Evas Reference Manual

Generated by Doxygen 1.5.1

Wed Mar 28 00:01:31 2007

Contents

Chapter 1

Evas Module Index

1.1 Evas Modules

Here is a list of all modules:

Chapter 2

Evas Data Structure Index

2.1 Evas Data Structures

Here are the data structures with brief descriptions:

Chapter 3

Evas File Index

3.1 Evas File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Evas Page Index

4.1 Evas Related Pages

Here is a list of all related documentation pages:

Chapter 5

Evas Module Documentation

5.1 Object Callback Functions

Functions that add and remove callbacks to evas objects.

Functions

- EAPI void evas_object_event_callback_add (Evas_Object *obj, Evas_Callback_Type type, void(*func)(void *data, Evas *e, Evas_Object *obj, void *event_info), const void *data)

  Add a callback function to an object.

- EAPI void * evas_object_event_callback_del (Evas_Object *obj, Evas_Callback_Type type, void(*func)(void *data, Evas *e, Evas_Object *obj, void *event_info))

  Delete a callback function from an object.

5.1.1 Detailed Description

Functions that add and remove callbacks to evas objects.

5.1.2 Function Documentation

5.1.2.1 EAPI void evas_object_event_callback_add (Evas_Object * *obj*, Evas_Callback_Type *type*, void(*)(void *data, Evas *e, Evas_Object *obj, void *event_info) *func*, const void * *data*)

Add a callback function to an object.

Parameters:

obj Object to attach a callback to

type The type of event that will trigger the callback

func The function to be called when the event is triggered

data The data pointer to be passed to `func`

This function adds a function callback to an object when the event of type `type` occurs on object `obj`. The function is `func`.

In the event of a memory allocation error during addition of the callback to the object, evas_alloc_error() should be used to determine the nature of the error, if any, and the program should sensibly try and recover.

The function will be passed the pointer `data` when it is called. A callback function must look like this:

```
void callback (void *data, Evas *e, Evas_Object *obj, void *event_info);
```

The first parameter `data` in this function will be the same value passed to evas_object_event_callback_add() as the `data` parameter. The second parameter is a convenience for the programmer to know what evas canvas the event occured on. The third parameter `obj` is the Object handle on which the event occured. The foruth parameter `event_info` is a pointer to a data structure that may or may not be passed to the callback, depending on the event type that triggered the callback.

The event type `type` to trigger the function may be one of EVAS_CALLBACK_MOUSE_IN, EVAS_CALLBACK_MOUSE_OUT, EVAS_CALLBACK_MOUSE_DOWN, EVAS_CALLBACK_MOUSE_UP, EVAS_CALLBACK_MOUSE_MOVE, EVAS_CALLBACK_MOUSE_WHEEL, EVAS_CALLBACK_FREE, EVAS_CALLBACK_KEY_DOWN, EVAS_CALLBACK_KEY_UP, EVAS_CALLBACK_FOCUS_IN, EVAS_CALLBACK_FOCUS_OUT, EVAS_CALLBACK_SHOW, EVAS_CALLBACK_HIDE, EVAS_CALLBACK_MOVE, EVAS_CALLBACK_RESIZE or EVAS_CALLBACK_RESTACK. This determines the kind of event that will trigger the callback to be called. The `event_info` pointer passed to the callback will be one of the following, depending on the event triggering it:

EVAS_CALLBACK_MOUSE_IN: event_info = pointer to Evas_Event_Mouse_In

This event is triggered when the mouse pointer enters the region of the object `obj`. This may occur by the mouse pointer being moved by evas_event_feed_mouse_move() or evas_event_feed_mouse_move_data() calls, or by the object being shown, raised, moved, resized, or other objects being moved out of the way, hidden, lowered or moved out of the way.

EVAS_CALLBACK_MOUSE_OUT: event_info = pointer to Evas_Event_Mouse_Out

This event is triggered exactly like EVAS_CALLBACK_MOUSE_IN is, but occurs when the mouse pointer exits an object. Note that no out events will be reported if the mouse pointer is implicitly grabbed to an object (the mouse buttons are down at all and any were pressed on that object). An out event will be reported as soon as the mouse is no longer grabbed (no mouse buttons are depressed). Out events will be reported once all buttons are released, if the mouse has left the object.

EVAS_CALLBACK_MOUSE_DOWN: event_info = pointer to Evas_Event_Mouse_Down

This event is triggered by a mouse button being depressed while over an object. This causes this object to passively grab the mouse until all mouse buttons have been released. That means if this mouse button is the first to be pressed, all future mouse events will be reported to only this object

until no buttons are down. That includes mouse move events, in and out events, and further button presses. When all buttons are released, event propagation occurs as normal.

EVAS_CALLBACK_MOUSE_UP: event_info = pointer to Evas_Event_Mouse_Up

This event is triggered by a mouse button being released while over an object or when passively grabbed to an object. If this is the last mouse button to be raised on an object then the passive grab is released and event processing will continue as normal.

EVAS_CALLBACK_MOUSE_MOVE: event_info = pointer to Evas_Event_Mouse_Move

This event is triggered by the mouse pointer moving while over an object or passively grabbed to an object.

EVAS_CALLBACK_MOUSE_WHEEL: event_info = pointer to Evas_Event_Mouse_Wheel

This event is triggered by the mouse wheel being rolled while over an object or passively grabbed to an object.

EVAS_CALLBACK_FREE: event_info = NULL

This event is triggered just before Evas is about to free all memory used by an object and remove all references to it. This is useful for programs to use if they attached data to an object and want to free it when the object is deleted. The object is still valid when this callback is called, but after this callback returns, there is no guarantee on the object's validity.

EVAS_CALLBACK_KEY_DOWN: event_info = pointer to Evas_Event_Key_Down

This callback is called when a key is pressed and the focus is on the object, or a key has been grabbed to a particular object which wants to intercept the key press regardless of what object has the focus.

EVAS_CALLBACK_KEY_UP: event_info = pointer to Evas_Event_Key_Up

This callback is called when a key is released and the focus is on the object, or a key has been grabbed to a particular object which wants to intercept the key release regardless of what object has the focus.

EVAS_CALLBACK_FOCUS_IN: event_info = NULL

This event is called when an object gains the focus. When the callback is called the object has already gained the focus.

EVAS_CALLBACK_FOCUS_OUT: event_info = NULL

This event is triggered by an object losing the focus. When the callback is called the object has already lost the focus.

EVAS_CALLBACK_SHOW: event_info = NULL

This event is triggered by the object being shown by evas_object_show().

EVAS_CALLBACK_HIDE: event_info = NULL

This event is triggered by an object being hidden by evas_object_hide().

EVAS_CALLBACK_MOVE: event_info = NULL

This event is triggered by an object being moved. evas_object_move() can trigger this, as can any object-specific manipulations that would mean the object's origin could move.

EVAS_CALLBACK_RESIZE: event_info = NULL

This event is triggered by an object being resized. Resizes can be triggered by evas_object_resize() or by any object-specific calls that may cause the object to resize.

Example:

```
extern Evas_Object *object;
extern void *my_data;
void down_callback(void *data, Evas *e, Evas_Object *obj, void *event_info);
void up_callback(void *data, Evas *e, Evas_Object *obj, void *event_info);

evas_object_event_callback_add(object, EVAS_CALLBACK_MOUSE_UP, up_callback, my_data);
if (evas_alloc_error() != EVAS_ALLOC_ERROR_NONE)
  {
    fprintf(stderr, "ERROR: Callback registering failed! Abort!\n");
    exit(-1);
  }
evas_object_event_callback_add(object, EVAS_CALLBACK_MOUSE_DOWN, down_callback, my_data);
if (evas_alloc_error() != EVAS_ALLOC_ERROR_NONE)
  {
    fprintf(stderr, "ERROR: Callback registering failed! Abort!\n");
    exit(-1);
  }
```

5.1.2.2 EAPI void* evas_object_event_callback_del (Evas_Object * *obj*, Evas_Callback_Type *type*, void(*)(void *data, Evas *e, Evas_Object *obj, void *event_info) *func*)

Delete a callback function from an object.

Parameters:

obj Object to remove a callback from

type The type of event that was triggering the callback

func The function that was to be called when the event was triggered

Returns:

The data pointer that was to be passed to the callback

This function removes the most recently added callback from the object `obj` which was triggered by the event type `type` and was calling the function `func` when triggered. If the removal is successful it will also return the data pointer that was passed to evas_object_event_callback_add() when the callback was added to the object. If not successful NULL will be returned.

Example:

```
extern Evas_Object *object;
void *my_data;
void up_callback(void *data, Evas *e, Evas_Object *obj, void *event_info);

my_data = evas_object_event_callback_del(object, EVAS_CALLBACK_MOUSE_UP, up_callback);
```

5.2 Clip Functions

Functions that manage the clipping of objects by other objects.

Functions

- EAPI void evas_object_clip_set (Evas_Object *obj, Evas_Object *clip)
  Clip one object to another.
- EAPI Evas_Object * evas_object_clip_get (Evas_Object *obj)
  Get the object clipping this one (if any).
- EAPI void evas_object_clip_unset (Evas_Object *obj)
  Disable clipping for an object.
- EAPI const Evas_List * evas_object_clipees_get (Evas_Object *obj)
  Return a list of objects currently clipped by a specific object.

5.2.1 Detailed Description

Functions that manage the clipping of objects by other objects.

5.2.2 Function Documentation

5.2.2.1 EAPI Evas_Object* evas_object_clip_get (Evas_Object * *obj*)

Get the object clipping this one (if any).

Parameters:

obj The object to get the clipper from

This function returns the the object clipping `obj`. If `obj` not being clipped, NULL is returned. The object `obj` must be a valid object.

See also evas_object_clip_set(), evas_object_clip_unset() and evas_object_clipees_get().

Example:

```
extern Evas_Object *obj;
Evas_Object *clipper;

clipper = evas_object_clip_get(obj);
if (clipper) evas_object_show(clipper);
```

5.2.2.2 EAPI void evas_object_clip_set (Evas_Object * *obj*, Evas_Object * *clip*)

Clip one object to another.

Parameters:

obj The object to be clipped

clip The object to clip `obj` by

This function will clip the object `obj` to the area occupied by the object `clipper`. This means the object `obj` will only be visible within the area occupied by the clipping object (`clip`). The color of the object being clipped will be multiplied by the color of the clipping object, so the resulting color for the clipped object is RESULT = (OBJ * CLIP) / (255 * 255) per color element (red, green, blue and alpha). Clipping is recursive, so clip objects may be clipped by other objects, and their color will in tern be multiplied. You may NOT set up circular clipping lists (i.e. object 1 clips object 2 which clips object 1). The behavior of Evas is undefined in this case. Objects which do not clip others are visible as normal, those that clip 1 or more objects become invisible themselves, only affecting what they clip. If an object ceases to have other objects being clipped by it, it will become visible again. The visibility of an object affects the objects that are clipped by it, so if the object clipping others is not shown, the objects clipped will not be shown either. If the object was being clipped by another object when this function is called, it is implicitly removed from the clipper it is being clipped to, and now is made to clip its new clipper.

At the moment the only objects that can validly be used to clip other objects are rectangle objects. All other object types are invalid and the result of using them is undefined.

The clip object `clip` must be a valid object, but may also be NULL in which case the effect of this function is the same as calling evas_object_clip_unset() on the `obj` object.

Example:

```
extern Evas *evas;
extern Evas_Object *obj;
Evas_Object *clipper;

clipper = evas_object_rectangle_add(evas);
evas_object_color_set(clipper, 255, 255, 255, 255);
evas_object_move(clipper, 10, 10);
evas_object_resize(clipper, 20, 50);
evas_object_clip_set(obj, clipper);
evas_object_show(clipper);
```

5.2.2.3 EAPI void evas_object_clip_unset (Evas_Object * *obj*)

Disable clipping for an object.

Parameters:

obj The object to cease clipping on

This function disables clipping for the object `obj`, if it was already clipped. If it wasn't, this has no effect. The object `obj` must be a valid object.

See also evas_object_clip_set(), evas_object_clipees_get() and evas_object_clip_get().

Example:

```
extern Evas_Object *obj;
Evas_Object *clipper;

clipper = evas_object_clip_get(obj);
if (clipper)
  {
    evas_object_clip_unset(obj);
    evas_object_hide(obj);
  }
```

5.2.2.4 EAPI const Evas_List* evas_object_clipees_get (Evas_Object * *obj*)

Return a list of objects currently clipped by a specific object.

Parameters:

obj The object to get a list of clippees from

This returns the inernal list handle that contains all objects clipped by the object `obj`. If none are clipped, it returns NULL. This list is only valid until the clip list is changed and should be fetched again with another call to evas_object_clipees_get() if any objects being clipped by this object are unclipped, clipped by a new object, are deleted or the clipper is deleted. These operations will invalidate the list returned so it should not be used anymore after that point. Any use of the list after this may have undefined results, not limited just to strange behavior but possible segfaults and other strange memory errors. The object `obj` must be a valid object.

See also evas_object_clip_set(), evas_object_clip_unset() and evas_object_clip_get().

Example:

```
extern Evas_Object *obj;
Evas_Object *clipper;

clipper = evas_object_clip_get(obj);
if (clipper)
  {
    Evas_List *clippees, *l;

    clippees = evas_object_clipees_get(clipper);
    printf("Clipper clips %i objects\n", evas_list_count(clippees));
    for (l = clippees; l; l = l->next)
      {
        Evas_Object *obj_tmp;

        obj_tmp = l->data;
        evas_object_show(obj_tmp);
      }
  }
```

5.3 Object Data Functions

Functions that retrieve and set data associated attached to an evas object.

Functions

- EAPI void evas_object_data_set (Evas_Object *obj, const char *key, const void *data)
  Set an attached data pointer to an object with a given string key.
- EAPI void * evas_object_data_get (Evas_Object *obj, const char *key)
  Return an attached data pointer by its given string key.
- EAPI void * evas_object_data_del (Evas_Object *obj, const char *key)
  Delete at attached data pointer from an object.

5.3.1 Detailed Description

Functions that retrieve and set data associated attached to an evas object.

5.3.2 Function Documentation

5.3.2.1 EAPI void* evas_object_data_del (Evas_Object * *obj*, const char * *key*)

Delete at attached data pointer from an object.

