Intuition

COLLABORATORS

	TITLE : Intuition	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 13, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Intuition

1.1 Intuition-related Classes for AmigaTalk© 1998-2002:

Described herein are the classes & their methods for manipulating
Amiga-Intuition objects with AmigaTalk. Please be aware that each
Class that I've added to AmigaTalk is generic & all-purpose. So
there is a lot of methods that you won't have to use all of the time.
I would encourage you to write your own subClasses, & just use
your own standard Window sizes, Images, BitMaps, etc.
The class hierarchy is:
Workbench (parent class is Object) - New for V2.0+
BasicIFF (parent class is Object) - New for V1.9+
ExamineIFF - New for V1.9+
Glyph (parent class is Object)
GadTools - New for V1.9+
NewGadgets - New for V1.9+
NewMenus - New for V1.9+
Screen
Window
Menu
Gadget
Color
Requester
Border
BitMap
Painter
Image
IStruct -- NOT implemented yet!

Animation -- NOT implemented yet!

Font

IText

Icon - New for V1.8+

1.2 Workbench Class:

Workbench Class implements the functions that the AmigaOS uses to
interface to workbench. Dealings with Icons are performed in
Icon Class . Some methods in this Class
use Tags that are located in WorkbenchTags.st
The Methods are:
closeWorkbench
Returns true if the Workbench was closed, false if not.
openWorkbench
Returns true if the Workbench was opened, false if not.
workbenchToBack
Returns true if Workbench was moved, false if not.
workbenchToFront
Returns true if Workbench was moved, false if not.
addAppWindow: windowObj port: msgPort id: id data: userData tags: tagArray
Returns an appWindow Object or nil.
removeAppWindow: appWindowObject
Returns true if the AppWindow was removed or false, if not.
addAppIcon: text port: msgPort id: id data: userData lock: fileLock
icon: diskObj tags: tagArray
Returns an appIcon Object or nil.
removeAppIcon: appIconObject
Returns true if the AppIcon was removed, false if not.
addAppMenuItem: text port: msgPort id: id data: userData tags: tagArray
Returns an appMenuItem Object or nil.
removeAppMenuItem: appMenuItemObject
Returns true if the AppMenuItem was removed, false if not.
workbenchInfo: objName lock: fileLock screen: screenObject
<primitive 209 1 8 private fileLock objName screenObject>
openWorkbenchObject: objName tags: tagArray
Returns true or false.
closeWorkbenchObject: objName tags: tagArray
Returns true or false.

workbenchControl: objName tags: tagArray

Returns true or false.

addAppWindowDropZone: appWindow id: id data: userData tags: tagArray

Returns an appWindowDropZone Object or nil.

removeAppWindowDropZone: appWindow dropZone: appWindowDropZoneObject

Returns true or false.

changeWorkbenchSelection: objName hook: hookObject tags: tagArray

Returns true or false.

makeWorkbenchObjectVisible: objName tags: tagArray

Returns true or false.

1.3 BasicIFF Class:

The BasicIFF Class interfaces AmigaTalk to the iffparse.library. See
class IDNumbers in IFFConstants.st file for valid ID numbers that
identify valid IFF chunks that IFF files & Objects contain.
EXAMPLE: 16r424F4459 is 'BODY'
You should have access to the documentation for iffparse.library (or
wait for me to write some examples of how to use this Class ;). I'm
NOT going to re-hash the IFF documentation for iffparse.library. The
Help directory is getting complicated as it is.
The Methods are:

closeIFF

Dispose of the IFF Object that was in the System.

openIFF: iffFileName type: fileType mode: mode

Open a file that contain IFF object(s). fileType here means:
0 for a file, & 1 for a clipboard. mode is either
#IFFF_READ, #IFFF_WRITE or #IFFF_RWBITS.

initIFFHook: hookObj flags: flags

Add a hook to the IFF object.

initIFFAsDOS

Initialize an IFF object as an AmigaDOS file.

initIFFAsClip

Initialize an IFF object as an AmigaDOS Clipboard.

closeClipboard

Close the previously opened Clipboard.

openClipboard: clipUnitNumber

Open a Clipboard with the given clipUnitNumber. Range for
clipUnitNumber is 0 to 255. If you supply an out-of-range

clipUnitNumber, this method will open clip number 0.

parseIFF: mode

Parse the IFF object using the supplied mode. Valid values for

mode are:

#IFFPARSE_SCAN

#IFFPARSE_STEP

#IFFPARSE_RAWSTEP

readChunkBytes: byteArray size: numBytes

Read a chunk of bytes from the IFF object into the byteArray that's

numBytes in size.

readChunkRecords: byteArray size: numBytes number: numRecords

Read the given number of records (numRecords) into a chunk of bytes

from the IFF object into the byteArray that's numBytes in size.

writeChunkBytes: byteArray size: numBytes

Write a chunk of bytes from the byteArray that's

numBytes in size to the IFF object.

writeChunkRecords: byteArray size: numBytes number: numRecords

Write the given number of records (numRecords) from a chunk of bytes

to the IFF object. The byteArray is numBytes in size.

stopChunk: type id: id

Tell parseIFF: which chunks to stop upon entry into the given chunk.

The most common types are:

#ID_ILBM, #ID_FTXT, #ID_SMUS, #ID_8SVX, #ID_ANIM

See IDNumbers Class in IFFConstants.st for id values.

stopChunks: iffObj with: propertyArray size: numPairs

Do a bunch of stopChunk settings at once. The propertyArray

is constructed as follows:

ele[1] <- type, ele[2] <- id,

ele[3] <- nextType, ele[4] <- nextid,

...

currentChunk

Return the current chunk as an Object.

propertyChunk: type id: id

Tell parseIFF: what chunks this IFF object is supposed to have.

propertyChunks: iffObj with: propertyArray size: numPairs

Do a bunch of propertyChunk settings at once. The propertyArray

is constructed as follows:

ele[1] <- type, ele[2] <- id,

ele[3] <- nextType, ele[4] <- nextid,

...
findProperty: type id: id
Tell iffparse.library which chunk to find.
collectionChunk: type id: id
Collect ALL instances of a specified chunk.
collectionChunks: iffObj with: propertyArray size: numPairs
Do a bunch of collectionChunk settings at once. The propertyArray
is constructed as follows:
ele[1] <- type, ele[2] <- id,
ele[3] <- nextType, ele[4] <- nextid,
...
findCollection: type id: id
Find the collected chunks currently in scope.
stopOnExit: type id: id
Tell iffParse: to stop just before leaving the given chunk.
addEntryHandlerHook: hookObj for: anObject type: type id: id position: pos
Install a custom chunk entry handler that will be invoked after the parser
enters the given chunk.
WARNING: If you don’t know what this does (I certainly don’t!),
then DO NOT use it!
addExitHandlerHook: hookObj for: anObject type: type id: id position: pos
Install a custom chunk exit handler that will be invoked before the parser
leaves the given chunk.
WARNING: If you don’t know what this does (I certainly don’t!),
then DO NOT use it!
pushChunk: type id: id size: size
Tell iffParse: you are about to begin writing a new chunk.
size can also be #IFFSIZE_UNKNOWN if you dont know the size, but this
is slower than supplying the correct size.
popChunk
When you are through writing data to a chunk, complete the write
by using this method. Every call to pushChunk MUST have a
corresponding call to this method!
parentChunk
Find the parent of a relevant context node object.
allocateLocalItem: ident type: type id: id size: dataSize
Create a local item having the specified type, id, & ident values; and
with dataSize bytes of buffer space for your application to use.
getLocalItemData

Obtain a local item buffer Object. Remember that the size of this
buffer is what you asked for with allocateLocalItem:type:id:size:
storeLocalItem: position
Store a local item Object in a context node. The position argument
determines where the local item is stored. The possible values are:
#IFFSLI_ROOT, #IFFSLI_TOP & #IFFSLI_PROP.
storeItemInContext
Use this when you already have a context node Object to which you
want to attach a local item Object.
findPropertyContext
Find the topmost context Object for storeLocalItem:
findLocalItem: ident type: type id: id
After you've stored your local item Object in a context node Object,
you can retrieve it later with this method.
freeLocalItem
Dispose of the local item Object obtained via allocateLocalItem:type:id:size:
setLocalItemPurge: hookObj
Add a hook that the parser will call before deleting local item(s).
getErrorString: errorNumber
Return a String Object that describes the given errorNumber.
idToString: identifier
Convert the given 32-bit identifier into a String object. Probably
useful someday.
getPropertySize: propertyObject
Return an Integer describing the size of the propertyObject.
getPropertyData: propertyObject
Return an Integer describing the data location of the propertyObject.
getCollectionSize: collectionObject
Return an Integer describing the size of the collectionObject.
getCollectionData: collectionObject
Return an Integer describing the data location of the collectionObject.

