EAGUI

COLLABORATORS

	TITLE : EAGUI	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 22, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

EAGUI

1.1 EAGUI.library

1.2 EAGUI.library/--background--

```
NAME
     EAGUI.library -- Environment Adaptive Graphic User Interface. (V2)

PURPOSE
     This library was designed to provide a flexible layout engine for
     your graphic interfaces. It uses an object oriented approach to
     provide an abstract definition of the contents of a window. The
     library then can convert this definition into an ordinary list of
     gadgets, which you can use freely. An important design goal was to
     make the interface as transparent as possible. This allows you to
     adapt your existing code easily.

COPYRIGHT
     Environment Adaptive Graphic User Interface
```

1.3 EAGUI.library/--classes--

DESCRIPTION

This is a description of the available classes or object types. It particularly describes which tags can be used in which context. Although in most cases other tags are ignored, you shouldn't use them on objects that don't support them, or you'll be in for some nice side effects.

Not all tags can be used under all circumstances. These conditions are described in EAGUI.h. Each tag has three flags: I, S and G. If all of them are set, you can use this tag at Initialization time (when you're using ea_NewObject()), and to Get or Set attributes (when using ea_GetAttr() or ea_GetAttrs() and ea_SetAttr() or ea_SetAttrs()).

In addition to this documentation, you should also read the GadTools and BOOPSI class documentation, to see how these objects work, and what tags they need.

TYPE

global -- Tags that apply to all objects.

TAGS

EA_Parent - (struct ea_Object *) Pointer to the parent object. The current object is linked to this parent.

EA_Type - (ULONG) Type of the object. Normally, you specify this as a normal argument when using ea_NewObject().

EA_Disabled - (BOOL) Flag which indicates if the object is disabled or not. A disabled object is completely ignored and doesn't take up any space. Disabling objects is a simple mechanism for temporarily removing the objects from the tree.

EA_ID - (ULONG) ID of the object, which is also copied to any data the object generates. Gadgets for example will also have this ID stored in their GadgetID field.

EA_MinWidth - (LONG) Minimum width of an object.

EA_MinHeight - (LONG) Minimum height of an object.

EA_BorderLeft - (LONG) Left border thickness of an object.

EA_BorderRight - (LONG) Right border thickness of an object.

EA_BorderTop - (LONG) Top border thickness of an object.

EA_BorderBottom - (LONG) Bottom border thickness of an object.

EA_Left - (LONG) Left offset of an object relative to its parent.

EA_Top - (LONG) Top offset of an object relative to its parent.

EA_Width - (LONG) Actual width of the object.

EA_Height - (LONG) Actual height of the object.

EA_Weight - (ULONG) Weight of the object.

EA_UserData - (VOID *) Pointer to user supplied data, which can be used to add information to certain objects, and is mainly useful in conjunction with custom methods. (V2)

EA_MinSizeMethod - (struct Hook *) Pointer to an initialized Hook for the minimum size method. Hooks are described in the utility.library. This method takes the following arguments:

```
ULONG hookfunction(struct Hook *, struct ea_Object *, APTR);
```

The third argument is always NULL at the moment. The most important argument is the second one, which points at the object whose minimum size EAGUI wants determined. For details on how to implement such a method, take a look at the tutorial.

EA_BorderMethod - (struct Hook *) Pointer to an initialized Hook for the border method. Takes the same hook arguments as the EA_MinSizeMethod hook.

EA_RenderMethod - (struct Hook *) Pointer to an initialized Hook for the render method. This method takes the following arguments:

```
ULONG hookfunction(struct Hook *, struct ea_Object *,
    struct ea_RenderMessage *);
```

The third argument, a pointer to a ea_RenderMessage structure, contains additional information which is needed for rendering the object. The second one, which points at the object that should be rendered. For details on how to implement such a method, take a look at the tutorial. (V2)

EA_GetStateMethod - (struct Hook *) Pointer to an initialized Hook for the state retrieval method. Not implemented.

EA_SetStateMethod - (struct Hook *) Pointer to an initialized Hook for the state restoration method. Not implemented.