Parameters:

obj The object to delete the data pointer from

key The string key the data was stored under

Returns:

The original data pointer stored at `key` on `obj`

This will remove thee stored data pointer from `obj` stored under `key`, and return the original pointer stored under `key`, if any, nor NULL if nothing was stored under that key.

Example:

```
int *my_data;
extern Evas_Object *obj;

my_data = evas_object_data_del(obj, "name_of_my_data");
```

5.3.2.2 EAPI void* evas_object_data_get (Evas_Object * *obj*, const char * *key*)

Return an attached data pointer by its given string key.

Parameters:

obj The object to which the data was attached

key The string key the data was stored under

Returns:

The data pointer stored, or NULL if none was stored

This function will return the data pointer attached to the object `obj` stored using the string key `key`. If the object is valid and data was stored under the given key, the pointer that was stored will be reuturned. If this is not the case, NULL will be returned, signifying an invalid object or non-existent key. It is possible a NULL pointer was stored given that key, but this situation is non-sensical and thus can be considered an error as well. NULL pointers are never stored as this is the return value if an error occurs.

Example:

```
int *my_data;
extern Evas_Object *obj;

my_data = evas_object_data_get(obj, "name_of_my_data");
if (my_data) printf("Data stored was %p\n", my_data);
else printf("No data was stored on the object\n");
```

5.3.2.3 EAPI void evas_object_data_set (Evas_Object * *obj*, const char * *key*, const void * *data*)

Set an attached data pointer to an object with a given string key.

Parameters:

obj The object to attach the data pointer to

key The string key for the data to access it

data The ponter to the data to be attached

This attaches the pointer `data` to the object `obj` given the string `key`. This pointer will stay "hooked" to the object until a new pointer with the same string key is attached with evas_object_data_set() or it is deleted with evas_object_data_del(). On deletion of the object `obj`, the pointers will not be accessible from the object anymore.

You can find the pointer attached under a string key using evas_object_data_get(). It is the job of the calling application to free any data pointed to by `data` when it is no longer required.

If `data` is NULL, the old value stored at `key` will be removed but no new value will be stored. This is synonymous with calling evas_object_data_del() with `obj` and `key`.

Example:

```
int *my_data;
extern Evas_Object *obj;

my_data = malloc(500);
evas_object_data_set(obj, "name_of_data", my_data);
printf("The data that was attached was %p\n", evas_object_data_get(obj, "name_of_data"));
```

5.4 Evas Event Freezing Functions

Functions that deal with the freezing of event processing of an evas.

Functions

- EAPI void evas_event_freeze (Evas *e)

  Freeze all event processing.

- EAPI void evas_event_thaw (Evas *e)

  Thaw a canvas out after freezing.

- EAPI int evas_event_freeze_get (Evas *e)

  Return the freeze count of a given canvas.

5.4.1 Detailed Description

Functions that deal with the freezing of event processing of an evas.

5.4.2 Function Documentation

5.4.2.1 EAPI void evas_event_freeze (Evas * *e*)

Freeze all event processing.

Parameters:

e The canvas to freeze event processing on

This function will indicate to evas that the canvas `e` is to have all event processing frozen until a matching evas_event_thaw() function is called on the same canvas. Every freeze call must be matched by a thaw call in order to completely thaw out a canvas.

Example:

```
extern Evas *evas;
extern Evas_Object *object;

evas_event_freeze(evas);
evas_object_move(object, 50, 100);
evas_object_resize(object, 200, 200);
evas_event_thaw(evas);
```

5.4.2.2 EAPI int evas_event_freeze_get (Evas * e)

Return the freeze count of a given canvas.

Parameters:

e The canvas to fetch the freeze count from

This returns the number of times the canvas has been told to freeze events. It is possible to call evas_event_freeze() multiple times, and these must be matched by evas_event_thaw() calls. This call allows the program to discover just how many times things have been frozen in case it may want to break out of a deep freeze state where the count is high.

Example:

```
extern Evas *evas;

while (evas_event_freeze_get(evas) > 0) evas_event_thaw(evas);
```

5.4.2.3 EAPI void evas_event_thaw (Evas * e)

Thaw a canvas out after freezing.

Parameters:

e The canvas to thaw out

This will thaw out a canvas after a matching evas_event_freeze() call. If this call completely thaws out a canvas, events will start being processed again after this call, but this call will not invole any "missed" events to be evaluated.

See evas_event_freeze() for an example.

5.5 Evas Object Event Flag Functions

Functions that deal with how events on an Evas Object are processed.

Functions

- EAPI void evas_object_pass_events_set (Evas_Object *obj, Evas_Bool pass)
  Set an object's pass events state.
- EAPI Evas_Bool evas_object_pass_events_get (Evas_Object *obj)
  Determine whether an object is set to pass events.
- EAPI void evas_object_repeat_events_set (Evas_Object *obj, Evas_Bool repeat)
  Set an object's repeat events state.
- EAPI Evas_Bool evas_object_repeat_events_get (Evas_Object *obj)
  Determine whether an object is set to repeat events.
- EAPI void evas_object_propagate_events_set (Evas_Object *obj, Evas_Bool prop)
  Set whether events on a smart member object should propagate to its parent.
- EAPI Evas_Bool evas_object_propagate_events_get (Evas_Object *obj)
  Determine whether an object is set to propagate events.

5.5.1 Detailed Description

Functions that deal with how events on an Evas Object are processed.

5.5.2 Function Documentation

5.5.2.1 EAPI Evas_Bool evas_object_pass_events_get (Evas_Object * *obj*)

Determine whether an object is set to pass events.

Parameters:

obj

Returns:

pass events state

5.5.2.2 EAPI void evas_object_pass_events_set (Evas_Object * *obj*, Evas_Bool *pass*)

Set an object's pass events state.

Parameters:

obj the evas object

pass whether to pass events or not

If `pass` is true, this will cause events on `obj` to be ignored. They will be triggered on the next lower object (that is not set to pass events) instead.

If `pass` is false, events will be processed as normal.

5.5.2.3 EAPI Evas_Bool evas_object_propagate_events_get (Evas_Object * *obj*)

Determine whether an object is set to propagate events.

Parameters:

obj

Returns:

propogate events state

5.5.2.4 EAPI void evas_object_propagate_events_set (Evas_Object * *obj*, Evas_Bool *prop*)

Set whether events on a smart member object should propagate to its parent.

Parameters:

obj the smart member object

prop wheter to propagate events or not

This function has no effect if `obj` is not a member of a smart object.

If `prop` is true, events occuring on this object will propagate on to the smart object of which `obj` is a member.

If `prop` is false, events for which callbacks are set on the member object, `obj`, will not be passed on to the parent smart object.

The default value is true.

5.5.2.5 EAPI Evas_Bool evas_object_repeat_events_get (Evas_Object * *obj*)

Determine whether an object is set to repeat events.

Parameters:

obj

Returns:

repeat events state

5.5.2.6 EAPI void evas_object_repeat_events_set (Evas_Object * *obj*, Evas_Bool *repeat*)

Set an object's repeat events state.

Parameters:

obj the object

repeat wheter to repeat events or not

If `repeat` is true, this will cause events on `obj` to trigger callbacks, but also to be repeated on the next lower object in the stack.

If `repeat` is false, events occuring on `obj` will be processed normally.

5.6 Object Layer Functions

Functions that retrieve and set the layer that an evas object is on.

Functions

- EAPI void evas_object_layer_set (Evas_Object *obj, int l)
  Sets the layer of the evas that the given object will be part of.
- EAPI int evas_object_layer_get (Evas_Object *obj)
  Retrieves the layer of the evas that the given object is part of.

5.6.1 Detailed Description

Functions that retrieve and set the layer that an evas object is on.

Todo

Document which way layers go.

5.6.2 Function Documentation

5.6.2.1 EAPI int evas_object_layer_get (Evas_Object * *obj*)

Retrieves the layer of the evas that the given object is part of.

Parameters:

obj The given evas object.

Returns:

Number of the layer.

5.6.2.2 EAPI void evas_object_layer_set (Evas_Object * *obj*, int *l*)

Sets the layer of the evas that the given object will be part of.

Parameters:

obj The given evas object.
l The number of the layer to place the object on.

5.7 Evas Canvas

Functions that deal with the basic evas object.

Functions

- EAPI Evas * evas_new (void)

  Creates a new empty evas.

- EAPI void evas_free (Evas *e)

  Frees the given evas and any objects created on it.

5.7.1 Detailed Description

Functions that deal with the basic evas object.

They are the functions you need to use at a minimum to get a working evas, and to destroy it.

5.7.2 Function Documentation

5.7.2.1 EAPI void evas_free (Evas * e)

Frees the given evas and any objects created on it.

Any objects with 'free' callbacks will have those callbacks called in this function.

Parameters:

e The given evas.

5.7.2.2 EAPI Evas* evas_new (void)

Creates a new empty evas.

Note that before you can use the evas, you will to at a minimum:

- Set its render method with evas_output_method_set .
- Set its viewport size with evas_output_viewport_set .
- Set its size of the canvas with evas_output_size_set .
- Ensure that the render engine is given the correct settings with evas_engine_info_set .

This function should only fail if the memory allocation fails.

Returns:

A new uninitialised Evas canvas on success. Otherwise, `NULL`.

5.8 Evas Render Engine Functions

Functions that are used to set the render engine for a given function, and then get that engine working.

Functions

- EAPI void evas_output_method_set (Evas *e, int render_method)

  Sets the output engine for the given evas.

- EAPI int evas_output_method_get (Evas *e)

  Retrieves the number of the output engine used for the given evas.

- EAPI Evas_Engine_Info * evas_engine_info_get (Evas *e)

  Retrieves the current render engine info struct from the given evas.

- EAPI void evas_engine_info_set (Evas *e, Evas_Engine_Info *info)

  Applies the engine settings for the given evas from the given `Evas_Engine_Info` *structure.*

- EAPI int evas_render_method_lookup (const char *name)

  Look up a numeric ID from a string name of a rendering engine.

- EAPI Evas_List * evas_render_method_list (void)

  List all the rendering engines compiled into the copy of the Evas library.

- EAPI void evas_render_method_list_free (Evas_List *list)

  This function should be called to free a list of engine names.

5.8.1 Detailed Description

Functions that are used to set the render engine for a given function, and then get that engine working.

The following code snippet shows how they can be used to initialise an evas that uses the X11 software engine:

```
Evas *evas;
Evas_Engine_Info_Software_X11 *einfo;
extern Display *display;
extern Window win;

evas = evas_new();
evas_output_method_set(evas, evas_render_method_lookup("software_x11"));
evas_output_size_set(evas, 640, 480);
evas_output_viewport_set(evas, 0, 0, 640, 480);
einfo = (Evas_Engine_Info_Software_X11 *)evas_engine_info_get(evas);
einfo->info.display = display;
einfo->info.visual = DefaultVisual(display, DefaultScreen(display));
einfo->info.colormap = DefaultColormap(display, DefaultScreen(display));
```

```
einfo->info.drawable = win;
einfo->info.depth = DefaultDepth(display, DefaultScreen(display));
evas_engine_info_set(evas, (Evas_Engine_Info *)einfo);
```

5.8.2 Function Documentation

5.8.2.1 EAPI Evas_Engine_Info* evas_engine_info_get (Evas * e)

Retrieves the current render engine info struct from the given evas.

The returned structure is publicly modifiable. The contents are valid until either evas_engine_info_set or evas_render are called.

This structure does not need to be freed by the caller.

Parameters:

e The given evas.

Returns:

A pointer to the Engine Info structure. `NULL` is returned if an engine has not yet been assigned.

5.8.2.2 EAPI void evas_engine_info_set (Evas * e, Evas_Engine_Info * *info*)

Applies the engine settings for the given evas from the given `Evas_Engine_Info` structure.

To get the Evas_Engine_Info structure to use, call evas_engine_info_get . Do not try to obtain a pointer to an `Evas_Engine_Info` structure in any other way.

You will need to call this function at least once before you can create objects on an evas or render that evas. Some engines allow their settings to be changed more than once.

Once called, the `info` pointer should be considered invalid.

Example:

Parameters:

e The pointer to the Evas Canvas

info The pointer to the Engine Info to use

5.8.2.3 EAPI int evas_output_method_get (Evas * e)

Retrieves the number of the output engine used for the given evas.

Parameters:

e The given evas.

Returns:

The ID number of the output engine being used. `0` is returned if there is an error.

5.8.2.4 EAPI void evas_output_method_set (Evas * e, int render_method)

Sets the output engine for the given evas.

Once the output engine for an evas is set, any attempt to change it will be ignored. The value for `render_method` can be found using evas_render_method_lookup .

Parameters:

e The given evas.

render_method The numeric engine value to use.

5.8.2.5 EAPI Evas_List* evas_render_method_list (void)

List all the rendering engines compiled into the copy of the Evas library.

Returns:

A linked list whose data members are C strings of engine names

Calling this will return a handle (pointer) to an Evas linked list. Each node in the linked list will have the data pointer be a (char *) pointer to the string name of the rendering engine available. The strings should never be modified, neither should the list be modified. This list should be cleaned up as soon as the program no longer needs it using evas_render_method_list_free(). If no engines are available from Evas, NULL will be returned.

Example:

```
Evas_List *engine_list, *l;

engine_list = evas_render_method_list();
if (!engine_list)
  {
    fprintf(stderr, "ERROR: Evas supports no engines! Exit.\n");
    exit(-1);
  }
printf("Availible Evas Engines:\n");
for (l = engine_list; l; l = l->next)
  {
    char *engine_name;

    engine_name = l->data;
    printf("%s\n", engine_name);
  }
evas_render_method_list_free(engine_list);
```

5.8.2.6 EAPI void evas_render_method_list_free (Evas_List * *list*)

This function should be called to free a list of engine names.

Parameters:

list The Evas_List base pointer for the engine list to be freed

When this function is called it will free the engine list passed in as `list`. The list should only be a list of engines generated by calling evas_render_method_list(). If `list` is NULL, nothing will happen.

Example:

```
Evas_List *engine_list, *l;

engine_list = evas_render_method_list();
if (!engine_list)
  {
    fprintf(stderr, "ERROR: Evas supports no engines! Exit.\n");
    exit(-1);
  }
printf("Availible Evas Engines:\n");
for (l = engine_list; l; l = l->next)
  {
    char *engine_name;

    engine_name = l->data;
    printf("%s\n", engine_name);
  }
evas_render_method_list_free(engine_list);
```

5.8.2.7 EAPI int evas_render_method_lookup (const char * *name*)

Look up a numeric ID from a string name of a rendering engine.

