Features of V39 GadTools

by Mark Ricci

The V39 GadTools library has been upgraded with new features and capabilities that support the enhanced graphic capabilities of the AA chipset. Other improvements include support for `IDCMP_GADGETHELP` events, more support for `IDCMP_GADGETUP` events, scalable imagery and a new function, GT_GetGadgetAttrs(), to obtain the values of certain gadget attributes.

This article assumes you are familiar with the previous version of the GadTools library. If not, you should refer to the ‘‘GadTools Library’’ chapter of the Amiga ROM Kernel Reference Manual: Libraries for an in-depth explanation of GadTools. You should also refer to the GadTools Autodoc for complete explanations of the GadTools functions. All tags and constants referenced in this article are defined in <*libraries/gadtools.h*> and <*intuition/intuition.h*>.

The NewLook Menus

For V37, Intuition received a facelift. V37 introduced the ‘‘NewLook’’ rendering scheme, which improved the appearance of most of the elements of the Amiga user interface (windows, system gadgets, standard Boopsi classes, etc.). One element of the Amiga’s GUI that escaped the NewLook was the menu system.

For V39, Intuition added the NewLook to its menus. When Intuition renders NewLook menus, it gets the colors for the menu from the Palette Preferences set by the user. Before V39, Intuition used the window’s DetailPen and BlockPen to draw the menu and the colors of the MenuItems depended on the MenuItem’s IntuiText or Image structure.

GadTools fully supports Intuition’s NewLook menus. To make GadTools use the NewLook color scheme for a window’s menus, an application needs to do two things. First, it needs to tell Intuition that it wants NewLook menu imagery for the window. It does this when opening the window by passing the `{WA_NewLookMenus, TRUE}` tag/value pair to OpenWindowTags(). Second, the application needs to remind GadTools that the window uses NewLook imagery for its menus. The application does this by passing the `{GTMN_NewLookMenus, TRUE}` tag/value pair to LayoutMenus():

```
struct VisualInfo *vi;
LayoutMenus(menusready, vi, GTMN_NewLookMenus, TRUE, TAG_END);
```

Instead of using the GTMN_NewLookMenus tag, it's possible to set the menu colors yourself by using the GTMN_FrontPen tag to set the pen number for rendering menu text. In addition to this tag, you must set the screen's BARDETAILPEN and BARTRIMPEN to match the menu text pen and set the BARBLOCKPEN to the color you want for the title bar. However, you cannot control how Intuition complements menu items and you do not automatically get the user's menu color preferences, so you will not get as perfect NewLook menu as you would by leaving it to the system.

Gadgets

Most GadTools gadgets received new attributes in V39. These include new IDCMP events, the ability to specify rendering pens, and scalable imagery.

IDCMP Events

Button, integer and string gadgets can send IDCMP_GADGETDOWN events by setting the GA_Immediate tag to TRUE.

Disabling

MX and listview gadgets join the other GadTools gadgets in having a disable attribute. Set the GA_Disabled tag to TRUE to disable, and FALSE to re-enable.

Scalable Imagery

Checkbox and MX gadgets are no longer fixed in size. You can scale them to any width and height. If you do not specify scaling, GadTools uses its built-in default dimensions. These dimensions, which are defined in *<libraries/gadtools.h>* , are CHECKBOXWIDTH and CHECKBOXHEIGHT for checkbox gadgets, and MXWIDTH by MXHEIGHT for MX gadgets.

To scale a checkbox gadget, set the GTCB_Scaled tag to TRUE, and set the desired width and height in the ng_Width and ng_Height fields of the NewGadget structure. The default value of GTCB_Scaled is FALSE, meaning use the default size of 26 by 11, the values of CHECKBOXWIDTH and CHECKBOXHEIGHT, respectively.

To scale an MX gadget, set the GTMX_Scaled tag to TRUE, and set the desired width and height in the ng_Width and ng_Height fields of the NewGadget structure. The default value of GTMX_Scaled is FALSE, meaning use the default size of 17 by 9, the values of MXWIDTH by MXHEIGHT, respectively. If you scale MX gadgets, you must also set the GTMX_Spacing tag to NewGadget.ng_TextAttr->ta_YSize + 1 to properly space the buttons with respect to the font.