EA_FirstChild - (struct ea_Object *) Pointer to the first child of the object. Returns NULL if the object has no children.

EA_NextObject - (struct ea_Object *) Pointer to the next object on the same level. Returns NULL if there is no next object (ie. when the current object is the last object in the list).

EA_StandardMethod - (ULONG) Set of flags, that determine if the object should use any standard built-in methods. Flags can be OR'ed together. Currently, the following flags are supported:

```
            EASM_NONE - Use no standard methods.
            EASM_MINSIZE - Use a standard method to determine the minimum
                size of the object.
            EASM_BORDER - Use a standard method to determine the size of
                the borders around an object.

        Please note that not all object types have standard methods.
        Currently, all GadTools gadgets have EASM_MINSIZE and
        EASM_BORDER methods.


TYPE
     EA_TYPE_VGROUP -- A vertical group of objects.
     EA_TYPE_HGROUP -- A horizontal group of objects.

TAGS
     EA_Child - (struct ea_Object *) Pointer to a child object. If you
         pass a NULL pointer here, the whole object creation process will
         fail. This means that if you're creating a group with children,
         and one of them can't be created, the creation of the group will
         also fail.


TYPE
     EA_TYPE_GTGADGET -- A GadTools gadget.

TAGS
     EA_Instance - (APTR) Pointer to a generated object. This can be a
         gadget or image pointer, depending on the object type.

     EA_InstanceAddress - (APTR *) Address of a pointer where the pointer
         to the generated object is stored.

     EA_GTType - (ULONG) Type of the GadTools gadget (#?_KIND) (see also
         gadtools.library/CreateGadget()).

     EA_GTTagList - (struct TagItem *) Pointer to an array of tags (see
         also gadtools.library/CreateGadget()).

     EA_GTText - (STRPTR) Gadget text (see also
         gadtools.library/CreateGadget()).

     EA_GTTextAttr - (struct TextAttr *) Pointer to a text attribute
         structure (see also gadtools.library/CreateGadget()).

     EA_GTFlags - (ULONG) Special flags (see also
         gadtools.library/CreateGadget()).


TYPE
     EA_TYPE_BOOPSIGADGET -- A BOOPSI gadget.

TAGS
     EA_Instance - (APTR) Pointer to a Gadget structure. In true OO style,
         this should only be used to identify the object. That's why it's
         cast as an APTR.
```

```
     EA_InstanceAddress - (APTR *) Address of a pointer where the pointer
         to the generated gadget is stored.

     EA_BOOPSIPrivClass - (APTR) Pointer to a generated private BOOPSI
         class (see also intuition.library/NewObject()).

     EA_BOOPSIPubClass - (STRPTR) Pointer to the name of a public BOOPSI
         class (see also intuition.library/NewObject()).

     EA_BOOPSITagList - (struct TagItem *) Pointer to an array of tags.


TYPE
     EA_TYPE_CUSTOMIMAGE -- A custom image.

TAGS
     none

NOTES
     This object type was designed for custom rendering. You can use the
     Render method to link your rendering function to the object. Also,
     you can use this object as an empty spaceholder.


TYPE
     EA_TYPE_BOOPSIIMAGE -- (not implemented)

TAGS
     none
```

1.4 EAGUI.library/ea_CreateGadgetList()

```
NAME
     ea_CreateGadgetList -- Create the gadgets as defined in the objects
         (V1)

SYNOPSIS
     result = ea_CreateGadgetList(obj_ptr, glist_ptr_ptr, vi_ptr,
     D0                           A0       A1             A2
         draw_ptr)
         A3

     LONG ea_CreateGadgetList(struct ea_Object *, struct Gadget **, APTR,
         struct DrawInfo *);

FUNCTION
     This function goes through the tree of objects and creates BOOPSI and
     GadTools gadgets. All gadgets are created using the VisualInfo and
     DrawInfo that are passed as arguments to this function.

     The result is a pointer to a gadgetlist that can be added to a
     window. This pointer will be stored in the address that was passed as
     an argument to this function.

INPUTS
```

```
     obj_ptr - Root object of the hierarchical structure to be searched
         for gadgets.
     glist_ptr_ptr - Valid pointer to the place where the pointer to the
         gadgetlist should be stored.
     vi_ptr - Valid pointer to a VisualInfo structure.
     draw_ptr - Valid pointer to a DrawInfo structure. Note that if you
         do not use any BOOPSI gadgets or images, this pointer may be
         NULL. This is discouraged, and may change in the future.

RESULT
     result - returncode, as defined in EAGUI.h.

SEE ALSO
     ea_FreeGadgetList()
```