1.4 ExamineIFF Class:

The ExamineIFF Class allows the User to obtain various chunks from an
IFF file (NOT clipboards).
Available methods for this class are:
initialize
Use this method after: myIFFExaminer <- ExamineIFF new

privateObtainChunk: chunkType from: fileName id: chunkID parent: pID

You DO NOT need to use this method!

obtainBMHD: fileName

Return an Integer Object that describes the BitMap Header chunk found

in the given fileName.

obtainCMAP: fileName

Return an Integer Object that describes the ColorMap chunk found

in the given fileName.

obtainCAMG: fileName

Return an Integer Object that describes the ViewModes chunk found

in the given fileName.

obtainPixelData: fileName

Return an Integer Object that describes the Pixel data chunk found

in the given fileName.

obtainCHRS: fileName

Return an Integer Object that describes the Text data chunk found

in the given fileName.

obtainVHDR: fileName

Return an Integer Object that describes the Voice Header chunk found

in the given fileName.

obtainVoiceData: fileName

Return an Integer Object that describes the Voice data chunk found

in the given fileName.

1.5 Glyph Class:

Class Glyph is an abstract class that serves as a parent class for

all Intuition-related classes.

glyphType

Return the receiver class asString. In class Glyph, this

method simply returns an error message.

isDisplayed

Return true if the object is being displayed, else return false.

For class Glyph, this method simply returns as error message.

1.6 GadTools Class:

GadTools class is the Parent class that interfaces AmigaTalk to the
gadtools.library in AmigaDOS. Most of the methods in this class do NOT
have to be used by the programmer. The methods you should use are:
drawBoxFrom: sPoint to: ePoint tags: tagArray
Draw a beveled box in the previously registered window Object.
The tags supplied will say whether it's recessed or not.
WARNING: The coordinates you supply are NOT checked against window
boundaries!
windowIs
Tell subclasses what Window they are attached to.
registerTo: aWindowObject
Set the instance variable windowObj to aWindowObject.
This method is usually over-ridden by a subclass.
visualInfoObject
Tell subclasses what VisualInfo they have to use.
freeVisualInfo
Dispose of the VisualInfo Object. visualInfoObj cannot be used
after this unless you perform getVisualInfo:tags: again.
This method is usually called from a subclass.
getVisualInfo: screenObj tags: tagArray
Get VisualInfo information from the screenObj supplied & specified
by the tagArray. This method is usually called from a subclass.
xxxWaitForSelection
Smalltalk code has to call this inside a loop if there is more than
one IDCMP event expected. You do NOT need to use beginRefresh or
endRefresh around this method. This method will return an Array
Object with two elements:
rval at: 1 -- value of Gadget (Boolean, String, Prop value or an
appropriate value from a NewGadget) or Menu String.
rval at: 2 -- Gadget or Menu UserData field.
Subclass methods utilize this method, so you do NOT have to use it.
Methods that the casual (read in-expert) User has no need for:
beginRefresh
Internally, this is a call to GT_BeginRefresh().
endRefresh: completeFlag
Internally, this is a call to GT_EndRefresh().
getIMsg

Internally, this is a call to GT_GetIMsg().

replyIMsg

Internally, this is a call to GT_ReplyIMsg().

refreshWindow

Internally, this is a call to GT_RefreshWindow().

postFilterIMsg

Internally, this is a call to GT_PostFilterIMsg().

filterIMsg

Internally, this is a call to GT_FilterIMsg().

SEE ALSO, NewGadgets , NewMenus

1.7 NewGadgets Class:

NewGadgets Class is the class that interfaces AmigaTalk to the
new gadgets portion of gadtools.library. The methods for this
class are:

disposeGadgetList: gadgetListObj

Remove a gadgetListObj from AmigaTalk memory.

allocateGadgetList

Make a gadgetListObj.

createGadgetList

Initialize a gadgetList Object.

disposeNewGadget: unNeededNewGadgetObj

You will have to keep track of every newGadgetObj returned
from makeNewGadget: & use this method on ALL of them
(unless you have memory to burn). Once you've called
addGadgetToList:type:tags:, a newGadgetObj is no longer
needed & perhaps you should use this method after addGadgetToList:type:tags:

makeNewGadget: structureArray

structureArray is an Array Object with the following
elements in the given order:
ele[1] <- ng_LeftEdge, ele[2] <- ng_TopEdge,
ele[3] <- ng_Width, ele[4] <- ng_Height,
ele[5] <- ng_GadgetText, ele[6] <- ng_TextAttr, (TextAttr can be 0)
ele[7] <- ng_GadgetID, ele[8] <- ng_Flags,
ele[9] <- ng_VisualInfo, ele[10] <- ng_UserData
ele[11] <- NewGadget Type Tag
ele[10] can be any AmigaTalk Object (especially useful
is a Symbol describing a method to call!).

newStructArray: initArray

This method is used to create a structureArray for makeNewGadget:

Example usage:

gType <- intuition getGadgetType: #BUTTON_KIND

newGadget <- NewGadgets new

vi <- newGadget visualInfoObject

newStruct <- newStructArray: #(10 40 100 20 'My Gadget'

textAttrObj id myFlags vi userData gType)

newGadgetObj <- newGadget makeNewGadget: newStruct

addGadgetToList: gadgetObj type: gType tags: tagArray

Add a new Gadget to the gadgetList Object.

setGadgetAttrs: tagArray

Change the attributes (as specified by tagArray) of a Gadget.

getGadgetAttrs: tagArray

Return some attributes (as specified by tagArray) of the Gadget.

registerTo: aWindowObject

Tell the instance of this Class what Window Object to use.

waitForGadgetValue

Smalltalk code has to call this inside a loop if there

is more than one IDCMP event expected. You do NOT

need to use beginRefresh or endRefresh arround this

method.

Use the returned Object (or copy it) BEFORE using any waitFor...

method (such as waitForGadgetUserData or waitForMenuString,

etc) again!

waitForGadgetUserData

Smalltalk code has to call this inside a loop if there

is more than one IDCMP event expected. You do NOT

need to use beginRefresh or endRefresh arround this

method. Any AmigaTalk Object is valid as the

UserData stored in the NewGadget (especially useful

is a Symbol describing a method to call!).

Example:

rval <- thisGadget waitForGadgetUserData

amigatalk perform: rval withArguments: #(onObject parm1 parm2 ...)

Use the returned Object (or copy it) BEFORE using any waitFor...

method (such as waitForMenuString, waitForGadgetValue, etc) again!

