editor_gc

COLLABORATORS

	TITLE : editor_gc		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		November 29, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

editor_gc

1.1 editor.gadget

```
editor.gadget/editor.gadget()
editor.gadget/EDIT_GetClass()
```

1.2 editor.gadget/editor.gadget()

```
  NAME
editor.gadget -- create multiline text entry BOOPSI objects
     (V1)

  LEGAL
editor.gadget is © 1994 Mark Thomas
All rights reserved.

  FUNCTION
The editor class allows you create an area on your screen for text
entry.  The class supports a number of features including unlimited
or limited text entry, specifying the font to use, specifying the
colors for different parts (text, background, and lines), two types
of borders (with option to invert the borders for a total of 4 types
of borders) or no border, text left/center/right justification,
vertical centering, and other options.

The editor class gadget should be used for relatively small areas
where you want to allow text entry.  Typically a size below 320 x 200
is a fairly reasonable limit, but larger sizes will work.  In other
words it is not intended to be a whole editor like Ed.  Please note
that the vertical size has more affect on speed than the horizontal
size.

To use the class you must open the "editor.gadget" library like so:
OpenLibrary("gadgets/editor.gadget", 0).  If that was successful, then
you need to get the class pointer with the EDIT_GetClass() function.
You do not have to check to make sure that the pointer returned by
EDIT_GetClass() is valid (see EDIT_GetClass description).  You do not
have to return the pointer with any function, just do not use the
```

pointer after you close the "editor.gadget" library.

If you use SAS/C 6.50 or greater, then you can use the EditorAuto.c provided to have the library open automatically at startup, and close automatically at termination so that you will not have to take care of these yourself.

Future versions of the editor.gadget intend to offer a public class name that you can use, but that will have to wait until (or if) Commodore or CATS return from the dead since I have to register the class name. At which time that a public class name is offered, both the old EDIT_GetClass() and the class name can be used with the CreateObject() function so that old and new programs will work.

The class will work on OS 2.04 and up, but for OS 2.04 through 2.1 there is no standard place to put .gadget libraries, so on these systems the gadget should be installed in a drawer named "Gadgets" in the same directory of the program that uses it. See the example program on how to open the class. For OS 3.0 and up, just install the editor.gadget file in "SYS:Classes/Gadgets" like normal.

Unlike the Amiga's string gadget, you get a gadget up message from this gadget when you hit the right mouse button, or either Amiga keys and the right alt keys pressed together.

This gadget does support the GA_Image tag for rendering a linked list of images attached to the gadget. Borders structures are not supported. (Complain if you want it.)

This may or may not help you, but the render order for the elements of this gadget is as follows:

```
    Border (if enabled)
    Lines of text
    Image  (from GA_Image)
```

So, images do overwrite borders and the lines of text. Also note that the gadget will render at least one line of text, and at least one character per line, so restrict the width and height if this is a problem.

IMPORTANT:
(It's anyone's guess as to what will happen if you don't follow the rules below.)

Do NOT do an GetAttr() on EDIT_Text unless the gadget is turned off with OffGadget() or disabled with GA_Disabled, TRUE. Also, do not use the value returned from GetAttr() on EDIT_Text after the gadget is turned back on with OnGadget() or enabled with GA_Disabled, TRUE. You MUST GetAttr() each time you want to get access to the text buffer. And you must NOT alter the data in the buffer. It is READ ONLY. You will get a NULL if you attempt to get the EDIT_Text variable without disabling the gadget. Also, if the EDIT_Size is 0, a NULL is returned for EDIT_Text, so don't try to reference the value you get back.

Also, to find out how much text is in the buffer, use GetAttr() on EDIT_Size after the gadget is turned off with OffGadget() or disabled with GA_Disabled, TRUE. You should NOT read past the end of the buffer (past the size returned by GetAttr() on EDIT_Size). The buffer is not 0 terminated. You are given the specific size.

The text you get from EDIT_Text only contains values from 0x0a, 0x20 - 0x7f, and 0xa0 - 0xff. In other words you get the printable characters, plus '\n'. No tabs are supported.

