

in

COLLABORATORS

	TITLE : in		
ACTION	NAME	DATE	SIGNATURE
WRITTEN BY		July 31, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	in	1
1.1	bgui.guide	1
1.2	bgui.library/BGUI_AllocBitMap	1
1.3	bgui.library/BGUI_CreateRPortBitMap	3
1.4	bgui.library/BGUI_DoGadgetMethodA	4
1.5	bgui.library/BGUI_FillRectPattern	5
1.6	bgui.library/BGUI_FreeBitMap	6
1.7	bgui.library/BGUI_FreeClass	7
1.8	bgui.library/BGUI_FreeRPortBitMap	7
1.9	bgui.library/BGUI_GetCatalogStr	8
1.10	bgui.library/BGUI_GetClassPtr	10
1.11	bgui.library/BGUI_GetLocaleStr	10
1.12	bgui.library/BGUI_Help	12
1.13	bgui.library/BGUI_InfoText	12
1.14	bgui.library/BGUI_InfoTextSize	13
1.15	bgui.library/BGUI_LockWindow	14
1.16	bgui.library/BGUI_MakeClassA	15
1.17	bgui.library/BGUI_NewObjectA	17
1.18	bgui.library/BGUI_PackStructureTags	18
1.19	bgui.library/BGUI_PostRender	19
1.20	bgui.library/BGUI_RequestA	21
1.21	bgui.library/BGUI_UnlockWindow	23
1.22	bgui.library/BGUI_UnpackStructureTags	24

Chapter 1

in

1.1 bgui.guide

Search

TABLE OF CONTENTS

```
bgui.library/BGUI_AllocBitMap
bgui.library/BGUI_CreateRPortBitMap
bgui.library/BGUI_DoGadgetMethodA
bgui.library/BGUI_FillRectPattern
bgui.library/BGUI_FreeBitMap
bgui.library/BGUI_FreeClass
bgui.library/BGUI_FreeRPortBitMap
bgui.library/BGUI_GetCatalogStr
bgui.library/BGUI_GetClassPtr
bgui.library/BGUI_GetLocaleStr
bgui.library/BGUI_Help
bgui.library/BGUI_InfoText
bgui.library/BGUI_InfoTextSize
bgui.library/BGUI_LockWindow
bgui.library/BGUI_MakeClassA
bgui.library/BGUI_NewObjectA
bgui.library/BGUI_PackStructureTags
bgui.library/BGUI_PostRender
bgui.library/BGUI_RequestA
bgui.library/BGUI_UnlockWindow
bgui.library/BGUI_UnpackStructureTags
```

1.2 bgui.library/BGUI_AllocBitMap

NAME

BGUI_AllocBitMap -- Allocate a BitMap.

** V40.4 **

SYNOPSIS

```
bitmap = BGUI_AllocBitMap( width, height, depth, flags, friend )
```

D0 D0 D1 D2 D3 A0

```
struct BitMap *BGUI_AllocBitMap( ULONG, ULONG, ULONG, ULONG,
                                struct BitMap * )
```

FUNCTION

Allocates and initializes a BitMap structure and its bitplanes. Under OS 3.0 or later it reverts to the graphics.library/AllocBitMap() routine. On earlier versions of the OS it will allocate and initialize the BitMap structure and bitplanes itself.

INPUTS

`width` - The width in pixels of the desired BitMap.

height - The height in pixels of the desired BitMap.

depth - The number of bitplanes of the desired BitMap.

flags - BMF_CLEAR to specify that the allocated raster should be filled with color 0.

BMF_DISPLAYABLE to specify that this bitmap data should be allocated in such a manner that it can be displayed. Displayable data has more severe alignment restrictions than non-displayable data in some systems.

BMF_INTERLEAVED tells graphics that you would like your bitmap to be allocated with one large chunk of display memory for all bitplanes. This minimizes color flashing on deep displays. If there is not enough contiguous RAM for an interleaved bitmap, graphics.library will fall back to a non-interleaved one.

BMF_MINPLANES causes graphics to only allocate enough space in the bitmap structure for "depth" plane pointers. This is for system use and should not be used by applications use as it is inefficient, and may waste memory.

friend - See `graphics.library/AllocBitMap()`.