SEE ALSO, NewMenus , GadTools

1.8 NewMenus Class:

NewMenus Class is the class that interfaces AmigaTalk to the
new Menus portion of gadtools.library. Read the Test file in
AmigaTalk:TestFiles/TestNewMenu for an example on how to use this
Class.
The methods for this Class are:
disposeMenu
Wipe the instance from AmigaTalk memory.
allocateNewMenu: numItems
Allocate an Array of NewMenu objects & place it in newMenuArrayObj.
endOfMenuArray: intuitionObj
Make an Array Object for the #NM_END NewMenu terminator that's
required by gadtools.library.
xxxMakeArray: t k: k f: f x: ex data: data
Do NOT call this method, use the initMenuArray: (etc) methods instead!
initMenuArray: intObj title: title key: commKey flags: flags
exclude: mx data: userData
Make a new Menu structureArray for fillNewMenuItem:with:
initMenuItemArray: intObj title: title key: commKey flags: flags
exclude: mx data: userData
Make a new MenuItem structureArray for fillNewMenuItem:with:
initSubItemArray: intObj title: title key: commKey flags: flags
exclude: mx data: userData
Make a new SubItem structureArray for fillNewMenuItem:with:
initMenuImageArray: intObj title: title key: commKey flags: flags
exclude: mx data: userData
Make a new MenuItem (Image) structureArray for fillNewMenuItem:with:
initSubImageArray: intObj title: title key: commKey flags: flags
exclude: mx data: userData
Make a new SubItem (Image) structureArray for fillNewMenuItem:with:
fillNewMenuItem: itemNumber with: structureArray
structureArray is an Array Object with the following elements
in the given order:
ele[1] <- nm_Type, ele[2] <- nm_Label,
ele[3] <- nm_CommKey, ele[4] <- nm_Flags,
ele[5] <- nm_MutualExclude, ele[6] <- nm_UserData
ele[6] can be any AmigaTalk Object (especially useful
is a Symbol describing a method to call!).

NOTE: Use the initMenuArray: (etc) methods to create structureArray.
createMenuStrip: tagArray
Create the menu Object that Window Objects can use to display Menus.
This means that you MUST hang onto the return value from this
method until you use windowObj addMenuStrip: yourMenuStrip
Valid tags for this method are:
#GTMN_FrontPen (Pen number for text & separators (default: 0))
#GTMN_FullMenu (true or false (default: false))
#GTMN_NewLookMenus (true or false)
#GTMN_CheckMark (value is an Image Object)
#GTMN_AmigaKey (value is an Image Object)
#GTMN_SecondaryError (value is an Integer Object)
initializeMenus: tagArray
This method returns true if successful, false if the menus
could NOT be laid-out, nil if there is an error condition.
Valid tags for the method are:
#GTMN_TextAttr (value is a TextAttr Object)
waitForMenuString
Smalltalk code has to call this inside a loop if there
is more than one IDCMP event expected.
Use the returned Object (or copy it) BEFORE using any waitFor...
method again!
waitForMenuUserData
Smalltalk code has to call this inside a loop if there
is more than one IDCMP event expected.
Make sure that you use only AmigaTalk Objects as the UserData
stored in the NewMenu. This method will return nil if the
Menu Item selected was NULL.
Use the returned Object (or copy it) BEFORE using any waitFor...
method again!
Any AmigaTalk Object is valid as the UserData stored in the
NewMenu (especially useful is a Symbol describing a method to call!).
Example:
rval <- thisMenu waitForMenuUserData
amigatalk perform: rval withArguments: #(onObject parm1 parm2 ...)
registerTo: aWindowObject
Tell the instance of this Class what Window Object to use.
visualInfo
Ask the Parent Class what the visualInfo Object is. You do NOT
need to use this method, it's for other methods in this Class.
SEE ALSO, NewGadgets , GadTools

1.9 Screen Class:

Class Screen allows the AmigaTalk system to manipulate Amiga screens.

The Methods are:

new: newScreenModeID

Create a new Screen Object with the mode of newScreenModeID.

NOTE: The order of methods for making a Screen is as follows:

scr <- Screen new

scr setTitle: 'My Screen Title'

scr new: screenMode

or:

scr <- Screen new

"Set any Screen parameters in here."

scr openScreen: screenMode title: 'My Screen Title'

openScreen: screenMode title: screenTitle

Create a new Screen Object with the given screenMode & screenTitle &

open it.

setScreenModeID: newScreenModeID

Set the Screen Object's ModeID to newScreenModeID.

getScreenModeID

Return the current screenModeID.

close

Close the Screen Object.

pullScreenUp: numLines

Move a screen up by numLines (see intuition.library

function MoveScreen()). numLines has to be <= 0.

pushScreenDown: numLines

Move a screen down by numLines (see intuition.library

function MoveScreen()). numLines has to be >= 0.

redrawScreen

Re-draw the Screen Object (see intuition.library functions

MakeScreen() & RethinkDisplay()).

reOpenScreen

Re-open the screen with any new parameters that were changed by the

user, such as a new screenModeID, Depth, etc.

displayBeep

Call the intuition function DisplayBeep() for the given screen.

screenToBack

Place the given screen behind all other open screens (see

intuition.library function ScreenToBack()).

screenToFront

Place the given screen in front of all other open screens (see intuition.library function ScreenToFront()).

turnOffTitle

Blank out the screenTitle (see intuition.library function ShowTitle()).

showTitle

Enable the screenTitle display (see intuition.library function ShowTitle()).

NOTE: All of the set Parameter methods won't take effect until a call to openScreen: or reOpenScreen is given.

In general, it's best to perform all of the set parameter methods before you open the screen, then a call to reOpenScreen isn't necessary.

setOrigin: aPoint

Set the starting point of the screen to the given point aPoint.

setScreenSize: sizePoint

Set the width & height of the Screen to (sizePoint x) & (sizePoint y) respectively.

setScreenPens: pensPoint

Set the Foreground & Background pens to (pensPoint x) & (pensPoint y) respectively.

setTitle: newTitle

Change the title of the Screen to newTitle.

setDepth: newDepth

Change the number of bit-planes that the screen will use. The depth of a screen determines how many colors it has.

setFont: newFontName

Change the Font used to render text in the screen.

setBitMap: newBitMap

Change the bitmap of the given screen.

getOrigin

Return the current value of the LeftEdge & TopEdge of the screen as a Point .

getScreenPens

Return the current value of the foreground & background pens for the screen as a Point .

getFlags

Return the current value of the Flags for the screen.
getType
Return the current value of the Type of the screen.
getViewMode
Return the current value of the ViewMode of the screen.
getTitle
Return the current value of the Title of the screen.
getDepth
Return the current value of the Depth of the screen.
getFontName
Return the name of the current Font for the screen.
getBitMapName
Return the name of the current bitmap for the screen.
openScreenWithTags: tagArray
Open a screen with the given tagArray Object. Valid tag
values for Screens are documented in ScreenTags.st &
the ScreenTags class is instantiated by the Intuition Class,
located in SetupIntuition.st. You use the getScreenTag:
method in Intuition to obtain the correct tag value for a given
ScreenTag Symbol from ScreenTags as follows:

```
intuition <- Intuition new
tag1 <- intuition getScreenTag: #SA_Top
data1 <- myScreenTopInteger "Usually 0"
...
tagArray <- Array new: (howManyTagPairs * 2)
tagArray at: 1 put: tag1
tagArray at: 2 put: data1
...
myScreen <- openScreenWithTags: tagArray
```

NOTE:
This method assumes you included an SA_Title tag!
This might seem to be a complicated way to open a screen,
but if you make your own subClass of Screen & bury all of
this in a method, you only have to write the method once
& you can use it over & over (the Data is encapsulated,
do I have to teach you Object-Oriented programming too?)
getScreenErrorString: errorNumber
Translate an errorNumber into a String Object
selectAndOpenScreen
Query the User for a Screen title & a ScreenModeID
from the ScreenMode ASL requester & open a new Screen.

1.10 Window Class:

Class Window allows the AmigaTalk system to manipulate Amiga windows.

The Methods are:

new: newWindowTitle

Create a new Window Object & set the title to newWindowTitle.

openOnScreen: screenObject

Open a new window on the given screen (screenObject is what you get

from screenObject <- Screen new: 16r40D20001 for example).

close

Close the given window.

setPointer: spriteObject size: sizePt offset: offPt

Change the mouse pointer to the given sprite spriteObject.

sizePt & offPt are Points .

addGadget: gadgetObject

Add a gadgetObject to the given window.

setFirstGadget: newGadget

Change the FirstGadget to newGadget.

