Smart Layout Gadgets

Prior to Release 3, Intuition took care of laying out GREL gadgets whenever the dimensions of the window change. This was a limitation for a GREL Boopsi gadget that wants to perform its own layout. Because Intuition didn't tell the Boopsi gadget that the window's dimensions changed, the gadget cannot react to the change.

As of Release 3, Intuition uses a new gadget method called GM_LAYOUT to tell a GREL Boopsi gadget that its window's dimensions have changed. The GM_LAYOUT method uses the gpLayout structure (defined in <*intuition/gadgetclass.h*>) as its message:

```
struct gpLayout
{
    ULONG               MethodID;
    struct GadgetInfo  *gpl_GInfo;
    ULONG               gpl_Initial; /* This field is non-zero if this method was invoked
                                      * during AddGList() or OpenWindow().  zero if this
                                      * method was invoked during window resizing.
                                      */
};
```

For GREL gadgets, Intuition sends a GM_LAYOUT message just after erasing the gadget's bounding box. Intuition does not touch the values of the gadget's bounding box or hit box, it lets the gadget handle recalculating these values. Thee gadget can lay itself out based on the new window (or requester) dimensions found in the GadgetInfo structure passed in the GM_LAYOUT message (gpl_GInfo from the gpLayout structure). Intuition also adds the old and new bounding box of the gadget to the window's damaged regions so that the gadget will appear is its new position. The gadget must not perform any rendering inside the GM_LAYOUT method. Intuition will send a GM_RENDER message to the gadget when its time to redraw the gadget in its new position.

There are two cases where Intuition sends a GM_LAYOUT message:

When the gadget's window is resized
When Intuition adds the gadget to a window

There are two ways for a window to gain gadgets. An application can explicitly add gadgets using the AddGList() function. The window can also gain gadgets during it initialization in the OpenWindow() call.

For most GREL Boopsi gadgets, Intuition expects the values in the LeftEdge, TopEdge, Width, and Height fields to follow the existing convention for regular GREL gadgets. Each of these fields has a flag in the Gadget.Flags field (GFLG_RELRIGHT, GFLG_RELBOTTOM, GFLG_RELWIDTH, and GFLG_RELHEIGHT, respectively). If that field's flag is set, Intuition expects the value in that field to be relative to either the window border or the window dimensions. For example, if GFLG_RELRIGHT is set, Intuition expects the value in the gadget's LeftEdge field to be relative to the right window border.

There is a special kind of GREL Boopsi gadget called a *custom relativity* gadget. For this type of gadget, Intuition expects the values in the LeftEdge, TopEdge, Width, and Height fields to be absolute measurements, just like the values Intuition expects in these fields for non-GREL gadgets.

Setting the `GFLG_RELSPECIAL` bit in the Gadget.Flags field marks the gadget as a *custom relativity* gadget. The best way to set this bit is by setting the `GA_RelSpecial` attribute to `TRUE` when creating the gadget.

Disabled Imagery

Certain gadget types support using a Boopsi Image as its imagery. These gadget types include the non-Boopsi boolean gadget, the frbuttonclass gadget, and the buttongclass gadget (This can also include private gadget classes that support the `GA_Image` attribute). The Gadget structure contains a field called GadgetRender which can point to a Boopsi Image object (the `GFLG_GADGIMAGE` bit in the Gadget.Flags field should also be set, see the‘‘Intuition Gadgets’’ chapter of the *RKRM:Libraries* for more information). Intuition uses this image as the gadget’s imagery.

When a gadget uses this scheme for its imagery, Intuition sends `IM_DRAW` (and `IM_DRAWFRAME`) messages to the image object. These imageclass methods not only tell a Boopsi image to render itself, they also tell the image which state to draw itself. Some examples of image states include normal, selected, and disabled.

Prior to Release 3, when Intuition sent a draw message to one of these images, it only used the normal and selected states. If the image was disabled, Intuition told the image to draw itself as normal or selected and would render the standard ‘‘ghosted’’ imagery over the gadget.

Under Release 3, Intuition will use the disabled state as well, but only if the gadget’s image object supports it. When Intuition adds the gadget to a window, Intuition tests the `IA_SupportsDisabled` attribute of the gadget’s image object. This imageclass attribute was introduced in Release 3. If this attribute is `TRUE`, the image supports disabled imagery (both the `IDS_DISABLED` and `IDS_SELECTEDDISABLED` states), so Intuition sets the `GFLG_IMAGEDISABLE` flag in the Gadget structure’s Flags field. Intuition uses this flag to tell if it should use the image’s disabled state in the `IM_DRAW`/`IM_DRAWFRAME` message.

As the `IA_SupportsDisabled` attribute is a fixed property of an image object, it is only accessible using the `OM_GET` method. An application cannot set it or clear it.