Parameters:

name The string name of an engine

Returns:

A numeric (opaque) ID for the rendering engine

This function looks up a numeric return value for the named engine in the string `name`. This is a normal C string, NUL byte terminated. The name is case sensitive. If the rendering engine is available, a numeric ID for that engine is returned that is not 0. If the engine is not available, 0 is returned, indicating an invalid engine.

The programmer should NEVER rely on the numeric ID of an engine unless it is returned by this function. Programs should NOT be written accessing render method ID's directly, without first obtaining it from this function.

Example:

```
int engine_id;
Evas *evas;
```

```
evas = evas_new();
if (!evas)
  {
    fprintf(stderr, "ERROR: Canvas creation failed. Fatal error.\n");
    exit(-1);
  }
engine_id = evas_render_method_lookup("software_x11");
if (!engine_id)
  {
    fprintf(stderr, "ERROR: Requested rendering engine is absent.\n");
    exit(-1);
  }
evas_output_method_set(evas, engine_id);
```

5.9 Evas Output and Viewport Resizing Functions

Functions that set and retrieve the output and viewport size of an evas.

Functions

- EAPI void evas_output_size_set (Evas *e, int w, int h)

  Sets the output size of the render engine of the given evas.

- EAPI void evas_output_size_get (Evas *e, int *w, int *h)

  Retrieve the output size of the render engine of the given evas.

- EAPI void evas_output_viewport_set (Evas *e, Evas_Coord x, Evas_Coord y, Evas_Coord w, Evas_Coord h)

  Sets the output viewport of the given evas in evas units.

- EAPI void evas_output_viewport_get (Evas *e, Evas_Coord *x, Evas_Coord *y, Evas_Coord *w, Evas_Coord *h)

  Get the render engine's output viewport co-ordinates in canvas units.

5.9.1 Detailed Description

Functions that set and retrieve the output and viewport size of an evas.

5.9.2 Function Documentation

5.9.2.1 EAPI void evas_output_size_get (Evas * *e*, int * *w*, int * *h*)

Retrieve the output size of the render engine of the given evas.

The output size is given in whatever the output units are for the engine.

If either `w` or `h` is `NULL`, then it is ignored. If `e` is invalid, the returned results are undefined.

Parameters:

e The given evas.

w The pointer to an integer to store the width in.

h The pointer to an integer to store the height in.

5.9.2.2 EAPI void evas_output_size_set (Evas * e, int w, int h)

Sets the output size of the render engine of the given evas.

The evas will render to a rectangle of the given size once this function is called. The output size is independent of the viewport size. The viewport will be stretched to fill the given rectangle.

The units used for `w` and `h` depend on the engine used by the evas.

Parameters:

- ***e*** The given evas.
- ***w*** The width in output units, usually pixels.
- ***h*** The height in output units, usually pixels.

5.9.2.3 EAPI void evas_output_viewport_get (Evas * e, Evas_Coord * x, Evas_Coord * y, Evas_Coord * w, Evas_Coord * h)

Get the render engine's output viewport co-ordinates in canvas units.

Parameters:

- ***e*** The pointer to the Evas Canvas
- ***x*** The pointer to a x variable to be filled in
- ***y*** The pointer to a y variable to be filled in
- ***w*** The pointer to a width variable to be filled in
- ***h*** The pointer to a height variable to be filled in

Calling this function writes the current canvas output viewport size and location values into the variables pointed to by `x`, `y`, `w` and `h`. On success the variables have the output location and size values written to them in canvas units. Any of `x`, `y`, `w` or `h` that are NULL will not be written to. If `e` is invalid, the results are undefined.

Example:

```
extern Evas *evas;
Evas_Coord x, y, width, height;

evas_output_viewport_get(evas, &x, &y, &w, &h);
```

5.9.2.4 EAPI void evas_output_viewport_set (Evas * e, Evas_Coord x, Evas_Coord y, Evas_Coord w, Evas_Coord h)

Sets the output viewport of the given evas in evas units.

The output viewport is the area of the evas that will be visible to the viewer. The viewport will be stretched to fit the output target of the evas when rendering is performed.

Note:

The coordinate values do not have to map 1-to-1 with the output target. However, it is generally advised that it is done for ease of use.

Parameters:

e The given evas.

x The top-left corner x value of the viewport.

y The top-left corner y value of the viewport.

w The width of the viewport. Must be greater than 0.

h The height of the viewport. Must be greater than 0.

5.10 Evas Coordinate Mapping Functions

Functions that are used to map coordinates from the canvas to the screen or the screen to the canvas.

Functions

- EAPI Evas_Coord evas_coord_screen_x_to_world (Evas *e, int x)
  Convert/scale an ouput screen co-ordinate into canvas co-ordinates.
- EAPI Evas_Coord evas_coord_screen_y_to_world (Evas *e, int y)
  Convert/scale an ouput screen co-ordinate into canvas co-ordinates.
- EAPI int evas_coord_world_x_to_screen (Evas *e, Evas_Coord x)
  Convert/scale a canvas co-ordinate into output screen co-ordinates.
- EAPI int evas_coord_world_y_to_screen (Evas *e, Evas_Coord y)
  Convert/scale a canvas co-ordinate into output screen co-ordinates.

5.10.1 Detailed Description

Functions that are used to map coordinates from the canvas to the screen or the screen to the canvas.

5.10.2 Function Documentation

5.10.2.1 EAPI Evas_Coord evas_coord_screen_x_to_world (Evas * *e*, int *x*)

Convert/scale an ouput screen co-ordinate into canvas co-ordinates.

Parameters:

- ***e*** The pointer to the Evas Canvas
- ***x*** The screen/output x co-ordinate

Returns:

The screen co-ordinate translated to canvas unit co-ordinates

This function takes in a horizontal co-ordinate as the `x` parameter and converts it into canvas units, accounting for output size, viewport size and location, returning it as the function return value. If `e` is invalid, the results are undefined.

Example:

```
extern Evas *evas;
extern int screen_x;
Evas_Coord canvas_x;

canvas_x = evas_coord_screen_x_to_world(evas, screen_x);
```

5.10.2.2 EAPI Evas_Coord evas_coord_screen_y_to_world (Evas * e, int y)

Convert/scale an ouput screen co-ordinate into canvas co-ordinates.

Parameters:

- ***e*** The pointer to the Evas Canvas
- ***y*** The screen/output y co-ordinate

Returns:

The screen co-ordinate translated to canvas unit co-ordinates

This function takes in a vertical co-ordinate as the `y` parameter and converts it into canvas units, accounting for output size, viewport size and location, returning it as the function return value. If `e` is invalid, the results are undefined.

Example:

```
extern Evas *evas;
extern int screen_y;
Evas_Coord canvas_y;

canvas_y = evas_coord_screen_y_to_world(evas, screen_y);
```

5.10.2.3 EAPI int evas_coord_world_x_to_screen (Evas * e, Evas_Coord x)

Convert/scale a canvas co-ordinate into output screen co-ordinates.

Parameters:

- ***e*** The pointer to the Evas Canvas
- ***x*** The canvas x co-ordinate

Returns:

The output/screen co-ordinate translated to output co-ordinates

This function takes in a horizontal co-ordinate as the `x` parameter and converts it into output units, accounting for output size, viewport size and location, returning it as the function return value. If `e` is invalid, the results are undefined.

Example:

```
extern Evas *evas;
int screen_x;
extern Evas_Coord canvas_x;

screen_x = evas_coord_world_x_to_screen(evas, canvas_x);
```

5.10.2.4 EAPI int evas_coord_world_y_to_screen (Evas * e, Evas_Coord y)

Convert/scale a canvas co-ordinate into output screen co-ordinates.

Parameters:

e The pointer to the Evas Canvas

y The canvas y co-ordinate

Returns:

The output/screen co-ordinate translated to output co-ordinates

This function takes in a vertical co-ordinate as the `x` parameter and converts it into output units, accounting for output size, viewport size and location, returning it as the function return value. If `e` is invalid, the results are undefined.

Example:

```
extern Evas *evas;
int screen_y;
extern Evas_Coord canvas_y;

screen_y = evas_coord_world_y_to_screen(evas, canvas_y);
```

5.11 Evas Pointer Functions

Functions that deal with the status of the pointer.

Functions

- EAPI void evas_pointer_output_xy_get (Evas *e, int *x, int *y)

  This function returns the current known pointer co-ordinates.

- EAPI void evas_pointer_canvas_xy_get (Evas *e, Evas_Coord *x, Evas_Coord *y)

  This function returns the current known pointer co-ordinates.

- EAPI int evas_pointer_button_down_mask_get (Evas *e)

  Returns a bitmask with the mouse buttons currently pressed, set to 1.

- EAPI Evas_Bool evas_pointer_inside_get (Evas *e)

  Returns whether the mouse pointer is logically inside the canvas.

5.11.1 Detailed Description

Functions that deal with the status of the pointer.

5.11.2 Function Documentation

5.11.2.1 EAPI int evas_pointer_button_down_mask_get (Evas * *e*)

Returns a bitmask with the mouse buttons currently pressed, set to 1.

Parameters:

e The pointer to the Evas Canvas

Returns:

A bitmask of the currently depressed buttons on the cavas

Calling this function will return a 32-bit integer with the appropriate bits set to 1 that correspond to a mouse button being depressed. This limits Evas to a mouse devices with a maximum of 32 buttons, but that is generally in excess of any host system's pointing device abilities.

A canvas by default begins with no mouse buttons being pressed and only calls to evas_event_feed_mouse_down(), evas_event_feed_mouse_down_data(), evas_event_feed_mouse_up() and evas_event_feed_mouse_up_data() will alter that.

The least significant bit corresponds to the first mouse button (button 1) and the most significant bit corresponds to the last mouse button (button 32).

If `e` is not a valid canvas, the return value is undefined.

Example:

```
extern Evas *evas;
int button_mask, i;

button_mask = evas_pointer_button_down_mask_get(evas);
printf("Buttons currently pressed:\n");
for (i = 0; i < 32; i++)
  {
    if ((button_mask & (1 << i)) != 0) printf("Button %i\n", i + 1);
  }
```

5.11.2.2 EAPI void evas_pointer_canvas_xy_get (Evas * e, Evas_Coord * x, Evas_Coord * y)

This function returns the current known pointer co-ordinates.

Parameters:

- ***e*** The pointer to the Evas Canvas
- ***x*** The pointer to a Evas_Coord to be filled in
- ***y*** The pointer to a Evas_Coord to be filled in

This function returns the current known canvas unit co-ordinates of the mouse pointer and sets the contents of the Evas_Coords pointed to by `x` and `y` to contain these co-ordinates. If `e` is not a valid canvas the results of this function are undefined.

Example:

```
extern Evas *evas;
Evas_Coord mouse_x, mouse_y;

evas_pointer_output_xy_get(evas, &mouse_x, &mouse_y);
printf("Mouse is at canvas position %f, %f\n", mouse_x, mouse_y);
```

5.11.2.3 EAPI Evas_Bool evas_pointer_inside_get (Evas * e)

Returns whether the mouse pointer is logically inside the canvas.

Parameters:

- ***e*** The pointer to the Evas Canvas

Returns:

An integer that is 1 if the mouse is inside the canvas, 0 otherwise

When this function is called it will return a value of either 0 or 1, depending on if evas_event_feed_mouse_in(), evas_event_feed_mouse_in_data(), or evas_event_feed_mouse_out(), evas_event_feed_mouse_out_data() have been called to feed in a mouse enter event into the canvas.

A return value of 1 indicates the mouse is logically inside the canvas, and 0 implies it is logically outside the canvas.

A canvas begins with the mouse being assumed outside (0).

If `e` is not a valid canvas, the return value is undefined.

Example:

```
extern Evas *evas;

if (evas_pointer_inside_get(evas)) printf("Mouse is in!\n");
else printf("Mouse is out!\n");
```

5.11.2.4 EAPI void evas_pointer_output_xy_get (Evas * e, int * x, int * y)

This function returns the current known pointer co-ordinates.

Parameters:

- ***e*** The pointer to the Evas Canvas
- ***x*** The pointer to an integer to be filled in
- ***y*** The pointer to an integer to be filled in

This function returns the current known screen/output co-ordinates of the mouse pointer and sets the contents of the integers pointed to by `x` and `y` to contain these co-ordinates. If `e` is not a valid canvas the results of this function are undefined.

Example:

```
extern Evas *evas;
int mouse_x, mouse_y;

evas_pointer_output_xy_get(evas, &mouse_x, &mouse_y);
printf("Mouse is at screen position %i, %i\n", mouse_x, mouse_y);
```

5.12 Object Name Function

Functions that retrieve and set the name of an evas object.

Functions

- EAPI void evas_object_name_set (Evas_Object *obj, const char *name)
  Sets the name of the given evas object to the given name.
- EAPI const char * evas_object_name_get (Evas_Object *obj)
  Retrieves the name of the given evas object.
- EAPI Evas_Object * evas_object_name_find (Evas *e, const char *name)
  Retrieves the object on the given evas with the given name.

5.12.1 Detailed Description

Functions that retrieve and set the name of an evas object.

5.12.2 Function Documentation

5.12.2.1 EAPI Evas_Object* evas_object_name_find (Evas * *e*, const char * *name*)

Retrieves the object on the given evas with the given name.

Parameters:

e The given evas.

name The given name.

Returns:

If successful, the evas object with the given name. Otherwise, `NULL`.

5.12.2.2 EAPI const char* evas_object_name_get (Evas_Object * *obj*)

Retrieves the name of the given evas object.

Parameters:

obj The given object.

Returns:

The name of the object. `NULL` if no name has been given to the object.

5.12.2.3 EAPI void evas_object_name_set (Evas_Object * *obj*, const char * *name*)

Sets the name of the given evas object to the given name.

Parameters:

obj The given object.

name The given name.

5.13 Evas Gradient Object Functions

Functions that work on evas gradient objects.

Functions

- EAPI Evas_Object * evas_object_gradient_add (Evas *e)

  Adds a gradient object to the given evas.

- EAPI void evas_object_gradient_color_stop_add (Evas_Object *obj, int r, int g, int b, int a, int delta)

  Adds a color stop to the given evas gradient object.

- EAPI void evas_object_gradient_alpha_stop_add (Evas_Object *obj, int a, int delta)

  Adds an alpha stop to the given evas gradient object.