1.5 EAGUI.library/ea_DisposeObject()

```
NAME
     ea_DisposeObject -- Deletes an object. (V1)

SYNOPSIS
     ea_DisposeObject(obj_ptr)
                      A0

     VOID ea_DisposeObject(struct ea_Object *);

FUNCTION
     Deletes an object and all of its auxiliary data. These objects are
     all created by ea_NewObject(). Objects of certain classes "own" other
     objects, which will also be deleted when the object is passed to
     ea_DisposeObject(). Read the per-class documentation carefully to be
     aware of these instances.

INPUTS
     obj_ptr - Pointer to the object to be deleted.

SEE ALSO
     ea_NewObject()
```

1.6 EAGUI.library/ea_FreeGadgetList()

```
NAME
     ea_FreeGadgetList -- Free all gadgets in a list. (V1)

SYNOPSIS
     ea_FreeGadgetList(obj_ptr, glist_ptr)
                       A0       A1

     VOID ea_FreeGadgetList(struct ea_Object *, struct Gadget *);

FUNCTION
     All BOOPSI and GadTools gadgets that were created by
```

```
     ea_CreateGadgetList() are cleaned up. It is important that the
     gadgetlist is not connected to a window. The objects are not cleaned
     up.

INPUTS
     obj_ptr - Root object of the hierarchical structure to be searched
         for gadgets.
     glist_ptr - Valid pointer to the the gadgetlist.

SEE ALSO
     ea_CreateGadgetList(), intuition.library/RemoveGList()
```

1.7 EAGUI.library/ea_GetAttr()

```
NAME
     ea_GetAttr -- Inquire the value of an attribute of an object. (V1)

SYNOPSIS
     value = ea_GetAttr(obj_ptr, attribute)
     D0                 A0       D0

     ULONG ea_GetAttr(struct ea_Object *, ULONG);

FUNCTION
     Inquires from the specified object the value of an attribute.

INPUTS
     obj_ptr - Pointer to the object.
     attribute - Tag value of the attribute.

RESULT
     value - Value of the attribute.

SEE ALSO
     ea_GetAttrs(), --classes--
```

1.8 EAGUI.library/ea_GetAttrsA()

```
NAME
     ea_GetAttrsA -- Inquire the value of some attributes of an object.
         (V1)
     ea_GetAttrs -- Varargs stub for ea_GetAttrsA(). (V1)

SYNOPSIS
     count = ea_GetAttrsA(obj_ptr, taglist_ptr)
     D0                   A0       A1

     ULONG ea_GetAttrsA(struct ea_Object *, struct TagItem *);

     count = ea_GetAttrs(obj_ptr, firsttag, ...)

     ULONG ea_GetAttrs(struct ea_Object *, ULONG, ...);
```

```
FUNCTION
     Inquires from the specified object the value of the attributes
     specified in the taglist.

     You always pass the addresses of the long variables, which will
     receive the same value that would be passed to ea_SetAttrs() in the
     ti_Data portion of a TagItem element.

     Not all attributes will respond to this function. Those that will are
     documented on a class-by-class basis.

     The function returns the number of attributes it obtained. It tries
     to obtain as many attributes as possible, so don't think that if for
     example 3 out of 4 arguments were obtained, that the last one was
     _not_ obtained. It may just as well be the first (or second, or
     third) one.

INPUTS
     obj_ptr - Pointer to the object.
     taglist_ptr - Pointer to the taglist.

RESULT
     count - Number of attributes obtained.

SEE ALSO
     ea_GetAttr(), ea_SetAttrs(), --classes--
```