Rendering Pens, Display Format, Justification, Clipping and Display Size

Application writers use number, text, and to some degree, slider gadgets, to present information to the user. The new attributes for these gadgets improve the presentation of the information.

You can specify any pen for rendering the foreground and the background of number and text gadgets. This allows you to use color for emphasis or aesthetics. For example, you might use red as the foreground color in a text gadget for emphasis. You can also use C-language-like formatting strings for the displays, e.g., %lx would give you hexadecimal output in a number gadget. The Exec RawDoFmt() function processes this string, so refer to that function for details.

Text clipping is another new attribute. It clips the display to fit within the borders of the gadget. This prevents the text or number from running over the border of a gadget if it's too large for the gadget. Along with text clipping, the number, text and slider gadgets can left, center and right justify their displays.

Finally, you can specify the size of the string buffer that number and slider gadgets use when formatting their text. This buffer must be large enough to hold the entire string generated by the gadget, as the gadget doesn't do any bounds checking on the buffer. Underestimating the size of the buffer will cause the gadget to overwrite the buffer, overwriting whatever happens to follow the buffer in memory.

For number gadgets, the `GTNM_FrontPen` tag specifies the pen to use for rendering the number. The default is DrawInfo->dri_Pens[TEXTPEN]). `GTNM_BackPen` specifies the pen to use for rendering the background of the number. The default is to use the background color of the gadget's window.

`GTNM_Justification` specifies the type of justification within the gadget box. `GTJ_LEFT` renders the number flush with the left side of the gadget, `GTJ_RIGHT` renders the number flush with the right side, and `GTJ_CENTER` centers the number. The default is `GTJ_LEFT`.

`GTNM_Format` specifies the C-language-like formatting string used to display the number. You must use the ‘‘l’’ (long) modifier. The default is ‘‘%ld’’. The gadget passes the formatting string to the exec.library function RawDoFmt(), which generates the string that appears in the number gadget. See the Autodoc for RawDoFmt() for more information on the formatting string.

If an application changes the default formatting string with the `GTNM_Format` tag, the string that the number gadget displays may become significantly longer. For example, if the application uses the format string ‘‘my number is %ld’’, when the gadget processes the formatting string with RawDoFmt(), RawDoFmt() generates a string at least 14 characters long. By default, the gadget sets aside a 10 byte buffer for RawDoFmt() to use, so the formatting string above is too long for the default buffer. The tag `GTNM_MaxNumberLen` sets the size of this buffer.

GTNM_Clipped controls text clipping. The default behavior differs between number gadgets with borders and number gadgets without borders. For bordered number gadgets, the default is TRUE--clip the display to fit within the gadget borders. For gadgets without borders, the default is FALSE--no clipping.

Text gadgets also have a new set of attributes that correspond to the new number gadget attributes. These corresponding tags are: GTTX_FrontPen, GTTX_BackPen, GTTX_Justification, and GTTX_Clipped. There is no corresponding text gadget tag for GTMN_MaxNumberLen or GTMN_Format.

Slider gadgets use the GTSL_Justification tag with the constants listed above (GTJ_LEFT, GTJ_RIGHT, and GTJ_CENTER) to control justification of the gadgets numeric display. The GTSL_MaxPixelLen tag specifies the maximum pixel size of the level display for any value of the slider. This is primarily useful when dealing with proportional fonts. The default is the font's character width (from the font's TextFont->tf_XSize field) multiplied by the value in the GTSL_MaxLevelLen tag.

New Gadget-Specific Attributes

MX gadgets can now have titles. The gadget obtains its title from the ng_GadgetText field of the NewGadget structure. The GTMX_TitlePlace tag specifies where to place the title. The possible values are:

PLACETEXT_LEFT	Right-align text on left side
PLACETEXT_RIGHT	Left-align text on right side
PLACETEXT_ABOVE	Center text above
PLACETEXT_BELOW	Center text below

For compatibility reasons, GadTools will not put a title on the MX gadget if the GTMX_TitlePlace tag is not present.