When setting attributes, the gadget will always render the changes set, except if the gadget is moved, in which case it will return a non-zero value to let you know changes need to be rendered by refreshing the gadget.

DOCS FOR USERS

You can mark text for cutting, copying, and erasing by double-clicking in the gadget. Double-clicking again will turn off marking mode. RIGHT-AMIGA b can also be used to turn marking mode on and off.

For key sequences, the Amiga Style Guide was followed. Anywhere the undelete buffer is mentioned, the statement is only valid if the ClipStream2 is supplied (see tag section below).

Key Sequence	Function
TAB	Activate next gadget (if GA_TabCycle)
SHIFT TAB	Activate previous gadget (if GA_TabCycle)
SHIFT cursor up	Move to the top line in the current page, or scroll up one page if cursor is on top line
SHIFT cursor down	Move to the bottom line in the current page, or scroll down one page if cursor is on top line
CTRL or SHIFT cursor right	Move to the right end of the current line
CTRL or SHIFT cursor left	Move to the left end of the current line
SHIFT backspace	Delete all text to the left of cursor on the current line
SHIFT delete	Delete all text to the right of the cursor on the current line (in block cursor mode this also includes the highlighted character)
CTRL cursor up	Move to the top line of the text
CTRL cursor down	Move to the bottom line of the text
ALT cursor right	Move to the next word (using the delimiter characters provided by the programmer)

```
    ALT cursor left        Move to the previous word (using the
                           delimiter characters provided by the
                           programmer)

    ALT cursor up          Move to first character in gadget

    ALT cursor down        Move to last character in gadget

  RAMIGA [                 Switch to left justification

  RAMIGA \ or
  RAMIGA =                 Switch to center justification

  RAMIGA ]                 Switch to right justification

  RAMIGA e                 Erase all text in gadget (saved in undelete
                           buffer)

  RAMIGA b                 Turn marking mode on and off

  RAMIGA v                 Paste text from clipboard to current cursor
                           position

  RAMIGA a                 Mark all text

  RAMIGA u                 Undelete the last block of text marked,
                           or recover from RAMIGA e

  When marking mode is on the following keys have functions:

         backspace         Erase marked text (save in undelete buffer)

         delete            Erase marked text (save in undelete buffer)

  RAMIGA x                 Cut marked text to clipboard

  RAMIGA c                 Copy marked text to clipboard

  RAMIGA v                 Replace marked text with text from
                           clipboard (save marked text in undelete
                           buffer)

         (any text key)    Replace marked text with that character
  ----------------------------------------------------------------------

    TAGS
  GA_Left (WORD) -- Specifies the left edge of the gadget.

  GA_Top (WORD) -- Specified the top edge of the gadget.

  GA_Width (WORD) -- Specifies the width of the gadget.  If a border is
    chosen, it will be inside this value.

  GA_Height (WORD) -- Specifies the height of the gadget.  If a border
    is chosen, it will be inside this value.

  GA_RelRight (WORD) -- Specifies the gadget as being relative to the
```

```
    right border of whatever the gadget is attached to.  See the
    BOOPSI Class Reference.

  GA_RelBottom (WORD) -- Specifies the gadget as being relative to the
    bottom border of whatever the gadget is attached to.  See the
    BOOPSI Class Reference.

  GA_RelWidth (WORD) -- Specifies the gadget as being relative to the
    width of whatever it is attached to.  See the BOOPSI Class
    Reference.

  GA_RelHeight (WORD) -- Specifies the gadget as being relative to the
    height of whatever it is attached to.  See the BOOPSI Class
    Reference.

  GA_Image (struct Image *) -- Pass a pointer to a linked list of images
    and this class will render them.  See rendering order above.

  GA_Disabled (BOOL) -- TRUE disables the gadget, not allowing input,
    and FALSE enables the gadget for input.  When the gadget is
    disables, it usually is ghosted, see EDIT_NoGhost for other
    conditions.

  GA_TabCycle (BOOL) -- Turns on tab cycling.  See the BOOPSI Class