- EAPI void evas_object_gradient_clear (Evas_Object *obj)

  Deletes all stops set for the given evas gradient object or any set data.

- EAPI void evas_object_gradient_color_data_set (Evas_Object *obj, void *data, int len, Evas_Bool has_alpha)

  Sets color data for the given evas gradient object.

- EAPI void evas_object_gradient_alpha_data_set (Evas_Object *obj, void *data, int len)

  Sets alpha data for the given evas gradient object.

- EAPI void evas_object_gradient_fill_angle_set (Evas_Object *obj, Evas_Angle angle)

  Sets the angle at which the given evas gradient object's fill sits clockwise from vertical.

- EAPI Evas_Angle evas_object_gradient_fill_angle_get (Evas_Object *obj)

  Retrieves the angle at which the given evas gradient object's fill sits clockwise from the vertical.

- EAPI void evas_object_gradient_fill_spread_set (Evas_Object *obj, int spread)

  Sets the tiling mode for the given evas gradient object's fill.

- EAPI int evas_object_gradient_fill_spread_get (Evas_Object *obj)

  Retrieves the spread (tiling mode) for the given gradient object's fill.

- EAPI void evas_object_gradient_angle_set (Evas_Object *obj, Evas_Angle angle)

  Sets the angle at which the given evas gradient sits, relative to whatever intrinsic orientation of the grad type.

- EAPI Evas_Angle evas_object_gradient_angle_get (Evas_Object *obj)

  Retrieves the angle at which the given evas gradient object sits rel to its intrinsic orientation.

- EAPI void evas_object_gradient_offset_set (Evas_Object *obj, float offset)

Sets the offset of the given evas gradient object's spectrum.

- EAPI float evas_object_gradient_offset_get (Evas_Object *obj)

  Retrieves the spectrum's offset.

- EAPI void evas_object_gradient_direction_set (Evas_Object *obj, int direction)

  Sets the direction of the given evas gradient object's spectrum.

- EAPI int evas_object_gradient_direction_get (Evas_Object *obj)

  Retrieves the evas gradient object's spectrum direction.

5.13.1 Detailed Description

Functions that work on evas gradient objects.

The following example shows how

5.13.2 Function Documentation

5.13.2.1 EAPI Evas_Object* evas_object_gradient_add (Evas * *e*)

Adds a gradient object to the given evas.

Parameters:

e The given evas.

Returns:

A new evas gradient object if successful. Otherwise, `NULL`.

5.13.2.2 EAPI void evas_object_gradient_alpha_data_set (Evas_Object * *obj*, void * *data*, int *len*)

Sets alpha data for the given evas gradient object.

If alpha data is so set, any existing gradient stops will be cleared, The data is not copied, so if it was allocated, do not free it while it's set.

Parameters:

obj The given evas gradient object.

data The alpha data to be set, in a8 format.

len The length of the data pointer - multiple of the pixel size.

5.13.2.3 EAPI void evas_object_gradient_alpha_stop_add (Evas_Object * *obj*, int *a*, int *delta*)

Adds an alpha stop to the given evas gradient object.

The `delta` parameter determines the proportion of the gradient object that is to be set to the alpha value.

Alphas are added from the top downwards.

Parameters:

obj The given evas gradient object.

a Alpha value.

delta Proportion of the gradient object that is this alpha.

5.13.2.4 EAPI Evas_Angle evas_object_gradient_angle_get (Evas_Object * *obj*)

Retrieves the angle at which the given evas gradient object sits rel to its intrinsic orientation.

Parameters:

obj The given evas gradient object.

Returns:

The current angle if successful. `0.0` otherwise.

5.13.2.5 EAPI void evas_object_gradient_angle_set (Evas_Object * *obj*, Evas_Angle *angle*)

Sets the angle at which the given evas gradient sits, relative to whatever intrinsic orientation of the grad type.

Used mostly by 'linear' kinds of gradients.

Parameters:

obj The given evas gradient object.

angle Angle in degrees. Can be negative.

5.13.2.6 EAPI void evas_object_gradient_clear (Evas_Object * *obj*)

Deletes all stops set for the given evas gradient object or any set data.

Parameters:

obj The given evas gradient object.

5.13.2.7 EAPI void evas_object_gradient_color_data_set (Evas_Object * *obj*, void * *data*, int *len*, Evas_Bool *has_alpha*)

Sets color data for the given evas gradient object.

If data is so set, any existing gradient stops will be deleted, The data is not copied, so if it was allocated, do not free it while it's set.

Parameters:

obj The given evas gradient object.

data The color data to be set. Should be in argb32 pixel format.

len The length of the data pointer - multiple of the pixel size.

has_alpha A flag indicating if the data has alpha or not.

5.13.2.8 EAPI void evas_object_gradient_color_stop_add (Evas_Object * *obj*, int *r*, int *g*, int *b*, int *a*, int *delta*)

Adds a color stop to the given evas gradient object.

The `delta` parameter determines the proportion of the gradient object that is to be set to the color. For instance, if red is added with `delta` set to 2, and green is added with `delta` set to 1, two-thirds will be red or reddish and one-third will be green or greenish.

Colors are added from the top downwards.

Parameters:

obj The given evas gradient object.

r Red component of the given color.

g Green component of the given color.

b Blue component of the given color.

a Alpha component of the given color.

delta Proportion of the gradient object that is this color.

5.13.2.9 EAPI int evas_object_gradient_direction_get (Evas_Object * *obj*)

Retrieves the evas gradient object's spectrum direction.

Parameters:

obj The given evas gradient object.

Returns:

The current gradient direction if successful. 1 otherwise.

5.13.2.10 EAPI void evas_object_gradient_direction_set (Evas_Object * *obj*, int *direction*)

Sets the direction of the given evas gradient object's spectrum.

Parameters:

obj The given evas gradient object.

direction Values are either 1 (the default) or -1.

5.13.2.11 EAPI Evas_Angle evas_object_gradient_fill_angle_get (Evas_Object * *obj*)

Retrieves the angle at which the given evas gradient object's fill sits clockwise from the vertical.

Parameters:

obj The given evas gradient object.

Returns:

The current angle if successful. `0.0` otherwise.

5.13.2.12 EAPI void evas_object_gradient_fill_angle_set (Evas_Object * *obj*, Evas_Angle *angle*)

Sets the angle at which the given evas gradient object's fill sits clockwise from vertical.

Parameters:

obj The given evas gradient object.

angle Angle in degrees. Can be negative.

5.13.2.13 EAPI int evas_object_gradient_fill_spread_get (Evas_Object * *obj*)

Retrieves the spread (tiling mode) for the given gradient object's fill.

Parameters:

obj The given evas gradient object.

Returns:

The current spread mode of the gradient object.

5.13.2.14 EAPI void evas_object_gradient_fill_spread_set (Evas_Object * *obj*, int *spread*)

Sets the tiling mode for the given evas gradient object's fill.

Parameters:

obj The given evas gradient object.

spread One of EVAS_TEXTURE_REFLECT, EVAS_TEXTURE_REPEAT, EVAS_TEXTURE_RESTRICT, EVAS_TEXTURE_RESTRICT_REFLECT, EVAS_TEXTURE_RESTRICT_REPEAT, or EVAS_TEXTURE_PAD.

5.13.2.15 EAPI float evas_object_gradient_offset_get (Evas_Object * *obj*)

Retrieves the spectrum's offset.

Parameters:

obj The given evas gradient object.

Returns:

The current gradient offset if successful. `0.0` otherwise.

5.13.2.16 EAPI void evas_object_gradient_offset_set (Evas_Object * *obj*, float *offset*)

Sets the offset of the given evas gradient object's spectrum.

Parameters:

obj The given evas gradient object.

offset Values can be negative.

5.14 Gradient Object Fill Rectangle Functions

Functions that deal with what areas of the gradient object are to be tiled with the gradient.

Functions

- EAPI void evas_object_gradient_fill_set (Evas_Object *obj, Evas_Coord x, Evas_Coord y, Evas_Coord w, Evas_Coord h)

  Sets the rectangle on the gradient object that the gradient will be drawn to.

- EAPI void evas_object_gradient_fill_get (Evas_Object *obj, Evas_Coord *x, Evas_Coord *y, Evas_Coord *w, Evas_Coord *h)

  Retrieves the dimensions of the rectangle on the gradient object that the gradient will use as its fill rect.

5.14.1 Detailed Description

Functions that deal with what areas of the gradient object are to be tiled with the gradient.

5.14.2 Function Documentation

5.14.2.1 EAPI void evas_object_gradient_fill_get (Evas_Object * *obj*, Evas_Coord * *x*, Evas_Coord * *y*, Evas_Coord * *w*, Evas_Coord * *h*)

Retrieves the dimensions of the rectangle on the gradient object that the gradient will use as its fill rect.

See evas_object_gradient_fill_set for more details.

Parameters:

obj The given evas gradient object.

x Pointer to an Evas_Coord to store the X coordinate in.

y Pointer to an Evas_Coord to store the Y coordinate in.

w Pointer to an Evas_Coord to store the width in.

h Pointer to an Evas_Coord to store the height in.

5.14.2.2 EAPI void evas_object_gradient_fill_set (Evas_Object * *obj*, Evas_Coord *x*, Evas_Coord *y*, Evas_Coord *w*, Evas_Coord *h*)

Sets the rectangle on the gradient object that the gradient will be drawn to.

Note that the gradient may be tiled around this one rectangle, according to its spread value - restrict, repeat, or reflect. To have only one 'cycle' of the gradient drawn, the spread value must be set to restrict, or `x` and `y` must be 0 and `w` and `h` need to be the width and height of the gradient object respectively.

The default values for the fill parameters is `x` = 0, `y` = 0, `w` = 32 and `h` = 32.

Parameters:

obj The given evas gradient object.

x The X coordinate for the top left corner of the rect.

y The Y coordinate for the top left corner of the rect.

w The width of the rect.

h The height of the rect.

5.15 Gradient Object Type Functions

Functions that set or get a gradient's geometric type.

Functions

- EAPI void evas_object_gradient_type_set (Evas_Object *obj, const char *name, const char *params)

  Sets the geometric type displayed by the given gradient object.

- EAPI void evas_object_gradient_type_get (Evas_Object *obj, char **name, char **params)

  Retrieves the type name and params of the given gradient object.

5.15.1 Detailed Description

Functions that set or get a gradient's geometric type.

Examples are "linear", "linear.diag", "linear.codiag", "radial", "rectangular", "angular", "sinusoidal", ... Some types may accept additional parameters to further specify the look.

5.15.2 Function Documentation

5.15.2.1 EAPI void evas_object_gradient_type_get (Evas_Object * *obj*, char ** *name*, char ** *params*)

Retrieves the type name and params of the given gradient object.

Parameters:

obj The given gradient object.

name Pointer to a character pointer to store the pointer to the type name in.

params Pointer to a character pointer to store the pointer to the type params string in.

5.15.2.2 EAPI void evas_object_gradient_type_set (Evas_Object * *obj*, const char * *name*, const char * *params*)

Sets the geometric type displayed by the given gradient object.

Parameters:

obj The given gradient object.

name Name of the geometric type that the gradient is to be drawn as.

params List of allowable params that the given gradient type allows. Can be NULL.

5.16 Image Object Functions

Functions used to create and manipulate image objects.

Functions

- EAPI Evas_Object * evas_object_image_add (Evas *e)

  Creates a new image object on the given evas.

- EAPI int evas_object_image_load_error_get (Evas_Object *obj)

  Retrieves a number representing any error that occurred during the last load for the given image object.

5.16.1 Detailed Description

Functions used to create and manipulate image objects.

5.16.2 Function Documentation

5.16.2.1 EAPI Evas_Object* evas_object_image_add (Evas * *e*)

Creates a new image object on the given evas.

Parameters:

e The given evas.

Returns:

The created image object.

5.16.2.2 EAPI int evas_object_image_load_error_get (Evas_Object * *obj*)

Retrieves a number representing any error that occurred during the last load for the given image object.

Parameters:

obj The given image object.

Returns:

A value giving the last error that occurred. It should be one of the `EVAS_LOAD_ERROR_*` values. `EVAS_LOAD_ERROR_NONE` is returned if there was no error.

5.17 Image Object File Functions

Functions that write to or retrieve images from files.

Functions

- EAPI void evas_object_image_file_set (Evas_Object *obj, const char *file, const char *key)

  Sets the image displayed by the given image object.

- EAPI void evas_object_image_file_get (Evas_Object *obj, const char **file, const char **key)

  Retrieves the filename and key of the given image object.

5.17.1 Detailed Description

Functions that write to or retrieve images from files.

5.17.2 Function Documentation

5.17.2.1 EAPI void evas_object_image_file_get (Evas_Object * *obj*, const char ** *file*, const char ** *key*)

Retrieves the filename and key of the given image object.

Parameters:

obj The given image object.

file Pointer to a character pointer to store the pointer to the file name in.

key Pointer to a character pointer to store the pointer to the key string in.

5.17.2.2 EAPI void evas_object_image_file_set (Evas_Object * *obj*, const char * *file*, const char * *key*)

Sets the image displayed by the given image object.

Parameters:

obj The given image object.

file Name of the file that the image exists in.

key Can be NULL.

5.18 Image Object Border Functions

Functions that adjust the unscaled image border of image objects.

Functions

- EAPI void evas_object_image_border_set (Evas_Object *obj, int l, int r, int t, int b)

  Sets how much of each border of the given evas image object is not to be scaled.

- EAPI void evas_object_image_border_get (Evas_Object *obj, int *l, int *r, int *t, int *b)

  Retrieves how much of each border of the given evas image is not to be scaled.

- EAPI void evas_object_image_border_center_fill_set (Evas_Object *obj, Evas_Bool fill)

  Sets if the center part of an image (not the border) should be drawn.

- EAPI Evas_Bool evas_object_image_border_center_fill_get (Evas_Object *obj)

  Retrieves If the center of an image object is to be filled or not.

5.18.1 Detailed Description

Functions that adjust the unscaled image border of image objects.

5.18.2 Function Documentation

5.18.2.1 EAPI Evas_Bool evas_object_image_border_center_fill_get (Evas_Object * *obj*)

Retrieves If the center of an image object is to be filled or not.

See evas_object_image_border_set for more information.

Parameters:

obj The given evas image object.

Returns:

If the center is to be filled or not.

