
class CFigureElementTemplate (see page 133)	This is friend friend class CFigureElementTemplate.

Legend

	protected
	Data Member
	Method
V	virtual
A	abstract

1.2.34.1 Data Members

1.2.34.1.1 CStyleListener::template_ Data Member

```
CFigureElementTemplate* template_;
```

Remarks

This is template_, a member of class CStyleListener.

1.2.34.2 Methods

1.2.34.2.1 CStyleListener::onChange Method

```
virtual void __cdecl onChange(CGCBase* sender, CGCBase* origin, TGCChangeReason reason);
```

1.2.34.2.2 CStyleListener::onDestroy Method

```
virtual void __cdecl onDestroy(CGCBase* sender);
```

1.2.34.2.3 CStyleListener::onError Method

```
virtual void __cdecl onError(CGCBase* sender, CGCBase* origin, const char* message);
```

1.2.34.3 Friends

1.2.34.3.1 friend class CFigureElementTemplate Friend

```
friend class CFigureElementTemplate;
```

Remarks

This is friend friend class CFigureElementTemplate.

1.2.35 CSVGParser Class

Class Hierarchy


```
class CSVGParser;
```

File

myx_gc_svgparser.h (see page 230)

Remarks

CSVGParser is the main svg parser class. It converts an element description into OpenGL calls.

note Not all possible subelements/attributes can be parsed by this class. If they are specified then they will be ignored. See Generic Canvas documentation for more details.

Constructors

Constructor	Description
CSVGParser (see page 134)	CSVGParser

Destructors

Destructor	Description
~CSVGParser (see page 134)	

Members

Constructors

Constructor	Description
CSVGParser (see page 134)	CSVGParser

Destructors

Destructor	Description
~CSVGParser (see page 134)	

Methods

Method	Description
convert (see page 135)	Parses the given svg xml description and converts it to OpenGL calls. This method can be called within an active OpenGL display list compilation (but not between glBegin/glEnd).
parseCircle (see page 135)	Parses the content of a circle definition.
parseDefinition (see page 135)	Parses the given style definition and creates a new style, which is then added to the given model.
parseElement (see page 135)	Recursively called function that parses the given svg xml element and converts it to OpenGL calls.
parseGroup (see page 136)	Collects all data in an element.
parseImage (see page 136)	Parses the content of an image definition.
parseLine (see page 136)	Parses the content of a line definition.
parsePolygon (see page 136)	Parses the content of a polygon definition.
parsePolyline (see page 137)	Parses the content of a polyline definition.
parseRectangle (see page 137)	Parses the content of a rectangle definition.
parseText (see page 137)	Takes a text node and gets all attributes for direct or dynamic rendering. This function is called recursively and can take either a or a node.
parseTransformation (see page 137)	Parsers the given string and interprets the content as one or more transformation of the form: translate(x, y, z) scale(x, y, z) etc.
renderVertices (see page 138)	Renders the given set of vertices.

Legend

	Method
	virtual
	protected

1.2.35.1 Constructors

1.2.35.1.1 CSVGParser::CSVGParser Constructor

```
CSVGParser(void);
```

Remarks

CSVGParser

1.2.35.2 Destructors

1.2.35.2.1 CSVGParser::~CSVGParser Destructor

```
virtual ~CSVGParser(void);
```

1.2.35.3 Methods

1.2.35.3.1 CSVGParser::convert Method

```
void convert(xmlNodePtr svg, GLuint DisplayList, CBoundingBoxComputer* boundingBox);
```

Parameters

Parameters	Description
xmlNodePtr svg	The svg top level element ().
GLuint DisplayList	The OpenGL display list to render into.
CBoundingBoxComputer* boundingBox	The bounding box computer to use to compute the overall bounding box of the element. note In order to render correctly the svg:svg element must have width and height attributes set. Particularly the height attribute is needed to convert from svg's top-down to OpenGL's bottom-up coordinate system. Without height attribute the element will be drawn head-down.

Remarks

Parses the given svg xml description and converts it to OpenGL calls. This method can be called within an active OpenGL display list compilation (but not between glBegin/glEnd).

1.2.35.3.2 CSVGParser::parseCircle Method

```
void parseCircle(xmlNodePtr svg, bool doFill, GLubyte* fillColor, bool doStroke, GLubyte*
strokeColor, float strokeWidth, CBoundingBoxComputer* boundingBox);
```

Remarks

Parses the content of a circle definition.

Related Topics

parseElement for a description of the parameters.

1.2.35.3.3 CSVGParser::parseDefinition Method

```
void parseDefinition(xmlNodePtr definition, CGCModel* model);
```

Parameters

Parameters	Description
xmlNodePtr definition	The definition to parse. model The model to which the new style is to be added.

Remarks

Parses the given style definition and creates a new style, which is then added to the given model.

1.2.35.3.4 CSVGParser::parseElement Method

```
GLuint parseElement(xmlNodePtr svg, bool doFill, GLubyte* fillColor, bool doStroke,
GLubyte* strokeColor, float strokeWidth, float MasterAlpha, CBoundingBoxComputer*
boundingBox);
```

Parameters

Parameters	Description
xmlNodePtr svg	Any of the supported primitives.
bool doFill	The parent's fill flag. If this is true then also this element is filled.
GLubyte* fillColor	The parent's fill color. Only used if the we have a local opacity.
bool doStroke	The parent's outline flag. If this true then also this element is outlined.
GLubyte* strokeColor	The parent's stroke color. Only used if the we have a local opacity.
float strokeWidth	The parent's stroke width. Used only if we draw an outline at all and no local width is given.
CBoundingBoxComputer* boundingBox	This calculator is used to determine the overall bounding box of the element being parsed.
masterAlpha	The accumulated alpha value, which is currently active. Used in conjunction with a local opacity.

Returns

Returns a new display list for the content of this element.

Remarks

Recursively called function that parses the given svg xml element and converts it to OpenGL calls.

1.2.35.3.5 CSVGParser::parseGroup Method

```
GLuint parseGroup(xmlNodePtr svg, bool doFill, GLubyte* fillColor, bool doStroke, GLubyte*
strokeColor, float strokeWidth, float MasterAlpha, CBoundingBoxComputer* boundingBox);
```

Returns

A new display list comprising all subelements.

Remarks

Collects all data in an element.

Related Topics

parseElement for the description of the parameters.

1.2.35.3.6 CSVGParser::parseImage Method

```
void parseImage(xmlNodePtr svg, CBoundingBoxComputer* boundingBox, bool render);
```

Parameters

Parameters	Description
`xmlNodePtr svg`	The XML svg element containing the definition.
`bool render`	If true then the appropriate OpenGL calls for rendering are issued, otherwise setup is performed.

Remarks

Parses the content of an image definition.

1.2.35.3.7 CSVGParser::parseLine Method

```
void parseLine(xmlNodePtr svg, bool doStroke, GLubyte* strokeColor, float strokeWidth,
CBoundingBoxComputer* boundingBox);
```

Parameters

Parameters	Description
`xmlNodePtr svg`	The XML svg element containing the definition.
`bool doStroke`	Flag to indicate if the line is to be drawn or not.
`GLubyte* strokeColor`	The color to be used for the line (if set).
`float strokeWidth`	The width of the line.
`CBoundingBoxComputer* boundingBox`	This calculator is used to determine the overall bounding box of the element being parsed.

Remarks

Parses the content of a line definition.

1.2.35.3.8 CSVGParser::parsePolygon Method

```
void parsePolygon(xmlNodePtr svg, bool doFill, GLubyte* fillColor, bool doStroke, GLubyte*
strokeColor, float strokeWidth, CBoundingBoxComputer* boundingBox);
```

Remarks

Parses the content of a polygon definition.

Related Topics

parseElement for a description of the parameters.

1.2.35.3.9 CSVGParser::parsePolyline Method

```
void parsePolyline(xmlNodePtr svg, bool doFill, GLubyte* fillColor, bool doStroke, GLubyte*
strokeColor, float strokeWidth, CBoundingBoxComputer* boundingBox);
```

Remarks

Parses the content of a polyline definition.

Related Topics

parseElement for a description of the parameters.

1.2.35.3.10 CSVGParser::parseRectangle Method

```
void parseRectangle(xmlNodePtr svg, bool doFill, GLubyte* fillColor, bool doStroke,
GLubyte* strokeColor, float strokeWidth, CBoundingBoxComputer* boundingBox);
```

Remarks

Parses the content of a rectangle definition.

Related Topics

parseElement for a description of the parameters.

1.2.35.3.11 CSVGParser::parseText Method

```
GLuint parseText(xmlNodePtr svg, bool doFill, GLubyte* fillColor, bool doStroke, GLubyte*
strokeColor, float strokeWidth, float MasterAlpha, CBoundingBoxComputer* boundingBox);
```

Parameters

Parameters	Description
`xmlNodePtr svg`	The text or tspan node to parse.
`GLubyte* fillColor`	The color for the text interior.
`GLubyte* strokeColor`	The color of the outline.
`CBoundingBoxComputer* boundingBox`	This calculator is used to determine the overall bounding box of the element being parsed.
`The`	width of the outlines.
`masterAlpha`	Only passed through because text nodes can have children.

Returns

A new display list comprising all subelements.

Remarks

Takes a text node and gets all attributes for direct or dynamic rendering. This function is called recursively and can take either a or a node.

1.2.35.3.12 CSVGParser::parseTransformation Method

```
void parseTransformation(char* Transformation);
```

Parameters

Parameters	Description
`char* Transformation`	The textual representation of the transformation to parse and perform.

Remarks

Parsers the given string and interprets the content as one or more transformation of the form: translate(x, y, z) scale(x, y, z)

etc.

1.2.35.3.13 CSVGParser::renderVertices Method

```
void renderVertices(bool doFill, GLubyte* fillColor, bool doStroke, GLubyte* strokeColor,
const CVertexVector& Vertices, CVertexVector* TextureCoordinates, float strokeWidth, bool
CloseShape, CBoundingBoxComputer* boundingBox);
```

Parameters

Parameters	Description
`bool doFill`	Filled output is only done if this flag is true. For strokes the existance of the color as such is used as indicator.
`GLubyte* fillColor`	If not NULL then it gives the local color for this call, otherwise the current color stays untouched.
`GLubyte* strokeColor`	If not nULL then the vertices are render again as lines but using that color.
`CVertexVector* TextureCoordinates`	If given (can be NULL) then there must be exactly the same number of texture coordinates as there are vertices.
`float strokeWidth`	The width of the border if one is rendered.
`CBoundingBoxComputer* boundingBox`	This calculator is used to determine the overall bounding box of the element being parsed.
`vertices`	The vertex data to render.
`closeShape`	Determines whether the border line (if strokeColor is given) is closed (that is, the last point is connected to the first point).

Remarks

Renders the given set of vertices.

1.2.36 CTextureManager Class

Class Hierarchy


```
class CTextureManager;
```

File

myx_gc_texture.h (see page 230)

Remarks

This is class CTextureManager.

Destructors

Destructor	Description
~CTextureManager (see page 139)	

Members

Destructors

Destructor	Description
~CTextureManager (see page 139)	

Methods

Method	Description
ClearTextures (see page 139)	
CreateTextureEntry (see page 139)	Creates a new texture entry and adds the entry to the texture list. No image data is loaded yet as this will happen when the texture is used the first time.
FindTexture (see page 139)	Looks throught the textures and attempts to find one with the given name.

SetPathPrefix (see page 139)	

Legend

	Method
	virtual

1.2.36.1 Destructors

1.2.36.1.1 CTextureManager::~CTextureManager Destructor

```
virtual ~CTextureManager(void);
```

1.2.36.2 Methods

1.2.36.2.1 CTextureManager::ClearTextures Method

```
void ClearTextures(void);
```

1.2.36.2.2 CTextureManager::CreateTextureEntry Method

```
CGCTexture* CreateTextureEntry(const TLODList& LODData, const string& ID, const string&
WrapH, const string& WrapV, const string& MinificationFilterStr, const string&
MagnificationFilterStr, int Dimensions, const string& Mode);
```

Parameters

Parameters	Description
`const TLODList& LODData`	A list of level identifiers and names of files, which contain the image data for that level.
`const string& ID`	The identifier (name) of the texture.
`const string& WrapH`	Horizontal wrap mode as string.
`const string& WrapV`	Vertical wrap mode as string.
`int Dimensions`	The number of dimensions of the image data (must be 1 or 2).
`const string& Mode`	The mode how the texture must be applied as string. return The newly created texture object.
`MinificationFilter`	The filter mode for minification as string.
`MagnificationFilter`	The filter mode for magnification as string.

Remarks

Creates a new texture entry and adds the entry to the texture list. No image data is loaded yet as this will happen when the texture is used the first time.

1.2.36.2.3 CTextureManager::FindTexture Method

```
CGCTexture* FindTexture(const string& name);
```

Remarks

Looks throught the textures and attempts to find one with the given name.

1.2.36.2.4 CTextureManager::SetPathPrefix Method

```
void SetPathPrefix(const string& Prefix);
```

1.2.37 LayoutMapper Class

Class Hierarchy


```
class LayoutMapper;
```

File

myx_gc_layout.h (see page 227)

Remarks

The layout mapper provides a simple way of getting a layouter class for a particular layout.

Members

Methods

Method	Description
layouterForElement (see page 140)	Static helper method to create a concrete layouter class for a figure element.

Legend

	Method

1.2.37.1 Methods

1.2.37.1.1 LayoutMapper::layouterForElement Method

```
static CLayouter* layouterForElement(CFigureElement* Element);
```

Parameters

Parameters	Description
`CFigureElement* Element`	The element for which an instance of a layouter is returned.

Returns

An instance of a new layouter class (or NULL if not supported). note The caller is responsible to free the returned instance.

Remarks

Static helper method to create a concrete layouter class for a figure element.

1.2.38 StringTokenizer Class

Class Hierarchy


```
class StringTokenizer;
```

File

myx_gc_utilities.h (see page 232)

Remarks

Simple tokenizer class that works similar as Java's StringTokenizer.

Constructors

Constructor	Description
StringTokenizer (see page 141)	This is StringTokenizer, a member of class StringTokenizer.

Members

Constructors

Constructor	Description
StringTokenizer (see page 141)	This is StringTokenizer, a member of class StringTokenizer.

Methods

Method	Description
hasMoreTokens (see page 141)	This is hasMoreTokens, a member of class StringTokenizer.
lastDelimiter (see page 141)	This is lastDelimiter, a member of class StringTokenizer.
nextToken (see page 141)	This is nextToken, a member of class StringTokenizer.
nextTokenAsFloat (see page 142)	This is nextTokenAsFloat, a member of class StringTokenizer.
nextTokenAsInt (see page 142)	This is nextTokenAsInt, a member of class StringTokenizer.

Legend

	Method

1.2.38.1 Constructors

1.2.38.1.1 StringTokenizer::StringTokenizer Constructor

```
StringTokenizer(string Text, string Delimiters);
```

Remarks

This is StringTokenizer, a member of class StringTokenizer.

1.2.38.2 Methods

1.2.38.2.1 StringTokenizer::hasMoreTokens Method

```
bool hasMoreTokens();
```

Remarks

This is hasMoreTokens, a member of class StringTokenizer.

1.2.38.2.2 StringTokenizer::lastDelimiter Method

```
char lastDelimiter(void);
```

Remarks

This is lastDelimiter, a member of class StringTokenizer.

1.2.38.2.3 StringTokenizer::nextToken Method

```
string nextToken();
```

Remarks

This is nextToken, a member of class StringTokenizer.

1.2.38.2.4 StringTokenizer::nextTokenAsFloat Method

```
float nextTokenAsFloat();
```

Remarks

This is nextTokenAsFloat, a member of class StringTokenizer.

1.2.38.2.5 StringTokenizer::nextTokenAsInt Method

```
int nextTokenAsInt();
```

Remarks

This is nextTokenAsInt, a member of class StringTokenizer.

1.2.39 tagBoundingBox Struct

Class Hierarchy

```
struct tagBoundingBox {
  TVertex upper;
  TVertex lower;
};
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is class tagBoundingBox.

Constructors

Constructor	Description
tagBoundingBox (see page 143)	This is tagBoundingBox, a member of class tagBoundingBox.

Members

Data Members

Data Member	Description
lower (see page 143)	This is lower, a member of class tagBoundingBox.
upper (see page 143)	This is upper, a member of class tagBoundingBox.

Constructors

Constructor	Description
tagBoundingBox (see page 143)	This is tagBoundingBox, a member of class tagBoundingBox.

Legend

	Data Member
	Method

1.2.39.1 Data Members

1.2.39.1.1 tagBoundingBox::lower Data Member

```
TVertex lower;
```

Remarks

This is lower, a member of class tagBoundingBox.

1.2.39.1.2 tagBoundingBox::upper Data Member

```
TVertex upper;
```

Remarks

This is upper, a member of class tagBoundingBox.

1.2.39.2 Constructors

1.2.39.2.1 tagBoundingBox

1.2.39.2.1.1 tagBoundingBox::tagBoundingBox Constructor ()

```
tagBoundingBox();
```

Remarks

This is tagBoundingBox, a member of class tagBoundingBox.

1.2.39.2.1.2 tagBoundingBox::tagBoundingBox Constructor (TVertex, TVertex)

```
tagBoundingBox(TVertex aUpper, TVertex aLower);
```

Remarks

This is tagBoundingBox, a member of class tagBoundingBox.

1.2.40 tagConstraints Struct

Class Hierarchy

```
struct tagConstraints {
  float maxHeight;
  float maxWidth;
  float minHeight;
  float minWidth;
};
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is class tagConstraints.

Constructors

Constructor	Description
tagConstraints (see page 145)	This is tagConstraints, a member of class tagConstraints.

Members

Data Members

Data Member	Description
maxHeight (see page 144)	This is maxHeight, a member of class tagConstraints.
maxWidth (see page 144)	This is maxWidth, a member of class tagConstraints.
minHeight (see page 144)	This is minHeight, a member of class tagConstraints.
minWidth (see page 144)	This is minWidth, a member of class tagConstraints.

Constructors

Constructor	Description
tagConstraints (see page 145)	This is tagConstraints, a member of class tagConstraints.