RESULT

A pointer to the allocated and initialized BitMap structure or NULL in case of a failure.

NOTES

Under OS 2.04 the "friend" parameter has no meaning. The only flag available under OS 2.04 is the BMF_CLEAR flag which will clear the bitplane data.

BUGS

None known.

SEE ALSO

```
bgui.library/BGUI_FreeBitMap(), graphics.library/AllocBitMap()
```

1.3 bgui.library/BGUI_CreateRPortBitMap

NAME

```
BGUI_CreateRPortBitMap -- Create a RastPort and BitMap at the same time.
```

```
** V40.4 **
```

SYNOPSIS

```
rport = BGUI_CreateRPortBitMap( src, width, height, depth )
D0                                A0    D0    D1    D2

struct RastPort *BGUI_CreateRPortBitMap( struct RastPort *, ULONG,
                                         ULONG, ULONG )
```

FUNCTION

To allocate and initialize a RastPort and BitMap. The routine will setup the RastPort attributes according to the source RastPort when available. If no source RastPort is made available the resulting RastPort is initialized by graphics.library/InitRastPort().

INPUTS

src	-	A pointer to the source RastPort from which the attributes for the created RastPort are copied. When this is not specified, the created RastPort is initialized with the graphics.library/InitRastPort() routine.
width	-	The width in pixels of the BitMap attached to the RastPort.
height	-	The height in pixels of the BitMap attached to the RastPort.
depth	-	The number of bitplanes of the BitMap attached to the RastPort.

RESULT

A pointer to the created RastPort structure or NULL upon failure.

NOTES

The created RastPort does not have a layer attached to it, so clipping is not possible. You are responsible to ensure that any rendering which occurs in the RastPort remains inside the RastPort bounds.

BUGS

None known.

SEE ALSO

```
bgui.library/BGUI_FreeRPortBitMap()
```

1.4 bgui.library/BGUI_DoGadgetMethodA

NAME

BGUI_DoGadgetMethodA -- Invoke a method on a BOOPSI gadget.
 BGUI_DoGadgetMethod -- Varargs version.

SYNOPSIS

```
result = BGUI_DoGadgetMethodA( gad, win, req, msg )
D0                                A0    A1    A2    A3

ULONG BGUI_DoGadgetMethodA( Object *, struct Window *,
                           struct Requester *, Msg )

ULONG BGUI_DoGadgetMethod( Object *, struct Window *,
                          struct Requester *, ULONG, ... )
```

FUNCTION

To invoke a method on a BOOPSI gadget in a way similar to the DoMethod() function in amiga.lib, with the advantage that context information for the gadget is included in the method by means of a GadgetInfo structure.

INPUTS

gad - A pointer to the BOOPSI gadget on which the method is invoked.

win - A pointer to the window on which the BOOPSI gadget is located.

req - Normally a pointer to the requester on which the BOOPSI gadget is located. Since BGUI does not support objects on requesters you should put NULL here.

msg - A pointer to the message to send to the BOOPSI gadget.

RESULT

The object does whatever it needs to do which may include updating it's visuals. The return value is defined per each method.

NOTES

This function invokes the specified method with a GadgetInfo derived from the 'win' and 'req' pointers. The GadgetInfo is passed as the second parameter of the message, except for OM_NEW, OM_SET, OM_NOTIFY, and OM_UPDATE, where the GadgetInfo is passed as the third parameter.

Custom gadget implemetors should take note of this. Always put the GadgetInfo in the second parameter of the message.

SEE ALSO

`intuition.library/DoGadgetMethod()`, `intuition.library/SetGadgetAttrs()`

1.5 bgui.library/BGUI_FillRectPattern

NAME

`BGUI_FillRectPattern` -- Fill a rectangle with a pattern

**** V41.3 ****

SYNOPSIS

```
BGUI_FillRectPattern( rport, pat, X1, Y1, X2, Y2)
                     A1      A0   D0  D1  D2  D3
```

```
void BGUI_FillRectPattern( struct RastPort *, struct bguiPattern *,
                          ULONG, ULONG, ULONG, ULONG)
```

FUNCTION

This function fills a given `RastPort` with the given `bguiPattern` within the constraints provided.