NOTE: Make sure that the newGadget is chained to all the other

gadgets you want for the window (See setNextGadget method

for Gadget).

refreshWindowFrame

Execute a call to RefreshWindowFrame() (see intuition.library).

refreshGadgets

Execute a call to RefreshGadgets() (see intuition.library).

removeGadget: gadgetObject

Delete a Gadget from the Window Object.

reportMouse: boolvalue

Turn reportMouse events on or off.

getMouseCoords

Return the current x & y-coordinates of the mouse position as a

Point .

printIText: iTextObject at: aPoint

Display the given IText structure in the Window at the

given Point .

handleIntuition

Wait for the user to select a Gadget or MenuItem from the window.

NOTE: Only IDCMP_CLOSEWINDOW, IDCMP_GADGETUP & IDCMP_MENUPICK events

are currently recognized. This method will return the name of

the first Gadget or MenuItem selected by the user.

windowToBack

Place the Window behind all other windows.

windowToFront

Place the Window in front of all other windows.

showRequester: requesterObject

Display the given Requester to the user.

addMenuStrip: menuObject

Display the given menu(s) in the Window.

removeMenuStrip

Remove the current menu strip from the Window.

moveWindow: deltaPoint

Move the Window to the new Point of origin.

infoReq: msg title: t

Display an information Requester to the user.

yesNoReq: msg title: t

Obtain a yes or no response from the user.

getUserChoice: msg title: t choices: bstr

Obtain a choice from the user from the given bstr (Button strings).

Example: 'YES|NO|MAYBE'

setWindowSize: sizePoint

Change the size of the Window to the given values (sizePoint x) &

(sizePoint y).

getOrigin

Return the current LeftEdge & TopEdge values for the Window

as a Point .

getWindowSize

Return the current Width & Height values for the Window

as a Point .

getWindowPens

Return the current foreground & background pen values for the Window

as a Point .

getFlags: windowTitle

Return the current Flags value for the given Window.

getIDCMPFlags: windowTitle

Return the current IDCMPFlags value for the given Window.

getTitle: windowTitle

Return the current Title value for the given Window.

changeTitle: newTitle

Change the title of the Window to newTitle.

getScreenTitle

Get the title of the Screen that the given Window is attached to.

beginRefresh

Execute a call to BeginRefresh() for the Window.

(see intuition.library)

endRefresh

Execute a call to EndRefresh() for the Window.

(see intuition.library)

remakeDisplay

Execute a call to RemakeDisplay() for the Window.

(see intuition.library)

rethinkDisplay

Execute a call to RethinkDisplay() for the Window.

(see intuition.library)

NOTE: All of the set Parameter methods won't take effect until a

call to openOnScreen:{UB} is given.

setWindowOrigin: aPoint

Change the LeftEdge & TopEdge of the receiver to the new

values.

setWindowPens: pensPoint

Change the foreground & background pens of the receiver to the new

values (foreground <- pensPoint x) & (background <- pensPoint y).

setFlags: newFlags

Change the Flags of the Window to the new value.

setIDCMPFlags: windowTitle to: newIDCMP

Change the IDCMPFlags of the Window to the new value.

setMinSize: newMinPoint

Change the MinWidth & MinHeight of the Window to the new

values.

setMaxSize: newMaxPoint

Change the MaxWidth & MaxHeight of the Window to the new

values.

getReqCount

Return a count of the Requesters (reqcount)for the Window.

getPointerSize

Return the width & height of the mouse pointer for the Window as a

Point .

setCheckMark: newCheckMark

Change the CheckMark Image of the Window to the
new value specified by 'newCheckMark', which is an Image .
getWindowOffset
Return the current x & y-offset coordinates for the Window as a
Point .
setBitMap: newBitMap
Change the BitMap of the Window to the new value specified
by 'newBitMap'.
changeWindowSize: deltaPoint
Ask Intuition to size the Window the specified amounts.
NOTE: This method is different from setWindowSize:
changeWindowSize: operates on a Window that's open.

1.11 Menu Class:

Class Menu allows the AmigaTalk system to manipulate Amiga menus.
Actual display of any Menus are taken care of in the Window Class by
setMenuStrip & removeMenuStrip.
The Methods for the Menu Class are:
new: newMenuName
Add a menu to the AmigaTalk internal system list.
remove
Remove a Menu from the AmigaTalk internal system list.
registerTo: windowTitle
Inform the AmigaTalk internal system which window is the parent of
the Menu.
getStartPoint
Return the LeftEdge & TopEdge of the Menu.
getMenuSize
Return the Width & Height of the Menu.
getFlags
Return the Flags of the Menu.
getNextMenu
Return the name of the NextMenu for the Menu.
getFirstItem
Return the name of the FirstItem for the Menu.
getMenuName
Return the name of the Menu. This is a silly function, since
you have to know the name of the menu to use it!

setStartPoint: newPoint

Set the LeftEdge & TopEdge of the Menu to the given values.

setMenuSize: sizePoint

Set the Width & Height of the Menu to the given values.

setFlags: newFlags

Set the Flags of the Menu to the given value.

setNextMenu: newNextMenu

Set the NextMenu of the Menu to the given value.

setFirstItem: newFirstItem

Set the FirstItem of the Menu to the given value.

setMenuName: newMenuName

Set the name of the Menu to the new name given.

See Also MenuItems , SubItems

1.12 MenuItem Class:

Class MenuItem allows the AmigaTalk system to manipulate Amiga MenuItems. Actual display of any Menus are taken care of in the Window Class by setMenuStrip & removeMenuStrip.

The Methods for the MenuItem Class are:

new: newMenuItemName

Add a MenuItem to the AmigaTalk internal system list.

remove

Remove a MenuItem from the AmigaTalk internal system list.

getStartPoint

Return the LeftEdge & TopEdge of the MenuItem.

getItemSize

Return the Width & Height of the MenuItem.

setStartPoint: newPoint

Set the LeftEdge & TopEdge of the MenuItem to the given values.

setItemSize: sizePoint

Set the Width & Height of the MenuItem to the given values.

getFlags

Return the Flags of the MenuItem.

setFlags: newFlags

Set the Flags of the MenuItem to the given value.

getMutualExclude

Return the MutualExclude value of the MenuItem.

setMutualExclude: newMutualExclude

Set the MutualExclude of the MenuItem to the given value.

getCommand

Return the Command (menu key-equivalent) value of the

MenuItem.

setCommand: newCommand

Set the Command (menu key-equivalent) of the MenuItem

to the given value.

getNextItem

Return the name of the NextItem from the MenuItem.

setNextItem: newNextItem

Set the NextItem of the MenuItem to the given value.

setItemFill: newItemFill

Set the ItemFill (IText or Image name) of the MenuItem.

setSelectFill: newSelectFill

Set the SelectFill (IText or Image name) of the MenuItem.

setSubItem: newSubItem

Set the SubItem of the MenuItem to the given value.

getSubItem

Return the name of the first SubItem attached to the

MenuItem.

getItemFill

Return the name of the ItemFill (either IText or Image) from

the MenuItem.

getSelectFill

Return the name of the SelectFill (either IText or Image)

from the MenuItem.

See Also Menus , SubItems

1.13 SubItem Class:

Class SubItem allows the AmigaTalk system to manipulate Amiga

Menu SubItems. Actual display of any Menus are taken care of in the

Window Class by setMenuStrip & removeMenuStrip.