Listviews

Listview gadget improvements include forced display of an item, highlighting of the selected item in the listview and custom callback hooks.

The GTLV_MakeVisible tag specifies the number of the item that should be forced into the viewing area. It overrides the GTLV_Top tag.

In V37, a listview identified its currently selected item by displaying it in a text gadget beneath the listview.   The tag GTLV_ShowSelected specifies how the listview indicates which item is the currently selected item. If set to NULL, the listview places a highlight bar over the currently selected item. If GTLV_ShowSelected points to a string gadget, the listview displays the selected item in the string gadget in addition to a highlight bar.

You can use callback hooks to render a listview's items. The listview in the V39 Palette Preferences editor uses a callback hook to display the standard user interface pens. The callback hook makes it possible for each item to display the pen color in addition to the pen name. The pen color appears in a little box at the left edge of each item in the listview.

Palette

`GTPA_NumColors` sets the number of colors to display in the palette gadget. This overrides the `GTPA_Depth` tag and allows numbers that are not powers of 2 (which was a limitation of the V37 palette gadget). The default is 2.

Palette gadgets can now have their own array of pen numbers. `GTPA_ColorTable` is a pointer to a array of pen numbers. The array must contain at least as many entries as there are colors displayed in the palette gadget. The default is `NULL` which causes a 1-to-1 mapping of pen numbers. That means if you don't specify this tag, your palette will use pen number 0 through pen n-1, where n is the number of colors you set in `GTPA_NumColors`. For example, if you set `GTPA_NumColors` to 6 and don't specify `GTPA_ColorTable`, you will get a pen array of pens 0, 1, 2, 3, 4, and 5.

A pen array you specify looks like this:

```
UBYTE colors[8]= {1,2,15,6,0,3,8};
```

Setting `GTPA_ColorTable` to this array indicates that the palette pens are 1, 2, 15, 6, 0, 3 and 8, respectively. Note that you have to allocate pens to be certain you can use them and control them. See the *Amiga OS V39 Developer Release Notes* for information on allocating and sharing pens.

You cannot use the `GTPA_ColorTable` tag with the `GTPA_ColorOffset` tag. The `GTPA_ColorOffset` tag specifies which pen to use as the first pen of a palette. For example, if you set it to 4, the first pen used in the palette will be pen 4. If you use the `GTPA_ColorTable` tag, the `GTPA_ColorOffset` tag is ignored.

However, you can do color offsets with a color table by setting the value of the `GTPA_ColorTable` tag to point to the color table plus the offset you want. In the table above, you could specify pen 6 (the fourth member of the table) as the first pen to use by setting `GPTA_ColorTable` in this manner:

```
CreateGadget(PALETTE_KIND, gad, &ng,
             GTPA_ColorTable, &colors[3],
             TAG_END);
```

Note that the `GTPA_NumColors` tag is not present in the call above. This will default the number of colors to two, so the palette gadget created above will be a two color gadget that uses pens 6 and 0, the fourth and fifth pens in the array.

Obtaining Gadget Attributes

Prior to V39, the only way to obtain the attributes of a GadTools gadget was through the Code field of an IDCMP_GADGETUP IntuiMessage. In addition to having to wait until the user clicked on a gadget, this method limited you to obtaining one attribute at a time.

A new GadTools function, GT_GetGadgetAttrs() obtains a gadget's attributes. You may call the function at any time after creating a gadget and for as many attributes as the function supports for that gadget.

