GadLayout

COLLABORATORS			
	TITLE : GadLayout		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 27, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

GadLayout

1.1 gadlayout.doc

```
FreeLayoutGadgets()
GadgetArrayIndex()
GL_SetGadgetAttrsA()
LayoutGadgetsA()
```

1.2 gadlayout/FreeLayoutGadgets

```
NAME
    FreeLayoutGadgets -- Frees gadgets laid out with LayoutGadgets().

SYNOPSIS
    FreeLayoutGadgets(gad_info);
    VOID FreeLayoutGadgets(APTR);

FUNCTION
    Frees all resources used in creating and laying out gadgets with
    LayoutGadgets().  This frees all gadgets as well as other
    resources used.  Generally this will be called after a call to
    CloseWindow() in Intuition.

INPUTS
    gad_info - The pointer returned by LayoutGadgets().

SEE ALSO
    LayoutGadgetsA()
```

1.3 gadlayout/GadgetArrayIndex

```
NAME
    GadgetArrayIndex -- Get a gadget's index in the LayoutGadget array.

SYNOPSIS
    i = GadgetArrayIndex(gad_id, gadgets)
```

```
    WORD GadgetArrayIndex(WORD, struct LayoutGadget *)

FUNCTION
    Given a gadget ID, returns the index of that gadget's definition
    in the LayoutGadget array.  For example, in cases where you need
    to know a gadget's Gadget structure (eg. if you wanted to use
    the Intuition function ActivateGadget() to make a string or an
    integer gadget active), you would need to lookup the lg_Gadget
    field in the LayoutGadget array.  You MUST NOT GIVE THE ARRAY
    INDEX YOURSELF, THIS IS NOT GUARUNTEED TO REMAIN VALID!  Instead,
    pass the id of the gadget that you want and this function will
    return the array index for you.

INPUTS
    gad_id - The ID of the gadget you want to find.
    gadgets - The LayoutGadget array that this gadget is defined in.

RESULT
    i - The index into the LayoutGadget array of the entry of the
       gadget ID you asked for.
```

1.4 gadlayout/GL_SetGadgetAttrsA

```
NAME
    GL_SetGadgetAttrsA -- Change attributes of a GadLayout gadget.
    GL_SetGadgetAttrs -- Varargs stub for GL_SetGadgetAttrsA.

SYNOPSIS
    GL_SetGadgetAttrsA(gad_info, gad, win, req, taglist)
    VOID GL_SetGadgetAttrsA(APTR, struct Gadget *, struct Window *,
                            struct Requester *, struct TagItem *)

    GL_SetGadgetAttrs(gad_info, gad, win, req, firsttag, ...)
    VOID GL_SetGadgetAttrs(APTR, struct Gadget *, struct Window *,
                           struct Requester *, Tag *, ...)

FUNCTION
    Changes attributes for one of the GadLayout gadget kinds according
    according to the attributes chosen in the tag list.

INPUTS
    gad_info - The value returned by LayoutGadgetsA().
    gad - Pointer to the gadget in question.
    win - Pointer to the window containing the gadget.
    req - Pointer to the requester containing the gadget, or NULL if
        not in a requester. (Not implemented yet, use NULL.)
    taglist - Pointer to a TagItem list.

TAGS
    IMAGEBUTTON_KIND:
    GLIM_Image (struct Image *) - Changes the image displayed in the
        gadget.

BUGS
    This function is not compatable with itself in versions releases
```

```
    1.5 and lower, because of the new pi parameter!  ALL OLD CODE
    WILL HAVE TO BE CHANGED!!!

    Attributes not pertaining to a specific gadget kind will not
    always be ignored, so you will need to be careful that you only
    try to change attributes that are valid for the gadget's kind.
```