This function is similar to `graphics.library/BlitBitMapRastPort()` but allows for additional convenience and flexibility.

INPUTS

`rport` - A pointer to the `RastPort` that you wish to copy the pattern to.

`pat` - A `bguiPattern` structure that you must initialize before use:

`ULONG bp_Flags`

0, or one of the following flags:

`BPF_RELATIVE_ORIGIN`

Origin is relative to the box constraints given below.

`UWORD bp_Left, UWORD bp_Top`

Offset into the bitmap which follows.

`UWORD bp_Width, UWORD bp_Height`

Defines the size of the 'cut' taken from the bitmap.

`struct BitMap *bp_BitMap`

Pointer to the `BitMap` structure you wish to use in the fill. Naturally, it must be initialized and valid.

Object *bp_Object

Not currently used.

ULONG X1 -

ULONG Y1 - The upper-left corner of the RastPort you are filling.

ULONG X2 -

ULONG Y2 - The lower-right corner of the RastPort you are filling.

RESULT

No return value. If successful, the RastPort will be filled with the desired portion of the desired pattern.

NOTES

BUGS

None known.

SEE ALSO

libraries/bgui.h, graphics.library/BltBitMapRastPort()

1.6 bgui.library/BGUI_FreeBitMap

NAME

BGUI_FreeBitMap -- Free a BitMap created with BGUI_AllocBitMap().

** V40.4 **

SYNOPSIS

```
BGUI_FreeBitMap( bitmap )
                  A0
```

```
VOID BGUI_FreeBitMap( struct BitMap * )
```

FUNCTION

To deallocate a BitMap structure and bitplane data previously created with BGUI_AllocBitMap(). Before the bitmap is actually freed a call to WaitBlit() is made to make sure it is safe.

INPUTS

bitmap - A pointer to the BitMap structure.

BUGS

None known.

SEE ALSO

bgui.library/BGUI_AllocBitMap(), graphics.library/WaitBlit()

1.7 bgui.library/BGUI_FreeClass

NAME

BGUI_FreeClass -- Free a class created with BGUI_MakeClassA().

** V41.7 **

SYNOPSIS

```
success = BGUI_FreeClass ( class )
D0                                A0
```

```
BOOL BGUI_FreeClass ( Class * )
```

FUNCTION

Free memory and resources for a class that was created with BGUI_MakeClassA(). The class OM_DISPOSE method will be called before disposing the class as discussed in the description of the CLASS_ClassDispatcher parameter of BGUI_MakeClassA() function.

INPUTS

class - The class you wish to free up.

RESULT

TRUE if success, FALSE if not.

NOTES

BUGS

None known.

SEE ALSO

libraries/bgui.h, bgui.library/BGUI_MakeClassA()

1.8 bgui.library/BGUI_FreeRPortBitMap

NAME

BGUI_FreeRPortBitMap -- Free a RastPort created by BGUI_CreateRPortBitMap()

** V40.4 **

SYNOPSIS

```
BGUI_FreeRPortBitMap( rport )
                     A0
```

```
VOID BGUI_FreeRPortBitMap( struct RastPort * )
```

FUNCTION

To free the RastPort and BitMap created by an earlier call to the BGUI_CreateRPortBitMap() function.

INPUTS

rport - A pointer to the RastPort structure to free.

BUGS

None known.

SEE ALSO

bgui.library/BGUI_CreateRPortBitMap()

1.9 bgui.library/BGUI_GetCatalogStr

NAME

BGUI_GetCatalogStr -- Obtain a pointer to a localized string from a message catalog.

** V41.3 **

SYNOPSIS

```
String = BGUI_GetCatalogStr( bgui_loc, ID, default )
D0                      A0          D0  A1
```

```
STRPTR *BGUI_GetCatalogStr( struct bguiLocale *, ULONG, STRPTR );
```

FUNCTION

This function fetches a locale string from a specific localization catalog. If the localized string indicated by ID is not found, the default string is returned.

This function is similar to locale.library/GetCatalogStr() but allows for additional flexibility via hooks.