1.9 EAGUI.library/ea_GetMinSizes()

```
NAME
     ea_GetMinSizes -- Calculate minimum object sizes. (V1)

SYNOPSIS
     ea_GetMinSizes(obj_ptr)
                    A0

     VOID ea_GetMinSizes(struct ea_Object *);

FUNCTION
     This function calculates the minimum sizes of all of the objects that
     are in the hierarchical tree of which you pass the root object. For
     doing this, ea_GetMinSizes() uses the methods you supplied when
     defining the object tree. A number of methods can be found in the
     EAGUI_METHODS, but it is possible to write your own methods,
     following a few simple rules.

     The minimum size of an object does not include the space that is
     needed for its borders.

     The minimum sizes are directly filled into the objects. They may only
     be read using ea_GetAttrs().

INPUTS
     obj_ptr - Pointer to the root object of the hierarchical tree. If
```

```
          this pointer is NULL, the function returns immediately.

SEE ALSO
     ea_LayoutObjects()
```

1.10 EAGUI.library/ea_GetObjectLeft()

```
NAME
     ea_GetObjectLeft -- Get the left offset to an object (V1)

SYNOPSIS
     left = ea_GetObjectLeft(root_ptr, obj_ptr)
     D0                      A0        A1

     LONG ea_GetObjectLeft(struct ea_Object *, struct ea_Object *);

FUNCTION
     This function returns the left offset from the upperleft corner of
     one object to the upperleft corner of another. The second object
     should branch down from the first object, otherwise the function will
     return garbage. Please make sure that this is the case!

INPUTS
     root_ptr - Pointer to the object the left offset should be calculated
         from. It should be higher in the hierarchical tree. If this
         pointer is NULL, the function returns immediately.
     obj_ptr - Pointer to the object the left offset should be calculated
         to. It should be lower in the hierarchical tree. If this
         pointer is NULL, the function returns immediately.

RESULT
     left - The left offset from a root object to another object.

SEE ALSO
     ea_GetObjectTop()
```

1.11 EAGUI.library/ea_GetObjectTop()

```
NAME
     ea_GetObjectTop -- Get the top offset to an object (V1)

SYNOPSIS
     top = ea_GetObjectTop(root_ptr, obj_ptr)
     D0                    A0        A1

     LONG ea_GetObjectTop(struct ea_Object *, struct ea_Object *);

FUNCTION
     This function returns the top offset from the upperleft corner of one
     object to the upperleft corner of another. The second object should
     branch down from the first object, otherwise the function will return
     garbage. Please make sure that this is the case!
```

```
INPUTS
     root_ptr - Pointer to the object the top offset should be calculated
         from. It should be higher in the hierarchical tree. If this
         pointer is NULL, the function returns immediately.
     obj_ptr - Pointer to the object the top offset should be calculated
         to. It should be lower in the hierarchical tree. If this
         pointer is NULL, the function returns immediately.

RESULT
     top - The top offset from a root object to another object.

SEE ALSO
     ea_GetObjectLeft()
```

1.12 EAGUI.library/ea_LayoutObjects()

```
NAME
     ea_LayoutObjects -- Calculates the actual object dimensions and
         positions. (V1)

SYNOPSIS
     ea_LayoutObjects(obj_ptr)
                      A0

     VOID ea_LayoutObjects(struct ea_Object *);

FUNCTION
     This function calculates the dimensions of all objects in the
     hierarchical tree of which you pass the root object. The dimensions
     of the root object should be filled in before you call this function.
     This function is normally called after you've opened a new window, or
     after someone has resized a window. The minimum sizes of the objects
     should already be calculated, using ea_GetMinSizes().

     For every horizontal or vertical group that is in the tree, this
     function does the following. The available height or width of the
     (vertical or horizontal) group object is decreased by the space
     needed for the borders of the child objects. Also the objects with
     weight 0 get their minimum dimensions, and the available space is
     decreased accordingly.

     The space that is left, is divided amongst the remaining children.
     Each child gets dimensions depending on its weight factor, but if
     this would result in an object which would be too small, the minimum
     size of the object is used instead. The other dimension of the
     children (the height for horizontal groups, the width for vertical
     ones) is always equal to the height or width of the group object
     decreased by the child objects' borders.

     After the space has been divided, the positions of the objects
     relative to their parents are calculated. This position is given as
     an offset from the top-left corner of the parent object to the
     top-left corner of the child object's border. Note that if you want
     to calculate the position of the object itself (again, relative to
```

```
     its parent), you should add the left and top border to the offset
     values.

INPUTS
     obj_ptr - Pointer to the root object of the hierarchical tree. If
         this pointer is NULL, the function returns immediately.

SEE ALSO
     ea_GetMinSizes()
```