5.18.2.2 EAPI void evas_object_image_border_center_fill_set (Evas_Object * *obj*, Evas_Bool *fill*)

Sets if the center part of an image (not the border) should be drawn.

See evas_object_image_border_set for more information.

When rendering, the image may be scaled to fit the size of the image object. This function sets if the center part of the scaled image is to be drawn or left completely blank. Very useful for frames and decorations.

Parameters:

obj The given evas image object.

fill If the center of the image object should be drawn/filled

5.18.2.3 EAPI void evas_object_image_border_get (Evas_Object * *obj*, int * *l*, int * *r*, int * *t*, int * *b*)

Retrieves how much of each border of the given evas image is not to be scaled.

See evas_object_image_border_set for more information.

If any of `l`, `r`, `t` or `b` are `NULL`, then the `NULL` parameter is ignored.

Parameters:

obj The given evas image object.

l Pointer to an integer to store the left border width in.

r Pointer to an integer to store the right border width in.

t Pointer to an integer to store the top border width in.

b Pointer to an integer to store the bottom border width in.

5.18.2.4 EAPI void evas_object_image_border_set (Evas_Object * *obj*, int *l*, int *r*, int *t*, int *b*)

Sets how much of each border of the given evas image object is not to be scaled.

When rendering, the image may be scaled to fit the size of the image object. This function sets what area around the border of the image is not to be scaled. This sort of function is useful for widget theming, where, for example, buttons may be of varying sizes, but the border size must remain constant.

The units used for `l`, `r`, `t` and `b` are output units.

Parameters:

obj The given evas image object.

l Distance of the left border that is not to be stretched.

r Distance of the right border that is not to be stretched.

t Distance of the top border that is not to be stretched.

b Distance of the bottom border that is not to be stretched.

5.19 Image Object Fill Rectangle Functions

Functions that deal with what areas of the image object are to be tiled with the given image.

Functions

- EAPI void evas_object_image_fill_set (Evas_Object *obj, Evas_Coord x, Evas_Coord y, Evas_Coord w, Evas_Coord h)

  Sets the rectangle on the image object that the image will be drawn to.

- EAPI void evas_object_image_fill_get (Evas_Object *obj, Evas_Coord *x, Evas_Coord *y, Evas_Coord *w, Evas_Coord *h)

  Retrieves the dimensions of the rectangle on the image object that the image will be drawn to.

5.19.1 Detailed Description

Functions that deal with what areas of the image object are to be tiled with the given image.

5.19.2 Function Documentation

5.19.2.1 EAPI void evas_object_image_fill_get (Evas_Object * *obj*, Evas_Coord * *x*, Evas_Coord * *y*, Evas_Coord * *w*, Evas_Coord * *h*)

Retrieves the dimensions of the rectangle on the image object that the image will be drawn to.

See evas_object_image_fill_set for more details.

Parameters:

obj The given evas image object.

x Pointer to an integer to store the X coordinate in.

y Pointer to an integer to store the Y coordinate in.

w Pointer to an integer to store the width in.

h Pointer to an integer to store the height in.

5.19.2.2 EAPI void evas_object_image_fill_set (Evas_Object * *obj*, Evas_Coord *x*, Evas_Coord *y*, Evas_Coord *w*, Evas_Coord *h*)

Sets the rectangle on the image object that the image will be drawn to.

Note that the image will be tiled around this one rectangle. To have only one copy of the image drawn, `x` and `y` must be 0 and `w` and `h` need to be the width and height of the image object respectively.

The default values for the fill parameters is `x` = 0, `y` = 0, `w` = 32 and `h` = 32.

Parameters:

obj The given evas image object.

x The X coordinate for the top left corner of the image.

y The Y coordinate for the top left corner of the image.

w The width of the image.

h The height of the image.

5.20 Image Object Image Size Functions

Functions that change the size of the image used by an image object.

Functions

- EAPI void evas_object_image_size_set (Evas_Object *obj, int w, int h)
  Sets the size of the image to be display by the given image object.
- EAPI void evas_object_image_size_get (Evas_Object *obj, int *w, int *h)
  Retrieves the size of the image displayed by the given image object.

5.20.1 Detailed Description

Functions that change the size of the image used by an image object.

5.20.2 Function Documentation

5.20.2.1 EAPI void evas_object_image_size_get (Evas_Object * *obj*, int * *w*, int * *h*)

Retrieves the size of the image displayed by the given image object.

Parameters:

obj The given image object.
w A pointer to an integer in which to store the width. Can be `NULL`.
h A pointer to an integer in which to store the height. Can be `NULL`.

5.20.2.2 EAPI void evas_object_image_size_set (Evas_Object * *obj*, int *w*, int *h*)

Sets the size of the image to be display by the given image object.

This function will scale down or crop the image so that it is treated as if it were at the given size. If the size given is smaller than the image, it will be cropped. If the size given is larger, then the image will be treated as if it were in the upper left hand corner of a larger image that is otherwise transparent.

Parameters:

obj The given image object.
w The new width.
h The new height.

5.21 Line Functions

Functions used to deal with evas line objects.

Functions

- EAPI Evas_Object * evas_object_line_add (Evas *e)

  Adds a new evas line object to the given evas.

- EAPI void evas_object_line_xy_set (Evas_Object *obj, Evas_Coord x1, Evas_Coord y1, Evas_Coord x2, Evas_Coord y2)

  Sets the coordinates of the end points of the given evas line object.

- EAPI void evas_object_line_xy_get (Evas_Object *obj, Evas_Coord *x1, Evas_Coord *y1, Evas_Coord *x2, Evas_Coord *y2)

  Retrieves the coordinates of the end points of the given evas line object.

5.21.1 Detailed Description

Functions used to deal with evas line objects.

5.21.2 Function Documentation

5.21.2.1 EAPI Evas_Object* evas_object_line_add (Evas * *e*)

Adds a new evas line object to the given evas.

Parameters:

e The given evas.

Returns:

The new evas line object.

5.21.2.2 EAPI void evas_object_line_xy_get (Evas_Object * *obj*, Evas_Coord * *x1*, Evas_Coord * *y1*, Evas_Coord * *x2*, Evas_Coord * *y2*)

Retrieves the coordinates of the end points of the given evas line object.

Parameters:

obj The given line object.

x1 Pointer to an integer in which to store the X coordinate of the first end point.

y1 Pointer to an integer in which to store the Y coordinate of the first end point.

x2 Pointer to an integer in which to store the X coordinate of the second end point.

y2 Pointer to an integer in which to store the Y coordinate of the second end point.

5.21.2.3 EAPI void evas_object_line_xy_set (Evas_Object * *obj*, Evas_Coord *x1*, Evas_Coord *y1*, Evas_Coord *x2*, Evas_Coord *y2*)

Sets the coordinates of the end points of the given evas line object.

Parameters:

obj The given evas line object.

x1 The X coordinate of the first point.

y1 The Y coordinate of the first point.

x2 The X coordinate of the second point.

y2 The Y coordinate of the second point.

5.22 Generic Object Functions

Functions that manipulate generic evas objects.

Functions

- EAPI void evas_object_del (Evas_Object *obj)
  Deletes the given evas object and frees its memory.
- EAPI void evas_object_move (Evas_Object *obj, Evas_Coord x, Evas_Coord y)
  Moves the given evas object to the given location.
- EAPI void evas_object_resize (Evas_Object *obj, Evas_Coord w, Evas_Coord h)
  Changes the size of the given evas object.
- EAPI void evas_object_geometry_get (Evas_Object *obj, Evas_Coord *x, Evas_Coord *y, Evas_Coord *w, Evas_Coord *h)
  Retrieves the position and rectangular size of the given evas object.
- EAPI void evas_object_color_set (Evas_Object *obj, int r, int g, int b, int a)
  Sets the general colour of the given evas object to the given colour.
- EAPI void evas_object_color_get (Evas_Object *obj, int *r, int *g, int *b, int *a)
  Retrieves the general colour of the given evas object.
- EAPI void evas_object_anti_alias_set (Evas_Object *obj, Evas_Bool anti_alias)
  Sets whether or not the given evas object is to be drawn anti_aliased.
- EAPI Evas_Bool evas_object_anti_alias_get (Evas_Object *obj)
  Retrieves whether or not the given evas object is to be drawn anti_aliased.
- EAPI void evas_object_color_interpolation_set (Evas_Object *obj, int color_space)
  Sets the color_space to be used for linear interpolation of colors.
- EAPI int evas_object_color_interpolation_get (Evas_Object *obj)
  Retrieves the current value of the color space used for linear interpolation.
- EAPI void evas_object_render_op_set (Evas_Object *obj, Evas_Render_Op render_op)
  Sets the render_op to be used for rendering the evas object.
- EAPI Evas_Render_Op evas_object_render_op_get (Evas_Object *obj)
  Retrieves the current value of the operation used for rendering the evas object.
- EAPI Evas * evas_object_evas_get (Evas_Object *obj)
  Retrieves the evas that the given evas object is on.

- EAPI const char * evas_object_type_get (Evas_Object *obj)

  Retrieves the name of the type of the given evas object.

5.22.1 Detailed Description

Functions that manipulate generic evas objects.

5.22.2 Function Documentation

5.22.2.1 EAPI Evas_Bool evas_object_anti_alias_get (Evas_Object * *obj*)

Retrieves whether or not the given evas object is to be drawn anti_aliased.

Parameters:

obj The given evas object.

Returns:

`1` if the object is to be anti_aliased. `0` otherwise.

5.22.2.2 EAPI void evas_object_anti_alias_set (Evas_Object * *obj*, Evas_Bool *anti_alias*)

Sets whether or not the given evas object is to be drawn anti_aliased.

Parameters:

obj The given evas object.

anti_alias. 1 if the object is to be anti_aliased, 0 otherwise.

5.22.2.3 EAPI void evas_object_color_get (Evas_Object * *obj*, int * *r*, int * *g*, int * *b*, int * *a*)

Retrieves the general colour of the given evas object.

Note that if any of `r`, `g`, `b` or `a` are `NULL`, then the `NULL` parameters are ignored.

Parameters:

obj The given evas object.

r Pointer to an integer in which to store the red component of the colour.

g Pointer to an integer in which to store the green component of the colour.

b Pointer to an integer in which to store the blue component of the colour.

a Pointer to an integer in which to store the alpha component of the colour.

5.22.2.4 EAPI int evas_object_color_interpolation_get (Evas_Object * *obj*)

Retrieves the current value of the color space used for linear interpolation.

Parameters:

obj The given evas object.

Returns:

`EVAS_COLOR_SPACE_ARGB` or EVAS_COLOR_SPACE_AHSV.

5.22.2.5 EAPI void evas_object_color_interpolation_set (Evas_Object * *obj*, int *color_space*)

Sets the color_space to be used for linear interpolation of colors.

Parameters:

obj The given evas object.

color_space,one of EVAS_COLOR_SPACE_ARGB or EVAS_COLOR_SPACE_AHSV.

5.22.2.6 EAPI void evas_object_color_set (Evas_Object * *obj*, int *r*, int *g*, int *b*, int *a*)

Sets the general colour of the given evas object to the given colour.

Parameters:

obj The given evas object.

r The red component of the given colour.

g The green component of the given colour.

b The blue component of the given colour.

a The alpha component of the given colour.

5.22.2.7 EAPI void evas_object_del (Evas_Object * *obj*)

Deletes the given evas object and frees its memory.

The object's 'free' callback is called when this function is called. If the object currently has the focus, its 'focus out' callback is also called.

Parameters:

obj The given evas object.

5.22.2.8 EAPI Evas* evas_object_evas_get (Evas_Object * *obj*)

Retrieves the evas that the given evas object is on.

Parameters:

obj The given evas object.

Returns:

The evas that the object is on.

5.22.2.9 EAPI void evas_object_geometry_get (Evas_Object * *obj*, Evas_Coord * *x*, Evas_Coord * *y*, Evas_Coord * *w*, Evas_Coord * *h*)

Retrieves the position and rectangular size of the given evas object.

Note that if any of `x`, `y`, `w` or `h` are `NULL`, the `NULL` parameters are ignored.

Parameters:

obj The given evas object.
x Pointer to an integer in which to store the X coordinate of the object.
y Pointer to an integer in which to store the Y coordinate of the object.
w Pointer to an integer in which to store the width of the object.
h Pointer to an integer in which to store the height of the object.

5.22.2.10 EAPI void evas_object_move (Evas_Object * *obj*, Evas_Coord *x*, Evas_Coord *y*)

Moves the given evas object to the given location.

Parameters:

obj The given evas object.
x X position to move the object to, in canvas units.
y Y position to move the object to, in canvas units.

5.22.2.11 EAPI Evas_Render_Op evas_object_render_op_get (Evas_Object * *obj*)

Retrieves the current value of the operation used for rendering the evas object.

Parameters:

obj The given evas object.

Returns:

one of the enumerated values in Evas_Render_Op.

5.22.2.12 EAPI void evas_object_render_op_set (Evas_Object * *obj*, Evas_Render_Op *render_op*)

Sets the render_op to be used for rendering the evas object.

Parameters:

obj The given evas object.

render_op one of the Evas_Render_Op values.

5.22.2.13 EAPI void evas_object_resize (Evas_Object * *obj*, Evas_Coord *w*, Evas_Coord *h*)

Changes the size of the given evas object.

Parameters:

obj The given evas object.

w The new width of the evas object.

h The new height of the evas object.

5.22.2.14 EAPI const char* evas_object_type_get (Evas_Object * *obj*)

Retrieves the name of the type of the given evas object.

Parameters:

obj The given object.

Returns:

The name.

5.23 Generic Object Visibility Functions

Functions that deal with the visibility of evas objects.

Functions

- EAPI void evas_object_show (Evas_Object *obj)

  Makes the given evas object visible.

- EAPI void evas_object_hide (Evas_Object *obj)

  Makes the given evas object invisible.

- EAPI Evas_Bool evas_object_visible_get (Evas_Object *obj)

  Retrieves whether or not the given evas object is visible.

5.23.1 Detailed Description

Functions that deal with the visibility of evas objects.

5.23.2 Function Documentation

5.23.2.1 EAPI void evas_object_hide (Evas_Object * *obj*)

Makes the given evas object invisible.

Parameters:

obj The given evas object.

5.23.2.2 EAPI void evas_object_show (Evas_Object * *obj*)

Makes the given evas object visible.

Parameters:

obj The given evas object.