Frameiclass Frame Types

For Release 3, the Boopsi image class frameiclass acquired a new attribute, IA_FrameType. This attribute allows applications to choose a frame style for a frame image. Currently there are four styles available:

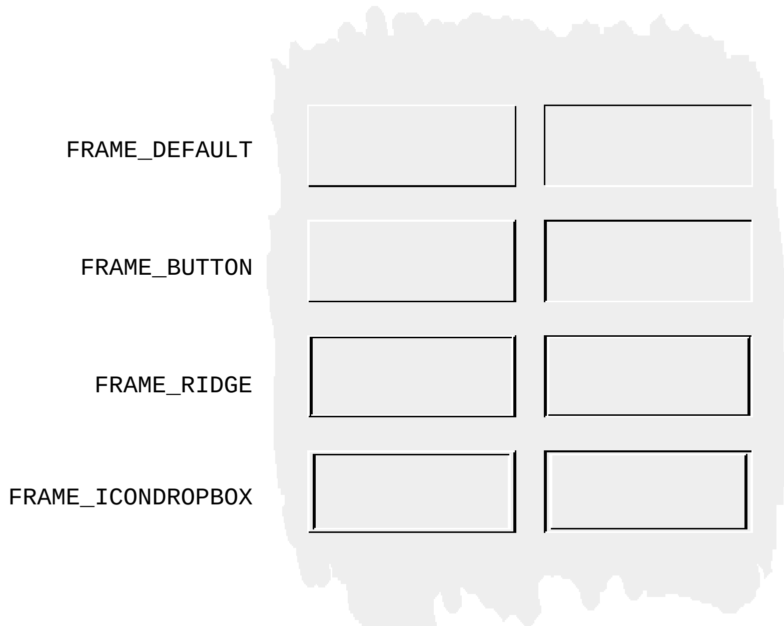

Figure 3 - frameiclass frame styles

FRAME_DEFAULT - This is the default frame which was the only available frame in V37. It has thin edges.

FRAME_BUTTON - This is the imagery for the standard button gadget. It has thicker sides and nicely edged corners.

FRAME_RIDGE - This is a ridge commonly used by standard string gadgets. When recessed (using the frameiclass attribute IA_Recessed), this image appears as a groove. Applications and utilities commonly use the recessed image to visually group objects on the display.

FRAME_ICONDROPBOX - This broad ridge is the system standard imagery for icon ‘‘drop boxes’’ inside of an AppWindow (see the ‘‘Workbench and Icon Library’’ chapter of the *RKRM:Libraries* for more information on AppWindows). The recessed version has no standard purpose.

Additions to Sysiclass

The class of system images, sysiclass, gained two new images for Release 3, a menu check mark image and an Amiga key image (respectively, MENUCHECK and AMIGAKEY from *<intuition/imageclass.h>*):

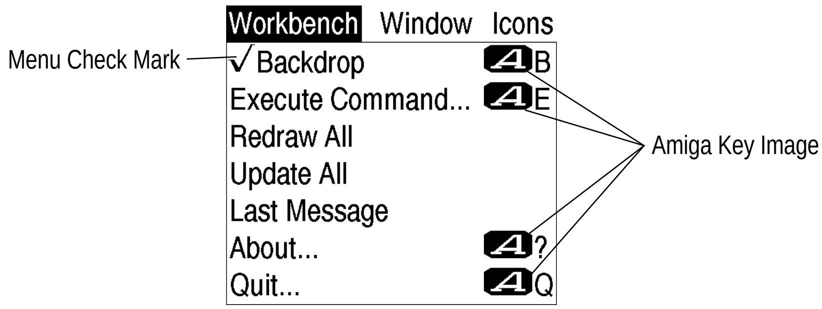

Figure 4 - Menu Check Mark and Amiga Key Images

This class also gained a new attribute called `SYSIA_ReferenceFont`. This attribute, which is only accessible with the `OM_NEW` method, points to a TextFont structure. The `MENUCHECK` and `AMIGAKEY` images will use this font to figure out how large to make the image. This attribute overrides the `SYSIA_DrawInfo` font and `SYSIA_Size` attributes when calculating the dimensions of the image.

About the Example

The example at the end of this article, *relative.c*, uses three features mentioned in this article. The example implements a private subclass of gadgetclass. The new class utilizes the frameiclass `IA_FrameType` attribute to create a button gadget. The class also takes advantage of the custom relativity feature. The example opens a window placing one of these gadgets in the middle of the window, sizing the gadget relative to the window's dimensions.

The example also illustrates the gadget help feature of Intuition. When the private class dispatcher receives a `GM_HELPTEST` message, it returns the bitwise AND of `GMR_HELPCODE` and the value 0xC0DE. When the example receives an `IDCMP_GADGETHELP` message at the window's message port, the example examines the message to find out what object triggered the help message and printf()s the results.