The Methods for the SubItem Class are:

new: newSubItemName

Add a SubItem to the AmigaTalk internal system list.

remove

Remove a SubItem from the AmigaTalk internal system list.

getStartPoint

Return the LeftEdge & TopEdge of the SubItem.

getSubSize

Return the Width & Height of the SubItem.

setStartPoint: newPoint

Set the LeftEdge & TopEdge of the SubItem to the given values.

setSubSize: sizePoint

Set the Width & Height of the SubItem to the given values.

getFlags

Return the Flags of the SubItem.

setFlags: newFlags

Set the Flags of the SubItem to the given value.

getMutualExclude

Return the MutualExclude value of the SubItem.

setMutualExclude: newMutualExclude

Set the MutualExclude of the SubItem to the given value.

getCommand

Return the Command (menu key-equivalent) value of the

SubItem.

setCommand: newCommand

Set the Command (menu key-equivalent) of the SubItem

to the given value.

getNextItem

Return the name of the NextItem from the SubItem.

setNextItem: newNextItem

Set the NextItem of the SubItem to the given value.

setItemFill: newItemFill

Set the ItemFill (IText or Image name) of the SubItem.

setSelectFill: newSelectFill

Set the SelectFill (IText or Image name) of the SubItem.

getItemFill

Return the name of the ItemFill (either IText or Image) from

the SubItem.

getSelectFill

Return the name of the SelectFill (either IText or Image)

from the SubItem.

See Also Menus , MenuItems

1.14 Gadget Class:

Class Gadget is an abstract class that serves as a parent class for
all Gadget-related classes. The methods it defines are only useful
for system-wide purposes:
gadgetTypeIs: gadgetName
Return the type of the gadget given by gadgetName. The return
values are:
BOOLEAN = 1
PORPORTIONAL = 3
STRING = 4
new: newGadgetName
Create a new Gadget to the newGadgetName & the default
type of BoolGadget.
SubClasses: BoolGadget
StrGadget
PropGadget

1.15 Boolean Gadget Class:

Class BoolGadget allows the AmigaTalk system to man-
ipulate Amiga Boolean Gadgets. The Methods are:
new: newGadgetName
Add a BoolGadget to the AmigaTalk system. This method allocates
an internal memory structure to the AmigaTalk system.
remove
Remove a BoolGadget from the AmigaTalk system.
registerTo: windowTitle
Set the name of the gadget's parent to the windowTitle.
setStartPoint: newPoint
Set the origin of the gadget to the given point value.
setGadgetSizeTo: sizePoint
Set the size of the gadget to the given width & height.
getLeftEdge
Return the LeftEdge value of the BoolGadget.
getTopEdge
Return the TopEdge value of the BoolGadget.
getWidth
Return the Width value of the BoolGadget.

getHeight

Return the Height value of the BoolGadget.

getFlags

Return the Flags value of the BoolGadget.

getActivation

Return the Activation value of the BoolGadget.

getGadgetType

Return the Type of the BoolGadget.

NOTE: only needed because of GZZGADGET & REQGADGET type flags.

getGadgetID

Return the GadgetID number for the BoolGadget.

getNextGadgetName

Return the name of the NextGadget for the BoolGadget.

getITextName

Return the name of the IText attached to the BoolGadget.

getRenderName

Return the name of the gadget rendering (IText or Image) for the

BoolGadget.

getSelectName

Return the name of the gadget selection rendering (IText or

Image) for the BoolGadget.

setFlags: newFlags

Set the gadget Flags to the new value(s).

setActivation: newActivation

Set the gadget Activation to the new value.

setGadgetType: newGadgetType

Set the gadget Type to the new value.

NOTE: only needed because of GZZGADGET & REQGADGET type flags.

setGadgetID: newGadgetID

Set the GadgetID to the new value.

setNextGadgetName: newNextGadgetName

Set the NextGadget to the Gadget attached to newNextGadgetName.

setITextName: newITextName

Set the IText to the new value.

setRenderName: newRenderName

Set the BoolGadget rendering to the name of the IText or Image

supplied.

setSelectName: newSelectName

Set the gadget selection rendering to the name of the IText

or Image supplied.

1.16 String Gadget Class:

Class StrGadget allows the AmigaTalk system to man-
ipulate Amiga String Gadgets. The Methods are:
new: newGadgetName
Add a StrGadget to the AmigaTalk system. This method allocates
an internal memory structure to the AmigaTalk system.
remove
Remove a StrGadget from the AmigaTalk system.
registerTo: windowTitle
Set the name of the StrGadget's parent to the windowTitle.
setStartPoint: newPoint
Set the origin of the StrGadget to the given point value.
setGadgetSize: sizePoint
Set the size of the StrGadget to the given point value.
changeBufferSize: newSize
Change the internal buffer size for the StrGadget.
getBufferSize
Return the size of the StrGadget buffer (in bytes).
getLeftEdge
Return the LeftEdge value of the StrGadget.
getTopEdge
Return the TopEdge value of the StrGadget.
getWidth
Return the Width value of the StrGadget.
getHeight
Return the Height value of the StrGadget.
getFlags
Return the Flags value of the StrGadget.
getActivation
Return the Activation value of the StrGadget.
getGadgetType
Return the Type of the StrGadget.
NOTE: only needed because of GZZGADGET & REQGADGET type flags.
getGadgetID
Return the GadgetID number for the StrGadget.
getNextGadgetName
Return the name of the NextGadget for the StrGadget.
getITextName

Return the name of the IText attached to the StrGadget.

getRenderName

Return the name of the gadget rendering (IText or Image) for the

StrGadget.

getSelectName

Return the name of the gadget selection rendering (IText

or Image) for the StrGadget.

setFlags: newFlags

Set the StrGadget Flags to the new value(s).

setActivation: newActivation

Set the StrGadget Activation to the new value.

setGadgetType: newGadgetType

Set the StrGadget Type to the new value.

NOTE: only needed because of GZZGADGET & REQGADGET type flags.

setGadgetID: newGadgetID

Set the GadgetID to the new value.

setNextGadgetName: newNextGadgetName

Set the NextGadget to the Gadget attached to newNextGadgetName.

setITextName: newITextName

Set the @{ "IText " LINK "ITextClass"} to the new value.

setRenderName: newRenderName

Set the StrGadget rendering to the name of the IText or Image

supplied.

setSelectName: newSelectName

Set the StrGadget selection rendering to the name of the IText

or Image supplied.

1.17 Proportional Gadget Class:

Class PropGadget allows the AmigaTalk system to man-

ipulate Amiga Proportional Gadgets. The Methods are:

new: gadgetName

Add a PropGadget to the AmigaTalk system. This method allocates

an internal memory structure to the AmigaTalk system.

remove

Remove a PropGadget from the AmigaTalk system.

registerTo: windowTitle

Set the name of the PropGadget's parent to the windowTitle.

setStartPoint: newPoint

Set the origin of the PropGadget to the given point value.

setGadgetSize: sizePoint

Set the size of the PropGadget to the given width & height.

modifyProps: newFlags hPot: hp vPot: vp hBody: hb vBody: vb

windowName: windowTitle

Change the given porportional values for the PropGadget.

setProps: gadgetName flags: newFlags hPot: hp vPot: vp

hBody: hb vBody: vb

Initialize the porportional gadget values.

getPropFlags

Return the PropFlags value for the PropGadget.

getHPot

Return the HPot value of the PropGadget.

getVPot

Return the VPot value of the PropGadget.

getHBody

Return the HBody value of the PropGadget.

getVBody

Return the VBody value of the PropGadget.

getLeftEdge

Return the LeftEdge value of the PropGadget.

getTopEdge

Return the TopEdge value of the PropGadget.

getWidth

Return the Width value of the PropGadget.

getHeight

Return the Height value of the PropGadget.

getFlags

Return the Flags value of the PropGadget.

getActivation

Return the Activation value of the PropGadget.

getGadgetType

Return the Type of the PropGadget.

NOTE: only needed because of GZZGADGET & REQGADGET type flags.

getGadgetID

Return the GadgetID number for the PropGadget.

getNextGadgetName

Return the name of the NextGadget for the PropGadget.

getITextName

Return the name of the @{ " IText " LINK "ITextClass"} attached to the PropGadget.

getRenderName

Return the name of the gadget rendering (IText or Image) for the

PropGadget.

getSelectName

Return the name of the gadget selection rendering (IText

or Image) for the PropGadget.

setFlags: newFlags

Set the gadget Flags to the new value(s).

setActivation: newActivation

Set the PropGadget Activation to the new value.

setGadgetType: newGadgetType

Set the PropGadget Type to the new value.