Legend

	Data Member
	Method

1.2.40.1 Data Members

1.2.40.1.1 tagConstraints::maxHeight Data Member

```
float maxHeight;
```

Remarks

This is maxHeight, a member of class tagConstraints.

1.2.40.1.2 tagConstraints::maxWidth Data Member

```
float maxWidth;
```

Remarks

This is maxWidth, a member of class tagConstraints.

1.2.40.1.3 tagConstraints::minHeight Data Member

```
float minHeight;
```

Remarks

This is minHeight, a member of class tagConstraints.

1.2.40.1.4 tagConstraints::minWidth Data Member

```
float minWidth;
```

Remarks

This is minWidth, a member of class tagConstraints.

1.2.40.2 Constructors

1.2.40.2.1 tagConstraints::tagConstraints Constructor

```
tagConstraints();
```

Remarks

This is tagConstraints, a member of class tagConstraints.

1.2.41 tagGCVariant Struct

Class Hierarchy

```
struct tagGCVariant {
  TGCVariantType type;
  bool b;
  int i;
  float f;
  string s;
  CGCBase* reference;
};
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is class tagGCVariant.

Constructors

Constructor	Description
tagGCVariant (see page 146)	This is tagGCVariant, a member of class tagGCVariant.

Members

Data Members

Data Member	Description
b (see page 146)	This is b, a member of class tagGCVariant.
f (see page 146)	This is f, a member of class tagGCVariant.
i (see page 146)	Values are not in a union because of the string entry.
reference (see page 146)	This is reference, a member of class tagGCVariant.
s (see page 146)	This is s, a member of class tagGCVariant.
type (see page 146)	This is type, a member of class tagGCVariant.

Constructors

Constructor	Description
tagGCVariant (see page 146)	This is tagGCVariant, a member of class tagGCVariant.

Operators

Operator	Description
= (see page 147)	This is =, a member of class tagGCVariant.

Operators

Operator	Description
= (see page 147)	This is =, a member of class tagGCVariant.

Legend

	Data Member
	Method

1.2.41.1 Data Members

1.2.41.1.1 tagGCVariant::b Data Member

```
bool b;
```

Remarks

This is b, a member of class tagGCVariant.

1.2.41.1.2 tagGCVariant::f Data Member

```
float f;
```

Remarks

This is f, a member of class tagGCVariant.

1.2.41.1.3 tagGCVariant::i Data Member

```
int i;
```

Remarks

Values are not in a union because of the string entry.

1.2.41.1.4 tagGCVariant::reference Data Member

```
CGCBase* reference;
```

Remarks

This is reference, a member of class tagGCVariant.

1.2.41.1.5 tagGCVariant::s Data Member

```
string s;
```

Remarks

This is s, a member of class tagGCVariant.

1.2.41.1.6 tagGCVariant::type Data Member

```
TGCVariantType type;
```

Remarks

This is type, a member of class tagGCVariant.

1.2.41.2 Constructors

1.2.41.2.1 tagGCVariant::tagGCVariant Constructor

```
tagGCVariant();
```

Remarks

This is tagGCVariant, a member of class tagGCVariant.

1.2.41.3 Operators

1.2.41.3.1 tagGCVariant::= Operator

```
tagGCVariant& operator =(const tagGCVariant& other);
```

Remarks

This is =, a member of class tagGCVariant.

1.2.42 tagVertex Struct

Class Hierarchy

```
struct tagVertex {
  float x;
  float y;
  float z;
  float w;
};
```

File

myx_gc_datatypes.h (see page 220)

Remarks

Some geometric data types.

Constructors

Constructor	Description
tagVertex (see page 148)	This is tagVertex, a member of class tagVertex.

Members

Data Members

Data Member	Description
w (see page 148)	This is w, a member of class tagVertex.
x (see page 148)	This is x, a member of class tagVertex.
y (see page 148)	This is y, a member of class tagVertex.
z (see page 148)	This is z, a member of class tagVertex.

Constructors

Constructor	Description
tagVertex (see page 148)	This is tagVertex, a member of class tagVertex.

Legend

	Data Member
	Method

1.2.42.1 Data Members

1.2.42.1.1 tagVertex::w Data Member

```
float w;
```

Remarks

This is w, a member of class tagVertex.

1.2.42.1.2 tagVertex::x Data Member

```
float x;
```

Remarks

This is x, a member of class tagVertex.

1.2.42.1.3 tagVertex::y Data Member

```
float y;
```

Remarks

This is y, a member of class tagVertex.

1.2.42.1.4 tagVertex::z Data Member

```
float z;
```

Remarks

This is z, a member of class tagVertex.

1.2.42.2 Constructors

1.2.42.2.1 tagVertex

1.2.42.2.1.1 tagVertex::tagVertex Constructor ()

```
tagVertex();
```

Remarks

This is tagVertex, a member of class tagVertex.

1.2.42.2.1.2 tagVertex::tagVertex Constructor (double, double, double)

```
tagVertex(double aX, double aY, double aZ);
```

Remarks

This is tagVertex, a member of class tagVertex.

1.2.42.2.1.3 tagVertex::tagVertex Constructor (float, float, float, float)

```
tagVertex(float aX, float aY, float aZ, float aW);
```

Remarks

This is tagVertex, a member of class tagVertex.

1.2.43 tagViewport Struct

Class Hierarchy

```
struct tagViewport {
  int left, top, width, height;
};
```

File

myx_gc_datatypes.h (see page 220)

Remarks

ifdef _WINDOWS (see page 210)

Constructors

Constructor	Description
tagViewport (see page 150)	This is tagViewport, a member of class tagViewport.

Members

Data Members

Data Member	Description
height (see page 149)	This is height, a member of class tagViewport.
left (see page 149)	This is left, a member of class tagViewport.
top (see page 150)	This is top, a member of class tagViewport.
width (see page 150)	This is width, a member of class tagViewport.

Constructors

Constructor	Description
tagViewport (see page 150)	This is tagViewport, a member of class tagViewport.

Legend

	Data Member
	Method

1.2.43.1 Data Members

1.2.43.1.1 tagViewport::height Data Member

```
int height;
```

Remarks

This is height, a member of class tagViewport.

1.2.43.1.2 tagViewport::left Data Member

```
int left;
```

Remarks

This is left, a member of class tagViewport.

1.2.43.1.3 tagViewport::top Data Member

```
int top;
```

Remarks

This is top, a member of class tagViewport.

1.2.43.1.4 tagViewport::width Data Member

```
int width;
```

Remarks

This is width, a member of class tagViewport.

1.2.43.2 Constructors

1.2.43.2.1 tagViewport

1.2.43.2.1.1 tagViewport::tagViewport Constructor ()

```
tagViewport();
```

Remarks

This is tagViewport, a member of class tagViewport.

1.2.43.2.1.2 tagViewport::tagViewport Constructor (int, int, int, int)

```
tagViewport(int iLeft, int iTop, int iWidth, int iHeight);
```

Remarks

This is tagViewport, a member of class tagViewport.

1.3 Functions

1.3.1 boundsAreEmpty Function

```
bool boundsAreEmpty(const TBoundingBox& Bounds);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
bounds	The bounds to examine.

Returns

True if the bounds are empty, false otherwise.

Remarks

Determines whether bounds are empty.

Examines the given bounds and returns whether it is empty or not.

1.3.2 boundsContainPoint Function

```
bool boundsContainPoint(const TBoundingBox& Bounds, const float X, const float Y);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
bounds	The bounds to check the point against.
x	The horizontal coordinate to check.
y	The vertical coordinate to check.

Returns

True if the point is within the bounds, otherwise false.

Remarks

Checks if a given point is within the given bounds.

Determines whether the given bounds include the given point.

1.3.3 boundsIntersect Function

```
bool boundsIntersect(const TBoundingBox& Bounds1, const TBoundingBox& Bounds2);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
const TBoundingBox& Bounds1	One of the bounds to compare.
const TBoundingBox& Bounds2	The other bounds to compare.

Returns

True if both bounds overlap each other, otherwise false.

Remarks

Determines whether both bounds overlap.

Determines whether both bounds overlap.

1.3.4 colorByName Function

```
bool colorByName(string name, GLubyte* Color);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`string name`	The name of the color to find.
`GLubyte* Color`	[out] The color data if it could be found. returns true if the color could be found, otherwise false;

Remarks

Find a color by name.

Searchs the predifined colors and tries to find one with the given name.

1.3.5 colorToString Function

```
string colorToString(GLfloat* Color);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`GLfloat* Color`	The color to convert.

Returns

The given color as HTML string.

Remarks

Converts a (float) color to a string.

Converts a color to a string in the form #RRGGBB.

1.3.6 colorToString Function

```
string colorToString(GLubyte* Color);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`GLubyte* Color`	The color to convert.

Returns

The given color as HTML string.

Remarks

Converts a (byte) color to a string.

Converts a color to a string in the form #RRGGBB.

1.3.7 convertColor Function

```
int convertColor(xmlNodePtr Element, const char* name, GLubyte* Color);
```

File

myx_gc_gl_helper.h (see page 225)

Parameters

Parameters	Description
`xmlNodePtr Element`	The XML element to parse.
`const char* name`	The name of the color attribute.
`GLubyte* Color`	[out] The converted color.

Returns

0 - If a color could be found and converted. 1 - If a color could be found but a conversion error occured. 2 - No color was given. 3 - The special color "none" was found.

Remarks

Reads the attribute with the given name and treats it as color value.

Reads attribute name from Element and tries to treat the string as a color. The allowed syntax for colors is (as given by the SVG specification) either an HTML like value (e.g. #FFFFFF, #FFF) or a function like form (e.g. rgb(100, 255, 255), rgb(10%, 100%, 0%)).

1.3.8 convertFontWeight Function

```
int convertFontWeight(const string Weight);
```

File

myx_gc_font_manager.h (see page 224)

Remarks

Converts the given string into a font weight value. Allowed values are: normal | bold | bolder | lighter | 100 | 200 | 300| 400 | 500 | 600 | 700 | 800 | 900 | inherit

1.3.9 CreateGenericCanvas Function

```
GENERIC_CANVAS_API CGenericCanvas* CreateGenericCanvas(GCContext Context, char* name);
```

File

myx_gc_canvas.h (see page 218)

Remarks

Factory function to create a generic canvas. This function is exported and must be used by the viewer implementations to actually create a canvas instance. This is the only way to get hold of a generic canvas instance for non-C++ languages.

Factory function for a canvas.

1.3.10 DefaultFontFamily Function

```
const string DefaultFontFamily("Arial");
```

File

myx_gc_const.h (see page 219)

Remarks

Default values for text.

1.3.11 DefaultFontStyle Function

```
const string DefaultFontStyle("normal");
```

File

myx_gc_const.h (see page 219)

Remarks

This is function DefaultFontStyle.

1.3.12 DefaultFontWeight Function

```
const string DefaultFontWeight("400");
```

File

myx_gc_const.h (see page 219)

Remarks

Must be a string as we get it from an attribute that can contain strings.

1.3.13 DefaultTextureMagFilter Function

```
const string DefaultTextureMagFilter("nearest");
```

File

myx_gc_texture.h (see page 230)

Remarks

This is function DefaultTextureMagFilter.

1.3.14 DefaultTextureMinFilter Function

```
const string DefaultTextureMinFilter("nearest");
```

File

myx_gc_texture.h (see page 230)

Remarks

This is function DefaultTextureMinFilter.

1.3.15 DefaultTextureMode Function

```
const string DefaultTextureMode("decal");
```

File

myx_gc_texture.h (see page 230)

Remarks

This is function DefaultTextureMode.

1.3.16 DefaultTextureWrapMode Function

```
const string DefaultTextureWrapMode("clamp");
```

File

myx_gc_texture.h (see page 230)

Remarks

Default values for texturing.

1.3.17 extractFilePath Function

```
GENERIC_CANVAS_API string extractFilePath(const string& Filename);
```

File

myx_gc_utilities.h (see page 232)

Remarks

Extracts the drive and path from the given file name.

Javadoc Summary

ExtractFilePath extracts the drive and directory parts of the given filename. The resulting string is the leftmost characters of FileName, up to and including the colon or backslash that separates the path information from the name and extension. The resulting string is empty if FileName contains no drive and directory parts.

- @param Filename The file name (ANSI encoded) of which the path is to be extracted.
- @return The extracted path part (ANSI encoded).

1.3.18 fontManager Function

```
CFontManager* fontManager(void);
```

File

myx_gc_font_manager.h (see page 224)

Remarks

Returns the singleton font manager instance.

Returns the current font manager (there is always only one).

1.3.19 freeImage Function

```
GENERIC_CANVAS_API void freeImage(TImage* Image);
```

File

myx_gc_utilities.h (see page 232)

Remarks

Releases the given image.

1.3.20 getContainerID Function

```
GENERIC_CANVAS_API TContainerID getContainerID(const string& container);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`const string& container`	The name of the container.

Returns

An identifier that specifies which container was given.

Remarks

Converts a container name to an identifier suitable for quick lookup.

Looks the given container name up and returns an identifier for it that can be used for quick lookup/handling.

1.3.21 getCurrentDir Function

```
GENERIC_CANVAS_API string getCurrentDir(void);
```

File

myx_gc_utilities.h (see page 232)

Returns

The current working folder.

Remarks

Returns the current working folder.

Javadoc Summary

Returns the current working folder (ANSI encoded).

1.3.22 getEntryIndex Function

```
int getEntryIndex(string& Path);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`string& Path`	[in, out] The path to examine. It must start with a slash to be valid. On return it contains a new path without the index part (can be empty then).

Returns

The integer value extracted from the top path part.

Remarks

Returns the index value for a given property.

Treats the given path as property name preceeded with a slash. The first (or only) subpath must be an integer number denoting an index in a list.

1.3.23 getFloatAttribute Function

```
bool getFloatAttribute(xmlNodePtr Element, const char* name, float& Value);
```

File

myx_gc_gl_helper.h (see page 225)

Remarks

Reads the attribute with the given name (if it exists) and converts it to a float value.

Helper method to retrieve a float attribute.

1.3.24 getFloatAttributeDef Function

```
float getFloatAttributeDef(xmlNodePtr Element, const char* name, float Default);
```

File

myx_gc_gl_helper.h (see page 225)

Remarks

Like GetFloatAttribute but with a default value in case the attribute does not exist.

Helper method to retrieve an integer attribute. If it cannot be found a default value will be used instead.

1.3.25 getIntAttribute Function

```
bool getIntAttribute(xmlNodePtr Element, const char* name, int& Value);
```

File

myx_gc_gl_helper.h (see page 225)

Remarks

Reads the attribute with the given name (if it exists) and converts it to an integer value.

Helper method to retrieve an integer attribute.

1.3.26 getIntAttributeDef Function

```
int getIntAttributeDef(xmlNodePtr Element, const char* name, int Default);
```

File

myx_gc_gl_helper.h (see page 225)

Remarks

Like GetIntAttribute but with a default value in case the attribute does not exist.

Helper method to retrieve an integer attribute. If it cannot be found a default value will be used instead.

1.3.27 getPropertyID Function

```
GENERIC_CANVAS_API TPropertyID getPropertyID(const string& property);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`const string& property`	The property name to look up.

Returns

An identifier that specifies which property was given.

Remarks

Returns an identifier for a given property name.

Looks up the property name and returns an identifier for it.

1.3.28 getStringAttribute Function

```
bool getStringAttribute(xmlNodePtr Element, const char* name, string& Value);
```

File

myx_gc_gl_helper.h (see page 225)

Remarks

Reads the attribute with the given name (if it exists) and returns it.

Helper method to retrieve a string attribute. If the attribute could be found then true is returned and Value is set to the value of the attribute. Otherwise false is returned and Value is not touched.

1.3.29 getStringAttributeDef Function

```
string getStringAttributeDef(xmlNodePtr Element, const char* name, const string Default);
```

File

myx_gc_gl_helper.h (see page 225)

Remarks

Like GetStringAttribute but with a default value in case the attribute does not exist.

Helper method to retrieve a string attribute. If the attribute is empty or cannot be found then a default value is returned.

1.3.30 loadPNG Function

```
GENERIC_CANVAS_API TImage* loadPNG(const string& Filename);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`const string& Filename`	An ANSI encoded file name (can contain path info) to the png file.

Returns

Returns a pointer to a TImage (see page 188) structure containing the image data. note The return memory must be freed using FreeImage().

Remarks

Loads the given PNG image from disk.

Javadoc Summary

Loads a png file.

1.3.31 lockFontManager Function

```
void lockFontManager(void);
```

File

myx_gc_font_manager.h (see page 224)

Remarks

Increase lock count for the manager.

Increases the lock count of the font manager. If the manager does not yet exist it is created.

1.3.32 matrixMultiply Function

```
static void matrixMultiply(TMatrix product, const TMatrix a, const TMatrix b);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`TMatrix product`	will receive the product of a and b. warning Is assumed that product != b. product == a is allowed. note KW: 4*16 = 64 multiplications
`const TMatrix a`	matrix.
`const TMatrix b`	matrix.

Remarks

Matrix code

Perform a full 4x4 matrix multiplication.

Author

This function was taken from Mesa3D (http://www.mesa3d.org/).

1.3.33 matrixRotate Function

```
void matrixRotate(TMatrix mat, GLfloat angle, GLfloat x, GLfloat y, GLfloat z);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`TMatrix mat`	The target matrix to multiply the rotation into.
`GLfloat angle`	The angle around which to rotate.
`GLfloat x`	The x coordinate of the axis to turn around.
`GLfloat y`	The y coordinate of the axis to turn around.
`GLfloat z`	The z coordinate of the axis to turn around.

Remarks

Generate a 4x4 transformation matrix from glRotate parameters, and post-multiply the input matrix by it.

Author

This function was taken from Mesa3D (http://www.mesa3d.org/).

1.3.34 matrixScale Function

```
void matrixScale(TMatrix mat, GLfloat x, GLfloat y, GLfloat z);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`TMatrix mat`	matrix.
`GLfloat x`	x axis scale factor.
`GLfloat y`	y axis scale factor.
`GLfloat z`	z axis scale factor. Multiplies in-place the elements of mat by the scale factors.