    Reference.

  EDIT_Text (char *) -- This set/replaces text.  NULL means no change.
    To set the buffer empty pass "" (pointer to empty string).
    When you use it to get text see special conditions under
    IMPORTANT above.

    Default for this tag is NULL.  Applicability is (ISG U).

  EDIT_InsertText (char *) -- This inserts text at current cursor
    position.

    Default for this tag is NULL.  Applicability is ( S  U).

  EDIT_TextFont (struct TextFont *) -- Sets the font for the gadget to
    use.  Pass the object a pointer to a TextFont structure.  This
    supersedes EDIT_TextAttr below.  Please do not close this font
    while the gadget is using it. :)  The default font is your
    screen's current font.

    Default for this tag is NULL. Applicability is (IS  U).

  EDIT_TextAttr (struct TextAttr *) -- Sets the font the gadget is to
    use.  Pass the gadget a pointer to a TextAttr structure.  This
    is superseded by EDIT_TextFont.  The default font is the
    screen's current font.

    Default for this tag is NULL.  Applicability is (IS  U).

  EDIT_FontStyle (ULONG) -- The style will get set to what you pass
    here.  The font style automatically gets reset when
    EDIT_TextFont or EDIT_TextAttr is set.
    Default for this tag is FSF_PLAIN.  Applicability is (IS  U).

  EDIT_Delimiters (char *) -- You get the default if you pass NULL.
    Words break after these and "\n".  You will probably want at
    least " ", the space.

    Default for this tag is ",)!@^&*_=+\|<>?/ ".  Applicability
    is (IS  U).

  EDIT_BlinkRate (ULONG) -- This sets the number of microseconds between
    a cursor on-off, or off-on transition.  A value of 0 means do
    not blink.  Realistically, this should be set to 100000 or
    higher since BOOPSI objects don't get idle messages any faster
    than about once every 10th of a second, but any value between
    0 and 100000 will just make the cursor blink as fast as it
    can.  If you give the user an option of blink speed, suggest
    values: 0 for no blink, 750000 for a slow blink, 500000 for a
    medium blink, and 250000 for a fast blink.

    Default for this tag is 0.  Applicability is (IS  U).

  EDIT_BlockCursor (BOOL) -- Turn on/off block cursor mode.  You should
    not use a block cursor if your font is italic because it looks
    weird.

    Default for this tag is FALSE.  Applicability is (IS  U).

  EDIT_CursorPos (ULONG) -- Get/Set the cursor position.  The cursor
    position returned is always an exact offset into the buffer
    you get to read via EDIT_Text.  0 takes you to the first
    character in the gadget, and 0xFFFFFFFF takes you past the
    last character in the gadget.  In general, any value you pass
    that is larger than what's returned by EDIT_Size will end up
    just past the last character in the gadget.

    Applicability is (ISG U).

  EDIT_Size (ULONG) -- Returns the number of characters in the gadget's
    buffer, including \n characters.  This gives you the size when
    you want to use EDIT_Text to read the text in the gadget.

    Applicability is (  G  ).

  EDIT_MaxSize (ULONG) -- Limit the size of text entered into the
    gadget.  0 means unlimited, otherwise limits the buffer size
    to what you pass.  This includes \n characters.

    Default for this tag is UNLIMITED. Applicability is (I    ).

  EDIT_Visible (ULONG ) -- Get the current number of visible lines.  It
    always returns how many _could_ be displayed if there were
    enough characters to fill the display.  Use for notifying a
    BOOPSI prop gadget.  See example program.

    Applicability is (  GN ).

  EDIT_Lines (ULONG) -- Get the total number of lines in the buffer of
```

```
    the gadget.  Use this to also notify a BOOPSI prop gadget.
    See example program.

    Applicability is (  GN ).

  EDIT_Top (ULONG) -- Get or set ordinal value of top line.  Useful for
    ICA_MAP and ICA_TARGET when using the BOOPSI prop gadget.
    See sample program for example.

    Default for this tag is 0.  Applicability is ( SGNU).

  EDIT_Partial (BOOL) -- When this flag is set to TRUE, partial lines
    will be shown at the bottom of the gadget.  When this flag is
    set to false, then only whole lines will be shown in the
    gadget.  Note that having both EDIT_VCenter, and EDIT_Partial
    on is not allowed and doesn't make sense.  If both
    EDIT_VCenter and EDIT_Partial are turned on at the same time,