1.13 EAGUI.library/ea_NewObjectA()

```
NAME
     ea_NewObjectA -- Create a new object. (V1)
     ea_NewObject -- varargs stub for ea_NewObjectA(). (V1)

SYNOPSIS
     obj_ptr = ea_NewObjectA(type, taglist_ptr)
     D0                      D0    A0

     struct ea_Object *ea_NewObjectA(ULONG, struct TagItem *);

     obj_ptr = ea_NewObject(type, firsttag, ...)

     struct ea_Object *ea_NewObject(ULONG, ULONG, ...);

FUNCTION
     This is the general method of creating objects.

     You specify a class by its type, which is passed as a number.

     You further specify initial "create-time" attributes for the object
     via a TagItem list, and they are applied to the resulting generic
     data object that is returned. The attributes, their meanings,
     attributes applied only at create-time, and required attributes are
     all defined and documented on a class-by-class basis.

INPUTS
     type - Type of object class you want to create.
     taglist_ptr - Pointer to attay of TagItems containing attribute/value
         pairs to be applied to the object being created.

RESULT
     obj_ptr - An object, which may be manipulated by generic functions.
         You eventually free the object using ea_DisposeObject().

SEE ALSO
     EAGUI.h, EAGUI_macros.h, --classes--
```

1.14 EAGUI.library/ea_NewRelationA()

```
NAME
     ea_NewRelationA -- Create a new relation. (V1)
     ea_NewRelation -- Varargs stub for ea_NewRelationA. (V1)

SYNOPSIS
     result = ea_NewRelationA(obj_ptr, hook_ptr, taglist_ptr)
     D0                       A0       A1         A2

     LONG ea_NewRelationA(struct ea_Object *, struct Hook *,
         struct TagItem *);

     result = ea_NewRelation(obj_ptr, hook_ptr, firsttag, ...)

     LONG ea_NewRelation(struct ea_Object *, struct Hook *, ULONG, ...);

FUNCTION
     Create a relation between children of obj_ptr. The relation is passed
     through a hook. The children are passed via the tags. Please note
     that these children must all be direct descendants of obj_ptr.

     The hook function is called when ea_GetMinSizes() is invoked. Like
     all hooks, it receives the following arguments:

         HookFunc(struct Hook *hook_ptr, struct List *list_ptr, APTR
             msg_ptr)

     Currently, the msg_ptr is not used and should be ignored. Your method
     should walk through the list of ea_RelationObjects, and it may modify
     their EA_MinWidth and EA_MinHeight attributes.

     Returncodes from the hook function are currently ignored, but you
     should return 0 for success. Future versions may check for
     returncodes.

INPUTS
     obj_ptr - Pointer to the object to which the relation is added.
     hook_ptr - Pointer to a hook containing the relation method.
     taglist_ptr - Pointer to the taglist. The only supported tag is
         EA_Object. The data field should contain a pointer to an
         object.

RESULT
     result - returncode as described in EAGUI.h.

SEE ALSO
     utility/hooks.h
```