Remarks

Multiply a matrix with a general scaling matrix.

Author

This function was taken from Mesa3D (http://www.mesa3d.org/).

1.3.35 matrixTransform Function

```
TVertex matrixTransform(TMatrix M, TVertex V);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`TMatrix M`	The matrix containing the transformation parameters.
`TVertex V`	The vertex to transform.

Returns

The transformed vertex.

Remarks

Multiplies the given vertex by matrix M and returns the result.

1.3.36 matrixTranslate Function

```
void matrixTranslate(TMatrix mat, GLfloat x, GLfloat y, GLfloat z);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`TMatrix mat`	matrix.
`GLfloat x`	translation vector x coordinate.
`GLfloat y`	translation vector y coordinate.
`GLfloat z`	translation vector z coordinate. Adds the translation coordinates to the elements of mat in-place.

Remarks

Multiply a matrix with a translation matrix.

Author

This function was taken from Mesa3D (http://www.mesa3d.org/).

1.3.37 openFile Function

```
FILE* openFile(string Filename, const char* OpenMode);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`string Filename`	The name of file encode in UTF-8.
`const char* OpenMode`	The mode how to open the file (the same as used for fopen calls).

Returns

A pointer to a FILE structure if the file could be opened, NULL otherwise.

Remarks

Platform neutral file open function.

Javadoc Summary

Platform neutral file open routine.

1.3.38 parseTextureEntry Function

```
void parseTextureEntry(xmlNodePtr XML);
```

File

myx_gc_gl_helper.h (see page 225)

Remarks

Parses the given XML element for a texture definition.

Parses the given XML node for texture information and creates a new entry in the texture manager.

1.3.39 registerSystemColors Function

```
void registerSystemColors(const CColorMap & ColorMap);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`const CColorMap & ColorMap`	Colors (see page 200) to add to the predefined color list.

Remarks

Adds colors to the named color table.

Registers predifined colors.

1.3.40 setCurrentDir Function

```
GENERIC_CANVAS_API void setCurrentDir(const string& Folder);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`const string& Folder`	The new folder to be set.

Remarks

Sets the current working folder.

Javadoc Summary

Sets the current working folder (folder name must be ANSI encoded).

1.3.41 sortBounds Function

```
TBoundingBox sortBounds(const TBoundingBox& Bounds);
```

File

myx_gc_utilities.h (see page 232)

Remarks

Sorts the given bounds so that left <= right and bottom <= top.

Helper method to sort left/right and bottom/top coordinates so that for left/top are always smaller than right/bottom (origin is considered in the left-upper corner, +y pointing down).

1.3.42 stringToColor Function

```
int stringToColor(string ColorString, GLfloat* Color);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`string ColorString`	The string to parse. param [out] The color converted from the string. It must have room for at least 3 members.

Returns

0 - If a color could be found and converted. 1 - If a color could be found but a conversion error occured. 2 - No color was given. 3 - The special color "none" was found.

Remarks

Converts a string to color with float members.

Converts a string to color with float members. The allowed syntax for colors is (as given by the SVG specification) either an HTML like value (e.g. #FFFFFF, #FFF) or a function like form (e.g. rgb(100, 255, 255), rgb(10%, 100%, 0%)).

1.3.43 stringToColor Function

```
int stringToColor(string ColorString, GLubyte* Color);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`string ColorString`	The string to parse. param [out] The color converted from the string. It must have room for at least 3 members.

Returns

0 - If a color could be found and converted. 1 - If a color could be found but a conversion error occured. 2 - No color was given. 3 - The special color "none" was found.

Remarks

Converts a string to color with byte members.

Converts a string to a color with byte members. The allowed syntax for colors is (as given by the SVG specification) either an HTML like value (e.g. #FFFFFF, #FFF) or a function like form (e.g. rgb(100, 255, 255), rgb(10%, 100%, 0%)).

1.3.44 textureManager Function

```
CTextureManager* textureManager();
```

File

myx_gc_texture.h (see page 230)

Remarks

The one (and only) texture manager instance.

1.3.45 unlockFontManager Function

```
void unlockFontManager(void);
```

File

myx_gc_font_manager.h (see page 224)

Remarks

Decrease lock count for the manager.

Returns the current font manager (there is always only one).

1.3.46 utf16ToANSI Function

```
GENERIC_CANVAS_API string utf16ToANSI(const wstring& Source);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`const wstring& Source`	Contains the source string encoded in UTF-16.

Returns

The converted string in ANSI encoding. note The current user locale is used to convert the Unicode string to ANSI.

Remarks

Converts the given string into an ANSI string using the current system locale.

Javadoc Summary

Converts the given string into an ANSI string.

1.3.47 utf16ToUtf8 Function

```
GENERIC_CANVAS_API string utf16ToUtf8(const wstring& Source);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`const wstring& Source`	Contains the source string encoded in UTF-16.

Returns

The converted string in UTF-8 encoding.

Remarks

Converts the given UTF-16 string into an UTF-8 string.

Converts the given UTF-16 string into an UTF-8 string.

1.3.48 utf8ToANSI Function

```
GENERIC_CANVAS_API string utf8ToANSI(const string& Source);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`const string& Source`	Contains the source string encoded in UTF-8.

Returns

The converted string in ANSI encoding. note The current user locale is used to convert the Unicode string to ANSI.

Remarks

Converts the given string, which is supposed to be an UTF-8 encoded text into an ANSI string using the current system locale.

Javadoc Summary

Converts the given string, which is supposed to be an UTF-8 encoded text into an ANSI string.

1.3.49 utf8ToUtf16 Function

```
GENERIC_CANVAS_API wstring utf8ToUtf16(const string& Source);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`const string& Source`	Contains the source string encoded in UTF-8.

Returns

The converted string in UTF-16 encoding.

Remarks

Converts the given string, which is supposed to be an UTF-8 encoded text into an UTF-16 string.

Converts the given string, which is supposed to be an UTF-8 encoded text into an UTF-16 string.

1.3.50 variant Function

```
GENERIC_CANVAS_API TGCVariant variant(const bool Value);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`const bool Value`	The value to wrap in the variant.

Returns

A new variant carrying the given value.

Remarks

Creates a GC variant from the given value.

1.3.51 variant Function

```
GENERIC_CANVAS_API TGCVariant variant(const char* Value);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`const char* Value`	The value to wrap in the variant.

Returns

A new variant carrying the given value.

Remarks

Creates a GC variant from the given value.

1.3.52 variant Function

```
GENERIC_CANVAS_API TGCVariant variant(const float Value);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`const float Value`	The value to wrap in the variant.

Returns

A new variant carrying the given value.

Remarks

Creates a GC variant from the given value.

1.3.53 variant Function

```
GENERIC_CANVAS_API TGCVariant variant(const int Value);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`const int Value`	The value to wrap in the variant.

Returns

A new variant carrying the given value.

Remarks

Creates a GC variant from the given value.

1.3.54 variant Function

```
GENERIC_CANVAS_API TGCVariant variant(const string& Value);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`const string& Value`	The value to wrap in the variant.

Returns

A new variant carrying the given value.

Remarks

Creates a GC variant from the given value.

1.3.55 variantToBool Function

```
GENERIC_CANVAS_API bool variantToBool(const TGCVariant& Variant);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`const TGCVariant& Variant`	The variant (see page 166) to convert.

Returns

The value of the variant (see page 166) as bool.

Remarks

Converts a GC variant (see page 166) to a bool.

1.3.56 variantToFloat Function

```
GENERIC_CANVAS_API float variantToFloat(const TGCVariant& Variant);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`const TGCVariant& Variant`	The variant (see page 166) to convert.

Returns

The value of the variant (see page 166) as float.

Remarks

Converts a GC variant (see page 166) to a float.

1.3.57 variantToInt Function

```
GENERIC_CANVAS_API int variantToInt(const TGCVariant& Variant);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`const TGCVariant& Variant`	The variant (see page 166) to convert.

Returns

The value of the variant (see page 166) as integer.

Remarks

Converts a GC variant (see page 166) to an integer.

1.3.58 variantToString Function

```
GENERIC_CANVAS_API string variantToString(const TGCVariant& Variant);
```

File

myx_gc_utilities.h (see page 232)

Parameters

Parameters	Description
`const TGCVariant& Variant`	The variant (see page 166) to convert.

Returns

The value of the variant (see page 166) as string.

Remarks

Conversion functions for GC variants and simple values.

Converts a GC variant (see page 166) to a string.

1.4 Structs, Records, Enums

1.4.1 tagAction Struct

```
struct tagAction {
  TActionType type;
  CActionParameters parameters;
};
```

File

myx_gc_datatypes.h (see page 220)

Remarks

An action with associated parameters.

1.4.2 tagActionType Enumeration

```
enum tagActionType {
  GC_ACTION_NONE,
  GC_ACTION_TOGGLE,
  GC_ACTION_EXPAND,
  GC_ACTION_COLLAPSE,
  GC_ACTION_CHANGE_STYLE,
  GC_ACTION_DRAG_FIGURE,
  GC_ACTION_DRAG_ALL,
  GC_ACTION_RESIZE
};
```

File

myx_gc_datatypes.h (see page 220)

Members

Members	Description
GC_ACTION_NONE	No action is assigned.
GC_ACTION_TOGGLE	Switch between expanded and collapsed state (scope: element).
GC_ACTION_EXPAND	Expand the element (scope: element).
GC_ACTION_COLLAPSE	Collapse the element (scope: element).
GC_ACTION_CHANGE_STYLE	Switch to a new style (scope: element).
GC_ACTION_DRAG_FIGURE	Start dragging of the associated figure instance (scope: a single figure instance).
GC_ACTION_DRAG_ALL	Start dragging of all selected figure instances (scope: all selected figure instances on the associated layer).
GC_ACTION_RESIZE	The figure is resized.

Remarks

Determines the type of action possible with a particular figure instance/element.

1.4.3 tagBidiMode Enumeration

```
enum tagBidiMode {
  GC_BIDI_LEFT_TO_RIGHT,
  GC_BIDI_RIGHT_TO_LEFT
};
```

File

myx_gc_datatypes.h (see page 220)

Members

Members	Description
GC_BIDI_LEFT_TO_RIGHT	Standard directionality (most languages).
GC_BIDI_RIGHT_TO_LEFT	Used for arabic and hebrew text.

Remarks

Bidirectional mode

1.4.4 tagChangeReason Enumeration

```
enum tagChangeReason {
  GC_CHANGE_SELECTION_ADD,
  GC_CHANGE_SELECTION_CLEAR,
  GC_CHANGE_SELECTION_REMOVE,
  GC_CHANGE_SELECTION_CHANGE,
  GC_CHANGE_CANVAS_REFRESH,
  GC_CHANGE_CANVAS_PROPERTY,
  GC_CHANGE_CANVAS_ADD_VIEW,
  GC_CHANGE_CANVAS_ADD_LAYER,
  GC_CHANGE_CANVAS_SWITCH_VIEW,
  GC_CHANGE_CANVAS_REMOVE_VIEW,
  GC_CHANGE_CANVAS_REMOVE_LAYER,
  GC_CHANGE_CANVAS_CLEAR_CONTENT,
  GC_CHANGE_CANVAS_CLEAR_LAYOUTS,
  GC_CHANGE_CANVAS_CLEAR_STYLES,
  GC_CHANGE_MODEL_PROPERTY,
  GC_CHANGE_MODEL_ADD_FIGURE,
  GC_CHANGE_MODEL_REMOVE_FIGURE,
  GC_CHANGE_MODEL_ADD_STYLE,
  GC_CHANGE_MODEL_REMOVE_STYLE,
  GC_CHANGE_CAPTION_PROPERTY,
  GC_CHANGE_ELEMENT_PROPERTY,
```

```
    GC_CHANGE_ELEMENT_ADD_CHILD,
    GC_CHANGE_FIGURE_PROPERTY,
    GC_CHANGE_FIGURE_EXCHANGE,
    GC_CHANGE_FINSTANCE_PROPERTY,
    GC_CHANGE_VIEW_PROPERTY,
    GC_CHANGE_VIEW_ADD_LAYER,
    GC_CHANGE_VIEW_REMOVE_LAYER,
    GC_CHANGE_VIEW_CLEAR,
    GC_CHANGE_LAYER_CLEAR,
    GC_CHANGE_LAYER_VISIBILITY,
    GC_CHANGE_LAYER_PROPERTY,
    GC_CHANGE_LAYER_ADD_INSTANCE,
    GC_CHANGE_LAYER_REMOVE_INSTANCE,
    GC_CHANGE_LAYER_ADD_GROUP,
    GC_CHANGE_LAYER_REMOVE_GROUP,
    GC_CHANGE_CONNECTION_INSTANCE
  };
```

File

myx_gc_datatypes.h (see page 220)

Members

Members	Description
GC_CHANGE_SELECTION_ADD	One or more figure instances were added to the current selection.
GC_CHANGE_SELECTION_CLEAR	The current selection was cleared.
GC_CHANGE_SELECTION_REMOVE	One or more figure instances were removed from the current selection.
GC_CHANGE_SELECTION_CHANGE	One or more figure instances were added to or removed from the current selection.
GC_CHANGE_CANVAS_REFRESH	Used to indicate that the view must update the visual representation.
GC_CHANGE_CANVAS_PROPERTY	The value of a property has been changed.
GC_CHANGE_CANVAS_ADD_VIEW	A new view was added.
GC_CHANGE_CANVAS_ADD_LAYER	A new layer was added.
GC_CHANGE_CANVAS_SWITCH_VIEW	Another view was activated.
GC_CHANGE_CANVAS_REMOVE_VIEW	A view was removed.
GC_CHANGE_CANVAS_REMOVE_LAYER	A layer was removed.
GC_CHANGE_CANVAS_CLEAR_CONTENT	All figures have been removed.
GC_CHANGE_CANVAS_CLEAR_LAYOUTS	All layout definitions have been removed.
GC_CHANGE_CANVAS_CLEAR_STYLES	All styles have been removed.
GC_CHANGE_MODEL_PROPERTY	The value of a property has been changed.
GC_CHANGE_MODEL_ADD_FIGURE	A new figure was added.
GC_CHANGE_MODEL_REMOVE_FIGURE	A figure was removed.
GC_CHANGE_MODEL_ADD_STYLE	A new style was added.
GC_CHANGE_MODEL_REMOVE_STYLE	A style was removed.
GC_CHANGE_CAPTION_PROPERTY	The value of a property has been changed.
GC_CHANGE_ELEMENT_PROPERTY	The value of a property has been changed.
GC_CHANGE_ELEMENT_ADD_CHILD	The value of a property has been changed.
GC_CHANGE_FIGURE_PROPERTY	The value of a property has been changed.
GC_CHANGE_FIGURE_EXCHANGE	A figure is about to be replaced by another one.
GC_CHANGE_FINSTANCE_PROPERTY	The value of a property has been changed.
GC_CHANGE_VIEW_PROPERTY	The value of a property has been changed.
GC_CHANGE_VIEW_ADD_LAYER	A new layer was added to a view.
GC_CHANGE_VIEW_REMOVE_LAYER	A layer was removed.
GC_CHANGE_VIEW_CLEAR	The view was cleared.
GC_CHANGE_LAYER_CLEAR	All figure instances on the layer are removed.
GC_CHANGE_LAYER_VISIBILITY	The visibility of a layer has been changed.
GC_CHANGE_LAYER_PROPERTY	The value of a property has been changed.
GC_CHANGE_LAYER_ADD_INSTANCE	A new figure instance was added.
GC_CHANGE_LAYER_REMOVE_INSTANCE	A figure instance was removed.
GC_CHANGE_LAYER_ADD_GROUP	A new group was added to a view.
GC_CHANGE_LAYER_REMOVE_GROUP	A group was removed.
GC_CHANGE_CONNECTION_INSTANCE	Connections

Remarks

This is record tagChangeReason.

1.4.5 tagColorEntry Struct

```
struct tagColorEntry {
  char* name;
  GLubyte Color[4];
};
```

File

myx_gc_const.h (see page 219)

1.4.6 tagColorType Enumeration

```
enum tagColorType {
  COLOR_TYPE_PALETTE = PNG_COLOR_TYPE_PALETTE,
  COLOR_TYPE_GRAY = PNG_COLOR_TYPE_GRAY,
  COLOR_TYPE_GRAY_ALPHA = PNG_COLOR_TYPE_GRAY_ALPHA,
  COLOR_TYPE_RGB = PNG_COLOR_TYPE_RGB,
  COLOR_TYPE_RGB_ALPHA = PNG_COLOR_TYPE_RGB_ALPHA
};
```

File

myx_gc_utilities.h (see page 232)

1.4.7 tagConnectionDirection Enumeration

```
enum tagConnectionDirection {
  GC_DIR_NONE,
  GC_DIR_NORTH,
  GC_DIR_EAST,
  GC_DIR_SOUTH,
  GC_DIR_WEST
};
```

File

myx_gc_connection.h (see page 219)

Remarks

This is record tagConnectionDirection.

1.4.8 tagConnectionLineStyle Enumeration

```
enum tagConnectionLineStyle {
  GC_CONNECTION_STYLE_SOLID,
  GC_CONNECTION_STYLE_DOTTED
};
```

File

myx_gc_datatypes.h (see page 220)

Remarks

Style for a connection line.

1.4.9 tagContainerID Enumeration

```
enum tagContainerID {
  GC_CONTAINER_UNKNOWN,
  GC_CONTAINER_LAYERS,
  GC_CONTAINER_FEEDBACK,
  GC_CONTAINER_FIGURE_INSTANCES,
  GC_CONTAINER_FIGURE_CONTENT,
  GC_CONTAINER_VIEWS,
  GC_CONTAINER_MODEL,
  GC_CONTAINER_STYLE,
  GC_CONTAINER_STYLES,
  GC_CONTAINER_LAYOUTS,
  GC_CONTAINER_FIGURE,
  GC_CONTAINER_FIGURES,
  GC_CONTAINER_SCALING,
  GC_CONTAINER_TRANSLATION,
  GC_CONTAINER_ROTATION,
  GC_CONTAINER_GROUPS,
  GC_CONTAINER_CHILDREN,
  GC_CONTAINER_CAPTION,
  GC_CONTAINER_CONTENT
};
```

File

myx_gc_datatypes.h (see page 220)

Remarks

Identifier for containers. Used when parsing pathes for properties.

1.4.10 tagFeedbackInfo Enumeration

```
enum tagFeedbackInfo {
  GC_FI_NONE,
  GC_FI_ON_OBJECT,
  GC_FI_NORTH,
  GC_FI_NORTH_EAST,
  GC_FI_EAST,
  GC_FI_SOUTH_EAST,
  GC_FI_SOUTH,
  GC_FI_SOUTH_WEST,
  GC_FI_WEST,
  GC_FI_NORTH_WEST
};
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is record tagFeedbackInfo.