INPUTS

bgui_loc - A pointer to a bguiLocale structure that you have filled out:

```
struct Locale *bl_Locale
```

Not used by this function.

```
struct Catalog *bl_Catalog
```

A Catalog structure you initialize using the `locale.library` function `OpenCatalog()`. If it is `NULL`, the default string is returned.

```
struct Hook *blLocaleStrHook
```

Not used by this function.

```
struct Hook *blCatalogStrHook
```

If this is filled in, the function indicated is called to retrieve the string pointer. It is passed a `bguiCatalogStr` struct that contains the ID of the desired string and the default string itself.

If this is not filled in, `locale.library/GetCatalogStr()` is called.

ID - The number of the locale string you wish to retrieve.

default - The default string to return if the localized version is not found.

RESULT

A pointer to the desired string. This string is `READ ONLY` and cannot be changed by your program! This string is valid only as long as the catalog remains open.

If the ID is out of bounds, the default string will be returned.

NOTES

`locale.library` must be opened before using this function. If it is not, a `NULL` pointer will always be returned unless you use your own hook.

The catalog must be closed by you -- BGUI does not clean it up.

BUGS

None known.

SEE ALSO

`libraries/bgui.h`, `locale.library/OpenCatalog()`,
`locale.library/GetCatalogStr()`

1.10 bgui.library/BGUI_GetClassPtr

NAME

BGUI_GetClassPtr -- Obtain a pointer to a BGUI class.

SYNOPSIS

```
class = BGUI_GetClassPtr( classID )  
D0                                     D0
```

```
Class *BGUI_GetClassPtr( ULONG );
```

FUNCTION

This function is meant to provide class writers an easy way to obtain a pointer to one of the bgui.library classes. The pointer returned by this routine may only be used to subclass or to obtain objects from. Reading from or writing to the class structure is NOT allowed.

INPUTS

classID - The numeric ID of the class you need.

RESULT

A pointer to the requested class or NULL if the call was unsuccessful.

BUGS

None known.

SEE ALSO

libraries/bgui.h

1.11 bgui.library/BGUI_GetLocaleStr

NAME

BGUI_GetLocaleStr -- Obtain a pointer to a localized string.

** V41.3 **

SYNOPSIS

```
String = BGUI_GetLocaleStr( locale, ID )  
D0                                     A0      D0
```

```
STRPTR *BGUI_GetLocaleStr( struct bguiLocale *, ULONG );
```

FUNCTION

This function fetches a localized string for the given locale.

This function is similar to `locale.library/GetLocaleStr()` but allows for additional flexibility via hooks.

INPUTS

`bgui_loc` - A pointer to a `bguiLocale` structure that you have filled out:

```
struct Locale *bl_Locale
```

A `Locale` structure you initialize using the `locale.library` function `OpenLocale()`.

```
struct Catalog *bl_Catalog
```

Not used by this function.

```
struct Hook *blLocaleStrHook
```

If this is filled in, the function indicated is called to retrieve the string pointer. It is passed a `bguiLocaleStr` struct that contains the ID of the locale string.

If this is not filled in, `locale.library/GetLocaleStr()` is called.

```
struct Hook *blCatalogStrHook
```

Not used by this function.

`ID` - The number of the locale string you wish to retrieve.

RESULT

A pointer to the desired locale string. This string is READ ONLY and cannot be changed by your program! This string will remain valid until you close the locale.

If the ID is out of bounds, a NULL pointer will be returned.

NOTES

`locale.library` must be opened before using this function. If it is not, a NULL pointer will always be returned unless you use your own hook.

You are responsible for closing the locale once it is opened.

BUGS

None known.

SEE ALSO

`libraries/bgui.h`, `locale.library/OpenLocale()`,
`locale.library/GetLocaleStr()`

1.12 bgui.library/BGUI_Help

NAME

BGUI_Help -- Put up a simple synchronus AmigaGuide help file.

SYNOPSIS

```
success = BGUI_Help( win, file, node, line )  
D0          A0    A1    A2    D0
```

```
BOOL BGUI_Help( struct Window *, UBYTE *, UBYTE *, ULONG )
```

FUNCTION

To show additional online-help using the AmigaGuide system.

INPUTS

win - A pointer to the window from which the AmigaGuide help session is being invoked.

name - A pointer to the full path name of the amigaguide file.

node - A pointer to the node name to display.

line - The line number to display.