1.5 gadlayout/LayoutGadgetsA

```
NAME
    LayoutGadgetsA -- Formats an array of GadTools gadgets.
    LayoutGadgets -- Varargs stub for LayoutGadgetsA().

SYNOPSIS
    gad_info = LayoutGadgetsA(gad_list, gadgets, screen, taglist)
    APTR LayoutGadgetsA(struct Gadget **, struct LayoutGadget *,
                        struct Screen *, struct TagItem *)

    gad_info = LayoutGadgets(gad_list, gadgets, screen, firsttag, ...)
    APTR LayoutGadgets(struct Gadget **, struct LayoutGadget *,
                       struct Screen *, Tag *, ...)

FUNCTION
    Creates a laid-out gadget list from a LayoutGadget array, which
    describes each gadget you want to create.  Gadgets you create
    can be any of the gadget kinds supported by GadTools, as well
    as any of the extended gadget kinds provided by GadLayout.
    Gadgets can easily be defined so that the automatically adjust
    their sizes and positions to accomodate fonts of any size
    (including proportional fonts) and also to adapt to different
    locale strings.  The real power of GadLayout is that allows you
    to create a gadget layout that dynamically adjusts to different
    user's environments.

INPUTS
    gad_list - Pointer to the gadget list pointer, this will be
        ready to pass to OpenWindowTags() or AddGList().

    gadgets - An array of LayoutGadget structures.  Each element
        in the array describes one of the gadgets that you will
        be creating.  Each LayoutGadget structure in the array
        should be initialized as follows:

        lg_GadgetID - The ID for this gadget.
        lg_LayoutTags - A taglist consisting of the following tags:
            GL_GadgetKind (ULONG) - Which gadget kind to use.  This
                may be any of the GadTools gadget kinds (defined in
                libraries/gadtools.h), or one of the additional kinds
                provided by GadLayout, which are:
                IMAGEBUTTON_KIND : A button gadget with that uses an
                                   Intuition Image structure for its
                                   contents.  The image will be centred
                                   automatically.
                DRAWER_KIND : A drawer button gadget.  Use this to
                              allow the user to use the ASL file
```

```
                    requester to select a path.
        FILE_KIND : A file button gadget.  Use this to allow
                    the user to use the ASL file requester
                    to select a file.
        Additional kinds may be added in the future.
GL_Width (WORD) - Absolute gadget width, in pixels.
GL_DupeWidth (UWORD) - Duplicate the width of another
    gadget.
GL_AutoWidth (WORD) - Set width according to length of
    text label + ti_Data.  Note that this function
    does not take into account the amount of space any
    gadget imagery might take within the gadgets area.
GL_Columns (UWORD) - Set width of gadget so that
    approximately ti_Data columns of text with the
    gadget's font will fit.  This will only be an
    approximation, because with proportional fonts the
    width of character varies.  Note that this function
    does not take into account the amount of space any
    gadget imagery might take within the gadgets area.
GL_AddWidth (WORD) - Add some value to the total width
    calculation.
GL_MinWidth (WORD) - Make sure that the final width of
    the gadget is at least this.
GL_MaxWidth (WORD) - Make sure that the final width of
    the gadget is at most this.
GL_Height (WORD) - Absolute gadget width.
GL_HeightFactor (UWORD) - Make the gadget height a multiple
    of the font height (useful for LISTVIEW_KIND gadgets).
GL_AutoHeight (WORD) - Set height according to height of
    text font + ti_Data.
GL_AddHeight (WORD) - Add some value to the total height
    calculation.
GL_MinHeight (WORD) - Make sure that the final height of
    the gadget is at least this.
GL_MaxHeight (WORD) - Make sure that the final height of
    the gadget is at most this.
GL_Top (WORD) - Absolute top edge.
GL_TopRel (UWORD)  - Top edge relative to bottom edge of
    another gadget (specified by its gadget ID).
GL_AdjustTop (WORD) - ADD the height of the text font +
    ti_Data to the top edge (often used to to properrly
    position gadgets that have their label above).
GL_AddTop (WORD) -  Add some value to the final top edge
    calculation.
GL_Bottom (WORD) - Absolute bottom edge.
GL_BottomRel (UWORD) - Bottom edge relative to top edge of
    another gadget (specified by its gadget ID).
GL_AddBottom (WORD) - Add some value to the final bottom edge
    calculation.
GL_Left (WORD) - Absolute left edge.
GL_LeftRel (UWORD) - Left edge relative to right edge of
    another gadget (specified by its gadget ID).
GL_AdjustLeft (WORD) - ADD the width of the text label +
    ti_Data to the left edge.
GL_AlignLeft (UWORD) - Align the left edge of the gadget
    with the left edge of another gadget (specified by its
    gadget ID).
```

```
            GL_AddLeft (WORD) - Add some value to the final left edge
                calculation.
            GL_Right (WORD) - Absolute right edge.
            GL_RightRel (UWORD) - Right edge relative to left edge of
                another gadget (specified by its gadget ID).
            GL_AlignRight (UWORD) - Align the right edge of the gadget
                with the right edge of another gadget (specified by its
                gadget ID).
            GL_AddRight (WORD) - Add some value to the final right edge
                calculation.
            GL_GadgetText (STRPTR) - Gadget text label.
            GL_TextAttr (struct TextAttr *) - Desired font for gadget
                label, will override the GL_DefTextAttr if used.
            GL_Flags - (ULONG) Gadget flags.
            GL_UserData (VOID *)- Gadget UserData.
            GL_LocaleText - Gadget label taken from a locale catalog,
                you supply the locale string ID.  If you use this tag
                you MUST have used GL_AppStrings in your call to
                LayoutGadgets().

            If you've specified one of GadLayout's own gadget kinds
            with GL_GadgetKind, the following tags are available for
            defining attributes of those gadgets:

            GLIM_Image (struct Image *) - Provide a pointer to the
                Image structure to to be used in an IMAGEBUTTON_KIND
                structure.  This pointer only need be valid when
                LayoutGadgets() is called.
            GLIM_ReadOnly (BOOL) - Specifies that the gadget is read-
                only.  It will get a recessed border and will not be
                highlighted when clicked on.

            Generally you need only specify the tags when the data
            has changed from the previously gadget.  This gets a
            little tricky when you use the relation tags like
            GL_TopRel, as this means that gadgets will not be
            processed in sequential order necessarily.

        lg_GadToolsTags - When defining a GadTools gadgets, you
            can pass a GadTools taglist to set options for that
            gadget.  This would be the same set of tags that you
            might pass to CreateGadgetA() if you were using GadTools
            directly.
        lg_Gadget - The pointer to the Gadget structure created for
            this gadget will be placed here.  You should initialize
            this field to NULL.  WARNING: The gadget structure
            created READ-ONLY!

    screen - A pointer to the screen that the gadgets will be
        created for.  The is required so that the layout routines
        can get display info about the screen, no rendering will
        be done.

    taglist - Pointer to a TagItem list (see below for allowed tags)

TAGS
    GL_RightExtreme (LONG *) - A pointer to a LONG where GadLayout
```