5.23.2.3 EAPI Evas_Bool evas_object_visible_get (Evas_Object * *obj*)

Retrieves whether or not the given evas object is visible.

Parameters:

obj The given evas object.

Returns:

1 if the object is visible. 0 otherwise.

5.24 Object Finder Functions

Functions that determine what evas objects are at a given location or within a given region of an evas.

Functions

- EAPI Evas_Object * evas_object_top_at_xy_get (Evas *e, Evas_Coord x, Evas_Coord y, Evas_Bool include_pass_events_objects, Evas_Bool include_hidden_objects)

  To be documented.

- EAPI Evas_Object * evas_object_top_at_pointer_get (Evas *e)

  To be documented.

- EAPI Evas_Object * evas_object_top_in_rectangle_get (Evas *e, Evas_Coord x, Evas_Coord y, Evas_Coord w, Evas_Coord h, Evas_Bool include_pass_events_objects, Evas_Bool include_hidden_objects)

  To be documented.

- EAPI Evas_List * evas_objects_at_xy_get (Evas *e, Evas_Coord x, Evas_Coord y, Evas_Bool include_pass_events_objects, Evas_Bool include_hidden_objects)

  To be documented.

- EAPI Evas_List * evas_objects_in_rectangle_get (Evas *e, Evas_Coord x, Evas_Coord y, Evas_Coord w, Evas_Coord h, Evas_Bool include_pass_events_objects, Evas_Bool include_hidden_objects)

  To be documented.

5.24.1 Detailed Description

Functions that determine what evas objects are at a given location or within a given region of an evas.

5.24.2 Function Documentation

5.24.2.1 EAPI Evas_Object* evas_object_top_at_pointer_get (Evas * *e*)

To be documented.

FIXME: To be fixed.

5.24.2.2 EAPI Evas_Object* evas_object_top_at_xy_get (Evas * e, Evas_Coord x, Evas_Coord y, Evas_Bool include_pass_events_objects, Evas_Bool include_hidden_objects)

To be documented.

FIXME: To be fixed.

5.24.2.3 EAPI Evas_Object* evas_object_top_in_rectangle_get (Evas * e, Evas_Coord x, Evas_Coord y, Evas_Coord w, Evas_Coord h, Evas_Bool include_pass_events_objects, Evas_Bool include_hidden_objects)

To be documented.

FIXME: To be fixed.

5.24.2.4 EAPI Evas_List* evas_objects_at_xy_get (Evas * e, Evas_Coord x, Evas_Coord y, Evas_Bool include_pass_events_objects, Evas_Bool include_hidden_objects)

To be documented.

FIXME: To be fixed.

5.24.2.5 EAPI Evas_List* evas_objects_in_rectangle_get (Evas * e, Evas_Coord x, Evas_Coord y, Evas_Coord w, Evas_Coord h, Evas_Bool include_pass_events_objects, Evas_Bool include_hidden_objects)

To be documented.

FIXME: To be fixed.

5.25 Polygon Functions

Functions that operate on evas polygon objects.

Functions

- EAPI Evas_Object * evas_object_polygon_add (Evas *e)

  Adds a new evas polygon object to the given evas.

- EAPI void evas_object_polygon_point_add (Evas_Object *obj, Evas_Coord x, Evas_Coord y)

  Adds the given point to the given evas polygon object.

- EAPI void evas_object_polygon_points_clear (Evas_Object *obj)

  Removes all of the points from the given evas polygon object.

5.25.1 Detailed Description

Functions that operate on evas polygon objects.

5.25.2 Function Documentation

5.25.2.1 EAPI Evas_Object* evas_object_polygon_add (Evas * *e*)

Adds a new evas polygon object to the given evas.

Parameters:

e The given evas.

Returns:

A new evas polygon object.

5.25.2.2 EAPI void evas_object_polygon_point_add (Evas_Object * *obj*, Evas_Coord *x*, Evas_Coord *y*)

Adds the given point to the given evas polygon object.

Parameters:

obj The given evas polygon object.

x The X coordinate of the given point.

y The Y coordinate of the given point.

5.25.2.3 EAPI void evas_object_polygon_points_clear (Evas_Object * *obj*)

Removes all of the points from the given evas polygon object.

Parameters:

obj The given polygon object.

5.26 Evas Smart Object Functions

Functions dealing with evas smart objects.

Functions

- EAPI void evas_object_smart_data_set (Evas_Object *obj, void *data)

  Store a pointer to user data for a smart object.

- EAPI void * evas_object_smart_data_get (Evas_Object *obj)

  Retrieve user data stored on a smart object.

- EAPI Evas_Smart * evas_object_smart_smart_get (Evas_Object *obj)

  Get the Evas_Smart from which `obj` *was created.*

- EAPI void evas_object_smart_member_add (Evas_Object *obj, Evas_Object *smart_obj)

  Set an evas object as a member of a smart object.

- EAPI void evas_object_smart_member_del (Evas_Object *obj)

  Removes a member object from a smart object.

- EAPI Evas_Object * evas_object_smart_parent_get (Evas_Object *obj)

  Gets the smart parent of an Evas_Object.

- EAPI Evas_Object * evas_object_smart_add (Evas *e, Evas_Smart *s)

  Instantiates a new smart object described by `s`.

- EAPI void evas_object_smart_callback_add (Evas_Object *obj, const char *event, void(*func)(void *data, Evas_Object *obj, void *event_info), const void *data)

  Add a callback for the smart event specified by `event`.

- EAPI void * evas_object_smart_callback_del (Evas_Object *obj, const char *event, void(*func)(void *data, Evas_Object *obj, void *event_info))

  Remove a smart callback.

- EAPI void evas_object_smart_callback_call (Evas_Object *obj, const char *event, void *event_info)

  Call any smart callbacks on `obj` *for* `event`.

5.26.1 Detailed Description

Functions dealing with evas smart objects.

Smart objects are groupings of primitive evas objects that behave as a cohesive group. For instance, a file manager icon may be a smart object composed of an image object, a text label and two

rectangles that appear behind the image and text when the icon is selected. As a smart object, the normal evas api could be used on the icon object.

5.26.2 Function Documentation

5.26.2.1 EAPI Evas_Object* evas_object_smart_add (Evas * e, Evas_Smart * s)

Instantiates a new smart object described by `s`.

Parameters:

e the evas on which to add the object

s the Evas_Smart describing the smart object

Returns:

a new Evas_Object

5.26.2.2 EAPI void evas_object_smart_callback_add (Evas_Object * obj, const char * event, void(*)(void *data, Evas_Object *obj, void *event_info) func, const void * data)

Add a callback for the smart event specified by `event`.

Parameters:

obj a smart object

event the event name

func the callback function

data user data to be passed to the callback function

5.26.2.3 EAPI void evas_object_smart_callback_call (Evas_Object * obj, const char * event, void * event_info)

Call any smart callbacks on `obj` for `event`.

Parameters:

obj the smart object

event the event name

event_info an event specific struct of info to pass to the callback

This should be called internally in the smart object when some specific event has occured. The documentation for the smart object should include a list of possible events and what type of `event_info` to expect.

5.26.2.4 EAPI void* evas_object_smart_callback_del (Evas_Object * *obj*, const char * *event*, void(*)(void *data, Evas_Object *obj, void *event_info) *func*)

Remove a smart callback.

Removes a callback that was added by evas_object_smart_callback_add()

Parameters:

obj a smart object

event the event name

func the callback function

Returns:

the data pointer

5.26.2.5 EAPI void* evas_object_smart_data_get (Evas_Object * *obj*)

Retrieve user data stored on a smart object.

Parameters:

obj The smart object

Returns:

A pointer to data stored using evas_object_smart_data_set(), or NULL if none has been set.

5.26.2.6 EAPI void evas_object_smart_data_set (Evas_Object * *obj*, void * *data*)

Store a pointer to user data for a smart object.

Parameters:

obj The smart object

data A pointer to user data

5.26.2.7 EAPI void evas_object_smart_member_add (Evas_Object * *obj*, Evas_Object * *smart_obj*)

Set an evas object as a member of a smart object.

Parameters:

obj The member object

smart_obj The smart object

Members will automatically be stacked and layered with the smart object. The various stacking function will operate on members relative to the other members instead of the entire canvas.

Non-member objects can not interleave a smart object's members.

5.26.2.8 EAPI void evas_object_smart_member_del (Evas_Object * *obj*)

Removes a member object from a smart object.

Parameters:

obj the member object

This removes a member object from a smart object. The object will still be on the canvas, but no longer associated with whichever smart object it was associated with.

5.26.2.9 EAPI Evas_Object* evas_object_smart_parent_get (Evas_Object * *obj*)

Gets the smart parent of an Evas_Object.

Parameters:

obj the Evas_Object you want to get the parent

Returns:

Returns the smart parent of *obj*, or NULL if *obj* is not a smart member of another Evas_Object

5.26.2.10 EAPI Evas_Smart* evas_object_smart_smart_get (Evas_Object * *obj*)

Get the Evas_Smart from which `obj` was created.

Parameters:

obj a smart object

Returns:

the Evas_Smart

5.27 Font Path Functions

Functions that edit the paths being used to load fonts.

Functions

- EAPI void evas_font_path_clear (Evas *e)
  Removes all font paths loaded into memory for the given evas.
- EAPI void evas_font_path_append (Evas *e, const char *path)
  Appends a font path to the list of font paths used by the given evas.
- EAPI void evas_font_path_prepend (Evas *e, const char *path)
  Prepends a font path to the list of font paths used by the given evas.
- EAPI const Evas_List * evas_font_path_list (Evas *e)
  Retrieves the list of font paths used by the given evas.

5.27.1 Detailed Description

Functions that edit the paths being used to load fonts.

5.27.2 Function Documentation

5.27.2.1 EAPI void evas_font_path_append (Evas * *e*, const char * *path*)

Appends a font path to the list of font paths used by the given evas.

Parameters:

e The given evas.
path The new font path.

5.27.2.2 EAPI void evas_font_path_clear (Evas * *e*)

Removes all font paths loaded into memory for the given evas.

Parameters:

e The given evas.

5.27.2.3 EAPI const Evas_List* evas_font_path_list (Evas * e)

Retrieves the list of font paths used by the given evas.

Parameters:

e The given evas.

Returns:

The list of font paths used.

5.27.2.4 EAPI void evas_font_path_prepend (Evas * e, const char * path)

Prepends a font path to the list of font paths used by the given evas.

Parameters:

e The given evas.

path The new font path.

5.28 Evas Smart Functions

Functions that deal with Evas_Smart's.

5.29 Hash Data Functions

Functions that add, access or remove data from hashes.

Functions

- EAPI Evas_Hash * evas_hash_add (Evas_Hash *hash, const char *key, const void *data)

  Adds an entry to the given hash table.

- EAPI Evas_Hash * evas_hash_direct_add (Evas_Hash *hash, const char *key, const void *data)

  Adds an entry to the given hash table and does not duplicate the string key.

- EAPI Evas_Hash * evas_hash_del (Evas_Hash *hash, const char *key, const void *data)

  Removes the entry identified by `key` *or* `data` *from the given hash table.*

- EAPI void * evas_hash_find (Evas_Hash *hash, const char *key)

  Retrieves a specific entry in the given hash table.

- EAPI void * evas_hash_modify (Evas_Hash *hash, const char *key, const void *data)

  Modifies the entry pointer at the specified key and returns the old entry.

5.29.1 Detailed Description

Functions that add, access or remove data from hashes.

The following example shows how to add and then access data in a hash table:

```
Evas_Hash *hash = NULL;
extern void *my_data;

hash = evas_hash_add(hash, "My Data", my_data);
if (evas_hash_alloc_error())
  {
    fprintf(stderr, "ERROR: Memory is low. Hash allocation failed.\n");
    exit(-1);
  }
if (evas_hash_find(hash, "My Data") == my_data)
  {
    printf("My Data inserted and successfully found.\n");
  }
```

What follows is another example, showing how the evas_hash_del function is used:

```
extern Evas_Hash *hash;
extern void *data;

printf("Insert some data...\n");
hash = evas_hash_add(hash, "My Data", my_data);
printf("Removing by key...\n");
```

```
hash = evas_hash_del(hash, "My Data", NULL);
printf("Insert some more data as a NULL key...\n");
hash = evas_hash_add(hash, NULL, my_data);
printf("Removing by data as a NULL key...\n");
hash = evas_hash_del(hash, NULL, my_data);
```

5.29.2 Function Documentation

5.29.2.1 EAPI Evas_Hash* evas_hash_add (Evas_Hash * *hash*, const char * *key*, const void * *data*)

Adds an entry to the given hash table.

`key` is expected to be a unique string within the hash table. Otherwise, you cannot be sure which inserted data pointer will be accessed with evas_hash_find , and removed with evas_hash_del .

Key strings are case sensitive.

evas_hash_alloc_error should be used to determine if an allocation error occurred during this function.

Parameters:

hash The given hash table. Can be `NULL`, in which case a new hash table is allocated and returned.

key A unique string. Can be `NULL`.

data Data to associate with the string given by `key`.

Returns:

Either the given hash table, or if the given value for `hash` is `NULL`, then a new one. `NULL` will be returned if memory could not be allocated for a new table.

5.29.2.2 EAPI Evas_Hash* evas_hash_del (Evas_Hash * *hash*, const char * *key*, const void * *data*)

Removes the entry identified by `key` or `data` from the given hash table.

If `key` is `NULL`, then `data` is used to find a match to remove.

Parameters:

hash The given hash table.

key The key string. Can be `NULL`.

data The data pointer to remove if `key` is `NULL`. Otherwise, not required and can be `NULL`.

Returns:

The modified hash table. If there are no entries left, the hash table will be freed and `NULL` will be returned.

5.29.2.3 EAPI Evas_Hash* evas_hash_direct_add (Evas_Hash * *hash*, const char * *key*, const void * *data*)

Adds an entry to the given hash table and does not duplicate the string key.

`key` is expected to be a unique string within the hash table. Otherwise, you cannot be sure which inserted data pointer will be accessed with evas_hash_find , and removed with evas_hash_del . This call does nto make a copy of the key so it must be a string constant or stored elsewhere (in the object being added) etc.

Key strings are case sensitive.

evas_hash_alloc_error should be used to determine if an allocation error occurred during this function.

Parameters:

hash The given hash table. Can be `NULL`, in which case a new hash table is allocated and returned.

key A unique string. Can be `NULL`.

data Data to associate with the string given by `key`.

