drag_gc

COLLABORATORS

	TITLE : drag_gc		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 19, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

drag_gc

1.1 drag_gc.doc

1.2 drag.gadget/--drag.gadget--

```
NAME
    drag.gadget -- gadget class for drag & drop actions.

FUNCTION
    The drag gadget class provides simple access to drag and drop
    functionality. It supports notifying of target objects, dragging of
    gadget groups, animated images and relativity flags (GA_RelBottom and
    GA_RelRight).
    The IClass* can be obtained from the cl_Class field of the
    ClassLibrary structure.
    AmigaOS3.0 (V39) is required.

METHODS
    GM_RENDER - the gadget image is drawn according to the state of
         GFLG_SELECTED and GFLG_GADGHIGHBITS. The images may be provided
         with the GA_Image and GA_SelectRender tags, at least GA_Image
         must be provided. If the DGA_DragImage or DGA_DragAnim tags are
         not used, the SelectRender or normal image are also used for the
         drag gadget's bob. In this case the supplied image may not be a
         BOOPSI object.

    OM_DROPACTION - this method is called if another drag gadget is
         dropped on this gadget. For the drag gadget class, it invokes a
         hook function if one is supplied via DGA_DropActHook. It may be
```

```
        implemented for other gadget classes if they are used as drop
        targets.

   GM_GOACTIVE - the gadget changes its state to selected and activates
        its bob. If the gadget belongs to a drag group, all other members
        are deselected. If SHIFT was pressed, instead the bobs of all
        selected members are activated, too.

   GM_HANDLEINPUT - If the mouse is moved, all activated bobs are moved
        accordingly. If the right mouse button is pressed (cancel), all
        bobs are deactivated and the related gadgets deselected.

   GM_GOINACTIVE - If the action was not cancelled, its target is
        identified: if the mouse pointer is in a "known" window (see
        AddDropWindow()), that window is the target. If the pointer is
        placed over a BOOPSI gadget within such a window, the gadget is
        the target. Otherwise no target is found, only the pointer
        position is noted. The target position, type and object can be
        found in the DragInfo structure (see DGA_DragInfo). If the target
        is a DropWindow, a message is send to the related port. If the
        target is a BOOPSI gadget, the BOOPSI message OM_DROPACTION is
        send to that gadget.

ATTRIBUTES
   DGA_ExtSelect (I) - default value is FALSE. If this attribute is set,
        the gadget stays selected after dropping. If it belongs to a
        drag group and another gadget of that group is selected with
        SHIFT pressed, this gadget's bob is also activated.

   DGA_Context (I) - context for the gadget's bob. See CreateDContext().

   DGA_Screen/DGA_Window (I) - screen/window of the gadget's bob. These
        attributes and DGA_Context are mutually exclusive. With these
        attributes, no locking is provided, so it may only be used, if
        the application has exclusive access to the supplied
        window/screen. Additionally, the first gadget created with one
        of these tags must be disposed as the last one. In general, it is
        better to use DGA_Context.

   DGA_DragInfo (G) - after a drop action, the application finds details
        in the DragInfo structure, which is found using this tag.

   DGA_DragImage (IS) - if this attribute is set, the bob's imagery is
        not taken from GA_Image or GA_SelectRender, but from the supplied
        image. It may not be a BOOPSI object, i.e. its ImageData pointer
        must be initialised.

   DGA_DragAnim (IS) - with this tag a NULL-terminated array of image
        pointers can be supplied. They will be used as animated bob
        imagery. None of them may be a BOOPSI object.

   DGA_AnimSpeed (IS) - this tag controls the speed of the bob animation
        in ticks/second. There are approximately ten ticks per second, so
        a value of 1 (default) will result in ten pictures per second.
        A value of 0 causes the animation to stop, i.e. the current image
        will never be changed.
```

```
    DGA_DragGroup (IS) - Setting this attribute adds the gadget to a drag
         group (see NewDragGroup()). A value of NULL removes the gadget
         from its group. All member gadgets of a group should be within
         the same window/requester/domain.

    DGA_DropActHook (I) - this hook is called from the OM_DROPACTION
         method, if another drag gadget is dropped on this gadget.
         The hook function gets a pointer to this gadget in register A2
         (Object) and a pointer to the DragInfo structure of the dropped
         gadget in A1 (Message). Note that this hook function is executed
         in context of input.device, which is a task, not a process. That
         means no DOS functions may be called!

NOTES
    The driver software of some graphics cards seems to introduce a bug
    with the GELS animation system, causing the system to crash if
    animated bobs are used. If this is the case for your system, it can be
    resolved with the patchgels program (see patchgels drawer).
```

