

**titlebar**

<b>COLLABORATORS</b>
----------------------

	<i>TITLE :</i> titlebar		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		July 31, 2024	

<b>REVISION HISTORY</b>
-------------------------

NUMBER	DATE	DESCRIPTION	NAME

# Contents

<b>1</b>	<b>titlebar</b>	<b>1</b>
1.1	titlebar.doc	1
1.2	titlebar.image/--background--	1
1.3	Description	1
1.4	Class documentation	2
1.5	Programming example	4
1.6	To do	5
1.7	History	5
1.8	Author	5
1.9	titlebar.image/titlebar.image	6
1.10	titlebar.image/ObtainTBIClass	7

# Chapter 1

## titlebar

### 1.1 titlebar.doc

```
--background--  
titlebar.image()  
ObtainTBIClass()
```

### 1.2 titlebar.image/--background--

Description  
Class documentation  
Programming example  
Future plans  
History  
Author

### 1.3 Description

#### DESCRIPTION

The titlebar.image is a shared library which, when opened, adds to the system a public BOOPSI image class called "tbiclass". This class implements several images suitable for the use as imagery of gadgets added to the titlebar of a window, such as an "iconify" image, a "pop-up" image, and so on. Also, a general purpose "empty" (border-only) image is provided, to be used as the background of more specialized titlebar images.

The purpose of "tbiclass" is to offer developers an easy way to implement the most common additional titlebar gadgets without having to draw or code their imagery themselves, and with the added benefit of standardizing the size and appearance of this kind of gadgets. For instance, until now any developer needing an iconify gadget had to hard-code its imagery into his application, leading to a plethora of similar, but different, versions of the same gadget.

The library is freeware; if you use it, you are allowed to distribute it

---

with your software, typically as a stand-alone file (you'll just have to add a section for it in your application's installation script).

Using "tbiclass" is very simple: you just need to open the titlebar.image library in order to add the class to the system, and then close it when you don't need the class anymore (and have freed all of its instances).

Once the class has been added to the system, you can use NewObject() in the usual way to create new "tbiclass" instances. See the "Class documentation" section to learn which tags are recognized by the class, as well as more information about the class in general.

Important note: a more refined version of "tbiclass" is also added to the system at boot time by the VisualPrefs utility (a patch to modify the look of the Amiga GUI). If the titlebar.image library, when it's opened, finds that a class named "tbiclass" already exists, it won't create another, but still will open successfully, so your code does NOT have to change to take this special case into account.

The only thing to remember is, you should NOT open the titlebar.image library if your program is launched in the Startup-sequence before the line where VisualPrefs is usually started (which is just before IPrefs), or else the patch won't run correctly. If you absolutely need to place your program in the Startup-sequence before VisualPrefs/IPrefs, at least offer the user an option to avoid opening the titlebar.image library. Anyway, this shouldn't be a problem for the vast majority of applications...

The class documentation applies to both versions of "tbiclass".

## 1.4 Class documentation

### CLASS DOCUMENTATION

```
Class:      tbiclass
Superclass: imageclass
Include File: <images/titlebar.h>
```

This is a class of images suitable for use as imagery of gadgets added to a window titlebar. As of version 40, there are 6 possible "tbiclass" images to choose from:

POPUPIIMAGE	A MUI "pop-up" titlebar gadget image
MUIIMAGE	A MUI "settings" titlebar gadget image
SNAPSHOTIMAGE	A MUI "snapshot" titlebar gadget image
ICONIFYIMAGE	An "iconify" titlebar gadget image
PADLOCKIMAGE	A DirOpus "padlock" titlebar gadget image
TBFRAMEIMAGE	A general-purpose empty titlebar gadget image

These image types are an extension to those already offered by "sysiclass"; in fact, "tbiclass" is used mostly the same way as "sysiclass" (see below for its methods and attributes).

It's worth noticing that all "tbiclass" image instances will have an Image->LeftEdge value of -1. This shouldn't be modified, and you should place your titlebar gadgets accordingly. The reason for this apparently

---

strange behavior is that Intuition titlebar gadget images, too, work this way, and we should try to stay as compatible with Intuition as possible.

Also, make sure you adjust your gadget's size if necessary, to adapt it to the returned image's size.

There are actually two versions of this class: a disk-based one, implemented by the freely distributable "titlebar.image" library, and another, more refined, added to the system by the VisualPrefs program at boot time. Applications needing this class should always ship with the "titlebar.image" library, and use it without worrying about the presence of VisualPrefs in the system: if it's running, its "tbiclass" version will be used automatically.

The only exception to this rule is when your application is run in the Startup-sequence before VisualPrefs; in this case, you should offer the user an option to avoid opening "titlebar.image", otherwise the disk-based version of the class would be added to the system and VisualPrefs wouldn't be able to add its own version. This is, however, a very unlikely scenario; in all foreseeable cases, an application opening windows shouldn't be run so early in the Startup-sequence.

New Methods:

None.

Changed Methods:

OM\_NEW - After creation of the image by the superclass, this method initializes the "tbiclass" instance data on the basis of the contents of the attributes tag list.

OM\_DISPOSE - This method frees all the resources allocated in the OM\_NEW method.

OM\_SET - This method allows to change a few of the image attributes and checks their bounds integrity.

OM\_GET - This method allows to retrieve the value of the most relevant image attributes.

IM\_DRAW - This method does the actual rendering of the image, taking into account its various attributes and the screen's DrawInfo pens.