Returns:

Either the given hash table, or if the given value for `hash` is `NULL`, then a new one. `NULL` will be returned if memory could not be allocated for a new table.

5.29.2.4 EAPI void* evas_hash_find (Evas_Hash * *hash*, const char * *key*)

Retrieves a specific entry in the given hash table.

Parameters:

hash The given hash table.

key The key string of the entry to find.

Returns:

The data pointer for the stored entry, or `NULL` if not found.

5.29.2.5 EAPI void* evas_hash_modify (Evas_Hash * *hash*, const char * *key*, const void * *data*)

Modifies the entry pointer at the specified key and returns the old entry.

Parameters:

hash The given hash table.

key The key string of the entry to modify.

data The data to replace the old entry, if it exists.

Returns:

The data pointer for the old stored entry, or `NULL` if not found. If an existing entry is not found, nothing is added to the hash.

5.30 Hash General Functions

Miscellaneous functions that operate on hash objects.

Functions

- EAPI int evas_hash_size (Evas_Hash *hash)

  Retrieves the number of buckets available in the given hash table.

- EAPI void evas_hash_free (Evas_Hash *hash)

  Free an entire hash table.

- EAPI void evas_hash_foreach (Evas_Hash *hash, Evas_Bool(*func)(Evas_Hash *hash, const char *key, void *data, void *fdata), const void *fdata)

  Call a function on every member stored in the hash table.

- EAPI int evas_hash_alloc_error (void)

  Return memory allocation failure flag after an function requiring allocation.

5.30.1 Detailed Description

Miscellaneous functions that operate on hash objects.

5.30.2 Function Documentation

5.30.2.1 EAPI int evas_hash_alloc_error (void)

Return memory allocation failure flag after an function requiring allocation.

Returns:

The state of the allocation flag

This function returns the state of the memory allocation flag. This flag is set if memory allocations fail during evas_hash_add() calls. If they do, 1 will be returned, otherwise 0 will be returned. The flag will remain in its current state until the next call that requires allocation is called, and is then reset.

Example:

```
Evas_Hash *hash = NULL;
extern void *my_data;

hash = evas_hash_add(hash, "My Data", my_data);
if (evas_hash_alloc_error())
```

```
  {
    fprintf(stderr, "ERROR: Memory is low. Hash allocation failed.\n");
    exit(-1);
  }
if (evas_hash_find(hash, "My Data") == my_data)
  {
    printf("My Data inserted and successfully found.\n");
  }
```

5.30.2.2 EAPI void evas_hash_foreach (Evas_Hash * *hash*, Evas_Bool(*)(Evas_Hash *hash, const char *key, void *data, void *fdata) *func*, const void * *fdata*)

Call a function on every member stored in the hash table.

Parameters:

hash The hash table whose members will be walked

func The function to call on each parameter

fdata The data pointer to pass to the function being called

This function goes through every entry in the hash table `hash` and calls the function `func` on each member. The function should NOT modify the hash table contents if it returns 1. IF the hash table contents are modified by this function or the function wishes to stop processing it must return 0, otherwise return 1 to keep processing.

Example:

```
extern Evas_Hash *hash;

Evas_Bool hash_fn(Evas_Hash *hash, const char *key, void *data, void *fdata)
{
  printf("Func data: %s, Hash entry: %s / %p\n", fdata, key, data);
  return 1;
}

int main(int argc, char **argv)
{
  char *hash_fn_data;

  hash_fn_data = strdup("Hello World");
  evas_hash_foreach(hash, hash_fn, hash_fn_data);
  free(hash_fn_data);
}
```

5.30.2.3 EAPI void evas_hash_free (Evas_Hash * *hash*)

Free an entire hash table.

Parameters:

hash The hash table to be freed

This function frees up all the memory allocated to storing the specified hash tale pointed to by `hash`. Any entries in the table that the program has no more pointers for elsewhere may now be lost, so this should only be called if the program has lready freed any allocated data in the hash table or has the pointers for data in teh table stored elswehere as well.

Example:

```
extern Evas_Hash *hash;

evas_hash_free(hash);
hash = NULL;
```

5.30.2.4 EAPI int evas_hash_size (Evas_Hash * *hash*)

Retrieves the number of buckets available in the given hash table.

Parameters:

hash The given hash table.

Returns:

256 if `hash` is not `NULL`. 0 otherwise.

5.31 Linked List Creation Functions

Functions that add data to an Evas_List.

Functions

- EAPI Evas_List * evas_list_append (Evas_List *list, const void *data)

  Appends the given data to the given linked list.

- EAPI Evas_List * evas_list_prepend (Evas_List *list, const void *data)

  Prepends the given data to the given linked list.

- EAPI Evas_List * evas_list_append_relative (Evas_List *list, const void *data, const void *relative)

  Inserts the given data into the given linked list after the specified data.

- EAPI Evas_List * evas_list_prepend_relative (Evas_List *list, const void *data, const void *relative)

  Prepend a data pointer to a linked list before the memeber specified.

5.31.1 Detailed Description

Functions that add data to an Evas_List.

5.31.2 Function Documentation

5.31.2.1 EAPI Evas_List* evas_list_append (Evas_List * *list*, const void * *data*)

Appends the given data to the given linked list.

The following example code demonstrates how to ensure that the given data has been successfully appended.

```
Evas_List *list = NULL;
extern void *my_data;

list = evas_list_append(list, my_data);
if (evas_list_alloc_error())
  {
    fprintf(stderr, "ERROR: Memory is low. List allocation failed.\n");
    exit(-1);
  }
```

Parameters:

list The given list. If `NULL` is given, then a new list is created.

data The data to append.

Returns:

A new list pointer that should be used in place of the one given to this function if successful. Otherwise, the old pointer is returned.

5.31.2.2 EAPI Evas_List* evas_list_append_relative (Evas_List * *list*, const void * *data*, const void * *relative*)

Inserts the given data into the given linked list after the specified data.

If `relative` is not in the list, `data` is appended to the end of the list. If there are multiple instances of `relative` in the list, `data` is inserted after the first instance.

The following example code demonstrates how to ensure that the given data has been successfully inserted.

```
Evas_List *list = NULL;
extern void *my_data;
extern void *relative_member;

list = evas_list_append(list, relative_member);
if (evas_list_alloc_error())
  {
    fprintf(stderr, "ERROR: Memory is low. List allocation failed.\n");
    exit(-1);
  }
list = evas_list_append_relative(list, my_data, relative_member);
if (evas_list_alloc_error())
  {
    fprintf(stderr, "ERROR: Memory is low. List allocation failed.\n");
    exit(-1);
  }
```

Parameters:

list The given linked list.

data The given data.

relative The data to insert after.

Returns:

A new list pointer that should be used in place of the one given to this function if successful. Otherwise, the old pointer is returned.

5.31.2.3 EAPI Evas_List* evas_list_prepend (Evas_List * *list*, const void * *data*)

Prepends the given data to the given linked list.

The following example code demonstrates how to ensure that the given data has been successfully prepended.

Example:

```
Evas_List *list = NULL;
extern void *my_data;

list = evas_list_prepend(list, my_data);
if (evas_list_alloc_error())
  {
    fprintf(stderr, "ERROR: Memory is low. List allocation failed.\n");
    exit(-1);
  }
```

Parameters:

list The given list.

data The given data.

Returns:

A new list pointer that should be used in place of the one given to this function, if successful. Otherwise, the old pointer is returned.

5.31.2.4 EAPI Evas_List* evas_list_prepend_relative (Evas_List * *list*, const void * *data*, const void * *relative*)

Prepend a data pointer to a linked list before the memeber specified.

Parameters:

list The list handle to prepend `data` too

data The data pointer to prepend to list `list` before `relative`

relative The data pointer before which to insert `data`

Returns:

A new list handle to replace the old one

Inserts the given data into the given linked list before the member specified.

If `relative` is not in the list, `data` is prepended to the start of the list. If there are multiple instances of `relative` in the list, `data` is inserted before the first instance.

The following code example demonstrates how to ensure that the given data has been successfully inserted.

```
Evas_List *list = NULL;
extern void *my_data;
extern void *relative_member;

list = evas_list_append(list, relative_member);
if (evas_list_alloc_error())
  {
    fprintf(stderr, "ERROR: Memory is low. List allocation failed.\n");
    exit(-1);
  }
list = evas_list_prepend_relative(list, my_data, relative_member);
if (evas_list_alloc_error())
  {
    fprintf(stderr, "ERROR: Memory is low. List allocation failed.\n");
    exit(-1);
  }
```

Parameters:

list The given linked list.

data The given data.

relative The data to insert before.

Returns:

A new list pointer that should be used in place of the one given to this function if successful. Otherwise the old pointer is returned.

5.32 Linked List Remove Functions

Functions that remove data from linked lists.

Functions

- EAPI Evas_List * evas_list_remove (Evas_List *list, const void *data)

  Removes the first instance of the specified data from the given list.

- EAPI Evas_List * evas_list_remove_list (Evas_List *list, Evas_List *remove_list)

  Removes the specified data.

- EAPI Evas_List * evas_list_free (Evas_List *list)

  Free an entire list and all the nodes, ignoring the data contained.

5.32.1 Detailed Description

Functions that remove data from linked lists.

5.32.2 Function Documentation

5.32.2.1 EAPI Evas_List* evas_list_free (Evas_List * *list*)

Free an entire list and all the nodes, ignoring the data contained.

Parameters:

list The list to free

Returns:

A NULL pointer

This function will free all the list nodes in list specified by `list`.

Example:

```
extern Evas_List *list;

list = evas_list_free(list);
```

5.32.2.2 EAPI Evas_List* evas_list_remove (Evas_List * *list*, const void * *data*)

Removes the first instance of the specified data from the given list.

If the specified data is not in the given list, nothing is done.

Parameters:

list The given list.

data The specified data.

Returns:

A new list pointer that should be used in place of the one passed to this functions.

5.32.2.3 EAPI Evas_List* evas_list_remove_list (Evas_List * *list*, Evas_List * *remove_list*)

Removes the specified data.

Remove a specified member from a list

Parameters:

list The list handle to remove `remove_list` from

remove_list The list node which is to be removed

Returns:

A new list handle to replace the old one

Calling this function takes the list note `remove_list` and removes it from the list `list`, freeing the list node structure `remove_list`.

Example:

```
extern Evas_List *list;
Evas_List *l;
extern void *my_data;

for (l = list; l; l= l->next)
  {
    if (l->data == my_data)
      {
        list = evas_list_remove_list(list, l);
        break;
      }
  }
```

5.33 Linked List Find Functions

Functions that find specified data in a linked list.

Functions

- EAPI void * evas_list_find (Evas_List *list, const void *data)

  Find a member of a list and return the member.

- EAPI Evas_List * evas_list_find_list (Evas_List *list, const void *data)

  Find a member of a list and return the list node containing that member.

- EAPI void * evas_list_nth (Evas_List *list, int n)

  Get the nth member's data pointer in a list.

- EAPI Evas_List * evas_list_nth_list (Evas_List *list, int n)

  Get the nth member's list node in a list.

5.33.1 Detailed Description

Functions that find specified data in a linked list.

5.33.2 Function Documentation

5.33.2.1 EAPI void* evas_list_find (Evas_List * *list*, const void * *data*)

Find a member of a list and return the member.

Parameters:

list The list handle to search for `data`

data The data pointer to find in the list `list`

Returns:

The found member data pointer

A call to this function will search the list `list` from beginning to end for the first member whose data pointer is `data`. If it is found, `data` will be returned, otherwise NULL will be returned.

Example:

```
extern Evas_List *list;
extern void *my_data;

if (evas_list_find(list, my_data) == my_data)
  {
    printf("Found member %p\n", my_data);
  }
```

5.33.2.2 EAPI Evas_List* evas_list_find_list (Evas_List * *list*, const void * *data*)

Find a member of a list and return the list node containing that member.

Parameters:

list The list handle to search for `data`

data The data pointer to find in the list `list`

Returns:

The found members list node

A call to this function will search the list `list` from beginning to end for the first member whose data pointer is `data`. If it is found, the list node containing the specified member will be returned, otherwise NULL will be returned.

Example:

```
extern Evas_List *list;
extern void *my_data;
Evas_List *found_node;

found_node = evas_list_find_list(list, my_data);
if (found_node)
  {
    printf("Found member %p\n", found_node->data);
  }
```

5.33.2.3 EAPI void* evas_list_nth (Evas_List * *list*, int *n*)

Get the nth member's data pointer in a list.

Parameters:

list The list to get member number `n` from

n The number of the element (0 being the first)

Returns:

The data pointer stored in the specified element

This function returns the data pointer of element number `n`, in the list `list`. The first element in the array is element number 0. If the element number `n` does not exist, NULL will be returned.

Example:

```
extern Evas_List *list;
extern int number;
void *data;

data = evas_list_nth(list, number);
if (data)
  printf("Element number %i has data %p\n", number, data);
```

5.33.2.4 EAPI Evas_List* evas_list_nth_list (Evas_List * *list*, int *n*)

Get the nth member's list node in a list.

Parameters:

list The list to get member number `n` from

n The number of the element (0 being the first)

Returns:

The list node stored in the numbered element

This function returns the list node of element number `n`, in the list `list`. The first element in the array is element number 0. If the element number `n` does not exist, NULL will be returned.

Example:

```
extern Evas_List *list;
extern int number;
Evas_List *nth_list;

nth_list = evas_list_nth_list(list, number);
if (nth_list)
  printf("Element number %i has data %p\n", number, nth_list->data);
```

5.34 Linked List Traverse Functions

Functions that you can use to traverse a linked list.

Functions

- EAPI Evas_List * evas_list_last (Evas_List *list)
  Get the last list node in the list.
- EAPI Evas_List * evas_list_next (Evas_List *list)
  Get the next list node after the specified list node.
- EAPI Evas_List * evas_list_prev (Evas_List *list)
  Get the previous list node before the specified list node.

5.34.1 Detailed Description

Functions that you can use to traverse a linked list.

5.34.2 Function Documentation

5.34.2.1 EAPI Evas_List* evas_list_last (Evas_List * *list*)

Get the last list node in the list.

Parameters:

list The list to get the last list node from

Returns:

The last list node in the list `list`

This function will return the last list node in the list (or NULL if the list is empty).

NB: This is a order-1 operation (it takes the same short time regardless of the length of the list).