NOTE: only needed because of GZZGADGET & REQGADGET type flags.

setGadgetID: newGadgetID

Set the GadgetID to the new value.

setNextGadgetName: newNextGadgetName

Set the NextGadget to the Gadget attached to newNextGadgetName.

setITextName: newITextName

Set the IText to the new value.

setRenderName: newRenderName

Set the PropGadget rendering to the name of the IText or Image

supplied.

setSelectName: newSelectName

Set the PropGadget selection rendering to the name of the IText

or Image supplied.

1.18 Colors Class:

Class Colors allows the AmigaTalk system to manipulate Amiga Colors.

The Methods are:

make: colorMapName size: numColors

Allocate a new ColorMap.

dispose

Free the given ColorMap from the system.

loadColors: c from: colorMapFileName

Load the amount of color registers c with the values given in the

colorMapFileName file.

getColor: sourceType from: sourceName which: n

Return an RGB representation from the given source (window = 1 or
colormap) for the nth register.
setWindowColorReg: n red: r green: g blue: b
Set the color register n to the RGB values supplied.
setMapValue: sourceType from: source num: n red: r green: g blue: b
Set the color register n for the given source (window = 1 or
colormap) to the RGB values supplied.
copyMap: source to: dest sourceType: type
Copy a ColorMap from the source of sourceType (window = 1 or colormap)
to the destination (ColorMap).
saveColorsTo: colorMapFileName
Save the color register values to the given filename.

1.19 Requester Class:

Class Requester implements control of Amiga Requesters for AmigaTalk,
except for displaying them, which is done inside the Window class.
The methods are:
initialize: requesterName withArray: reqValues
Initialize a Requester for the AmigaTalk system.
reqValues is an Array with the following fields:
LeftEdge, TopEdge,
Width, Height,
RelLeft, RelTop,
ReqGadget, ReqBorder, ReqText,
Flags BackFill, ImageBMap
new: requesterName
Register a Requester with the AmigaTalk system.
remove
Remove a Requester from the AmigaTalk system.
getStartPoint
Return the LeftEdge & TopEdge of the receiver.
getReqSize
Return the Width & Height of the receiver.
getRelativePoint
Return the RelLeft & RelTop variable of the receiver.
getFlags
Return the Flags variable of the receiver.
getBackFill

Return the background pen number of the receiver.

getReqText

Return the name of the IText attached to the receiver.

getReqGadget

Return the name of the first Gadget attached to the receiver.

getReqBorder

Return the name of the Border attached to the receiver.

getReqBitMap

Return the name of the BitMap attached to the receiver.

setStartPoint: newPoint

Change the LeftEdge & TopEdge of the receiver to

the supplied values.

setReqSize: sizePoint

Change the Width & Height of the receiver to the

values supplied.

setRelativePoint: newRelPoint

Change the RelLeft & RelTop of the receiver to the values

given.

NOTE: You should also add the POINTREL value to the Flags of the

receiver in order to use this feature.

receiver in order to use this feature.

setFlags: newFlags

Change the Flags of the receiver to newFlags.

setBackFill: newBackFill

Change the background pen number to newBackFill for the

receiver.

setReqText: newReqText

Change the IText attached to the reciever to newReqText.

setReqBorder: newReqBorder

Change the Border attached to the receiver to newReqBorder.

setReqGadget: newReqGadget

Change the Gadget attached to the receiver to newReqGadget.

setReqBitMap: newReqBMap

Change the BitMap attached to the receiver to newReqBMap.

1.20 Alert Class:

Class Alert implements control of Amiga Alerts for the AmigaTalk system.
The message string that the user supplies will be truncated at 128
characters & the alert number will be prepended to it.
new: newAlertName
Add an Alert to the AmigaTalk system.
remove
Remove an Alert from the AmigaTalk system.
getAlertNumber
Return the alert number attached to the receiver.
getAlertHeight
Return the height of the receiver.
getAlertMessage
Return the alert String attached to the receiver.
setAlertNumber: num
Change the alert number of the receiver to num.
setAlertHeight: height
Change the alert height of the receiver to height.
setAlertMessage: newMsg
Change the alert String of the receiver to newMsg.
displayAlert
Display the receiver to the user.

1.21 Border Class:

The Class Border (in this implementation) is an abstract class that
normally is attached to other objects, such as Gadgets or Requesters.
This is why there is no method for actually drawing borders into
Windows in this Class. See drawPolygon in the Class Painter
if you need to draw a Border in a Window .
SubClasses: Line
Triangle
Rectangle
new: newBorderName
Add the receiver to the AmigaTalk system. Initialize the new
Border Object as follows:
name <- newBorderName
nextBorderName <- nil

leftEdge <- 0

topEdge <- 0

frontPen <- 1

backPen <- 0

drawMode <- 1

count <- 2

remove

Remove the receiver from the AmigaTalk system.

registerTo: windowTitle

Set the parent name of the receiver to windowTitle.

getLeftEdge

Return the LeftEdge of the receiver.

getTopEdge

Return the TopEdge of the receiver.

getFrontPen

Return the foreground pen number of the receiver.

getBackPen

Return the background pen number of the receiver.

getDrawMode

Return the drawing mode of the receiver. Currently known values

are:

JAM1 = 0

JAM2 = 1

COMPLEMENT = 2

INVERSEVID = 4

getCount

Return the number of points in the receiver.

getNextBorderName

Return the name of the next Border attached to the receiver.

setStartPoint: sPoint

Change the starting point of the receiver to sPoint.

This method sets LeftEdge & TopEdge.

setFrontPen: newFrontPen

Change the foreground pen number of the receiver.

setBackPen: newBackPen

Change the background pen number of the receiver.

setDrawMode: newDrawMode

Change the drawing mode of the receiver. Currently known values

are:

JAM1 = 0

JAM2 = 1

COMPLEMENT = 2

INVERSEVID = 4

setCount: newCount

Change the number of points in the receiver.

setNextBorderName: newBorder

Change the name of the next Border attached to the receiver.

setBorderPoint: thePt to: newPoint

Change the value of a point in the receiver.

1.22 Line Class (Border sub-class):

makeLine: lineName from: fPoint to: tPoint

Add a Border with two points & register it with the AmigaTalk system.

1.23 Triangle Class (Border sub-class):

makeTriangle: triangleName vert1: v1Point vert2: v2Point vert3: v3Point

Add a Border with 4 points & register it with the AmigaTalk system.

1.24 Rectangle Class (Border sub-class):

makeRectangle: rectangleName from: fPoint to: tPoint

Add a Border with 5 points & register it with the AmigaTalk system.

1.25 Font Class:

Class Font implements control of Amiga Fonts.

The methods for Font are:

new: newFontName

Add a Font to the AmigaTalk system.

remove

Remove an Font from the AmigaTalk system.

getName

Return the name of the Font.

getYSize

Return the Height of the Font (in pixels - ta_YSize).

getStyle

Return the style (ta_Style) of the Font such as PLAIN, BOLD,

UNDERLINED or ITALIC, etc.

getFlags

Return the Flags of the receiver (ta_Flags).

setName: newName

Change the name of the Font to newName (ta_Name <- newName).

This method is probably not really needed.

setYSize: newYSize

Set the Height of the Font (ta_YSize <- newYSize).

setStyle: newStyle

Change the style of the Font (ta_Style <- newStyle).

setFlags: newFlags

Change the Flags of the Font (ta_Flags <- newFlags).

See Also IText Class

1.26 IText Class:

Class IText implements control of Amiga IntuiText except for actually

displaying it, which is in the Window class.