will put the co-ordinate of the rightmost point where any imagery of the laid-out gadgets will be drawn. Use this to open a window exactly big enough to hold all your gadgets. Use this value alone with the WA_InnerWidth window tag and NOT WA_Width, since you do not know how big the window border will be.

GL_LowerExtreme (LONG *) - A pointer to a LONG where GadLayout will put the co-ordinate of the lowermost point where any imagery of the laid-out gadget will be drawn. Use this to open a window exactly big enough to hold all your gadgets. Use this value alone with the WA_InnerHeight window tag and NOT WA_Height, since you do not know how big the window border will be.

GL_DefTextAttr (struct TextAttr *) - Instead of having to indicate a TextAttr for each gadget, you can specify a font to be used by default for all your gadgets.

GL_Catalog (struct Catalog *) - Specify the locale catalog to use to get your strings from. If you wish to localize your gadget string via GL_LocaleText you MUST use this tag as well as GL_AppStrings. You must also make certain that locale.library has been opened successfully with LocaleBase pointing to the library base.

GL_AppStrings (struct AppString **) - If you wish to make your gadgets localized, you you must pass a list of strings and their IDs. The format of these strings is an array of structures, with a LONG that contains the ID and a STRPTR pointing to the string, i.e.:

```
struct AppString
{
    LONG   as_ID;
    STRPTR as_Str;
};
```

These strings serve as the default language for the gadgets. See locale.library documentation for more information on localizing applications. You MUST use this tag in addition to GL_Catalog if you wish to use GL_LocaleText to localize your gadgets.

GL_NoCreate (BOOL) - Set to TRUE if you don't want the layout routine to actually create any gadgets. This is used when you want to use the GL_RightExtreme and GL_LowerExtreme tags to find out how much space your gadgets will take, but don't actually want to create the gadgets just yet.

GL_BorderTop (UWORD) - The size of the top border of your window. If your window does not have the WFLG_GIMMEZEROZERO flag set, it will be necessary to pass the size of the window borders. This value can be gotten either from the Window structure of your window (if it is already open), or from the Screen structure of your screen (see intuition/screens.h for details about this). NOTE: This value is NOT added to the value returned by GL_LowerExtreme!

GL_BorderLeft (UWORD) - The size of the left border of your window. If your window does not have the WFLG_GIMMEZEROZERO flag set, it will be necessary to pass the size of the window borders. This value can be gotten either from the Window structure of your window (if it is already open), or from the Screen structure of your screen (see intuition/screens.h for details about this). NOTE: This value is NOT added to the value

returned by GL_RightExtreme.

RESULT

gad_info - A pointer to a private structure. You must keep this value and pass it to FreeLayoutGadgets() later on in order to free up all resources used by your gadgets.

NOTES

You must be careful with the taglist in the lg_LayoutTags field. Tags are processed sequentally in the order you give them in, and if a tag references another gadget (eg. the GL_TopRel tag), then processing of the current gadget halts while the referenced gadget is processed (if it has not already been processed). Problems can arise if this gadget refers back to the original gadget that referenced it, if it is referring to a field that has not yet been processed in that gadget. For example, gadget GAD_BUTTON1 may use the GL_TopRel tag to refer to GAD_BUTTON2, which may subsequently make use of GL_LeftRel to refer back to GAD_BUTTON1. The gadgets left edge must already be defined in GAD_BUTTON1 (i.e. a tag such as GL_Left MUST appear before the GL_TopRel tag) if GAD_BUTTON2 is to get the left edge desired.

BUGS

Doesn't do any checking to make sure gadgets don't overlap. Essentially assumes you know what you're doing with the layout.

Bad things will happen if you provide an IMAGEBUTTON_KIND gadget with an image too big to fit within the dimensions you've provided for the gadget.

SEE ALSO

FreeLayoutGadgets(), gadlayout/gadlayout.h, libraries/gadtools.h, GadTools documentation.