1.3 drag.gadget/AddDropWindow

```
NAME
    AddDropWindow -- add a window to the dropwindow list of a drag
                     context.

SYNOPSIS
    dropwindow = AddDropWindow(context, id, userdata, window, msgport);
    D0                          A2       D0  D1        A0      A1

    APTR AddDropWindow(APTR, ULONG, ULONG, struct Window *,
                       struct MsgPort *);

FUNCTION
    Add the window to the context's list of dropwindows. All drag gadgets
    using the context now "know" this window; it may be target of drop
    actions. If the drag gadgets are using no context but the DGA_Window
    tag, that single window is automatically the only known window.
    If a message port is supplied, the application is notified if drag
    gadgets are dropped inside that window (not hitting any gadgets).
    The notification consists of a DropMessage (see drag.h) of type
    DMTYPE_DROPWINDOW.

INPUTS
    context - a context pointer (see CreateDContext()). The window is to
              be included into it's dropwindow list.
    id - with this variable you you can give the dropwindow an id to
         identify it in the dropmessage.
    userdata - this variable is for your own use.
    window - pointer to window to add.
    msgport - pointer to a message port or NULL.

RESULT
    dropwindow - pointer to a dropwindow structure. This is passed to
                 RemoveDropWindow to remove the window from the list of
                 dropwindows. NULL for failure.
```

```
NOTES

SEE ALSO
    RemoveDropWindow() ,CreateDContext(), DeleteDContext(),
    <drag.h>
```

1.4 drag.gadget/CreateDContext

```
NAME
    CreateDContext -- create a context for drag gadgets.

SYNOPSIS
    context = CreateDContext(screen);
     D0                       A0

    APTR CreateDContext(struct Screen *);

FUNCTION
    A context is created for drag gadgets used on a public (shared)
    screen. This includes a copy of the screen's rastport initialised
    with an own GelsInfo structure. The returned value may be used with
    the DGA_Context tag. The resulting bobs can be moved on the specified
    screen. If the bobs should be moveable only within a certain window,
    use the DGA_Window tag instead, which doesn't need a context.

INPUTS
    screen - Screen the bobs should appear on. The window the drag gadgets
             are placed in should be on the same screen.

RESULT
    context - pointer to be passed to drag gadgets via DGA_Context. NULL
              if failure.

NOTES

SEE ALSO
    DeleteDContext(), AddDropWindow(), RemoveDropWindow()
```

1.5 drag.gadget/DeleteDContext

```
NAME
    DeleteDContext -- delete a drag context.

SYNOPSIS
    DeleteDContext(context);
                   A0

    VOID DeleteDContext(APTR);

FUNCTION
    Frees any memory and resources allocated by CreateDContext(). All drag
    gadgets using the context must have been disposed. If any windows are
    still in the dropwindow list, they are removed from the list first.

INPUTS
    context - Pointer obtained by CreateDContext() or NULL.

NOTES

SEE ALSO
    CreateDContext(), AddDropWindow(), RemoveDropWindow()
```

1.6 drag.gadget/FreeDragGroup

```
NAME
    FreeDragGroup -- dispose a draggroup structure.

SYNOPSIS
    FreeDragGroup(draggroup);
                  A0

    VOID FreeDragGroup(APTR);

FUNCTION
    Remove a draggroup obtained by NewDragGroup(). All member gadgets of
    this group should have been removed by setting their DGA_DragGroup
    attribute to NULL.

INPUTS
    draggroup - pointer to a draggroup structure or NULL.

NOTES

SEE ALSO
    NewDragGroup()
```

1.7 drag.gadget/NewDragGroup

```
NAME
    NewDragGroup -- create a context to handle groups of drag gadgets.

SYNOPSIS
    draggroup = NewDragGroup();
    D0

    APTR NewDragGroup(VOID);

FUNCTION
    The draggroup context enables groups of drag gadgets to communicate
    with each other. If a drag gadget is selected, it deselects the other
    members of its group. If it is SHIFT-selected and dropped to a target,
    all selected members are dropped together, with their DropInfo
    structures linked. (See DGA_DragGroup)
```

```
RESULT
    draggroup - pointer to a draggroup structure which may be passed to
                drag gadgets via the DGA_DragGroup tag. NULL for failure.

NOTES

SEE ALSO
    FreeDragGroup()
```

1.8 drag.gadget/OM_DROPACTION

```
NAME
    OM_DROPACTION -- react if a drag gadget is dropped on this gadget.

FUNCTION
    This method is invoked if a drag gadget is dropped on this gadget. For
    the drag gadget class it calls a hook function if one is supplied via
    DGA_DropActHook.
    This method enables any gadget class to act as target for drop
    actions. It "simply" has to be implemented for the desired class.
    A common way of communicating to the target gadget is to supply
    information in the UserData field of the drag gadget which is copied
    into the DragInfo structure of this message.

RESULT
    TRUE if this method is implemented.

NOTES

SEE ALSO
    <drag.h>
```

1.9 drag.gadget/RemoveDropWindow

```
NAME
    RemoveDropWindow -- remove a window from it's list of dropwindows.

SYNOPSIS
    RemoveDropWindow(dropwindow);
                     A0

    VOID RemoveDropWindow(APTR);

FUNCTION
    Remove a dropwindow from the dropwindow list it has been added to.
    If a message port has been attached to the dropwindow, you must make
    sure that all dropmessages are replied after removing the window.

INPUTS
    dropwindow - pointer to a dropwindow structure returned by
                 AddDropWindow() or NULL.
```

```
NOTES

SEE ALSO
    AddDropWindow(), CreateDContext(), DeleteDContext()
```