RESULT

TRUE uppon success and FALSE if something went wrong.

BUGS

None known.

SEE ALSO

amigaguide.library/OpenAmigaGuideA()

1.13 bgui.library/BGUI_InfoText

NAME

BGUI_InfoText -- Render text using BGUI infoclass command sequences.

** V40.8 **

SYNOPSIS

```
BGUI_InfoText( rp, text, bounds, drawinfo )  
              A0  A1    A2    A3
```

```
void BGUI_InfoText( struct RastPort *, UBYTE *, struct IBox *,
```

```
struct DraInfo * )
```

FUNCTION

To render text with infoclass command sequences inside the given bounds. The rendering routine will automatically truncate all text that will not fit inside the given bounds.

Also the text, when possible, is centered vertically in the given bounds.

INPUTS

- rp - A pointer to the RastPort in which the text is to be rendered.
- text - A pointer to the text to render. All infoclass command sequences are supported. They are:
 - \33b - Bold text.
 - \33i - Italics text.
 - \33u - Underlined text.
 - \33n - Normal text.
 - \33c - Center this and the following text lines.
 - \33l - Left-justify this and the following text lines.
 - \33r - Right-justify this and the following text lines.
 - \33d<n> - Set drawinfo pen <n>.
 - \33p<n> - Set pen <n>.
 - \n - Start a new line of text.

Please note that the \33c, \33l and \33r command sequences can only be used at the beginning of a new line.

- bounds - A pointer to a struct IBox in which the bounds of the area in which is rendered are described.
- drawinfo - A pointer to the screen's DrawInfo structure. This is used to find pen information for the screen. When this is NULL a default set of pens is used.

RESULT

None.

BUGS

None known.

SEE ALSO

bgui.library/BGUI_InfoTextSize

1.14 bgui.library/BGUI_InfoTextSize

NAME

BGUI_InfoTextSize -- Get pixel size of text with command sequences.

** V40.8 **

SYNOPSIS

```
BGUI_InfoTextSize( rp, text, width, height )
                   A0  A1    A2    A3
```

```
void BGUI_InfoTextSize( struct RastPort *, UBYTE *, UWORD *, UWORD * )
```

FUNCTION

To compute the complete width and height of a text with BGUI infoclass command sequences. Result is in pixels.

INPUTS

rp - The RastPort which is used to base the computations on.

text - A pointer to the text.

width - A pointer to storage space to hold the pixel width of the text. May be NULL in which case the width is not computed.

height - A pointer to storage space to hold the pixel height of the text. May be NULL in which case the height is not computed.

RESULT

None

BUGS

None known.

SEE ALSO

bgui.library/BGUI_InfoText

1.15 bgui.library/BGUI_LockWindow

NAME

BGUI_LockWindow -- Disable a window from receiving IDCMP messages.

SYNOPSIS

```
lock = BGUI_LockWindow( win )
D0                       A0
```

```
APTR BGUI_LockWindow( struct Window * )
```

FUNCTION

To disable a window from receiving IDCMP messages at its message port. This is done by putting up an invisible requester and a busy pointer (same as the workbench uses). The only thing possible with locked windows is moving it with the dragbar and depth gadget.

INPUTS

win - A pointer to the window to lock.

RESULT

Lock will point to some private data if successfull or NULL if not.

BUGS

None known.

SEE ALSO

bgui.library/BGUI_UnlockWindow()

1.16 bgui.library/BGUI_MakeClassA

vNAME

```
BGUI_MakeClassA -- Set up a new class.
BGUI_MakeClass  -- VarArgs version

** V41.7 **
```

SYNOPSIS

```
BGUI_MakeClass ( tag1, ...)
                A0

BGUI_MakeClassA( tagitem )
                A0

Class *BGUI_MakeClass ( ULONG, ...)

Class *BGUI_MakeClassA( struct TagItem *)
```

FUNCTION

For class implementors only.

This function creates a new BGUI class. The superclass should be defined to be another BGUI class; all classes are decendants of the 'rootclass.'

INPUTS

The following tags are supported:

```
CLASS_SuperClassBGUI
    (ULONG) The class ID of the class' superclass IF the class is a
```

subclass of one of the known BGUI classes.