IM\_DRAWFRAME - Like IM\_DRAW, but uses the size specified in the impDraw message rather than the one in the image structure.

Attributes:

SYSIA\_DrawInfo (IS) - This is absolutely mandatory. You MUST pass a DrawInfo pointer to "tbiclass" or NewObject() will fail.

SYSIA\_Which (ISG) - To specify which image you want; currently there are six image types, defined in <images/titlebar.h>: POPUPIMAGE, MUIIMAGE, SNAPSHOTIMAGE, ICONIFYIMAGE, PADLOCKIMAGE and TBFRAMEIMAGE.

---

- IA\_Width, IA\_Height (ISG) - Only recognized by the TBFRAMEIMAGE type; the other image types ignore them and always have the same size of the depth gadget image.
- SYSIA\_ReferenceFont (IS) - This is only recognized by the TBFRAMEIMAGE type; the other image types ignore it and always have the same height of the depth gadget image. It's also ignored if IA\_Height is explicitly used.
- TBIA\_ContentsBox (G) - To ask the image about the position and size of its actual "contents box" relative to the image itself. Said box is the part where, if you need to, you can add any further custom imagery. This probably only makes sense with TBFRAMEIMAGE images. To position correctly the box, you should add its Left and Top offsets to those of the image. Note that this attribute is useless for the disk-based version of the class, as the returned size will be always equal to the full image size; it is only relevant for the "tbiclass" version created by VisualPrefs. The value you pass to this tag must be a pointer to an IBox structure. Do NOT pass a longword pointer! (V40.12)

## 1.5 Programming example

### PROGRAMMING EXAMPLE

An example of usage of "titlebar.image" and "tbiclass" could be given by the following code fragment:

```
...
#include "images/titlebar.h"
#include "clib/titlebarimage_protos.h"
#include "pragmas/titlebarimage_pragmas.h"

...

struct Library *TitlebarImageBase;

...

/* Open the library */

if (!(TitlebarImageBase = OpenLibrary("Images/titlebar.image",40)))
{
    ComplainAndExit();
}

...

/* Create the image */

if (!(iconifyimage = NewObject(NULL,"tbiclass",SYSIA_Which,ICONIFYIMAGE,
                                SYSIA_DrawInfo,dri,
                                TAG_END)))
```

```
{
    iconifyimage = builtin_iconifyimage; /* If really needed... */
}

/* Use the image */

gad->GadgetRender = iconifyimage;
...

/* Free the image */

if (iconifyimage != builtin_iconifyimage) DisposeObject(iconifyimage);

...

/* Close the library */

CloseLibrary(TitlebarImageBase);

...
```

A complete example program can also be found in the distribution archive.

## 1.6 To do

TO DO

Future releases should add the following features:

- o More image types.
- o Some more code optimization.

And of course, I'm open to any suggestion about how to improve the class.

## 1.7 History

HISTORY

40.12 (7.10.99)

Added a new attribute to ask the image about its inner dimensions.

40.11 (7.9.98)

First public release.

## 1.8 Author

AUTHOR

Massimo Tantignone

---



Via Campagnoli, 4  
28100 Novara (NO)  
ITALY

E-mail: tanti@intercom.it  
Web: <http://www.intercom.it/~amigaws>

## 1.9 titlebar.image/titlebar.image

### NAME

titlebar.image -- create an image suitable for titlebar gadgets

### FUNCTION

The titlebar.image shared library implements a public BOOPSI image class, called "tbiclass", which provides the most commonly used images for gadgets added by applications to the titlebar of their windows. Examples of this are the ubiquitous "iconify" gadget or DirOpus 5's "padlock" gadget.

All programs using such class will automatically get the same look for their titlebar images. This is probably better than having myriads of different versions of the same image...

The titlebar.image library is freeware; the idea is that if your software needs images for gadgets in window titlebars, you can simply place the library into your distribution archive and let the user install it to his machine alongside your application, which will then be able to open and use it.

You can use "tbiclass" just like "sysiclass"; they're both sub-classes of "imageclass". A "tbiclass" image can be created in the usual way, by calling NewObject() with the tags described in the class reference documentation.

Of course, if NewObject() fails, you should provide a built-in fallback image for your titlebar gadget. However, as this occurrence isn't very likely (the class requires very little memory), you can keep your built-in images very simple.

To be able to create and use "tbiclass" objects in an application, you first have to open the titlebar.image library with OpenLibrary(). Once you have done this, you shouldn't close the library until all of your "tbiclass" objects have been freed.

For a complete description of the "tbiclass" class and its features, see the class reference documentation in the --background-- paragraph.

### TAGS

For the full list of "tbiclass" attributes, see the class reference documentation in the --background-- paragraph.

### WARNING

Don't close the titlebar.image library if you still have some instances of "tbiclass" images hanging around!

## BUGS

None known.

## 1.10 titlebar.image/ObtainTBIClass

## NAME

ObtainTBIClass -- return a pointer to "tbiclass".

## SYNOPSIS

```
tbiclass = ObtainTBIClass()  
D0
```

```
Class *ObtainTBIClass(void);
```

## FUNCTION

Returns a pointer to the class implemented by the titlebar.image, called "tbiclass". Use this pointer with care, and do not reference it after you've closed the titlebar.image library.

## INPUTS

none

## RESULT

tbiclass - pointer to the class

## NOTES

If VisualPrefs is running, you will obtain a pointer to its special version of "tbiclass".

## BUGS

None known.

## SEE ALSO