GT_GetGadgetAttrs() accepts a taglist of the attributes to obtain. An application passes it the address of the gadget, the address of the window containing the gadget, a NULL field (this field is reserved for later use and must be NULL for now), and a taglist. Each attribute value pair in the taglist supplies the ID of the tag to get (in ti_Tag) and an address where GT_GetGadgetAttrs() can store the tag's value. Since all tag values are longwords, expect GT_GetGadgetAttrs() to copy a longword for each attribute you request. The function returns the number of attributes it obtained.

```
LONG how_many, top_item = 0, selected_item = 0;

how_many = GT_GetGadgetAttrs( listview_gad, win, NULL,
                              GTLV_Top, &top_item,
                              GTLV_Selected, &selected_item,
                              TAG_DONE );
```

How_many contains the number of gadget attributes GT_GetGadgetAttrs() obtained. GT_GetGadgetAttrs() copies the value of the listview top and the selected item into top_item and selected_item, respectively. *Use longword targets to hold the returned values.*

GT_GetGadgetAttrs() supports GA_Disabled (TRUE if the gadget is disabled; FALSE if the gadget is active) for all GadTools gadgets except text and number gadgets. Other attributes are specific to each gadget as follows:

Checkbox - GTCB_Checked returns TRUE if the checkbox gadget is checked, or FALSE if the gadget is not checked.

Cycle - GTCY_Active returns the ordinal number, counting up from zero, of the active choice. GTCY_Labels returns a pointer to the NULL-terminated array of choices.

Integer - GTIN_Number returns the contents of the gadget.

Listview - GTLV_Top returns the ordinal number, counting up from zero, of the top item. GTLV_Selected returns the ordinal number of the selected item. GTLV_Labels returns a pointer to the List structure associated with the gadget.

MX - GTMX_Active returns the ordinal number, counting up from zero, of the active selection.

Palette - `GTPA_Color` returns the ordinal number of the selected color. This number corresponds to the selected color's position in the palette's `GTPA_ColorTable` array. `GTPA_ColorOffset` returns the ordinal number of the first color used in the palette. `GTPA_ColorTable` returns a pointer to the table of pen numbers used for the gadget, if there is one.

Scroller - `GTSC_Top` returns the number of the top line visible in the scroller. `GTSC_Total` returns the total number of lines in the scroller. `GTSC_Visible` returns the number of visible lines in the scroller.

Slider - `GTSL_Min` returns the minimum value of the slider. `GTSL_Max` returns the maximum value of the slider. `GTSL_Level` returns the current level of the slider.

String - `GTST_String` returns a pointer to the gadget's buffer.

Text - `GTTX_Text` returns a pointer to the gadget's buffer.

Message Handling - ExtIntuiMessage Structure

For V39, the IntuiMessage structure has been extended to allow for tablet support. The GadTools message handling functions--GT_FilterIMsg(), GT_GetIMsg(), GT_PostFilterIMsg() and GT_ReplyIMsg()--work with the extended IntuiMessage structure, ExtIntuiMessage, but are backwards compatible with IntuiMessage.

Gadget Help

All GadTools gadgets support Intuition's gadget help feature. You enable this by calling the *intuition.library* function HelpControl(). Gadget help is either on or off for all GadTools gadgets in a window; you cannot selectively enable it for certain gadgets as you can with Intuition gadgets. For more information, refer to page IV-114 of the ‘‘Boopsi in Release 3’’ article in the January/February 1993 issue of Amiga Mail.

Bevel Box

Bevel boxes now come in three types that you specify with `GTBB_FrameType`. `BBFT_BUTTON` generates the type of box used around button gadgets. `BBFT_RIDGE` generates the type of box used for string gadgets. `BBFT_ICONDROPBOX` generates a box suitable for a standard icon drop box imagery. The default is `BBFT_BUTTON`. These frame types are similar to the Boopsi imageclass frame types `FRAME_BUTTON`, `FRAME_RIDGE`, and `FRAME_ICONDROPBOX`, that are pictured on page IV-120 of the ‘‘Boopsi in Release 3’’ article in the January/February 1993 issue of Amiga Mail.

Example Code - *NewGadgets.c*

The example code below is a demonstration of some of the V39 GadTools changes and how to specify them in your code. You'll notice that it lacks many of the features required to actually use the gadgets. This is by design. The intent is for you to modify an existing application using the example as a guide.

NewGadgets.c produced the screen shot below.

Example - *NewGadgets.c*