1.4.11 tagFigureElementLayout Enumeration

```
enum tagFigureElementLayout {
  GC_LAYOUT_ROW,
  GC_LAYOUT_COLUMN
};
```

File

myx_gc_datatypes.h (see page 220)

Remarks

Layout variants for a figure element.

1.4.12 tagFigureElementResize Enumeration

```
enum tagFigureElementResize {
  GC_RESIZE_NONE,
  GC_RESIZE_HORIZONTAL_ONLY,
  GC_RESIZE_VERTICAL_ONLY,
  GC_RESIZE_ALL
};
```

File

myx_gc_datatypes.h (see page 220)

Remarks

Resize variants for a figure element.

1.4.13 tagFontFileEntry Struct

```
struct tagFontFileEntry {
  int useCount;
  string entries[2][2];
};
```

File

myx_gc_font_manager.h (see page 224)

Remarks

This is record tagFontFileEntry.

1.4.14 tagGCError Enumeration

```
enum tagGCError {
  GC_NO_ERROR = 0,
  GC_CANT_OPEN_FILE,
  GC_XML_PARSE_ERROR,
  GC_XML_INVALID_DOCUMENT,
  GC_XML_EMPTY_DOCUMENT,
  GC_OBJECT_NOT_FOUND,
  GC_CANT_READ_FROM_FILE,
```

```
  GC_CHARSET_CONVERSION_ERROR,
  GC_CHARSET_WRONG_CHARSET_SPECIFIED
};
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is record tagGCError.

1.4.15 tagGCVariantType Enumeration

```
enum tagGCVariantType {
  GC_VAR_UNKNOWN,
  GC_VAR_BOOL,
  GC_VAR_INT,
  GC_VAR_FLOAT,
  GC_VAR_STRING,
  GC_VAR_LIST,
  GC_VAR_OBJECT
};
```

File

myx_gc_datatypes.h (see page 220)

Members

Members	Description
GC_VAR_UNKNOWN	The value type is unknown (e.g. because a property does not exist).
GC_VAR_BOOL	The value is a boolean.
GC_VAR_INT	The value is an integer number.
GC_VAR_FLOAT	The value is a floating point number.
GC_VAR_STRING	The value is a sequence of characters.
GC_VAR_LIST	The value is a collection of objects (e.g. layers).
GC_VAR_OBJECT	The value is an object with subproperties (e.g. a figure instance).

Remarks

A struct to transport certain base data that has no previously known type.

1.4.16 tagHitEntry Struct

```
struct tagHitEntry {
  CFigureInstance* Instance;
  double ZMin;
  double ZMax;
};
```

File

myx_gc_view.h (see page 234)

Remarks

Hit testing structures

1.4.17 tagImage Struct

```
struct tagImage {
  unsigned int Width;
  unsigned int Height;
  unsigned char* Data;
  TColorType ColorType;
  unsigned int Channels;
  GLenum Format;
};
```

File

myx_gc_utilities.h (see page 232)

Members

Members	Description
`unsigned int Width;`	The width of the image in pixels.
`unsigned int Height;`	The height of the image in pixels.
`unsigned char* Data;`	The image data.
`TColorType ColorType;`	Palette images are not supported.
`unsigned int Channels;`	Bytes per pixel.
`GLenum Format;`	OpenGL color format specifier. Set by the image user.

Remarks

This is record tagImage.

1.4.18 tagModifierKey Enumeration

```
enum tagModifierKey {
  GC_MODIFIER_NONE,
  GC_MODIFIER_ADD = 0x02,
  GC_MODIFIER_TOGGLE = 0x04,
  GC_MODIFIER_ALTERNATIVE = 0x08
};
```

File

myx_gc_datatypes.h (see page 220)

Members

Members	Description
`GC_MODIFIER_ADD = 0x02`	Add only modiifier (on Windows usually shift key)
`GC_MODIFIER_TOGGLE = 0x04`	Switch state modifier (on Windows usually ctrl key)
`GC_MODIFIER_ALTERNATIVE = 0x08`	Additional key (on Windows usually alt key)

Remarks

Modifier keys used when handling mouse input. Any value can be combined with other values.

1.4.19 tagMouseButton Enumeration

```
enum tagMouseButton {
  GC_MOUSE_BUTTON_NONE,
  GC_MOUSE_BUTTON_LEFT,
  GC_MOUSE_BUTTON_MIDDLE,
```

```
  GC_MOUSE_BUTTON_RIGHT
};
```

File

myx_gc_datatypes.h (see page 220)

Remarks

Used in mouse events to specify which mouse button is involved. For one button only systems like MacOS the left button inidicator is used for this (only) button.

1.4.20 tagMouseEvent Enumeration

```
enum tagMouseEvent {
  GC_MOUSE_DOWN,
  GC_MOUSE_UP,
  GC_MOUSE_MOVE
};
```

File

myx_gc_datatypes.h (see page 220)

Remarks

Indicates which mouse event is to be handled.

1.4.21 tagOccurence Enumeration

```
enum tagOccurence {
  GC_OCC_ONCE,
  GC_OCC_ZERO_OR_MORE,
  GC_OCC_ONE_OR_MORE
};
```

File

myx_gc_datatypes.h (see page 220)

Members

Members	Description
GC_OCC_ONCE	A single instance element (default).
GC_OCC_ZERO_OR_MORE	An element in a list that can appear in any number.
GC_OCC_ONE_OR_MORE	An element in a list that must exist at least once.

Remarks

Determines how often a figure element is allowed to appear.

1.4.22 tagPropertyID Enumeration

```
enum tagPropertyID {
  GC_PROPERTY_UNKNOWN,
  GC_PROPERTY_NAME,
  GC_PROPERTY_ID,
  GC_PROPERTY_WIDTH,
  GC_PROPERTY_HEIGHT,
  GC_PROPERTY_X,
  GC_PROPERTY_Y,
```

```
  GC_PROPERTY_Z,
  GC_PROPERTY_DESCRIPTION,
  GC_PROPERTY_ZOOMX,
  GC_PROPERTY_ZOOMY,
  GC_PROPERTY_COLOR,
  GC_PROPERTY_JITTER,
  GC_PROPERTY_ANGLE,
  GC_PROPERTY_VISIBLE,
  GC_PROPERTY_ENABLED,
  GC_PROPERTY_SELECTED,
  GC_PROPERTY_LAYOUT,
  GC_PROPERTY_RESIZABLE,
  GC_PROPERTY_EXPANDED,
  GC_PROPERTY_MIN_WIDTH,
  GC_PROPERTY_MIN_HEIGHT,
  GC_PROPERTY_MAX_WIDTH,
  GC_PROPERTY_MAX_HEIGHT,
  GC_PROPERTY_TEXT,
  GC_PROPERTY_FONT_FAMILY,
  GC_PROPERTY_FONT_SIZE,
  GC_PROPERTY_FONT_WEIGHT,
  GC_PROPERTY_FONT_STYLE,
  GC_PROPERTY_ALIGNMENT_VERTICAL,
  GC_PROPERTY_ALIGNMENT_HORIZONTAL,
  GC_PROPERTY_BIDI_MODE,
  GC_PROPERTY_OWNER
};
```

File

myx_gc_datatypes.h (see page 220)

Members

Members	Description
GC_PROPERTY_UNKNOWN	Simple properties.

Remarks

Identifier for properties. Used when parsing property specifications.

1.4.23 tagRRSelectionAction Enumeration

```
enum tagRRSelectionAction {
  GC_RRACTION_NONE,
  GC_RRACTION_SELECT,
  GC_RRACTION_SELECT_REMOVE,
  GC_RRACTION_TOGGLE
};
```

File

myx_gc_datatypes.h (see page 220)

Members

Members	Description
GC_RRACTION_NONE	Don't touch the selection state of any figure instance. Usually used for non-selecting rubber rectangless (e.g. for figure creation).
GC_RRACTION_SELECT	Always select figure instances if their bounding box intersects. Keep selected instances as their are if the do not intersect anymore. Usually used for rubber rectangles with pressed shift key modifier.
GC_RRACTION_SELECT_REMOVE	Select figure instances if they intersect, unselect those, which do not intersect. Most common rubber rectangle selection mode.
GC_RRACTION_TOGGLE	Revert the selection state of figure instances, which intersect. Don't touch the others. Usually used for rubber rectangles with pressed control key modifier.

Remarks

TRRSelectionAction (see page 190) (rubber rect selection action) determines how to manipulate the selection state of

figure instances with regard to their bounding box intersecting with the rubber rectangle.

1.4.24 tagRubberRectStyle Enumeration

```
enum tagRubberRectStyle {
  GC_RBSTYLE_SOLID_THIN,
  GC_RBSTYLE_SOLID_THICK,
  GC_RBSTYLE_DOTTED_THIN,
  GC_RBSTYLE_DOTTED_THICK,
  GC_RBSTYLE_BLENDED_CHECKERBOARD,
  GC_RBSTYLE_BLENDED_DIAGONALS
};
```

File

myx_gc_datatypes.h (see page 220)

Members

Members	Description
GC_RBSTYLE_SOLID_THIN	A simple black rectangle with a one pixel wide border.
GC_RBSTYLE_SOLID_THICK	A simple black rectangle with a 3 pixel wide border.
GC_RBSTYLE_DOTTED_THIN	A simple black rectangle with a one pixel wide dotted border.
GC_RBSTYLE_DOTTED_THICK	A simple black rectangle with a 3 pixel wide dotted border.
GC_RBSTYLE_BLENDED_CHECKERBOARD	A filled rectangle with a one pixel border and a translucent interior. The system's selection color is used. The interior is a checker board.
GC_RBSTYLE_BLENDED_DIAGONALS	A filled rectangle with a one pixel border and a translucent interior. The system's selection color is used. The interior consists of diagonal bands.

Remarks

TRubberRectStyle (see page 191) describes the look of the rubber rectangle in the selection layer.

1.4.25 tagSelectionEntry Struct

```
struct tagSelectionEntry {
  CFigureInstance* instance;
  bool dirty;
  TBoundingBox bounds;
};
```

File

myx_gc_layer.h (see page 226)

Remarks

Selection layer and associated structures

1.4.26 tagSystemColorEntry Struct

```
struct tagSystemColorEntry {
  char* name;
  int Value;
  GLubyte Color[4];
};
```

File

myx_gc_const.h (see page 219)

Remarks

This is record tagSystemColorEntry.

1.4.27 FontFileEntry Struct

```
typedef struct tagFontFileEntry {
  int useCount;
  string entries[2][2];
} FontFileEntry;
```

File

myx_gc_font_manager.h (see page 224)

Remarks

This is type FontFileEntry.

1.4.28 GC_PRIMITIVE Enumeration

```
typedef enum {
  GC_PRIMITIVE_UNKNOWN = -1,
  GC_PRIMITIVE_RECT,
  GC_PRIMITIVE_LINE,
  GC_PRIMITIVE_POLYLINE,
  GC_PRIMITIVE_POLYGON,
  GC_PRIMITIVE_CIRCLE,
  GC_PRIMITIVE_TEXT,
  GC_PRIMITIVE_TSPAN,
  GC_PRIMITIVE_GROUP,
  GC_PRIMITIVE_IMAGE
} GC_PRIMITIVE;
```

File

myx_gc_svgparser.cpp (see page 229)

1.4.29 TAction Struct

```
typedef struct tagAction {
  TActionType type;
  CActionParameters parameters;
} TAction;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

An action with associated parameters.

1.4.30 TActionType Enumeration

```
typedef enum tagActionType {
  GC_ACTION_NONE,
```

```
  GC_ACTION_TOGGLE,
  GC_ACTION_EXPAND,
  GC_ACTION_COLLAPSE,
  GC_ACTION_CHANGE_STYLE,
  GC_ACTION_DRAG_FIGURE,
  GC_ACTION_DRAG_ALL,
  GC_ACTION_RESIZE
} TActionType;
```

File

myx_gc_datatypes.h (see page 220)

Members

Members	Description
GC_ACTION_NONE	No action is assigned.
GC_ACTION_TOGGLE	Switch between expanded and collapsed state (scope: element).
GC_ACTION_EXPAND	Expand the element (scope: element).
GC_ACTION_COLLAPSE	Collapse the element (scope: element).
GC_ACTION_CHANGE_STYLE	Switch to a new style (scope: element).
GC_ACTION_DRAG_FIGURE	Start dragging of the associated figure instance (scope: a single figure instance).
GC_ACTION_DRAG_ALL	Start dragging of all selected figure instances (scope: all selected figure instances on the associated layer).
GC_ACTION_RESIZE	The figure is resized.

Remarks

Determines the type of action possible with a particular figure instance/element.

1.4.31 TAlignment Enumeration

```
typedef enum {
  GC_ALIGN_LEFT_TOP,
  GC_ALIGN_CENTER,
  GC_ALIGN_RIGHT_BOTTOM
} TAlignment;
```

File

myx_gc_figure.h (see page 222)

Remarks

Text alignment constants.

1.4.32 TBidiMode Enumeration

```
typedef enum tagBidiMode {
  GC_BIDI_LEFT_TO_RIGHT,
  GC_BIDI_RIGHT_TO_LEFT
} TBidiMode;
```

File

myx_gc_datatypes.h (see page 220)

Members

Members	Description
GC_BIDI_LEFT_TO_RIGHT	Standard directionality (most languages).
GC_BIDI_RIGHT_TO_LEFT	Used for arabic and hebrew text.

Remarks

Bidirectional mode

1.4.33 TBoundingBox Struct

```
typedef struct tagBoundingBox {
  TVertex upper;
  TVertex lower;
} TBoundingBox;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is type TBoundingBox.

1.4.34 TColorEntry Struct

```
typedef struct tagColorEntry {
  char* name;
  GLubyte Color[4];
} TColorEntry;
```

File

myx_gc_const.h (see page 219)

1.4.35 TColorType Enumeration

```
typedef enum tagColorType {
  COLOR_TYPE_PALETTE = PNG_COLOR_TYPE_PALETTE,
  COLOR_TYPE_GRAY = PNG_COLOR_TYPE_GRAY,
  COLOR_TYPE_GRAY_ALPHA = PNG_COLOR_TYPE_GRAY_ALPHA,
  COLOR_TYPE_RGB = PNG_COLOR_TYPE_RGB,
  COLOR_TYPE_RGB_ALPHA = PNG_COLOR_TYPE_RGB_ALPHA
} TColorType;
```

File

myx_gc_utilities.h (see page 232)

1.4.36 TConnectionDirection Enumeration

```
typedef enum tagConnectionDirection {
  GC_DIR_NONE,
  GC_DIR_NORTH,
  GC_DIR_EAST,
  GC_DIR_SOUTH,
  GC_DIR_WEST
} TConnectionDirection;
```

File

myx_gc_connection.h (see page 219)

Remarks

This is type TConnectionDirection.

1.4.37 TConnectionLineStyle Enumeration

```
typedef enum tagConnectionLineStyle {
  GC_CONNECTION_STYLE_SOLID,
  GC_CONNECTION_STYLE_DOTTED
} TConnectionLineStyle;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

Style for a connection line.

1.4.38 TConstraints Struct

```
typedef struct tagConstraints {
  float maxHeight;
  float maxWidth;
  float minHeight;
  float minWidth;
} TConstraints;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is type TConstraints.

1.4.39 TContainerID Enumeration

```
typedef enum tagContainerID {
  GC_CONTAINER_UNKNOWN,
  GC_CONTAINER_LAYERS,
  GC_CONTAINER_FEEDBACK,
  GC_CONTAINER_FIGURE_INSTANCES,
  GC_CONTAINER_FIGURE_CONTENT,
  GC_CONTAINER_VIEWS,
  GC_CONTAINER_MODEL,
  GC_CONTAINER_STYLE,
  GC_CONTAINER_STYLES,
  GC_CONTAINER_LAYOUTS,
  GC_CONTAINER_FIGURE,
  GC_CONTAINER_FIGURES,
  GC_CONTAINER_SCALING,
  GC_CONTAINER_TRANSLATION,
  GC_CONTAINER_ROTATION,
  GC_CONTAINER_GROUPS,
  GC_CONTAINER_CHILDREN,
  GC_CONTAINER_CAPTION,
  GC_CONTAINER_CONTENT
} TContainerID;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

Identifier for containers. Used when parsing pathes for properties.

1.4.40 TFeedbackInfo Enumeration

```
typedef enum tagFeedbackInfo {
  GC_FI_NONE,
  GC_FI_ON_OBJECT,
  GC_FI_NORTH,
  GC_FI_NORTH_EAST,
  GC_FI_EAST,
  GC_FI_SOUTH_EAST,
  GC_FI_SOUTH,
  GC_FI_SOUTH_WEST,
  GC_FI_WEST,
  GC_FI_NORTH_WEST
} TFeedbackInfo;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is type TFeedbackInfo.

1.4.41 TFigureElementLayout Enumeration

```
typedef enum tagFigureElementLayout {
  GC_LAYOUT_ROW,
  GC_LAYOUT_COLUMN
} TFigureElementLayout;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

Layout variants for a figure element.