    only EDIT_VCenter will get turned on.

    Default for this tag is FALSE.  Applicability is (IS  U).

  EDIT_NoGhost (BOOL) -- If TRUE, never ghost when gadget is disabled.
    If FALSE, then ghost when gadget is disabled.  You can use
    this to make a read-only multiline string gadget.  It has a
    special purpose, though.

    Normally you will want a gadget to be enabled when allowing
    text to be entered.  However, when you need to read the text
    from the gadget, you have to disable it.  But disabling a
    gadget ghosts it.  So, with this option, you can pass
    GA_Disabled, TRUE, EDIT_NoGhost, TRUE at the same time and it
    will disable without ever showing the ghosted pattern.  And
    likewise, passing the attributes GA_Disabled, FALSE,
    EDIT_NoGhost, FALSE will seamlessly reenable the gadget.
    While the gadget is disabled, read the text and then be on
    your way.  Also note that most S and U attributes are settable
    while the gadget is disabled, notably EDIT_Top.  This allows
    you to make a scrollable read-only multiline non-ghosted text,
    image capable, and border capable gadget.  Sounds useful to me!

    Default for this tag is FALSE.  Applicability is (IS  U).

  EDIT_Border (ULONG) -- Sets the border type.  See defines below.  The
    gadget offers a standard bevel, and standard double bevel.  If
    you need another type, your could always create an image, link
    it to the gadget with GA_Image, and set its top and left edges
    above and to the left of this gadget (negative in the image
    structure), and make the width and height larger than this
    gadget.

    Default for this tag is EDIT_BORDER_NONE.  Applicability is
    (IS  U).

  EDIT_Inverted (BOOL) -- If this flags is TRUE, the border is drawn
    inverted, if there is a border.  If FALSE, the border is drawn
    non-inverted.  This option is here in case you want to give
    the editor gadget a read-only look when used in conjunction
```

```
      with EDIT_NoGhost and GA_Disabled.

      Default for this tag is FALSE.  Applicability is (IS  U).

    EDIT_Up (ULONG) -- Moves the text up by one line.  You can pass
      anything, but it will only move the text up by a line, if it's
      not at the top already.  Useful BOOPSI notifications.

      Applicability is ( S  U).

    EDIT_Down (ULONG) -- Moves the text down by one line.  You can pass
      anything, but it will only move the text down by a line, if
      it's not at the bottom already.  Useful for BOOPSI
      notifications.

      Applicability is ( S  U).

    EDIT_Alignment (ULONG) -- Set/Get the line justification.  This
      gadget offers left, center, and right justification.  See defines
      below.

      Default for this tag is EDIT_ALIGN_LEFT.  Applicability is
      (ISG U).

    EDIT_VCenter (BOOL) -- Turn on/off vertical centering.  When on, the
      lines in the display are centered vertically.  If the total
      number of lines is less than the visible number of lines then
      the smaller number of lines are centered.  This allows you to
      center single lines of text within the gadget very easily.
      For normal text entry operation, it is best to leave this off.
      Also, check EDIT_Partial for possible conflicts when used with
      EDIT_VCenter.

      Default for this tag is FALSE.  Applicability is (IS  U).

    EDIT_UserAlign (BOOL) -- If this is set at creation, then the user
      will have control over the left/center/right justification of
      text through RIGHT-AMIGA [, =, ] keyboard shortcuts. If you
      want to save what the user has set the justification to, then
      do a GetAttr() on EDIT_Alignment.

      Default for this tag is FALSE.  Applicability is (I    ).

    EDIT_RuledPaper (BOOL) -- Lets you set whether the paper (background)
      has ruled horizontal lines under each line of text or not.

      Default for this tag is FALSE.  Applicability is (IS  U).

    EDIT_PaperPen (ULONG) -- This lets you specify the pen used for
      drawing the paper (background) of the gadget.  A value of -1
      means use default, which is BACKGROUNDPEN.

      Default for this tag is -1.  Applicability is (IS  U).

    EDIT_InkPen (ULONG) -- This lets you specify the pen used for drawing
      the text.  A value of -1 means use the default, which is
      SHADOWPEN.  If this pen, and the EDIT_LinePen are different,
```