CLASS_SuperClass

(ULONG) If CLASS_SuperClassBGUI is missing, this must be set to a pointer to an already created class (superclass).

CLASS_SuperClassID

(STRPTR) If CLASS_SuperClassBGUI is missing and if CLASS_SuperClass is missing or it is NULL, this may be set to the name (character string) of any existing superclass of the class to be created. The default value is "rootclass". This is the same as required for intuition.library/MakeClass() and will be passed to it.

CLASS_ClassID

(STRPTR) The name (character string) of the class. Only required if this class is to be created is to be made public. This is the same as required for intuition.library/MakeClass() and will be passed to it.

CLASS_ClassSize

(ULONG) The size in bytes of the class private data space. That memory space will be allocated and pointed to by the class cl_UserData field. Class implementors should take advantage of this data space to store class specific global data.

CLASS_ObjectSize

(ULONG) The size in bytes of the instance data of each object of the class.

CLASS_Flags

(ULONG) Flags; these are the same as required for intuition.library/MakeClass() and will be passed to it. Defaults to 0.

CLASS_Dispatcher

(HOOKFUNC) Pointer to the dispatcher function for objects of this class. This is NOT a callback hook, but a function pointer. The function pointed to must accept data as follows:

REG(A0) Class *cl, REG(A2) Object *obj, REG(A1) Msg msg

If this tag is not specified, the default jump table defined in CLASS_DFTable is used.

CLASS_DFTable

(DPFUNC *) Pointer to the dispatcher's function lookup table. This table lists all methods that the class can perform and a pointer to that method.

CLASS_ClassDispatcher

(HOOKFUNC) Pointer to the dispatcher for this class (same format as CLASS_Dispatcher). This dispatcher is used to call class specific methods. This is an extension to intuition BOOPSI classes. Besides the object methods, the classes may have their own methods.

Currently, only OM_NEW and OM_DISPOSE are supported by BGUI. OM_NEW is called after a class is successfully created right before it returns from BGUI_MakeClass. BGUI will pass to OM_NEW the pointer of the class just created and NULL as object instance pointer.

Class implementors should use this method to initialize the private class data structure (given by the cl_UserData field) and allocate any resources needed globally by the class' objects, like for instance external library pointers. If the initialization succeeds the method should return TRUE. Otherwise, BGUI_MakeClass will fail disposing the class that was created before calling OM_NEW.

OM_DISPOSE is called by BGUI_FreeClass. Class implementors should use it dispose any resources allocated during the class life time. If it fails the method should return FALSE, the class is not freed and BGUI_FreeClass fails.

CLASS_ClassDFTable

(DPFUNC *) Pointer to the class dispatcher's function lookup table array. This array lists all methods that the class supports and function pointers to each of the methods. (same format as CLASS_DFTable)

RESULT

A pointer to a valid Class structure, or NULL if the class could not be created.

NOTES

The global data pointer that is stored in register A4 is restored upon exiting this function; thus, you need not use __saveds in your method dispatcher functions.

BUGS

None known.

SEE ALSO

libraries/bgui.h, bgui.library/FreeClass()

1.17 bgui.library/BGUI_NewObjectA

NAME

BGUI_NewObjectA -- Create an object of a specific class.
BGUI_NewObject -- Varargs version.

SYNOPSIS

```

object = BGUI_NewObjectA( classID, tags )
D0                      D0          A0

Object *BGUI_NewObjectA( ULONG, struct TagItem * )

object = BGUI_NewObject( classID, tag1, ... )

Object *BGUI_NewObject( ULONG, Tag, ... )

```

FUNCTION

This routine is a replacement routine for Intuition's NewObjectA() call. It is an easy way to obtain an object from any of the BGUI classes. You pass it a classID and some create time attributes and the routine will return you a pointer to the created object.

INPUTS

```

classID  - The numeric ID of the class.
tags     - A set of create-time attributes which will be passed to the
           class from which the object is created.

```

RESULT

A pointer to the created object or NULL if an error occurred.

BUGS

None known.

SEE ALSO

intuition.library/NewObjectA(), libraries/bgui.h

1.18 bgui.library/BGUI_PackStructureTags

NAME

BGUI_PackStructureTags -- Pack a structure with values from a TagList.