1.4.42 TFigureElementResize Enumeration

```
typedef enum tagFigureElementResize {
  GC_RESIZE_NONE,
  GC_RESIZE_HORIZONTAL_ONLY,
  GC_RESIZE_VERTICAL_ONLY,
  GC_RESIZE_ALL
} TFigureElementResize;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

Resize variants for a figure element.

1.4.43 TGCChangeReason Enumeration

```
typedef enum tagChangeReason {
  GC_CHANGE_SELECTION_ADD,
  GC_CHANGE_SELECTION_CLEAR,
  GC_CHANGE_SELECTION_REMOVE,
  GC_CHANGE_SELECTION_CHANGE,
  GC_CHANGE_CANVAS_REFRESH,
  GC_CHANGE_CANVAS_PROPERTY,
  GC_CHANGE_CANVAS_ADD_VIEW,
  GC_CHANGE_CANVAS_ADD_LAYER,
  GC_CHANGE_CANVAS_SWITCH_VIEW,
  GC_CHANGE_CANVAS_REMOVE_VIEW,
  GC_CHANGE_CANVAS_REMOVE_LAYER,
  GC_CHANGE_CANVAS_CLEAR_CONTENT,
  GC_CHANGE_CANVAS_CLEAR_LAYOUTS,
  GC_CHANGE_CANVAS_CLEAR_STYLES,
  GC_CHANGE_MODEL_PROPERTY,
  GC_CHANGE_MODEL_ADD_FIGURE,
  GC_CHANGE_MODEL_REMOVE_FIGURE,
  GC_CHANGE_MODEL_ADD_STYLE,
  GC_CHANGE_MODEL_REMOVE_STYLE,
  GC_CHANGE_CAPTION_PROPERTY,
  GC_CHANGE_ELEMENT_PROPERTY,
  GC_CHANGE_ELEMENT_ADD_CHILD,
  GC_CHANGE_FIGURE_PROPERTY,
  GC_CHANGE_FIGURE_EXCHANGE,
  GC_CHANGE_FINSTANCE_PROPERTY,
  GC_CHANGE_VIEW_PROPERTY,
  GC_CHANGE_VIEW_ADD_LAYER,
  GC_CHANGE_VIEW_REMOVE_LAYER,
  GC_CHANGE_VIEW_CLEAR,
  GC_CHANGE_LAYER_CLEAR,
  GC_CHANGE_LAYER_VISIBILITY,
  GC_CHANGE_LAYER_PROPERTY,
  GC_CHANGE_LAYER_ADD_INSTANCE,
  GC_CHANGE_LAYER_REMOVE_INSTANCE,
  GC_CHANGE_LAYER_ADD_GROUP,
  GC_CHANGE_LAYER_REMOVE_GROUP,
  GC_CHANGE_CONNECTION_INSTANCE
} TGCChangeReason;
```

File

myx_gc_datatypes.h (see page 220)

Members

Members	Description
GC_CHANGE_SELECTION_ADD	One or more figure instances were added to the current selection.
GC_CHANGE_SELECTION_CLEAR	The current selection was cleared.
GC_CHANGE_SELECTION_REMOVE	One or more figure instances were removed from the current selection.
GC_CHANGE_SELECTION_CHANGE	One or more figure instances were added to or removed from the current selection.
GC_CHANGE_CANVAS_REFRESH	Used to indicate that the view must update the visual representation.
GC_CHANGE_CANVAS_PROPERTY	The value of a property has been changed.
GC_CHANGE_CANVAS_ADD_VIEW	A new view was added.
GC_CHANGE_CANVAS_ADD_LAYER	A new layer was added.
GC_CHANGE_CANVAS_SWITCH_VIEW	Another view was activated.
GC_CHANGE_CANVAS_REMOVE_VIEW	A view was removed.
GC_CHANGE_CANVAS_REMOVE_LAYER	A layer was removed.
GC_CHANGE_CANVAS_CLEAR_CONTENT	All figures have been removed.
GC_CHANGE_CANVAS_CLEAR_LAYOUTS	All layout definitions have been removed.
GC_CHANGE_CANVAS_CLEAR_STYLES	All styles have been removed.
GC_CHANGE_MODEL_PROPERTY	The value of a property has been changed.

GC_CHANGE_MODEL_ADD_FIGURE	A new figure was added.
GC_CHANGE_MODEL_REMOVE_FIGURE	A figure was removed.
GC_CHANGE_MODEL_ADD_STYLE	A new style was added.
GC_CHANGE_MODEL_REMOVE_STYLE	A style was removed.
GC_CHANGE_CAPTION_PROPERTY	The value of a property has been changed.
GC_CHANGE_ELEMENT_PROPERTY	The value of a property has been changed.
GC_CHANGE_ELEMENT_ADD_CHILD	The value of a property has been changed.
GC_CHANGE_FIGURE_PROPERTY	The value of a property has been changed.
GC_CHANGE_FIGURE_EXCHANGE	A figure is about to be replaced by another one.
GC_CHANGE_FINSTANCE_PROPERTY	The value of a property has been changed.
GC_CHANGE_VIEW_PROPERTY	The value of a property has been changed.
GC_CHANGE_VIEW_ADD_LAYER	A new layer was added to a view.
GC_CHANGE_VIEW_REMOVE_LAYER	A layer was removed.
GC_CHANGE_VIEW_CLEAR	The view was cleared.
GC_CHANGE_LAYER_CLEAR	All figure instances on the layer are removed.
GC_CHANGE_LAYER_VISIBILITY	The visibility of a layer has been changed.
GC_CHANGE_LAYER_PROPERTY	The value of a property has been changed.
GC_CHANGE_LAYER_ADD_INSTANCE	A new figure instance was added.
GC_CHANGE_LAYER_REMOVE_INSTANCE	A figure instance was removed.
GC_CHANGE_LAYER_ADD_GROUP	A new group was added to a view.
GC_CHANGE_LAYER_REMOVE_GROUP	A group was removed.
GC_CHANGE_CONNECTION_INSTANCE	Connections

Remarks

This is type TGCChangeReason.

1.4.44 TGCError Enumeration

```
typedef enum tagGCError {
  GC_NO_ERROR = 0,
  GC_CANT_OPEN_FILE,
  GC_XML_PARSE_ERROR,
  GC_XML_INVALID_DOCUMENT,
  GC_XML_EMPTY_DOCUMENT,
  GC_OBJECT_NOT_FOUND,
  GC_CANT_READ_FROM_FILE,
  GC_CHARSET_CONVERSION_ERROR,
  GC_CHARSET_WRONG_CHARSET_SPECIFIED
} TGCError;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is type TGCError.

1.4.45 TGCVariant Struct

```
typedef struct tagGCVariant {
  TGCVariantType type;
  bool b;
  int i;
  float f;
  string s;
  CGCBase* reference;
} TGCVariant;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is type TGCVariant.

1.4.46 TGCVariantType Enumeration

```
typedef enum tagGCVariantType {
  GC_VAR_UNKNOWN,
  GC_VAR_BOOL,
  GC_VAR_INT,
  GC_VAR_FLOAT,
  GC_VAR_STRING,
  GC_VAR_LIST,
  GC_VAR_OBJECT
} TGCVariantType;
```

File

myx_gc_datatypes.h (see page 220)

Members

Members	Description
GC_VAR_UNKNOWN	The value type is unknown (e.g. because a property does not exist).
GC_VAR_BOOL	The value is a boolean.
GC_VAR_INT	The value is an integer number.
GC_VAR_FLOAT	The value is a floating point number.
GC_VAR_STRING	The value is a sequence of characters.
GC_VAR_LIST	The value is a collection of objects (e.g. layers).
GC_VAR_OBJECT	The value is an object with subproperties (e.g. a figure instance).

Remarks

A struct to transport certain base data that has no previously known type.

1.4.47 TGCViewport Struct

```
typedef struct tagViewport {
  int left, top, width, height;
} TGCViewport;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

ifdef _WINDOWS (see page 210)

1.4.48 THitEntry Struct

```
typedef struct tagHitEntry {
  CFigureInstance* Instance;
  double ZMin;
  double ZMax;
} THitEntry;
```

File

myx_gc_view.h (see page 234)

Remarks

Hit testing structures

1.4.49 TImage Struct

```
typedef struct tagImage {
  unsigned int Width;
  unsigned int Height;
  unsigned char* Data;
  TColorType ColorType;
  unsigned int Channels;
  GLenum Format;
} TImage;
```

File

myx_gc_utilities.h (see page 232)

Members

Members	Description
`unsigned int Width;`	The width of the image in pixels.
`unsigned int Height;`	The height of the image in pixels.
`unsigned char* Data;`	The image data.
`TColorType ColorType;`	Palette images are not supported.
`unsigned int Channels;`	Bytes per pixel.
`GLenum Format;`	OpenGL color format specifier. Set by the image user.

Remarks

This is type TImage.

1.4.50 TModifierKey Enumeration

```
typedef enum tagModifierKey {
  GC_MODIFIER_NONE,
  GC_MODIFIER_ADD = 0x02,
  GC_MODIFIER_TOGGLE = 0x04,
  GC_MODIFIER_ALTERNATIVE = 0x08
} TModifierKey;
```

File

myx_gc_datatypes.h (see page 220)

Members

Members	Description
`GC_MODIFIER_ADD = 0x02`	Add only modiifier (on Windows usually shift key)
`GC_MODIFIER_TOGGLE = 0x04`	Switch state modifier (on Windows usually ctrl key)
`GC_MODIFIER_ALTERNATIVE = 0x08`	Additional key (on Windows usually alt key)

Remarks

Modifier keys used when handling mouse input. Any value can be combined with other values.

1.4.51 TMouseButton Enumeration

```
typedef enum tagMouseButton {
  GC_MOUSE_BUTTON_NONE,
  GC_MOUSE_BUTTON_LEFT,
  GC_MOUSE_BUTTON_MIDDLE,
  GC_MOUSE_BUTTON_RIGHT
} TMouseButton;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

Used in mouse events to specify which mouse button is involved. For one button only systems like MacOS the left button inidicator is used for this (only) button.

1.4.52 TMouseEvent Enumeration

```
typedef enum tagMouseEvent {
  GC_MOUSE_DOWN,
  GC_MOUSE_UP,
  GC_MOUSE_MOVE
} TMouseEvent;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

Indicates which mouse event is to be handled.

1.4.53 TOccurence Enumeration

```
typedef enum tagOccurence {
  GC_OCC_ONCE,
  GC_OCC_ZERO_OR_MORE,
  GC_OCC_ONE_OR_MORE
} TOccurence;
```

File

myx_gc_datatypes.h (see page 220)

Members

Members	Description
GC_OCC_ONCE	A single instance element (default).
GC_OCC_ZERO_OR_MORE	An element in a list that can appear in any number.
GC_OCC_ONE_OR_MORE	An element in a list that must exist at least once.

Remarks

Determines how often a figure element is allowed to appear.

1.4.54 TPropertyID Enumeration

```
typedef enum tagPropertyID {
  GC_PROPERTY_UNKNOWN,
  GC_PROPERTY_NAME,
  GC_PROPERTY_ID,
  GC_PROPERTY_WIDTH,
  GC_PROPERTY_HEIGHT,
  GC_PROPERTY_X,
  GC_PROPERTY_Y,
  GC_PROPERTY_Z,
  GC_PROPERTY_DESCRIPTION,
  GC_PROPERTY_ZOOMX,
  GC_PROPERTY_ZOOMY,
  GC_PROPERTY_COLOR,
  GC_PROPERTY_JITTER,
  GC_PROPERTY_ANGLE,
  GC_PROPERTY_VISIBLE,
  GC_PROPERTY_ENABLED,
  GC_PROPERTY_SELECTED,
  GC_PROPERTY_LAYOUT,
  GC_PROPERTY_RESIZABLE,
  GC_PROPERTY_EXPANDED,
  GC_PROPERTY_MIN_WIDTH,
  GC_PROPERTY_MIN_HEIGHT,
  GC_PROPERTY_MAX_WIDTH,
  GC_PROPERTY_MAX_HEIGHT,
  GC_PROPERTY_TEXT,
  GC_PROPERTY_FONT_FAMILY,
  GC_PROPERTY_FONT_SIZE,
  GC_PROPERTY_FONT_WEIGHT,
  GC_PROPERTY_FONT_STYLE,
  GC_PROPERTY_ALIGNMENT_VERTICAL,
  GC_PROPERTY_ALIGNMENT_HORIZONTAL,
  GC_PROPERTY_BIDI_MODE,
  GC_PROPERTY_OWNER
} TPropertyID;
```

File

myx_gc_datatypes.h (see page 220)

Members

Members	Description
GC_PROPERTY_UNKNOWN	Simple properties.

Remarks

Identifier for properties. Used when parsing property specifications.

1.4.55 TRRSelectionAction Enumeration

```
typedef enum tagRRSelectionAction {
  GC_RRACTION_NONE,
  GC_RRACTION_SELECT,
  GC_RRACTION_SELECT_REMOVE,
  GC_RRACTION_TOGGLE
} TRRSelectionAction;
```

File

myx_gc_datatypes.h (see page 220)

Members

Members	Description
GC_RRACTION_NONE	Don't touch the selection state of any figure instance. Usually used for non-selecting rubber rectangless (e.g. for figure creation).
GC_RRACTION_SELECT	Always select figure instances if their bounding box intersects. Keep selected instances as their are if the do not intersect anymore. Usually used for rubber rectangles with pressed shift key modifier.
GC_RRACTION_SELECT_REMOVE	Select figure instances if they intersect, unselect those, which do not intersect. Most common rubber rectangle selection mode.
GC_RRACTION_TOGGLE	Revert the selection state of figure instances, which intersect. Don't touch the others. Usually used for rubber rectangles with pressed control key modifier.

Remarks

TRRSelectionAction (rubber rect selection action) determines how to manipulate the selection state of figure instances with regard to their bounding box intersecting with the rubber rectangle.

1.4.56 TRubberRectStyle Enumeration

```
typedef enum tagRubberRectStyle {
  GC_RBSTYLE_SOLID_THIN,
  GC_RBSTYLE_SOLID_THICK,
  GC_RBSTYLE_DOTTED_THIN,
  GC_RBSTYLE_DOTTED_THICK,
  GC_RBSTYLE_BLENDED_CHECKERBOARD,
  GC_RBSTYLE_BLENDED_DIAGONALS
} TRubberRectStyle;
```

File

myx_gc_datatypes.h (see page 220)

Members

Members	Description
GC_RBSTYLE_SOLID_THIN	A simple black rectangle with a one pixel wide border.
GC_RBSTYLE_SOLID_THICK	A simple black rectangle with a 3 pixel wide border.
GC_RBSTYLE_DOTTED_THIN	A simple black rectangle with a one pixel wide dotted border.
GC_RBSTYLE_DOTTED_THICK	A simple black rectangle with a 3 pixel wide dotted border.
GC_RBSTYLE_BLENDED_CHECKERBOARD	A filled rectangle with a one pixel border and a translucent interior. The system's selection color is used. The interior is a checker board.
GC_RBSTYLE_BLENDED_DIAGONALS	A filled rectangle with a one pixel border and a translucent interior. The system's selection color is used. The interior consists of diagonal bands.

Remarks

TRubberRectStyle describes the look of the rubber rectangle in the selection layer.

1.4.57 TSelectionEntry Struct

```
typedef struct tagSelectionEntry {
  CFigureInstance* instance;
  bool dirty;
  TBoundingBox bounds;
} TSelectionEntry;
```

File

myx_gc_layer.h (see page 226)

Remarks

Selection layer and associated structures

1.4.58 TSystemColorEntry Struct

```
typedef struct tagSystemColorEntry {
  char* name;
  int Value;
  GLubyte Color[4];
} TSystemColorEntry;
```

File

myx_gc_const.h (see page 219)

Remarks

This is type TSystemColorEntry.

1.4.59 TVertex Struct

```
typedef struct tagVertex {
  float x;
  float y;
  float z;
  float w;
} TVertex;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

Some geometric data types.

1.5 Types

1.5.1 CActionParameters Type

```
typedef vector<wstring> CActionParameters;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is type CActionParameters.

1.5.2 CActions Type

```
typedef vector<TAction*> CActions;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is type CActions.

1.5.3 CColorMap Type

```
typedef map<string, unsigned char*> CColorMap;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is type CColorMap.

1.5.4 CColorMapIterator Type

```
typedef map<string, unsigned char*>::const_iterator CColorMapIterator;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is type CColorMapIterator.

1.5.5 CColorMapPair Type

```
typedef pair<string, unsigned char*> CColorMapPair;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is type CColorMapPair.

1.5.6 CConnectionInstanceList Type

```
typedef set<CConnectionInstance*> CConnectionInstanceList;
```

File

myx_gc_layer.h (see page 226)

Remarks

Connection layer and associated structures

1.5.7 CConnectionList Type

```
typedef vector<CConnection*> CConnectionList;
```

File

myx_gc_model.h (see page 228)

Remarks

This is type CConnectionList.

1.5.8 CElementList Type

```
typedef vector<CFigureElement*> CElementList;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is type CElementList.

1.5.9 CElementTemplateList Type

```
typedef vector<CFigureElementTemplate*> CElementTemplateList;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is type CElementTemplateList.

1.5.10 CFigureConnectionList Type

```
typedef hash_map<CFigureInstance*, CConnectionInstanceList> CFigureConnectionList;
```

File

myx_gc_layer.h (see page 226)

Remarks

This is type CFigureConnectionList.

1.5.11 CFigureElementMap Type

```
typedef hash_map<wstring, CFigureElement*> CFigureElementMap;
```

File

myx_gc_figure.h (see page 222)

1.5.12 CFigureInstances Type

```
typedef vector<CFigureInstance*> CFigureInstances;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is type CFigureInstances.

1.5.13 CFigureList Type

```
typedef vector<CFigure*> CFigureList;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is type CFigureList.

1.5.14 CGCListenerIterator Type

```
typedef set<CGCListener*>::iterator CGCListenerIterator;
```

File

myx_gc_base.h (see page 217)

Remarks

This is type CGCListenerIterator.

1.5.15 CGCListeners Type

```
typedef set<CGCListener*> CGCListeners;
```

File

myx_gc_base.h (see page 217)

Remarks

This is type CGCListeners.

1.5.16 CLayers Type

```
typedef vector<CLayer*> CLayers;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is type CLayers.

1.5.17 CLayoutList Type

```
typedef multimap<wstring, CFigureTemplate*> CLayoutList;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is type CLayoutList.