The methods for IText are:

new: newITextName

Add an IText (IntuiText) to the AmigaTalk system.

remove

Remove an IText from the AmigaTalk system.

registerTo: windowTitle

Set the parent name of the receiver to windowTitle.

getFrontPen

Return the foreground pen number of the receiver.

getBackPen

Return the background pen number of the receiver.

getDrawMode

Return the drawing mode of the receiver. Currently known values are:

JAM1 = 0

JAM2 = 1

COMPLEMENT = 2

INVERSEVID = 4

getLeftEdge

Return the LeftEdge of the receiver.

getTopEdge

Return the TopEdge of the receiver.

getFontName

Return the name of the rendering font of the receiver.

getIText

Return the String that the the receiver will display.

getNextText

Return the name of the next IntuiText of the receiver.

getTextLength

Return the length (in pixels) of the text of the receiver.

setFrontPen: newFrontPen

Change the foreground pen of the receiver.

setBackPen: newBackPen

Change the background pen of the receiver.

setDrawMode: newDrawMode

Change the drawing mode of the receiver. Currently known values are:

JAM1 = 0

JAM2 = 1

COMPLEMENT = 2

INVERSEVID = 4

setLeftEdge: newLeftEdge

Change the LeftEdge of the receiver.

setTopEdge: newTopEdge

Change the TopEdge of the receiver.

setFontName: newFontName

Change the name of the rendering font of the receiver.

setIText: iTextName to: newIText

Change the text String of the receiver.

setNextText: newNextText

Change the name of the next IText of the receiver.

See Also Font Class

1.27 Icon Class:

Class Icon implements control of Amiga Icons.

The methods for Icon are:

openIcon: iconFileName

Add an Icon to the AmigaTalk system.

new: iconFileName

Same as openIcon: method.

closeIcon
Remove an Icon from the AmigaTalk system.
editToolTypes
Display the ToolTypes in a GUI for editing purposes. Not all
Icon types have ToolTypes (just Tool & Project icon types).
displayIconInfo
Display a GUI containing information about the Icon.
NOTE: The Icon colors displayed might not be correct
(working on it!).
displayIconImages
Display only the Icon images. NOTE: The Icon colors displayed
might not be correct (working on it!).
setIconPosition: newPoint
Set the physical screen location of an Icon to newPoint .
moveIcon: deltaPoint
Move the physical screen location of an Icon by deltaPoint .
editIcon: externalEditorName
Invoke the named external Icon editor (such as Sys:Tools/IconEdit),
so that you can edit the Icon with the editor of your choosing.
addToolType: toolString
Add a ToolType to the Icon (only Tool & Project Icon types have
ToolTypes).
deleteToolType: toolString
Delete a ToolType from the Icon (only Tool & Project Icon types have
ToolTypes).
getIconWidth
Return the Width (in pixels) of the Icon.
getIconHeight
Return the Height (in pixels) of the Icon.
getIconFlags
Return the Flags (usually related to Display mode) of the Icon.
^ <primitive 219 12 private>
getIconImagePtr
Return the address (as an Integer) of the Icon Image.
getIconAlternateImagePtr
Return the address (as an Integer) of the Icon alternate Image.
getIconType
Return an Integer representing the type of the Icon. The returned
value is decoded as follows:

WBDISK = 1, WBDRAWER = 2

WBTOOL = 3, WBPROJECT = 4

WBGARBAGE = 5, WBDEVICE = 6

WBKICK = 7, WBAPPICON = 8

getDefaultTool

Return a String representing the Default Tool of the Icon (only Project & Disk Icon types have Default Tools).

getStackSize

Return an Integer representing the Stack Size of the Icon.

getWindowWidth

Return an Integer representing the Window Width of the Icon. Only Disk, Drawer & Garbage icon types have a valid value for this.

getWindowHeight

Return an Integer representing the Window Height of the Icon. Only Disk, Drawer & Garbage icon types have a valid value for this.

getWindowTopEdge

Return an Integer representing the Window TopEdge of the Icon. Only Disk, Drawer & Garbage icon types have a valid value for this.

getWindowLeftEdge

Return an Integer representing the Window LeftEdge of the Icon. Only Disk, Drawer & Garbage icon types have a valid value for this.

setIconWidth: newWidth

Change the Width of the Icon to newWidth.

WARNING: This method updates the Icon to disk, so do NOT change it unless you’re willing (or you have a backup) to live with the consequences.

setIconHeight: newHeight

Change the Height of the Icon to newHeight.

WARNING: This method updates the Icon to disk, so do NOT change it unless you’re willing (or you have a backup) to live with the consequences.

setIconFlags: newFlags

Change the Flags of the Icon to newFlags.

WARNING: This method updates the Icon to disk, so do NOT change it unless you’re willing (or you have a backup) to live with the consequences.

setIconImage: imageObject

Change the Image of the Icon to imageObject .

WARNING: This method updates the Icon to disk, so do NOT change

it unless you're willing (or you have a backup) to live
with the consequences.
setIconAlternateImage: imageObject
Change the alternate Image of the Icon to imageObject .
WARNING: This method updates the Icon to disk, so do NOT change
it unless you're willing (or you have a backup) to live
with the consequences.
setIconType: newType
Change the type of the Icon to newType.
WARNING: This method updates the Icon to disk, so do NOT change
it unless you're willing (or you have a backup) to live
with the consequences.
Valid types are:
WBDISK = 1, WBDRAWER = 2
WBTOOL = 3, WBPROJECT = 4
WBGARBAGE = 5, WBDEVICE = 6
WBKICK = 7, WBAPPICON = 8
setDefaultTool: newDefaultTool
Change the Default Tool of the Icon to newDefaultTool. Only
Disk & Project Icon types have a Default Tool.
WARNING: This method updates the Icon to disk, so do NOT change
it unless you're willing (or you have a backup) to live
with the consequences.
setStackSize: newStackSize
Change the Stack Size of the Icon to newStackSize.
WARNING: This method updates the Icon to disk, so do NOT change
it unless you're willing (or you have a backup) to live
with the consequences.
setWindowWidth: newWidth
Change the Width of the Icon Window to newWidth. Only Disk, Drawer
& Garbage Icon types have Windows.
WARNING: This method updates the Icon to disk, so do NOT change
it unless you're willing (or you have a backup) to live
with the consequences.
setWindowHeight: newHeight
Change the Height of the Icon Window to newHeight. Only Disk, Drawer
& Garbage Icon types have Windows.
WARNING: This method updates the Icon to disk, so do NOT change
it unless you're willing (or you have a backup) to live

with the consequences.
setWindowTopEdge: newTopEdge
Change the TopEdge of the Icon Window to newTopEdge. Only Disk,
Drawer & Garbage Icon types have Windows.
WARNING: This method updates the Icon to disk, so do NOT change
it unless you're willing (or you have a backup) to live
with the consequences.
setWindowLeftEdge: newLeftEdge
Change the LeftEdge of the Icon Window to newLeftEdge. Only Disk,
Drawer & Garbage Icon types have Windows.
WARNING: This method updates the Icon to disk, so do NOT change
it unless you're willing (or you have a backup) to live
with the consequences.
readInAsciiImage: fileName
The image read in will be placed in the Icon normal Image
(icon->do_Gadget.GadgetRender for you programmers), then the Icon
will be written to the file system.
The ASCII image file has the following format:
width, height, depth \n
datum \n datum \n datum \n ... \n<EOF>
Only writeAsciiImage: can create these types of files.
NOTE: This is NOT the same as an Image object (or an Intuition
Image)!
writeAsciiImage: fileName
Write the normal Icon Image (icon->do_Gadget.GadgetRender for you
programmers) to the fileName in the following format:
width, height, depth\n
datum \n datum \n datum \n ... <EOF>
Only readInAsciiImage: can make use of these types of files.
NOTE: This is NOT the same as an Image object (or an Intuition
Image)!
See Also Image Class

1.28 BitMap Class:

Class BitMap implements control of Amiga BitMaps for the Amigatalk
system. Valid values for Flags are:
BMB_CLEAR = 0
BMB_DISPLAYABLE = 1

BMB_INTERLEAVED = 2

BMB_STANDARD = 3

BMB_MINPLANES = 4

new: newBitMapName

Add an instance of a BitMap to the AmigaTalk system.