** V41.8 **

SYNOPSIS

```

num = BGUI_PackStructureTags (pack, pTable, taglist)
D0                      A0      A1      A2

ULONG BGUI_PackStructureTags( APTR, ULONG *, struct TagItem *)

```

FUNCTION

For each pTable entry, a FindTagItem() will be done and if the matching tag is found in the taglist, the data will be packed into the given

structure based on the packtable definition.

INPUTS

`pack` - A pointer to the data space to be packed into.

`pTable` - A pointer to the packing information table. See `utility/pack.h` for definition and macros.

`taglist` - A pointer to the `TagList` to pack from.

RESULT

The number of tags packed.

NOTES

if the user is using V39 or later of the OS, `utility.library/PackStructureTags()` will be used. Otherwise, internal `bgui.library` functions will be used.

BUGS

None known.

SEE ALSO

`libraries/bgui.h`, `bgui.library/BGUI_UnpackStructureTags()`, `utility/pack.h`, `utility.library/PackStructureTags()`

1.19 bgui.library/BGUI_PostRender

NAME

`BGUI_PostRender` -- Perform operations on a class after rendering.

**** V41.6 ****

SYNOPSIS

```
BGUI_PostRender( class, object, gp_render)
                 A0      A2      A1
```

```
void BGUI_PostRender( struct Class *, struct Object *,
                     struct gpRender * )
```

FUNCTION

This function is only for class implementors that subclass from baseclass and implement the `GM_RENDER` method.

It is called with the same arguments that `GM_RENDER` was called with, and should be called you exit the rendering function. Do not call it if the superclass method failed. Call it in all other cases.

INPUTS

class - A pointer to a Class structure that the object belongs to.

obj - A pointer to the object.

gpr - A pointer to a gpRender structure with the fields set up as appropriate:

- ULONG MethodID
Set to GM_RENDER
- struct GadgetInfo *gpr_GInfo
Set to point to the object's GadgetInfo structure.
- struct RastPort *gpr_RPort
Set to point to the object's RastPort
- LONG gpr_Redraw
Redraw method one of the following three values:
 - GREDRAW_REDRAW
Redraw the entire gadget
 - GREDRAW_UPDATE
The object's imagery has changed, possibly due to the user manipulating the object. Only the part of the imagery that was affected will be redrawn (e.g., a slider object)
 - GREDRAW_TOGGLE
If the object supports it, toggle between the selected and unselected state.

RESULT

No returned result. If successful, the object will re redrawn as needed.

NOTES

This function was implemented in V41.6 to support recursive clipping and buffering with virtual groups in mind.

Do not use objects in virtual groups if their classes do not call this function. If you must use an old pre-compiled class, you'll have to use it with the externalclass.

BUGS

None known.

SEE ALSO

libraries/bgui.h, intuition/gadgetclass.h

1.20 bgui.library/BGUI_RequestA

NAME

BGUI_RequestA -- Put up a text requester.
 BGUI_Request -- Varargs version.

SYNOPSIS

```
gadid = BGUI_RequestA( win, req, args )
D0                      A0  A1  A2

ULONG BGUI_RequestA( struct Window *, struct bguiRequest *, ULONG * )

gadid = BGUI_Request( win, req, arg1, ... )

ULONG BGUI_Request( struct Window *, struct bguiRequest *, ULONG, ... )
```

FUNCTION

To put up a requester. It is typically the same as Intuition's EasyRequestArgs() only this routine allows you to put InfoClass style command sequences in the body text and keyboard shortcuts for the gadgets.

INPUTS

win - A pointer to the window on which the requester will open. This may be NULL.

req - A pointer to an initialized bguiRequest structure. This structure is similar to Intuition's EasyStruct structure. It is used to control the general look of the requester. The structure is initialized with the following data:

br_Flags - This field can contain any of the following flags:

BREQF_CENTERWINDOW

This will center the requester over the window 'win' if a valid pointer to a window is passed.

BREQF_LOCKWINDOW

This will disable the window on which the requester appears from receiving any IDCMP messages. A busy pointer is also set on that window. NOTE: 'win' must point to a window for this to work.

BREQF_NO_PATTERN

This will suppress the backfill pattern.