1.15 EAGUI.library/ea_RenderObjects()

```
NAME
     ea_RenderObjects -- Renders all objects that have a render method.
         (V2)

SYNOPSIS
```

```
     ea_RenderObjects(obj_ptr, rastport_ptr)
                      A0       A1

     VOID ea_RenderObjects(struct ea_Object *, struct RastPort *);

FUNCTION
     Renders an object and all its children. For each object, the render
     method hook is called. This function should be used to render images,
     since gadgets are rendered by Intuition and GadTools.

INPUTS
     obj_ptr - Pointer to the object to be rendered. This must be the same
         object that was used to create the gadget list.

     rastport_ptr - Pointer to the RastPort the render methods will use
         for rendering.

SEE ALSO
     ea_CreateGadgetList()
```

1.16 EAGUI.library/ea_SetAttr()

```
NAME
     ea_SetAttr -- Specify attribute value for an object. (V1)

SYNOPSIS
     result = ea_SetAttr(obj_ptr, attribute, value)
     D0                  A0       D0         D1

     ULONG ea_SetAttr(struct ea_Object *, ULONG, ULONG);

FUNCTION
     Specifies an attribute/value pair with meaning as defined by
     an object's class.

     This function does not provide enough context information or
     arbitration for gadgets which are attached to windows or requesters.

INPUTS
     obj_ptr - Pointer to an object.
     attribute - Tag value of an attribute.
     value - Value of the attribute.

RESULT
     result - Returns 0 for success.

SEE ALSO
     ea_SetAttrs(), --classes--
```

1.17 EAGUI.library/ea_SetAttrsA()

```
NAME
     ea_SetAttrsA -- Specify attribute values for an object. (V1)
     ea_SetAttrs -- Varargs stub for ea_SetAttrsA(). (V1)

SYNOPSIS
     result = ea_SetAttrsA(obj_ptr, taglist_ptr)
     D0                    A0       A1

     ULONG ea_SetAttrsA(struct ea_Object *, struct TagItem *);

     result = ea_SetAttrs(obj_ptr, firsttag, ...)

     ULONG ea_SetAttrs(struct ea_Object *, ULONG, ...);


FUNCTION
     Specifies a set of attribute/value pairs with meaning as defined by
     an object's class.

     This function does not provide enough context information or
     arbitration for gadgets which are attached to windows or requesters.

INPUTS
     obj_ptr - Pointer to an object.
     taglist_ptr - Pointer to an taglist.

RESULT
     result - Returns 0 for success.

SEE ALSO
     ea_SetAttr(), ea_GetAttrs(), --classes--
```

1.18 EAGUI.library/ea_TextHeight()

```
NAME
     ea_TextHeight -- Determine the pixelheight of a text (V1)

SYNOPSIS
     height = ea_TextHeight(textattr_ptr)
     D0                     A0

     LONG ea_TextHeight(struct TextAttr *);

FUNCTION
     This function determines the height of the font in pixels.

INPUTS
     textattr_ptr - Pointer to a TextAttr structure. It is safe to pass a
         NULL pointer here, in which case the function will return zero.

RESULT
     height - Height in pixels, or zero if the height couldn't be
         determined.
```

```
SEE ALSO
     ea_TextLength()
```

1.19 EAGUI.library/ea_TextLength()

```
NAME
     ea_TextLength -- Determine the pixellength of a string. (V1)

SYNOPSIS
     length = ea_TextLength(textattr_ptr, string_ptr, underscore)
     D0                     A0            A1          D0

     LONG ea_TextLength(struct TextAttr *, STRPTR, UBYTE);

FUNCTION
     Determine the pixellength of a given string using the font described
     in the given TextAttr structure. If the string contains an underscore
     character that is used (by GadTools) as an underline marker, set the
     underscore argument to the character that is used as a marker.

INPUTS
     textattr_ptr - Pointer to a TextAttr structure. It is safe to pass a
         NULL pointer here, in which case the function will return zero.
     string_ptr - Pointer to the string. It is safe to pass a NULL pointer
         here, in which case the function will return zero.
     underscore - ASCII value of the character that is used to indicate an
         underlined character. If you don't want this correction, simply
         pass a value of zero.

RESULT
     length - The pixellength of the string, or zero if the length
         couldn't be determined.

SEE ALSO
     ea_TextHeight(), graphics.library/TextLength()
```