Default values are as follows:

name <- newBitMapName

width <- 1

height <- 1

depth <- 1

remove

Remove a BitMap from the AmigaTalk system.

getBitMapWidth

Return the width (in pixels) of the receiver.

getBitMapHeight

Return the height (in pixels) of the receiver.

getBitMapFlags

Return the Flags of the receiver.

getBitMapDepth

Return the depth (number of bitplanes) of the receiver.

setBitMapWidth: newWidth

Change the width (in pixels) of the receiver.

setBitMapHeight: newHeight

Change the height (in pixels) of the receiver.

setBitMapFlags: newFlags

Change the Flags of the receiver.

setBitMapDepth: newDepth

Change the depth (number of bitplanes) of the receiver.

readBitMapFile: bitMapFile

Load a BitMap from the given file. The file format is unique to

AmigaTalk.

writeBitMapFile: bitMapFile

Save a BitMap to the given file. The file format is unique to

AmigaTalk.

1.29 Painter Class:

Class Painter allows the user to draw simple graphics onto
Windows . The methods are:
new: newOwnerWindow
Initialize a new instance of Painter by setting the Window that will
be used by the Painter.
setLinePattern: newPatternMask
Set a pattern to use for drawing lines. For example, if
newPatternMask is 2r1100110011001100110011001100, a dashed
line will be drawn.
setAPen: pen
Change the foreground pen number of the given Window.
setBPen: pen
Change the background pen number of the given Window.
setOPen: pen
Change the outline pen number of the given Window.
setDrawMode: mode
Change the drawing mode of the given Window. Currently known
values are:
JAM1 = 0
JAM2 = 1
COMPLEMENT = 2
INVERSEVID = 4
getPens
Return the foreground pen & the background pen as a Point .
getOPen
Return an Integer representing the outline pen color Register number.
getDrawMode
Return an Integer representing the drawing mode (See setDrawMode:)
location
Return the coordinates, where the pen is located, as a Point .
ownerIs
Return an Object representing the Window that the Painter is attached to.
movePenTo: newPoint
Change the drawing Point (without drawing anything!) on the given
Window.
drawTo: aPoint
Draw a line from the current Window location to the given point.

drawLineFrom: fPoint to: tPoint

Draw a line from fPoint to tPoint.

drawBoxFrom: fPoint to: tPoint

Draw a box from fPoint to tPoint.

drawCircle: cPoint radius: r

Draw a circle of radius r with center point cPoint.

drawEllipse: cPoint minaxis: a maxaxis: b

Draw an ellipse at center point cPoint.

drawPolygon: borderObject

Draw a Border Object.

drawPixelAt: aPoint

Draw a single pixel at the point given.

drawText: iTextObject at: aPoint

Draw an IText at the point given.

NOTE: You should set the foreground pen, background pen & the

drawing mode before using this method!

initializeArea: numPoints tmpXSize: x tmpYSize: y

You MUST use this method BEFORE using any of the Area/Filled methods

down to disposeArea:y: (which MUST be used after you're done

with the Area/Filled method(s). x & y should specify the dimensions

of the largest rectangular area that will be drawn.

drawFilledEllipse: cPoint minaxis: a maxaxis: b

Same as drawEllipse:minaxis:maxaxis:, only the ellipse is a solid color.

drawFilledCircle: cPoint radius: r

Same as drawCircle:radius:, only the circle is a solid color.

areaMoveTo: newPoint

Same as movePenTo: method, only for Area/Filled methods.

areaDrawTo: aPoint

Same as drawTo: method, only for Area/Filled methods.

drawFilledBoxFrom: fPoint to: tPoint

This uses the RectFill() function.

floodFill: mode at: aPoint

If mode is 0 (outline mode), every pixel surrounding aPoint that is

NOT the outline Pen color will be changed to the flood Pen color (or

flood pattern). If the mode is 1 (color mode), whatever the color is

at aPoint and all surrounding pixels of the same color will be flood-

filled (or patterned).

areaEnd

Complete the Area/Filled polygons. Use this after

areaDrawTo:, drawFilledEllipse:minaxis:maxaxis:,
& drawFilledCircle:radius: only.
setAreaPattern: patternWords size: size
Similar to setLinePattern:
patternWords is a ByteArray that is divisible by two. Each pair of
bytes in patternWords is interpreted as a UWORD value.
size is a power of two, indicating how tall the pattern is,
which means that the number of elements in patternWords must
be 2 * 2 ^ size.
(example: 0 = two ByteArray elements (1 line),
1 = four elements (2 lines),
2 = 4 lines, 3 = 8 lines, 4 = 16, etc). No checking is done,
so get it right (or write your own method!).
outlineOff
Turn off usage of the outline Pen. Part of the Area/Filled methods.
outlineOn
Turn on usage of the outline Pen. Part of the Area/Filled methods.
disposeArea: xSize y: ySize
xSize & ySize MUST be the same dimensions that were used in the
initializeArea:tmpXSize:tmpYSize: method.

1.30 Image Class:

Class Image allows the user to draw Amiga Images
onto Windows . The methods are:
ownerIs
Return the Window object that contains the Image.
registerTo: windowObject
Set the parent Object of the receiver Image to windowObject.
disposeImage
Remove an Image from the AmigaTalk system.
drawImageAt: aPoint
Display an Image at the given aPoint coordinates.
drawImageAt: aPoint inState: state
Same as drawImageAt: except that the state of the Image can be
specified. Valid values for state are:
0 = IDS_NORMAL: // same as drawImageAt:
1 = IDS_SELECTED: // represents the 'selected state' of a Gadget
2 = IDS_DISABLED: // the 'ghosted state' of a gadget

3 = IDS_BUSY: // for future functionality

4 = IDS_INDETERMINATE: // for future functionality

5 = IDS_INACTIVENORMAL: // for gadgets in window border

6 = IDS_INACTIVESELECTED: // for gadgets in window border

7 = IDS_INACTIVEDISABLED: // for gadgets in window border

8 = IDS_SELECTEDDISABLED: // disabled and selected

pointInImage: testPoint

If the testPoint is inside the Image boundaries, return true.

eraseImageStartingAt: aPoint

Erase part or ALL of an Image.

setImageDataFrom: imageFile

Load Image data from the given file.

getStartPoint

Return the LeftEdge & TopEdge of the receiver as a Point .

getImageSize

Return the Width & Height of the receiver as a Point .

getImageDepth

Return the Depth of the receiver.

getImagePlanePick

Return the PlanePick variable of the receiver.

getImagePlaneOnOff

Return the PlaneOnOff variable of the receiver.

getNextImage

Return the next Image Object in the receiver.

setOrigin: aPoint

Set the LeftEdge & TopEdge of the receiver to the Point given.

setExtent: sizePoint

Set the Width & Height of the receiver to the Point given.

setImageDepth: newDepth

Set the Depth (range: 1 to 8) of the receiver.

setImagePlanePick: pp

Set the PlanePick variable of the receiver.

setImagePlaneOnOff: po

Set the PlaneOnOff variable of the receiver.

setNextImage: nextImage

Set the next Image Object in the receiver.

grabImageFrom: windowObj startPoint: s endPoint: e

Initialize the receiver data from the Window Object's display from

the given coordinates. WARNING: Make sure that the receiver

has enough room before you use this method. No checks for
overflowing the internal memory area can be performed!
getImageObject
Return an Image Object that represents the Receiver.
addImage: width height: h depth: d
Initialize the receiver Object. Use this method after:
myIMage <- Image new

1.31 IStruct Class:

Not Implemented yet!
Class IStruct allows access to various Amiga OS structures
used by Intuition. These are:
ViewPorts
Views
PlayFields
RastPorts
RasInfo
Blitter
Copper

1.32 Animation Class:

Not Implemented yet!
Class Anim