1.5.18 CLayoutPair Type

```
typedef pair<wstring, CFigureTemplate*> CLayoutPair;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is type CLayoutPair.

1.5.19 CSelection Type

```
typedef map<CFigureInstance*, TSelectionEntry*> CSelection;
```

File

myx_gc_layer.h (see page 226)

Remarks

This is type CSelection.

1.5.20 CSelectionIterator Type

```
typedef map<CFigureInstance*, TSelectionEntry*>::iterator CSelectionIterator;
```

File

myx_gc_layer.h (see page 226)

Remarks

This is type CSelectionIterator.

1.5.21 CSelectionIteratorReverse Type

```
typedef map<CFigureInstance*, TSelectionEntry*>::reverse_iterator CSelectionIteratorReverse;
```

File

myx_gc_layer.h (see page 226)

Remarks

This is type CSelectionIteratorReverse.

1.5.22 CStyleList Type

```
typedef hash_map<wstring, CGCStyle*> CStyleList;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is type CStyleList.

1.5.23 CTextureIterator Type

```
typedef map<string, CGCTexture*>::iterator CTextureIterator;
```

File

myx_gc_texture.h (see page 230)

Remarks

This is type CTextureIterator.

1.5.24 CTextures Type

```
typedef map<string, CGCTexture*> CTextures;
```

File

myx_gc_texture.h (see page 230)

Remarks

The list of textures is an associated list of names and CTexture classes.

1.5.25 CVertexVector Type

```
typedef vector<TVertex> CVertexVector;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is type CVertexVector.

1.5.26 CViews Type

```
typedef vector<CGCView*> CViews;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is type CViews.

1.5.27 FontFiles Type

```
typedef hash_map<string, FontFileEntry*> FontFiles;
```

File

myx_gc_font_manager.h (see page 224)

Remarks

This is type FontFiles.

1.5.28 Fonts Type

```
typedef hash_map<string, FTFont*> Fonts;
```

File

myx_gc_font_manager.h (see page 224)

1.5.29 GCContext Type

```
typedef GLXContext GCContext;
```

File

myx_gc_datatypes.h (see page 220)

Remarks

An opaque handle to a rendering context. Must be provided by the viewer.

1.5.30 THitEntries Type

```
typedef vector<CFigureInstance*> THitEntries;
```

File

myx_gc_view.h (see page 234)

Remarks

This is type THitEntries.

1.5.31 THitEntryIterator Type

```
typedef vector<CFigureInstance*>::iterator THitEntryIterator;
```

File

myx_gc_view.h (see page 234)

Remarks

This is type THitEntryIterator.

1.5.32 TLODList Type

```
typedef vector<string> TLODList;
```

File

myx_gc_texture.h (see page 230)

Remarks

A list of texture names each with the level-of-detail they are associated. The index in the vector is also the LOD they stand for.

1.5.33 TMatrix Type

```
typedef GLfloat TMatrix[16];
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is type TMatrix.

1.6 Variables

1.6.1 actionLookup Variable

```
static hash_map<wstring, TActionType> actionLookup;
```

File

myx_gc_figure_parser.cpp (see page 223)

Remarks

This is variable actionLookup.

1.6.2 alignmentLookup Variable

```
static map<string, TAlignment> alignmentLookup;
```

File

myx_gc_figure_parser.cpp (see page 223)

Remarks

This is variable alignmentLookup.

1.6.3 Colors Variable

```
static TColorEntry Colors[] = { {"aliceblue", {240, 248, 255, 255}}, {"antiquewhite", {250,
235, 215, 255}}, {"aqua", { 0, 255, 255, 255}}, {"aquamarine", {127, 255, 212, 255}},
{"azure", {240, 255, 255, 255}}, {"beige", {245, 245, 220, 255}}, {"bisque", {255, 228,
196, 255}}, {"black", { 0, 0, 0, 255}}, {"blanchedalmond", {255, 235, 205, 255}}, {"blue",
{ 0, 0, 255, 255}}, {"blueviolet", {138, 43, 226, 255}}, {"brown", {165, 42, 42, 255}},
{"burlywood", {222, 184, 135, 255}}, {"cadetblue", { 95, 158, 160, 255}}, {"chartreuse",
{127, 255, 0, 255}}, {"chocolate", {210, 105, 30, 255}}, {"coral", {255, 127, 80, 255}},
{"cornflowerblue", {100, 149, 237, 255}}, {"cornsilk", {255, 248, 220, 255}}, {"crimson",
{220, 20, 60, 255}}, {"cyan", { 0, 255, 255, 255}}, {"darkblue", { 0, 0, 139, 255}},
{"darkcyan", { 0, 139, 139, 255}}, {"darkgoldenrod", {184, 134, 11, 255}}, {"darkgray",
{169, 169, 169, 255}}, {"darkgreen", { 0, 100, 0, 255}}, {"darkgrey", {169, 169, 169,
255}}, {"darkkhaki", {189, 183, 107, 255}}, {"darkmagenta", {139, 0, 139, 255}},
{"darkolivegreen", { 85, 107, 47, 255}}, {"darkorange", {255, 140, 0, 255}}, {"darkorchid",
{153, 50, 204, 255}}, {"darkred", {139, 0, 0, 255}}, {"darksalmon", {233, 150, 122, 255}},
{"darkseagreen", {143, 188, 143, 255}}, {"darkslateblue", { 72, 61, 139, 255}},
{"darkslategray", { 47, 79, 79, 255}}, {"darkslategrey", { 47, 79, 79, 255}},
{"darkturquoise", { 0, 206, 209, 255}}, {"darkviolet", {148, 0, 211, 255}}, {"deeppink",
{255, 20, 147, 255}}, {"deepskyblue", { 0, 191, 255, 255}}, {"dimgray", {105, 105, 105,
255}}, {"dimgrey", {105, 105, 105, 255}}, {"dodgerblue", { 30, 144, 255, 255}},
{"firebrick", {178, 34, 34, 255}}, {"floralwhite", {255, 250, 240, 255}}, {"forestgreen", {
34, 139, 34, 255}}, {"fuchsia", {255, 0, 255, 255}}, {"gainsboro", {220, 220, 220, 255}},
{"ghostwhite", {248, 248, 255, 255}}, {"gold", {255, 215, 0, 255}}, {"goldenrod", {218,
165, 32, 255}}, {"gray", {128, 128, 128, 255}}, {"grey", {128, 128, 128, 255}}, {"green", {
0, 128, 0, 255}}, {"greenyellow", {173, 255, 47, 255}}, {"honeydew", {240, 255, 240, 255}},
{"hotpink", {255, 105, 180, 255}}, {"indianred", {205, 92, 92, 255}}, {"indigo", { 75, 0,
130, 255}}, {"ivory", {255, 255, 240, 255}}, {"khaki", {240, 230, 140, 255}}, {"lavender",
{230, 230, 250, 255}}, {"lavenderblush", {255, 240, 245, 255}}, {"lawngreen", {124, 252, 0,
```

```
255}}, {"lemonchiffon", {255, 250, 205, 255}}, {"lightblue", {173, 216, 230, 255}},
{"lightcoral", {240, 128, 128, 255}}, {"lightcyan", {224, 255, 255, 255}},
{"lightgoldenrodyellow", {250, 250, 210, 255}}, {"lightgray", {211, 211, 211, 255}},
{"lightgreen", {144, 238, 144, 255}}, {"lightgrey", {211, 211, 211, 255}}, {"lightpink",
{255, 182, 193, 255}}, {"lightsalmon", {255, 160, 122, 255}}, {"lightseagreen", { 32, 178,
170, 255}}, {"lightskyblue", {135, 206, 250, 255}}, {"lightslategray", {119, 136, 153,
255}}, {"lightslategrey", {119, 136, 153, 255}}, {"lightsteelblue", {176, 196, 222, 255}},
{"lightyellow", {255, 255, 224, 255}}, {"lime", { 0, 255, 0, 255}}, {"limegreen", { 50,
205, 50, 255}}, {"linen", {250, 240, 230, 255}}, {"magenta", {255, 0, 255, 255}},
{"maroon", {128, 0, 0, 255}}, {"mediumaquamarine", {102, 205, 170, 255}}, {"mediumblue", {
0, 0, 205, 255}}, {"mediumorchid", {186, 85, 211, 255}}, {"mediumpurple", {147, 112, 219,
255}}, {"mediumseagreen", { 60, 179, 113, 255}}, {"mediumslateblue", {123, 104, 238, 255}},
{"mediumspringgreen", { 0, 250, 154, 255}}, {"mediumturquoise", { 72, 209, 204, 255}},
{"mediumvioletred", {199, 21, 133, 255}}, {"midnightblue", { 25, 25, 112, 255}},
{"mintcream", {245, 255, 250, 255}}, {"mistyrose", {255, 228, 225, 255}}, {"moccasin",
{255, 228, 181, 255}}, {"navajowhite", {255, 222, 173, 255}}, {"navy", { 0, 0, 128, 255}},
{"oldlace", {253, 245, 230, 255}}, {"olive", {128, 128, 0, 255}}, {"olivedrab", {107, 142,
35, 255}}, {"orange", {255, 165, 0, 255}}, {"orangered", {255, 69 , 0, 255}}, {"orchid",
{218, 112, 214, 255}}, {"palegoldenrod", {238, 232, 170, 255}}, {"palegreen", {152, 251,
152, 255}}, {"paleturquoise", {175, 238, 238, 255}}, {"palevioletred", {219, 112, 147,
255}}, {"papayawhip", {255, 239, 213, 255}}, {"peachpuff", {255, 218, 185, 255}}, {"peru",
{205, 133, 63, 255}}, {"pink", {255, 192, 203, 255}}, {"plum", {221, 160, 221, 255}},
{"powderblue", {176, 224, 230, 255}}, {"purple", {128, 0, 128, 255}}, {"red", {255, 0, 0,
255}}, {"rosybrown", {188, 143, 143, 255}}, {"royalblue", { 65, 105, 225, 255}},
{"saddlebrown", {139, 69, 19, 255}}, {"salmon", {250, 128, 114, 255}}, {"sandybrown", {244,
164, 96, 255}}, {"seagreen", { 46, 139, 87, 255}}, {"seashell", {255, 245, 238, 255}},
{"sienna", {160, 82, 45, 255}}, {"silver", {192, 192, 192, 255}}, {"skyblue", {135, 206,
235, 255}}, {"slateblue", {106, 90, 205, 255}}, {"slategray", {112, 128, 144, 255}},
{"slategrey", {112, 128, 144, 255}}, {"snow", {255, 250, 250, 255}}, {"springgreen", { 0,
255, 127, 255}}, {"steelblue", { 70, 130, 180, 255}}, {"tan", {210, 180, 140, 255}},
{"teal", { 0, 128, 128, 255}}, {"thistle", {216, 191, 216, 255}}, {"tomato", {255, 99, 71,
255}}, {"turquoise", { 64, 224, 208, 255}}, {"violet", {238, 130, 238, 255}}, {"wheat",
{245, 222, 179, 255}}, {"white", {255, 255, 255, 255}}, {"whitesmoke", {245, 245, 245,
255}}, {"yellow", {255, 255, 0, 255}}, {"yellowgreen", {154, 205, 50, 255}} };
```

File

myx_gc_const.h (see page 219)

Remarks

This is variable Colors.

1.6.4 DefaultFontSize Variable

```
const int DefaultFontSize = 20;
```

File

myx_gc_const.h (see page 219)

Remarks

20pt

1.6.5 DefaultLayout Variable

```
const string DefaultLayout = "column";
```

File

myx_gc_figure.h (see page 222)

Remarks

The layout to be used for figure elements without any given layout.

1.6.6 DefaultResize Variable

```
const string DefaultResize = "none";
```

File

myx_gc_figure.h (see page 222)

Remarks

By default figure elements cannot be resized.

1.6.7 DefaultTextureDimensions Variable

```
const int DefaultTextureDimensions = 2;
```

File

myx_gc_texture.h (see page 230)

Remarks

This is variable DefaultTextureDimensions.

1.6.8 Identity Variable

```
const TMatrix Identity = { 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1 };
```

File

myx_gc_const.h (see page 219)

Remarks

This is variable Identity.

1.6.9 internalManager Variable

```
static CFontManager* internalManager;
```

File

myx_gc_font_manager.cpp (see page 224)

Remarks

CFontManager (see page 71)

1.6.10 InternalTextureManager Variable

```
CTextureManager InternalTextureManager;
```

File

myx_gc_texture.cpp (see page 230)

Remarks

Singleton texture manager instance.

1.6.11 layoutLookup Variable

```
static map<string, TFigureElementLayout> layoutLookup;
```

File

myx_gc_figure_parser.cpp (see page 223)

Remarks

This is variable layoutLookup.

1.6.12 lockCount Variable

```
static int lockCount = 0;
```

File

myx_gc_font_manager.cpp (see page 224)

Remarks

This is variable lockCount.

1.6.13 predefinedColors Variable

```
static CColorMap predefinedColors;
```

File

myx_gc_utilities.cpp (see page 231)

1.6.14 resizeLookup Variable

```
static map<string, TFigureElementResize> resizeLookup;
```

File

myx_gc_figure_parser.cpp (see page 223)

Remarks

This is variable resizeLookup.

1.6.15 SystemColors Variable

```
static TSystemColorEntry SystemColors[] = { {"ActiveBorder", COLOR_ACTIVEBORDER},
{"ActiveCaption", COLOR_ACTIVECAPTION}, {"AppWorkspace", COLOR_APPWORKSPACE},
{"Background", COLOR_BACKGROUND}, {"ButtonFace", COLOR_BTNFACE}, {"ButtonHighlight",
COLOR_BTNHIGHLIGHT}, {"ButtonShadow", COLOR_BTNSHADOW}, {"ButtonText", COLOR_BTNTEXT},
{"CaptionText", COLOR_CAPTIONTEXT}, {"GrayText", COLOR_GRAYTEXT}, {"Highlight",
COLOR_HIGHLIGHT}, {"HighlightText", COLOR_HIGHLIGHTTEXT}, {"Hotlight", COLOR_HOTLIGHT},
{"InactiveBorder", COLOR_INACTIVEBORDER}, {"InactiveCaption", COLOR_INACTIVECAPTION},
{"InactiveCaptionText", COLOR_INACTIVECAPTIONTEXT}, {"InfoBackground", COLOR_INFOBK},
{"InfoText", COLOR_INFOTEXT}, {"Menu", COLOR_MENU}, {"MenuText", COLOR_MENUTEXT},
{"Scrollbar", COLOR_SCROLLBAR}, {"ThreeDDarkShadow", COLOR_3DDKSHADOW}, {"ThreeDFace",
COLOR_3DFACE}, {"ThreeDHighlight", COLOR_3DHIGHLIGHT}, {"ThreeDLightShadow",
COLOR_3DLIGHT}, {"ThreeDShadow", COLOR_3DSHADOW}, {"Window", COLOR_WINDOW}, {"WindowFrame",
COLOR_WINDOWFRAME}, {"WindowText", COLOR_WINDOWTEXT} };
```

File

myx_gc_const.h (see page 219)

Remarks

This is variable SystemColors.

1.7 Macros

1.7.1 __cdecl Macro

```
#define __cdecl
```

File

myx_gc.h (see page 216)

Remarks

This is macro __cdecl.

1.7.2 __GC_BASE_H__ Macro

```
#define __GC_BASE_H__
```

File

myx_gc_base.h (see page 217)

Javadoc File

myx_gc_base.h (see page 217)

Javadoc Summary

Implementation of the GC base class from which most other GC classes are derived.

1.7.3 __GC_CANVAS_H__ Macro

```
#define __GC_CANVAS_H__
```

File

myx_gc_canvas.h (see page 218)

Javadoc File

myx_gc_canvas.h (see page 218)

Javadoc Summary

Generic canvas main class.

1.7.4 __GC_CONNECTION_H__ Macro

```
#define __GC_CONNECTION_H__
```

File

myx_gc_connection.h (see page 219)

Javadoc File

myx_gc_connection.h (see page 219)

Javadoc Summary

Implementation of the connectionclass.

1.7.5 __GC_DATATYPES_H__ Macro

```
#define __GC_DATATYPES_H__
```

File

myx_gc_datatypes.h (see page 220)

Javadoc File

myx_gc_datatypes.h (see page 220)

Javadoc Summary

Some commonly used data types.

1.7.6 __GC_FIGURE_H__ Macro

```
#define __GC_FIGURE_H__
```

File

myx_gc_figure.h (see page 222)

Javadoc File

myx_gc_figure.h (see page 222)

Javadoc Summary

Implementation of the model element class.

1.7.7 __GC_FONT_MANAGER_H__ Macro

```
#define __GC_FONT_MANAGER_H__
```

File

myx_gc_font_manager.h (see page 224)

Javadoc File

myx_gc_font_manager.cpp (see page 224)

Javadoc Summary

A class that manages shared display lists used for text output

1.7.8 __GC_GL_CONST_H__ Macro

```
#define __GC_GL_CONST_H__
```

File

myx_gc_const.h (see page 219)

Javadoc File

myx_gc_const.h (see page 219)

Javadoc Summary

Some commonly used constants.

1.7.9 __GC_GL_FIGURE_PARSER_H__ Macro

```
#define __GC_GL_FIGURE_PARSER_H__
```

File

myx_gc_figure_parser.h (see page 223)

Javadoc File

myx_gc_svgparser.h (see page 230)

Javadoc Summary

Parser for figure elements, which are converted from XML to our internal model.

1.7.10 __GC_GL_HELPER_H__ Macro

```
#define __GC_GL_HELPER_H__
```

File

myx_gc_gl_helper.h (see page 225)

Javadoc File

myx_gc_gl_helper.h (see page 225)

Javadoc Summary

Helper functions for creating OpenGL data and structures out of XML data.

1.7.11 __GC_GL_SVGPARSER_H__ Macro

```
#define __GC_GL_SVGPARSER_H__
```

File

myx_gc_svgparser.h (see page 230)

Javadoc File

myx_gc_svgparser.h (see page 230)

Javadoc Summary

Parser for svg elements, which are converted to OpenGL calls. note This parser does not handle a full svg description but only single svg elements.

1.7.12 __GC_GL_UTILITIES_H__ Macro

```
#define __GC_GL_UTILITIES_H__
```

File

myx_gc_utilities.h (see page 232)

Javadoc File

myx_gc_utilities.h (see page 232)

Javadoc Summary

Some common utility functions.

1.7.13 __GC_H__ Macro

