AmigaMail

COLLABORATORS

	TITLE : AmigaMail		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 14, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

AmigaMail

1.1 IV-57: Boopsi's Transparent Base Classes

by Peter Cherna

[Editor's note: This article assumes the reader is already familar with several boopsi concepts. For more information on boopsi, see the article "Introduction to Boopsi" from the March/April 1991 issue of Amiga Mail.]

Although it is normally considered a "programming sin", in some cases it is legal to directly manipulate some internal fields of certain boopsi objects. For compatibility reasons, a boopsi Image or Gadget object contains an actual Image or Gadget structure. To a large degree, a boopsi Image or Gadget looks like a regular Image or Gadget. Most fields in these embedded Image and Gadget structures generally contain information equivalent to the information in a conventional Image or Gadget (although not always). This means that many programs and routines that don't know about boopsi can still make some use of boopsi Images and Gadgets. These objects are instances of the Transparent Base Classes, imageclass and gadgetclass.

To change an attribute of a boopsi object, you normally invoke a set method (OM_SET). The Intuition functions SetAttrs() and SetGadgetAttrs() invoke this method. A boopsi class is informed of any attribute change at that time, allowing it to react to this change. The reaction can include validation of the changed attribute, changing other attributes to match, or informing other objects of the change. That is the inherent advantage of a function-access approach to changing attributes.

When using conventional Images and Gadgets, you generally modify the structure's fields directly. This operation is very fast. For such Images and Gadgets, there is no class that needs to know about the changes, so there is no problem. However, this is untrue of boopsi Images and Gadgets. Although directly modifying the boopsi object's internal structure would provide a performance increase over using the boopsi OM_SET mechanism, altering a boopsi object's internal structure directly will not give the class the opportunity to react to any structure changes. This violates the boopsi concept, and therefore cannot be done in general. In order to provide a balance between the flexibility of function access and the performance of direct access, the transparent base classes imageclass and gadgetclass will not depend on being informed of changes to certain fields in the internal Image and Gadget structures. This means that direct subclasses of imageclass or gadgetclass (for example, frameiclass, sysiclass, propgclass, buttongclass, or any direct subclass you create) are allowed to poke specific fields. Indirect subclasses of imageclass or gadgetclass (for example, subclasses of propgclass, frameiclass, or other indirect subclasses you create) may not poke those fields, since their parent classes may depend on hearing about changes to these fields, which the SetAttrs() call (or similar function) provides.

For images, the following Image structure fields are the only fields that may be poked by image classes whose immediate superclass is imageclass:

```
        LeftEdge
        TopEdge
        Width
        Height
        ImageData
        PlanePick
        PlaneOnOff
```

For gadgets, the following Gadget structure fields are the only fields that may be poked by gadget classes whose immediate superclass is gadgetclass:

```
        LeftEdge
        TopEdge
        Width
        Height
        Flags
        Activation
        GadgetType
        GadgetRender
        SelectRender
        GadgetText
        SpecialInfo
```

In all cases, the direct subclass must take care to maintain sensible values for these fields.

Under no circumstances may an indirect subclass poke one of these fields, even if the subclass knows the superclasses do not depend on notification for this field. This is the only way to preserve the possibility for future enhancements to that superclass.