then rendering speed is slowed down a bit. It is recommended that the line pen be left to -1.

Default for this tag is -1. Applicability is (IS U).

EDIT_LinePen (ULONG) -- This lets you specify the pen used for drawing the ruled lines, if EDIT_RuledPaper is TRUE. See EDIT_InkPen for possible speed problems when specifying this pen. A value of -1 means to use the same pen as EDIT_InkPen.

Default for this tag is -1. Applicability is (IS U).

EDIT_Spacing (UBYTE) -- Lets you set an extra amount of spacing between lines of text, for maybe doing 1-1/2 or double spacing. It's a pixel value between 0 and 255. The space is added to the top of each line. In other words, the baseline is moved down by the amount you specify.

Default for this tag is 0. Applicability is (IS U).

EDIT_ClipStream (struct ClipboardHandle *) -- This tag allows clipboard support in the gadget. Pass the pointer returned from the iffparse.library OpenClipboard() function. If a NULL is passed, the clipboard support is not allowed. Please supply this tag value. Don't leave users without clipboard support. It is recommended that the unit opened by OpenClipboard() be 0 or PRIMARY_UNIT, since that is the standard unit, but you can pick whatever unit you or your user wants. This stream can be safely given to multiple objects.

Default for this tag is NULL. Applicability is (I   ).

EDIT_ClipStream2 (struct ClipboardHandle *) -- ClipStream2 is used for the undo features of the editor class. It is obtained from the iffparse.library OpenClipboard() function. You should probably use a clipboard unit other than 0 to avoid conflicts with normal clips. This stream can be safely passed to multiple objects.

Default for this tag is NULL. Applicability (I   ).

BORDER REFERENCE

You can use the width and heights given when calculating window sizes and limits. To make the window's height minimal with respect to your font, use (window border top) + (window border bottom) + (num_lines * (font height)) + (gadget border height). Also, if you use EDIT_Spacing, you'll have to add that in too.

EDIT_BORDER_NONE	Border takes up: 0 width, 0 height
EDIT_BORDER_BEVEL	Border takes up: 8 width, 4 height
EDIT_BORDER_DOUBLEBEVEL	Border takes up: 12 width, 6 height

ALIGMENT REFERENCE

EDIT_ALIGN_LEFT	Cause text to be flush left
EDIT_ALIGN_CENTER	Cause text to be centered
EDIT_ALIGN_RIGHT	Cause text to be flush right

```
  ACKNOWLEDGEMENTS
I would like to thank the following people for volunteering to look-at
and test this class:

Peter Edward Janes
Wayne Robbins
Michael Wiedmer
Phill Coxon
Kenneth Ekman


I would like to thank the following people for suggestions:

Ben Owen
Chris Aldi
Timothy J. Aston
vaald  (I didn't get your name)
Markus Juhani Aalto
Manuel Lemos
David Junod

  BUGS
What bugs?  Please let me know if you find any.

  CONTACT
To contact me for reporting bugs or giving suggestions:

Mark Thomas
1515 Royal Crest Dr. #3259
Austin, TX  78741

or

mthomas@cs.utexas.edu
```

1.3 editor.gadget/EDIT_GetClass()

```
  NAME
EDIT_GetClass -- Gets the pointer to the editor class. (V1)

  SYNOPSIS
editor_class = EDIT_GetClass();
D0

Class *EDIT_GetClass(void);

  FUNCTION
Obtains the pointer to the editor.gadget class for use with
CreateObject().  This function always returns a valid pointer so
you do not beed to check it.  The reason is that if the library
opens fine, then the pointer returned is already setup.

  INPUTS
None.
```

```
  RESULT
editor_class - the pointer to the editor.gadget class.

  BUGS
None.
```