```
#define __GC_H__
```

File

myx_gc.h (see page 216)

Javadoc File

myx_gc.h (see page 216)

Javadoc Summary

Base configuration header. Here most of the platform specific switches are kept.

1.7.14 __GC_LAYER_H__ Macro

```
#define __GC_LAYER_H__
```

File

myx_gc_layer.h (see page 226)

Javadoc File

myx_gc_layer.h (see page 226)

Javadoc Summary

Implementation of the GC layer class.

1.7.15 __GC_LAYOUT_H__ Macro

```
#define __GC_LAYOUT_H__
```

File

myx_gc_layout.h (see page 227)

Javadoc File

myx_gc_layout.h (see page 227)

Javadoc Summary

Implementation of the layouter classes.

1.7.16 __GC_MODEL_H__ Macro

```
#define __GC_MODEL_H__
```

File

myx_gc_model.h (see page 228)

Javadoc File

myx_gc_model.h (see page 228)

Javadoc Summary

Implementation of the model that manages the visual representation in the generic canvas.

1.7.17 __GC_STYLE_H__ Macro

```
#define __GC_STYLE_H__
```

File

myx_gc_style.h (see page 229)

Javadoc File

myx_gc_style.h (see page 229)

Javadoc Summary

Implementation of the style class.

1.7.18 __GC_TEXTURE_H__ Macro

```
#define __GC_TEXTURE_H__
```

File

myx_gc_texture.h (see page 230)

Javadoc File

myx_gc_texture.h (see page 230)

Javadoc Summary

Implementation of a texture class.

1.7.19 __GC_VIEW_H__ Macro

```
#define __GC_VIEW_H__
```

File

myx_gc_view.h (see page 234)

Javadoc File

myx_gc_view.h (see page 234)

Javadoc Summary

Implementation of the view class.

1.7.20 __myhash Macro

```
#define __myhash
```

File

myx_gc.h (see page 216)

Remarks

This is macro __myhash.

1.7.21 _USE_MATH_DEFINES Macro

```
#define _USE_MATH_DEFINES
```

File

myx_gc.h (see page 216)

Remarks

fidef _WINDOWS (see page 210)

1.7.22 _WINDOWS Macro

```
#define _WINDOWS
```

File

myx_gc.h (see page 216)

Remarks

Just to ease life. This way the symbol has to be given during build. WIN__ and _WIN32 are implicitely defined on Windows, however _WINDOWS is not.

1.7.23 A Macro

```
#define A(row,col) a[(col<<2)+row]
```

File

myx_gc_utilities.cpp (see page 231)

Remarks

The following matrix code was taken from Mesa3D (http://www.mesa3d.org/).

1.7.24 B Macro

```
#define B(row,col) b[(col<<2)+row]
```

File

myx_gc_utilities.cpp (see page 231)

Remarks

This is macro B.

1.7.25 COLOR_COUNT Macro

```
#define COLOR_COUNT (sizeof(Colors) / sizeof(Colors[0]))
```

File

myx_gc_const.h (see page 219)

Remarks

This is macro COLOR_COUNT.

1.7.26 DEG2RAD Macro

```
#define DEG2RAD M_PI / 180
```

File

myx_gc_const.h (see page 219)

Remarks

Constant for angle conversion from radians to degrees.

1.7.27 EPSILON Macro

```
#define EPSILON 1E-6
```

File

myx_gc_const.h (see page 219)

Remarks

The distance two float values can have at most and are still considered as being the same.

1.7.28 EXPORT_IMPORT_TEMPLATE Macro

```
#define EXPORT_IMPORT_TEMPLATE extern
```

File

myx_gc.h (see page 216)

Remarks

This is macro EXPORT_IMPORT_TEMPLATE.

1.7.29 GC_FBSTATE_RUBBERBAND Macro

```
#define GC_FBSTATE_RUBBERBAND 0x0002
```

File

myx_gc_layer.h (see page 226)

Remarks

This is macro GC_FBSTATE_RUBBERBAND.

1.7.30 GC_FBSTATE_RUBBERRECT Macro

```
#define GC_FBSTATE_RUBBERRECT 0x0001
```

File

myx_gc_layer.h (see page 226)

Remarks

Interal states of the selection layer.

1.7.31 GC_MOUSE_RELATED_STATES Macro

```
#define GC_MOUSE_RELATED_STATES GC_STATE_DRAG_PENDING || GC_STATE_DRAGGING ||
GC_STATE_LBUTTON_DOWN || GC_STATE_MBUTTON_DOWN || GC_STATE_RBUTTON_DOWN
```

File

myx_gc_view.h (see page 234)

Remarks

For simple state checks.

1.7.32 GC_STATE_CLEAR_PENDING Macro

```
#define GC_STATE_CLEAR_PENDING 0x0100
```

File

myx_gc_view.h (see page 234)

Remarks

This is macro GC_STATE_CLEAR_PENDING.

1.7.33 GC_STATE_DRAG_PENDING Macro

```
#define GC_STATE_DRAG_PENDING 0x0002
```

File

myx_gc_view.h (see page 234)

Remarks

Certain states a view can enter.

1.7.34 GC_STATE_DRAGGING Macro

```
#define GC_STATE_DRAGGING 0x0004
```

File

myx_gc_view.h (see page 234)

Remarks

This is macro GC_STATE_DRAGGING.

1.7.35 GC_STATE_LBUTTON_DOWN Macro

```
#define GC_STATE_LBUTTON_DOWN 0x0008
```

File

myx_gc_view.h (see page 234)

Remarks

This is macro GC_STATE_LBUTTON_DOWN.

1.7.36 GC_STATE_MBUTTON_DOWN Macro

```
#define GC_STATE_MBUTTON_DOWN 0x0010
```

File

myx_gc_view.h (see page 234)

Remarks

This is macro GC_STATE_MBUTTON_DOWN.

1.7.37 GC_STATE_PENDING_ACTIVATION Macro

```
#define GC_STATE_PENDING_ACTIVATION 0x0001
```

File

myx_gc_canvas.h (see page 218)

Remarks

States the canvas can enter.

1.7.38 GC_STATE_RBUTTON_DOWN Macro

```
#define GC_STATE_RBUTTON_DOWN 0x0020
```

File

myx_gc_view.h (see page 234)

Remarks

This is macro GC_STATE_RBUTTON_DOWN.

1.7.39 GC_STATE_RESIZING Macro

```
#define GC_STATE_RESIZING 0x0200
```

File

myx_gc_view.h (see page 234)

Remarks

This is macro GC_STATE_RESIZING.

1.7.40 GC_STATE_RUBBER_BAND Macro

```
#define GC_STATE_RUBBER_BAND 0x0080
```

File

myx_gc_view.h (see page 234)

Remarks

This is macro GC_STATE_RUBBER_BAND.

1.7.41 GC_STATE_RUBBER_RECTANGLE Macro

```
#define GC_STATE_RUBBER_RECTANGLE 0x0040
```

File

myx_gc_view.h (see page 234)

Remarks

This is macro GC_STATE_RUBBER_RECTANGLE.

1.7.42 GENERIC_CANVAS_API Macro

```
#define GENERIC_CANVAS_API
```

File

myx_gc.h (see page 216)

Remarks

The export/import template macro is needed if any STL template is used across DLL boundaries. The only exceptions are string and wstring, as they are already exported. Make sure all particpating DLLs/apps are using the same shared library type (either debug or release) and they must use a shared (dynamic) runtime instead of a statically linked one.

1.7.43 MAX_PATH Macro

```
#define MAX_PATH PATH_MAX
```

File

myx_gc.h (see page 216)

Remarks

This is macro MAX_PATH.

1.7.44 P Macro

```
#define P(row,col) product[(col<<2)+row]
```

File

myx_gc_utilities.cpp (see page 231)

Remarks

This is macro P.

1.7.45 ROUND Macro

```
#define ROUND(X) (int)((X) < 0 ? (X) - 0.5 : (X) + 0.5)
```

File

myx_gc_datatypes.h (see page 220)

Remarks

There is no ANSI C rounding function for float numbers, so define our own.

1.7.46 stdext Macro

```
#define stdext __gnu_cxx
```

File

myx_gc.h (see page 216)

Remarks

This is macro stdext.

1.7.47 SYS_COLOR_COUNT Macro

```
#define SYS_COLOR_COUNT (sizeof(SystemColors) / sizeof(SystemColors[0]))
```

File

myx_gc_const.h (see page 219)

Remarks

This is macro SYS_COLOR_COUNT.

1.7.48 WIN32_LEAN_AND_MEAN Macro

```
#define WIN32_LEAN_AND_MEAN
```

File

myx_gc.h (see page 216)

Remarks

This is macro WIN32_LEAN_AND_MEAN.

1.7.49 XML_IS Macro