BREQF_XEN_BUTTONS

When set this flag will make the buttons framing appear as XEN style framing.

BREQF_AUTO_ASPECT

When set all the requester will make some aspect ratio dependant changes to the GUI like thin/thick frames etc.

BREQF_FAST_KEYS

This flag tells BGUI to use the Return and Esc key as default positive/negative response to the requester. Please note that no visual confirmation is given at this time when the key is pressed.

`br_Title` - A pointer to the title of the requester. If this is NULL the title of the window is used if one is present. As a final default, "BGUI Request" or its localized equivalent is used.

`br_GadgetFormat` - A pointer to the gadget label string. The gadget labels are separated by a '|' character. I.E "OK|Cancel" will give you an "OK" and a "Cancel" gadget.

When you precede the gadget label with a '*' the label of the gadget in question will be rendered as bold text, and also will automatically be the default response of the Return key when the BREQF_FAST_KEYS flag is set (see above).

`br_TextFormat` - A printf-style formatting string which may also contain InfoClass style command sequences.

`br_ReqPos` - The position at which the requester will be opened. There are three possibilities:

POS_CENTERSCREEN

Center requester on the window's (if any) screen.

POS_CENTERMOUSE

Center requester under mouse pointer, if possible.

POS_TOPLEFT

Open requester in the top left corner of

the window's screen (similar to how Intuition's EasyRequest does it).

NOTE: The BREQF_CENTERWINDOW flag will override this setting.

- br_Underscore - With this field you can set the character which preceeds the character to underline. The underlined character will automatically become the key which activates the gadget.
- br_Reserved0 - This field is for future expansion and `_must_` be set to zero.
- br_Screen - Here you can optionally specify the Screen on which the requester must appear. By default the given window's Screen is used if a valid Window pointer is given. If no Window is given then this Screen is used. If this field is NULL the default Public Screen is used.
- br_Reserved1 - These fields are for future expansion and `_must_` be zero'd.
- args - A pointer to an array of arguments for the C-style formatting codes.

RESULT

1, 2, 3, 4, 0.

You will be returned a value ranging from 0 to the count of gadgets minus one. NOTE: The right most gadget will always return 0.

SEE ALSO

intuition.library/EasyRequestArgs(), intuition/intuition.h,
libraries/bgui.h, infoclass.doc

1.21 bgui.library/BGUI_UnlockWindow

NAME

BGUI_UnlockWindow -- Enable a window that was previously disabled.

SYNOPSIS

```
BGUI_UnlockWindow( lock )
                  A0
```

```
VOID BGUI_UnlockWindow( APTR )
```

FUNCTION

To enable a window to receive IDCMP messages on its message port. This routine must be used to 'unlock' windows locked with `BGUI_LockWindow()`.

INPUTS

`lock` - A pointer to the data returned by `BGUI_LockWindow()`. This may be NULL.

RESULT

The window will be unlocked.

BUGS

None know.

SEE ALSO

`bgui.library/BGUI_LockWindow()`

1.22 `bgui.library/BGUI_UnpackStructureTags`

NAME

`BGUI_UnpackStructureTags` -- unpack a structure into values in a TagList.

**** V41.8 ****

SYNOPSIS

```
num = BGUI_UnpackStructureTags (pack, pTable, taglist)
D0                                A0    A1    A2
```

```
ULONG BGUI_UnpackStructureTags( APTR, ULONG *, struct TagItem *)
```

FUNCTION

For each `pTable` entry, a `FindTagItem()` will be done and if the matching tag is found in the taglist, the data in the structure will be placed into the memory pointed at by the tag's `ti_Data`. `ti_Data` MUST be a LONGWORD.

INPUTS

`pack` - A pointer to the data space to be unpacked.

`pTable` - A pointer to the packing information table. See `utility/pack.h` for definition and macros.

`taglist` - A pointer to the TagList to unpack into.

RESULT

The number of tags unpacked.

NOTES

if the user is using V39 or later of the OS,
utility.library/UnpackStructureTags() will be used. Otherwise, internal
bgui.library functions will be used.

BUGS

None known.

SEE ALSO

libraries/bgui.h, bgui.library/BGUI_PackStructureTags(), utility/pack.h,
utility.library/UnpackStructureTags()