```
#define XML_IS(node, type) (xmlStrcmp(node->name, (const xmlChar *) type) == 0)
```

File

myx_gc_datatypes.h (see page 220)

Remarks

This is macro XML_IS.

1.8 Files

1.8.1 myx_gc.h

Copyright (C) 2004 MySQL AB

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

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

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Macros

Macro	Description
__cdecl (see page 204)	This is macro __cdecl.
__GC_H__ (see page 207)	
__myhash (see page 209)	This is macro __myhash.
_USE_MATH_DEFINES (see page 209)	fidef _WINDOWS (see page 210)
_WINDOWS (see page 210)	Just to ease life. This way the symbol has to be given during build. WIN__ and _WIN32 are implicitely defined on Windows, however _WINDOWS is not.
EXPORT_IMPORT_TEMPLATE (see page 211)	This is macro EXPORT_IMPORT_TEMPLATE.
GENERIC_CANVAS_API (see page 214)	The export/import template macro is needed if any STL template is used across DLL boundaries. The only exceptions are string and wstring, as they are already exported. Make sure all particpating DLLs/apps are using the same shared library type (either debug or release) and they must use a shared (dynamic) runtime instead of a statically linked one.
MAX_PATH (see page 214)	This is macro MAX_PATH.
stdext (see page 215)	This is macro stdext.
WIN32_LEAN_AND_MEAN (see page 215)	This is macro WIN32_LEAN_AND_MEAN.

Namespaces

Namespace	Description
__gnu_cxx (see page 1)	This is namespace __gnu_cxx.

1.8.2 myx_gc_base.cpp

1.8.3 myx_gc_base.h

Classes

Class	Description
CGCBase (see page 73)	CGCBase serves as general base class for all generic canvas (see page 75) classes.
CGCListener (see page 77)	The general listener class is used to notify users of the canvas about general events like repaints and errors. This class is only an abstract class and must get a concrete implemention in the application. All Listener classes are meant to be a means for calling back the application. They are implemented and instantiated in the application and must be freed there. Don't forget to remove the listener class before you free it!

Macros

Macro	Description
__GC_BASE_H__ (see page 204)	

Types

Type	Description
CGCListenerIterator (see page 195)	This is type CGCListenerIterator.
CGCListeners (see page 195)	This is type CGCListeners.

1.8.4 myx_gc_canvas.cpp

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

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

1.8.5 myx_gc_canvas.h

Copyright (C) 2004 MySQL AB

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

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

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Classes

Class	Description
CCanvasListener (see page 4)	This is class CCanvasListener.
CFigureInstanceEnumerator (see page 62)	The CFigureInstanceEnumerator class is for quick access to all figure instances on all (common) layers. Enumeration happens depth-first. That means for each layer first all instances are enumerated before the next (see page 64) layer is taken.
CGenericCanvas (see page 101)	CGenericCanvas is the main class of the library and is the base for all further functionality (e.g. it creates and maintains the model). Instances are created via the exported CreateGenericCanvas (see page 153) function (if called from non C++ languages). CGenericCanvas serves as the controller in the model-view-controller pattern, which is used here and communicates with the viewer via callbacks. The viewer is platform specific and must be implemented individually. It is responsible to create a canvas (see page 75) controller class.

Functions

Function	Description
CreateGenericCanvas (see page 153)	Factory function to create a generic canvas. This function is exported and must be used by the viewer implementations to actually create a canvas instance. This is the only way to get hold of a generic canvas instance for non-C++ languages. Factory function for a canvas.

Macros

Macro	Description
__GC_CANVAS_H__ (see page 205)	
GC_STATE_PENDING_ACTIVATION (see page 213)	States the canvas can enter.

1.8.6 myx_gc_connection.cpp

Copyright (C) 2004 MySQL AB

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

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

more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

1.8.7 myx_gc_connection.h

Copyright (C) 2004 MySQL AB

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

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

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Classes

Class	Description
CConnection (see page 13)	A class comprising data for a connection.
CConnectionInstance (see page 16)	A concrete instance for a connection.

Enumerations

Enumeration	Description
tagConnectionDirection (see page 172)	This is record tagConnectionDirection.
TConnectionDirection (see page 182)	This is type TConnectionDirection.

Macros

Macro	Description
__GC_CONNECTION_H__ (see page 205)	

1.8.8 myx_gc_const.h

Copyright (C) 2004 MySQL AB

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

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

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Functions

Function	Description
DefaultFontFamily (see page 153)	Default values for text.
DefaultFontStyle (see page 154)	This is function DefaultFontStyle.
DefaultFontWeight (see page 154)	Must be a string as we get it from an attribute that can contain strings.

Macros

Macro	Description
__GC_GL_CONST_H__ (see page 206)	

COLOR_COUNT (see page 210)	This is macro COLOR_COUNT.
DEG2RAD (see page 210)	Constant for angle conversion from radians to degrees.
EPSILON (see page 211)	The distance two float values can have at most and are still considered as being the same.
SYS_COLOR_COUNT (see page 215)	This is macro SYS_COLOR_COUNT.

Structs

Struct	Description
tagColorEntry (see page 172)	
tagSystemColorEntry (see page 179)	This is record tagSystemColorEntry.
TColorEntry (see page 182)	
TSystemColorEntry (see page 192)	This is type TSystemColorEntry.

Variables

Variable	Description
Colors (see page 200)	This is variable Colors.
DefaultFontSize (see page 201)	20pt
Identity (see page 202)	This is variable Identity.
SystemColors (see page 204)	This is variable SystemColors.

1.8.9 myx_gc_datatypes.h

Enumerations

Enumeration	Description
tagActionType (see page 169)	Determines the type of action possible with a particular figure instance/element.
tagBidiMode (see page 170)	Bidirectional mode
tagChangeReason (see page 170)	This is record tagChangeReason.
tagConnectionLineStyle (see page 172)	Style for a connection line.
tagContainerID (see page 173)	Identifier for containers. Used when parsing pathes for properties.
tagFeedbackInfo (see page 173)	This is record tagFeedbackInfo.
tagFigureElementLayout (see page 174)	Layout variants for a figure element.
tagFigureElementResize (see page 174)	Resize variants for a figure element.
tagGCError (see page 174)	This is record tagGCError.
tagGCVariantType (see page 175)	A struct to transport certain base data that has no previously known type.
tagModifierKey (see page 176)	Modifier keys used when handling mouse input. Any value can be combined with other values.
tagMouseButton (see page 176)	Used in mouse events to specify which mouse button is involved. For one button only systems like MacOS the left button inidicator is used for this (only) button.
tagMouseEvent (see page 177)	Indicates which mouse event is to be handled.
tagOccurence (see page 177)	Determines how often a figure element is allowed to appear.
tagPropertyID (see page 177)	Identifier for properties. Used when parsing property specifications.
tagRRSelectionAction (see page 178)	TRRSelectionAction (see page 190) (rubber rect selection action) determines how to manipulate the selection state of figure instances with regard to their bounding box intersecting with the rubber rectangle.
tagRubberRectStyle (see page 179)	TRubberRectStyle (see page 191) describes the look of the rubber rectangle in the selection layer.

TActionType (see page 180)	Determines the type of action possible with a particular figure instance/element.
TBidiMode (see page 181)	Bidirectional mode
TConnectionLineStyle (see page 183)	Style for a connection line.
TContainerID (see page 183)	Identifier for containers. Used when parsing pathes for properties.
TFeedbackInfo (see page 184)	This is type TFeedbackInfo.
TFigureElementLayout (see page 184)	Layout variants for a figure element.
TFigureElementResize (see page 184)	Resize variants for a figure element.
TGCChangeReason (see page 185)	This is type TGCChangeReason.
TGCError (see page 186)	This is type TGCError.
TGCVariantType (see page 187)	A struct to transport certain base data that has no previously known type.
TModifierKey (see page 188)	Modifier keys used when handling mouse input. Any value can be combined with other values.
TMouseButton (see page 189)	Used in mouse events to specify which mouse button is involved. For one button only systems like MacOS the left button inidicator is used for this (only) button.
TMouseEvent (see page 189)	Indicates which mouse event is to be handled.
TOccurence (see page 189)	Determines how often a figure element is allowed to appear.
TPropertyID (see page 190)	Identifier for properties. Used when parsing property specifications.
TRRSelectionAction (see page 190)	TRRSelectionAction (rubber rect selection action) determines how to manipulate the selection state of figure instances with regard to their bounding box intersecting with the rubber rectangle.
TRubberRectStyle (see page 191)	TRubberRectStyle describes the look of the rubber rectangle in the selection layer.

Macros

Macro	Description
__GC_DATATYPES_H__ (see page 205)	
ROUND (see page 215)	There is no ANSI C rounding function for float numbers, so define our own.
XML_IS (see page 216)	This is macro XML_IS.

Structs

Struct	Description
tagAction (see page 169)	An action with associated parameters.
tagBoundingBox (see page 142)	This is class tagBoundingBox.
tagConstraints (see page 143)	This is class tagConstraints.
tagGCVariant (see page 145)	This is class tagGCVariant.
tagVertex (see page 147)	Some geometric data types.
tagViewport (see page 149)	ifdef _WINDOWS (see page 210)
TAction (see page 180)	An action with associated parameters.
TBoundingBox (see page 182)	This is type TBoundingBox.
TConstraints (see page 183)	This is type TConstraints.
TGCVariant (see page 186)	This is type TGCVariant.
TGCViewport (see page 187)	ifdef _WINDOWS (see page 210)
TVertex (see page 192)	Some geometric data types.

Types

Type	Description
CActionParameters (see page 192)	This is type CActionParameters.
CActions (see page 192)	This is type CActions.
CColorMap (see page 193)	This is type CColorMap.
CColorMapIterator (see page 193)	This is type CColorMapIterator.
CColorMapPair (see page 193)	This is type CColorMapPair.
CElementList (see page 194)	This is type CElementList.
CElementTemplateList (see page 194)	This is type CElementTemplateList.
CFigureInstances (see page 195)	This is type CFigureInstances.
CFigureList (see page 195)	This is type CFigureList.
CLayers (see page 196)	This is type CLayers.
CLayoutList (see page 196)	This is type CLayoutList.
CLayoutPair (see page 196)	This is type CLayoutPair.
CStyleList (see page 197)	This is type CStyleList.

CVertexVector (see page 198)	This is type CVertexVector.
CViews (see page 198)	This is type CViews.
GCContext (see page 198)	An opaque handle to a rendering context. Must be provided by the viewer.
TMatrix (see page 199)	This is type TMatrix.

1.8.10 myx_gc_figure.cpp

1.8.11 myx_gc_figure.h

Classes

Class	Description
CCaptionElement (see page 6)	Instance for figure elements and captions.
CCaptionElementTemplate (see page 10)	Special text element class.
CElementListener (see page 25)	This is class CElementListener.
CFigure (see page 33)	CFigure is the main element in the model (see page 38) and is created from a figure template. It cannot itself appear in a scene but is represented by one or more figure instances.
CFigureController (see page 41)	
CFigureElement (see page 43)	This is class CFigureElement.
CFigureElementListener (see page 49)	This is class CFigureElementListener.
CFigureElementTemplate (see page 51)	A figure element is one detail in a figure template and so also in a figure. There can be a hierarchy of figure elements to form complex figures.
CFigureInstance (see page 54)	The figure (see page 58) instance class is a proxy for a figure (see page 58) on a particular layer. There can be more than one instance pointing to the same figure (see page 58).
CFigureTemplate (see page 68)	CFigureTemplate is a description of how a concrete figure has to look and act. It is loaded from a description file and created by the figure parser.
CStyleListener (see page 132)	This is class CStyleListener.

Enumerations

Enumeration	Description
TAlignment (see page 181)	Text alignment constants.

Macros

Macro	Description
__GC_FIGURE_H__ (see page 205)	

Types

Type	Description
CFigureElementMap (see page 194)	

Variables

Variable	Description
DefaultLayout (see page 201)	The layout to be used for figure elements without any given layout.
DefaultResize (see page 202)	By default figure elements cannot be resized.

1.8.12 myx_gc_figure_parser.cpp

Variables

Variable	Description
actionLookup (see page 200)	This is variable actionLookup.
alignmentLookup (see page 200)	This is variable alignmentLookup.
layoutLookup (see page 203)	This is variable layoutLookup.
resizeLookup (see page 203)	This is variable resizeLookup.

1.8.13 myx_gc_figure_parser.h

Classes

Class	Description
CFigureParser (see page 64)	CFigureParser converts a figure descriptions given in XML to elements in our internal model.

Macros

Macro	Description
__GC_GL_FIGURE_PARSER_H__ (see page 206)	

1.8.14 myx_gc_font_manager.cpp

Variables

Variable	Description
internalManager (see page 202)	CFontManager (see page 71)
lockCount (see page 203)	This is variable lockCount.

1.8.15 myx_gc_font_manager.h

Classes

Class	Description
CFontManager (see page 71)	CFontManager is a helper class for text output in the generic canvas. It maps a description string for a font with attributes to a display list. If there is no display for a given font then one is created. The font manager is basically a singleton class. We only need one instance of it.

Functions

Function	Description
convertFontWeight (see page 153)	Converts the given string into a font weight value. Allowed values are: normal | bold | bolder | lighter | 100 | 200 | 300| 400 | 500 | 600 | 700 | 800 | 900 | inherit
fontManager (see page 155)	Returns the singleton font manager instance. Returns the current font manager (there is always only one).
lockFontManager (see page 159)	Increase lock count for the manager. Increases the lock count of the font manager. If the manager does not yet exist it is created.
unlockFontManager (see page 164)	Decrease lock count for the manager. Returns the current font manager (there is always only one).

Macros

Macro	Description
__GC_FONT_MANAGER_H__ (see page 206)	

Structs

Struct	Description
tagFontFileEntry (see page 174)	This is record tagFontFileEntry.
FontFileEntry (see page 180)	This is type FontFileEntry.

Types

Type	Description
FontFiles (see page 198)	This is type FontFiles.
Fonts (see page 198)	

1.8.16 myx_gc_gl_helper.cpp

1.8.17 myx_gc_gl_helper.h

Functions

Function	Description
convertColor (see page 152)	Reads the attribute with the given name and treats it as color value. Reads attribute name from Element and tries to treat the string as a color. The allowed syntax for colors is (as given by the SVG specification) either an HTML like value (e.g. #FFFFFF, #FFF) or a function like form (e.g. rgb(100, 255, 255), rgb(10%, 100%, 0%)).
getFloatAttribute (see page 157)	Reads the attribute with the given name (if it exists) and converts it to a float value. Helper method to retrieve a float attribute.
getFloatAttributeDef (see page 157)	Like GetFloatAttribute but with a default value in case the attribute does not exist. Helper method to retrieve an integer attribute. If it cannot be found a default value will be used instead.

getIntAttribute (see page 157)	Reads the attribute with the given name (if it exists) and converts it to an integer value. Helper method to retrieve an integer attribute.
getIntAttributeDef (see page 158)	Like GetIntAttribute but with a default value in case the attribute does not exist. Helper method to retrieve an integer attribute. If it cannot be found a default value will be used instead.
getStringAttribute (see page 158)	Reads the attribute with the given name (if it exists) and returns it. Helper method to retrieve a string attribute. If the attribute could be found then true is returned and Value is set to the value of the attribute. Otherwise false is returned and Value is not touched.
getStringAttributeDef (see page 158)	Like GetStringAttribute but with a default value in case the attribute does not exist. Helper method to retrieve a string attribute. If the attribute is empty or cannot be found then a default value is returned.
parseTextureEntry (see page 162)	Parses the given XML element for a texture definition. Parses the given XML node for texture information and creates a new entry in the texture manager.

Macros

Macro	Description
__GC_GL_HELPER_H__ (see page 206)	

1.8.18 myx_gc_layer.cpp

Copyright (C) 2004 MySQL AB

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

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

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

1.8.19 myx_gc_layer.h

Copyright (C) 2004 MySQL AB

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

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

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Classes

Class	Description
CConnectionLayer (see page 20)	The connection layer is a special layer variant (see page 166) that renders connections between figures.
CFeedbackLayer (see page 26)	The selection layer is a special layer variant (see page 166) that renders decorations for selected figures and can be queried for quick hit tests and lists of selected figures.
CGridLayer (see page 109)	The grid layer is a special layer variant (see page 166) that renders itself as grid.

CLayer (see page 115)	This is the base layer class, which is used by views to display their content. There are descendants for special things like feedback, grids and so on.
CPaperLayer (see page 126)	The paper layer is a normal layer but shows only one figure instace it creates implicitely given a certain figure. This layer does not take part in the usual input handling and is displayed as a ground layer on which all other layers render (see page 120) their stuff. This class is not exclusively managed by the view it belongs to.

Macros

Macro	Description
__GC_LAYER_H__ (see page 208)	
GC_FBSTATE_RUBBERBAND (see page 211)	This is macro GC_FBSTATE_RUBBERBAND.
GC_FBSTATE_RUBBERRECT (see page 211)	Interal states of the selection layer.

Structs

Struct	Description
tagSelectionEntry (see page 179)	Selection layer and associated structures
TSelectionEntry (see page 191)	Selection layer and associated structures

Types

Type	Description
CConnectionInstanceList (see page 193)	Connection layer and associated structures
CFigureConnectionList (see page 194)	This is type CFigureConnectionList.
CSelection (see page 196)	This is type CSelection.
CSelectionIterator (see page 196)	This is type CSelectionIterator.
CSelectionIteratorReverse (see page 197)	This is type CSelectionIteratorReverse.

1.8.20 myx_gc_layout.cpp

1.8.21 myx_gc_layout.h

Classes

Class	Description
CColumnLayouter (see page 11)	This is class CColumnLayouter.
CLayouter (see page 124)	Abstract base class for all layouter classes.
CRowLayouter (see page 130)	This is class CRowLayouter.
LayoutMapper (see page 140)	The layout mapper provides a simple way of getting a layouter class for a particular layout.

Macros

Macro	Description
__GC_LAYOUT_H__ (see page 208)	

1.8.22 myx_gc_model.cpp

Copyright (C) 2004 MySQL AB

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

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

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

1.8.23 myx_gc_model.h

Copyright (C) 2004 MySQL AB

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

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

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Classes

Class	Description
CGCModel (see page 78)	This is class CGCModel.

Macros

Macro	Description
__GC_MODEL_H__ (see page 208)	

Types

Type	Description
CConnectionList (see page 194)	This is type CConnectionList.

1.8.24 myx_gc_style.cpp

1.8.25 myx_gc_style.h

Classes

Class	Description
CGCStyle (see page 85)	A compiled style with its associated bounding box.

Macros

Macro	Description
__GC_STYLE_H__ (see page 208)	

1.8.26 myx_gc_svgparser.cpp

Enumerations

Enumeration	Description
GC_PRIMITIVE (see page 180)	

1.8.27 myx_gc_svgparser.h

Classes

Class	Description
CSVGParser (see page 133)	CSVGParser is the main svg parser class. It converts an element description into OpenGL calls. note Not all possible subelements/attributes can be parsed by this class. If they are specified then they will be ignored. See Generic Canvas documentation for more details.

Macros

Macro	Description
__GC_GL_SVGPARSER_H__ (see page 207)	

1.8.28 myx_gc_texture.cpp

Variables

Variable	Description
InternalTextureManager (see page 202)	Singleton texture manager instance.

1.8.29 myx_gc_texture.h

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Classes

Class	Description
CGCTexture (see page 88)	CGCTexture encapsulates a png image used to texture a figure in Generic Canvas. It loads the image data and manages it as well as the OpenGL properties for it.
CTextureManager (see page 138)	This is class CTextureManager.

Functions

Function	Description
DefaultTextureMagFilter (see page 154)	This is function DefaultTextureMagFilter.
DefaultTextureMinFilter (see page 154)	This is function DefaultTextureMinFilter.
DefaultTextureMode (see page 155)	This is function DefaultTextureMode.
DefaultTextureWrapMode (see page 155)	Default values for texturing.
textureManager (see page 164)	The one (and only) texture manager instance.

Macros

Macro	Description
__GC_TEXTURE_H__ (see page 209)	

Types

Type	Description
CTextureIterator (see page 197)	This is type CTextureIterator.
CTextures (see page 197)	The list of textures is an associated list of names and CTexture classes.
TLODList (see page 199)	A list of texture names each with the level-of-detail they are associated. The index in the vector is also the LOD they stand for.

Variables

Variable	Description
DefaultTextureDimensions (see page 202)	This is variable DefaultTextureDimensions.

1.8.30 myx_gc_utilities.cpp

Copyright (C) 2004 MySQL AB

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

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

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Macros

Macro	Description
A (see page 210)	The following matrix code was taken from Mesa3D (http://www.mesa3d.org/).
B (see page 210)	This is macro B.
P (see page 214)	This is macro P.

Variables

Variable	Description
predefinedColors (see page 203)	

1.8.31 myx_gc_utilities.h

Classes

Class	Description
CBoundingBoxComputer (see page 2)	The bounding box computer is a neat little helper class to construct a final bound box out of an arbitrary number of other boxes as well as (lists of) points.
StringTokenizer (see page 140)	Simple tokenizer class that works similar as Java's StringTokenizer.

Enumerations

Enumeration	Description
tagColorType (see page 172)	
TColorType (see page 182)	

Functions

Function	Description
boundsAreEmpty (see page 150)	Determines whether bounds are empty. Examines the given bounds and returns whether it is empty or not.
boundsContainPoint (see page 151)	Checks if a given point is within the given bounds. Determines whether the given bounds include the given point.
boundsIntersect (see page 151)	Determines whether both bounds overlap. Determines whether both bounds overlap.
colorByName (see page 151)	Find a color by name. Searchs the predifined colors and tries to find one with the given name.
colorToString (see page 152)	Converts a (float) color to a string. Converts a color to a string in the form #RRGGBB.
colorToString (see page 152)	Converts a (byte) color to a string. Converts a color to a string in the form #RRGGBB.
extractFilePath (see page 155)	Extracts the drive and path from the given file name.
freeImage (see page 156)	Releases the given image.
getContainerID (see page 156)	Converts a container name to an identifier suitable for quick lookup. Looks the given container name up and returns an identifier for it that can be used for quick lookup/handling.
getCurrentDir (see page 156)	Returns the current working folder.
getEntryIndex (see page 156)	Returns the index value for a given property. Treats the given path as property name preceeded with a slash. The first (or only) subpath must be an integer number denoting an index in a list.
getPropertyID (see page 158)	Returns an identifier for a given property name. Looks up the property name and returns an identifier for it.
loadPNG (see page 159)	Loads the given PNG image from disk.
matrixMultiply (see page 159)	Matrix code Perform a full 4x4 matrix multiplication.
matrixRotate (see page 160)	Generate a 4x4 transformation matrix from glRotate parameters, and post-multiply the input matrix by it.
matrixScale (see page 160)	Multiply a matrix with a general scaling matrix.
matrixTransform (see page 161)	Multiplies the given vertex by matrix M and returns the result.
matrixTranslate (see page 161)	Multiply a matrix with a translation matrix.
openFile (see page 161)	Platform neutral file open function.

registerSystemColors (see page 162)	Adds colors to the named color table. Registers predifined colors.
setCurrentDir (see page 162)	Sets the current working folder.
sortBounds (see page 163)	Sorts the given bounds so that left <= right and bottom <= top. Helper method to sort left/right and bottom/top coordinates so that for left/top are always smaller than right/bottom (origin is considered in the left-upper corner, +y pointing down).
stringToColor (see page 163)	Converts a string to color with float members. Converts a string to color with float members. The allowed syntax for colors is (as given by the SVG specification) either an HTML like value (e.g. #FFFFFF, #FFF) or a function like form (e.g. rgb(100, 255, 255), rgb(10%, 100%, 0%)).
stringToColor (see page 163)	Converts a string to color with byte members. Converts a string to a color with byte members. The allowed syntax for colors is (as given by the SVG specification) either an HTML like value (e.g. #FFFFFF, #FFF) or a function like form (e.g. rgb(100, 255, 255), rgb(10%, 100%, 0%)).
utf16ToANSI (see page 164)	Converts the given string into an ANSI string using the current system locale.
utf16ToUtf8 (see page 165)	Converts the given UTF-16 string into an UTF-8 string. Converts the given UTF-16 string into an UTF-8 string.
utf8ToANSI (see page 165)	Converts the given string, which is supposed to be an UTF-8 encoded text into an ANSI string using the current system locale.
utf8ToUtf16 (see page 165)	Converts the given string, which is supposed to be an UTF-8 encoded text into an UTF-16 string. Converts the given string, which is supposed to be an UTF-8 encoded text into an UTF-16 string.
variant (see page 166)	Creates a GC variant from the given value.
variant (see page 166)	Creates a GC variant from the given value.
variant (see page 167)	Creates a GC variant from the given value.
variant (see page 167)	Creates a GC variant from the given value.
variant (see page 167)	Creates a GC variant from the given value.
variantToBool (see page 168)	Converts a GC variant (see page 166) to a bool.
variantToFloat (see page 168)	Converts a GC variant (see page 166) to a float.
variantToInt (see page 168)	Converts a GC variant (see page 166) to an integer.
variantToString (see page 169)	Conversion functions for GC variants and simple values. Converts a GC variant (see page 166) to a string.

Macros

Macro	Description
__GC_GL_UTILITIES_H__ (see page 207)	

Structs

Struct	Description
tagImage (see page 176)	This is record tagImage.
TImage (see page 188)	This is type TImage.

1.8.32 myx_gc_view.cpp

1.8.33 myx_gc_view.h

Classes

Class	Description
CGCView (see page 90)	A view implements an association between a set of layers and their visual representation on screen. Views can have individual zoom and offset values, viewports and other properties. There can always be only one active view. Views are managed by the canvas (see page 75) class.
CHitResults (see page 113)	The CHitResult class is used to collect a number of figures that are located at a given point in the canvas. note Never hold the given hit results record for a long time. The referenced figure instances may disappear at any time.

Macros

Macro	Description
__GC_VIEW_H__ (see page 209)	
GC_MOUSE_RELATED_STATES (see page 212)	For simple state checks.
GC_STATE_CLEAR_PENDING (see page 212)	This is macro GC_STATE_CLEAR_PENDING.
GC_STATE_DRAG_PENDING (see page 212)	Certain states a view can enter.
GC_STATE_DRAGGING (see page 212)	This is macro GC_STATE_DRAGGING.
GC_STATE_LBUTTON_DOWN (see page 212)	This is macro GC_STATE_LBUTTON_DOWN.
GC_STATE_MBUTTON_DOWN (see page 213)	This is macro GC_STATE_MBUTTON_DOWN.
GC_STATE_RBUTTON_DOWN (see page 213)	This is macro GC_STATE_RBUTTON_DOWN.
GC_STATE_RESIZING (see page 213)	This is macro GC_STATE_RESIZING.
GC_STATE_RUBBER_BAND (see page 214)	This is macro GC_STATE_RUBBER_BAND.
GC_STATE_RUBBER_RECTANGLE (see page 214)	This is macro GC_STATE_RUBBER_RECTANGLE.

Structs

Struct	Description
tagHitEntry (see page 175)	Hit testing structures
THitEntry (see page 187)	Hit testing structures

Types

Type	Description
THitEntries (see page 199)	This is type THitEntries.
THitEntryIterator (see page 199)	This is type THitEntryIterator.

Index

A

B

C

D

E

F

G

H

I

L

M

O

P

R

S

T

U

V

W

X