Example:

```
extern Evas_List *list;
Evas_List *last, *l;

last = evas_list_last(list);
printf("The list in reverse:\n");
for (l = last; l; l = l->prev)
  {
    printf("%p\n", l->data);
  }
```

5.34.2.2 EAPI Evas_List* evas_list_next (Evas_List * *list*)

Get the next list node after the specified list node.

Parameters:

list The list node to get the next list node from

Returns:

The next list node, or NULL if no next list node exists

This function returns the next list node after the current one. It is equivalent to list->next.

Example:

```
extern Evas_List *list;
Evas_List *l;

printf("The list:\n");
for (l = list; l; l = evas_list_next(l))
  {
    printf("%p\n", l->data);
  }
```

5.34.2.3 EAPI Evas_List* evas_list_prev (Evas_List * *list*)

Get the previous list node before the specified list node.

Parameters:

list The list node to get the previous list node from

Returns:

The previous list node, or NULL if no previous list node exists

This function returns the previous list node before the current one. It is equivalent to list->prev.

Example:

```
extern Evas_List *list;
Evas_List *last, *l;

last = evas_list_last(list);
printf("The list in reverse:\n");
for (l = last; l; l = evas_list_prev(l))
  {
    printf("%p\n", l->data);
  }
```

5.35 Linked List General Functions

Miscellaneous functions that work on linked lists.

Functions

- EAPI void * evas_list_data (Evas_List *list)
  Get the list node data member.
- EAPI int evas_list_count (Evas_List *list)
  Get the count of the number of items in a list.
- EAPI int evas_list_alloc_error (void)
  Return the memory allocation failure flag after any operation needin allocation.

5.35.1 Detailed Description

Miscellaneous functions that work on linked lists.

5.35.2 Function Documentation

5.35.2.1 EAPI int evas_list_alloc_error (void)

Return the memory allocation failure flag after any operation needin allocation.

Returns:

The state of the allocation flag

This function returns the state of the memory allocation flag. This flag is set if memory allocations during evas_list_append(), evas_list_prepend(), evas_list_append_relative(), or evas_list_prepend_relative() fail. If they do fail, 1 will be returned, otherwise 0 will be returned. The flag will remain in its current state until the next call that requires allocation is called, and is then reset.

Example:

```
Evas_List *list = NULL;
extern void *my_data;

list = evas_list_append(list, my_data);
if (evas_list_alloc_error())
  {
    fprintf(stderr, "ERROR: Memory is low. List allocation failed.\n");
    exit(-1);
  }
```

5.35.2.2 EAPI int evas_list_count (Evas_List * *list*)

Get the count of the number of items in a list.

Parameters:

list The list whose count to return

Returns:

The number of members in the list `list`

This function returns how many members in the specified list: `list`. If the list is empty (NULL), 0 is returned.

NB: This is an order-1 operation and takes the same tiem regardless of the length of the list.

Example:

```
extern Evas_List *list;

printf("The list has %i members\n", evas_list_count(list));
```

5.35.2.3 EAPI void* evas_list_data (Evas_List * *list*)

Get the list node data member.

Parameters:

list The list node to get the data member of

Returns:

The data member from the list node `list`

This function returns the data member of the specified list node `list`. It is equivalent to list->data.

Example:

```
extern Evas_List *list;
Evas_List *l;

printf("The list:\n");
for (l = list; l; l = evas_list_next(l))
  {
    printf("%p\n", evas_list_data(l));
  }
```

5.36 Linked List Ordering Functions

Functions that change the ordering of data in a linked list.

Functions

- EAPI Evas_List * evas_list_reverse (Evas_List *list)

  Reverse all the elements in the list.

- EAPI Evas_List * evas_list_sort (Evas_List *list, int size, int(*func)(void *, void *))

  Sort a list according to the ordering func will return.

5.36.1 Detailed Description

Functions that change the ordering of data in a linked list.

5.36.2 Function Documentation

5.36.2.1 EAPI Evas_List* evas_list_reverse (Evas_List * *list*)

Reverse all the elements in the list.

Parameters:

list The list to reverse

Returns:

The list after it has been reversed

This takes a list `list`, and reverses the order of all elements in the list, so the last member is now first, and so on.

Example:

```
extern Evas_List *list;

list = evas_list_reverse(list);
```

5.36.2.2 EAPI Evas_List* evas_list_sort (Evas_List * *list*, int *size*, int(*)(void *, void *) *func*)

Sort a list according to the ordering func will return.

Parameters:

list The list handle to sort

size The length of the list to sort

func A function pointer that can handle comparing the list data nodes

Returns:

A new sorted list

This function sorts your list. The data in your nodes can be arbitrary, you just have to be smart enough to know what kind of data is in your lists

In the event of a memory allocation failure, It might segv.

Example:

```
int
sort_cb(void *d1, void *d2)
{
  const char *txt = NULL;
   const char *txt2 = NULL;

   if(!d1) return(1);
   if(!d2) return(-1);

   return(strcmp((const char*)d1, (const char*)d2));
}
extern Evas_List *list;

list = evas_list_sort(list, evas_list_count(list), sort_cb);
if (evas_list_alloc_error())
  {
    fprintf(stderr, "ERROR: Memory is low. List Sorting failed.\n");
    exit(-1);
  }
```

Chapter 6

Evas Data Structure Documentation

6.1 _Evas_Engine_Info Struct Reference

Generic engine information.

Data Fields

- int magic
  Magic number.

6.1.1 Detailed Description

Generic engine information.

Generic info is useless

6.2 _Evas_Event_Key_Down Struct Reference

Key press event.

Data Fields

- char * keyname

  The string name of the key pressed.

- const char * key

  The logical key : (eg shift+1 == exclamation).

- const char * string

  A UTF8 string if this keystroke has produced a visible string to be ADDED.

- const char * compose

  A UTF8 string if this keystroke has modified a string in the middle of being composed - this string replaces the previous one.

6.2.1 Detailed Description

Key press event.

6.3 _Evas_Event_Key_Up Struct Reference

Key release event.

Data Fields

- char * keyname

  The string name of the key released.

- const char * key

  The logical key : (eg shift+1 == exclamation).

- const char * string

  A UTF8 string if this keystroke has produced a visible string to be ADDED.

- const char * compose

  A UTF8 string if this keystroke has modified a string in the middle of being composed - this string replaces the previous one.

6.3.1 Detailed Description

Key release event.

6.4 _Evas_Event_Mouse_Down Struct Reference

Mouse button press event.

Data Fields

- int button

  Mouse button number that went down (1 - 32).

6.4.1 Detailed Description

Mouse button press event.

6.5 _Evas_Event_Mouse_In Struct Reference

Mouse enter event.

Data Fields

- int buttons

  Button pressed mask, Bits set to 1 are buttons currently pressed (bit 0 = mouse button 1, bit 1 = mouse button 2 etc.

6.5.1 Detailed Description

Mouse enter event.

6.5.2 Field Documentation

6.5.2.1 int _Evas_Event_Mouse_In::buttons

Button pressed mask, Bits set to 1 are buttons currently pressed (bit 0 = mouse button 1, bit 1 = mouse button 2 etc.

)

6.6 _Evas_Event_Mouse_Move Struct Reference

Mouse button down event.

Data Fields

- int buttons

  Button pressed mask, Bits set to 1 are buttons currently pressed (bit 0 = mouse button 1, bit 1 = mouse button 2 etc.

6.6.1 Detailed Description

Mouse button down event.

6.6.2 Field Documentation

6.6.2.1 int _Evas_Event_Mouse_Move::buttons

Button pressed mask, Bits set to 1 are buttons currently pressed (bit 0 = mouse button 1, bit 1 = mouse button 2 etc.

)

6.7 _Evas_Event_Mouse_Out Struct Reference

Mouse leave event.

Data Fields

- int buttons

  Button pressed mask, Bits set to 1 are buttons currently pressed (bit 0 = mouse button 1, bit 1 = mouse button 2 etc.

6.7.1 Detailed Description

Mouse leave event.

6.7.2 Field Documentation

6.7.2.1 int _Evas_Event_Mouse_Out::buttons

Button pressed mask, Bits set to 1 are buttons currently pressed (bit 0 = mouse button 1, bit 1 = mouse button 2 etc.

)

6.8 _Evas_Event_Mouse_Up Struct Reference

Mouse butotn relase event.

Data Fields

- int button

  Mouse button number that was raised (1 - 32).

6.8.1 Detailed Description

Mouse butotn relase event.

6.9 _Evas_Event_Mouse_Wheel Struct Reference

Wheel event.

6.9.1 Detailed Description

Wheel event.

6.10 _Evas_List Struct Reference

A linked list node.

Data Fields

- void * data
  Pointer to list element payload.
- Evas_List * next
  Next member in the list.
- Evas_List * prev
  Previous member in the list.
- void * accounting
  Private list accounting info - don't touch.

6.10.1 Detailed Description

A linked list node.

6.11 _Evas_Rectangle Struct Reference

A rectangle.

Data Fields

- int x
  top-left x co-ordinate of rectangle
- int y
  top-left y co-ordinate of rectangle
- int w
  width of rectangle
- int h
  height of rectangle

6.11.1 Detailed Description

A rectangle.

6.12 _Evas_Smart_Class Struct Reference

a smart object class

Data Fields

- int version

  the string name of the class

6.12.1 Detailed Description

a smart object class

Chapter 7

Evas File Documentation

7.1 Evas.h File Reference

These routines are used for Evas library interaction.

Data Structures

- struct _Evas_List
  A linked list node.
- struct _Evas_Rectangle
  A rectangle.
- struct _Evas_Smart_Class
  a smart object class
- struct _Evas_Engine_Info
  Generic engine information.
- struct _Evas_Event_Mouse_Down
  Mouse button press event.
- struct _Evas_Event_Mouse_Up
  Mouse butotn relase event.
- struct _Evas_Event_Mouse_In
  Mouse enter event.
- struct _Evas_Event_Mouse_Out
  Mouse leave event.
- struct _Evas_Event_Mouse_Move

Mouse button down event.

- struct _Evas_Event_Mouse_Wheel
  Wheel event.
- struct _Evas_Event_Key_Down
  Key press event.
- struct _Evas_Event_Key_Up
  Key release event.

Defines

- #define EVAS_LOAD_ERROR_NONE 0
  No error on load.
- #define EVAS_LOAD_ERROR_GENERIC 1
  A non-specific error occured.
- #define EVAS_LOAD_ERROR_DOES_NOT_EXIST 2
  File (or file path) does not exist.
- #define EVAS_LOAD_ERROR_PERMISSION_DENIED 3
  Permission deinied to an existing file (or path).
- #define EVAS_LOAD_ERROR_RESOURCE_ALLOCATION_FAILED 4
  Allocation of resources failure prevented load.
- #define EVAS_LOAD_ERROR_CORRUPT_FILE 5
  File corrupt (but was detected as a known format).
- #define EVAS_LOAD_ERROR_UNKNOWN_FORMAT 6
  File is not a known format.
- #define EVAS_ALLOC_ERROR_NONE 0
  No allocation error.
- #define EVAS_ALLOC_ERROR_FATAL 1
  Allocation failed despite attempts to free up memory.
- #define EVAS_ALLOC_ERROR_RECOVERED 2
  Allocation succeeded, but extra memory had to be found by freeing up speculative resources.
- #define EVAS_PIXEL_FORMAT_NONE 0
  No pixel format.
- #define EVAS_PIXEL_FORMAT_ARGB32 1
  ARGB 32bit pixel format with A in the high byte per 32bit pixel word.

- #define EVAS_PIXEL_FORMAT_YUV420P_601 2

  YUV 420 Planar format with CCIR 601 color encoding wuth contiguous planes in the order Y, U and V.

- #define EVAS_COLOR_SPACE_ARGB 0

  ARGB color space.

- #define EVAS_COLOR_SPACE_AHSV 1

  AHSV color space.

- #define EVAS_TEXTURE_REFLECT 0

  Gradient and image fill tiling mode - tiling reflects.

- #define EVAS_TEXTURE_REPEAT 1

  tiling repeats

- #define EVAS_TEXTURE_RESTRICT 2

  tiling clamps - range offset ignored

- #define EVAS_TEXTURE_RESTRICT_REFLECT 3

  tiling clamps and any range offset reflects

- #define EVAS_TEXTURE_RESTRICT_REPEAT 4

  tiling clamps and any range offset repeats

- #define EVAS_TEXTURE_PAD 5

  tiling extends with end values

Typedefs

- typedef enum _Evas_Callback_Type Evas_Callback_Type

  The type of event to trigger the callback.

- typedef enum _Evas_Button_Flags Evas_Button_Flags

  Flags for Mouse Button events.

- typedef enum _Evas_Font_Hinting_Flags Evas_Font_Hinting_Flags

  Flags for Font Hinting.

- typedef enum _Evas_Colorspace Evas_Colorspace

  Colorspaces for pixel data supported by Evas.

- typedef _Evas_List Evas_List

  A generic linked list node handle.

- typedef _Evas_Rectangle Evas_Rectangle

  A generic rectangle handle.

- typedef _Evas_Smart_Class Evas_Smart_Class

A smart object base class.

- typedef _Evas_Hash Evas_Hash
  A Hash table handle.
- typedef _Evas Evas
  An Evas canvas handle.
- typedef _Evas_Object Evas_Object
  An Evas Object handle.
- typedef void Evas_Performance
  An Evas Performance handle.
- typedef _Evas_Modifier Evas_Modifier
  An Evas Modifier.
- typedef _Evas_Lock Evas_Lock
  An Evas Lock.
- typedef _Evas_Smart Evas_Smart
  An Evas Smart Object handle.
- typedef _Evas_Native_Surface Evas_Native_Surface
  A generic datatype for engine specific native surface information.
- typedef unsigned long long Evas_Modifier_Mask
  An Evas modifier mask type.
- typedef _Evas_Pixel_Import_Source Evas_Pixel_Import_Source
  A source description of pixels for importing pixels.
- typedef _Evas_Engine_Info Evas_Engine_Info
  A generic Evas Engine information structure.
- typedef _Evas_Event_Mouse_Down Evas_Event_Mouse_Down
  Event structure for EVAS_CALLBACK_MOUSE_DOWN event callbacks.
- typedef _Evas_Event_Mouse_Up Evas_Event_Mouse_Up
  Event structure for EVAS_CALLBACK_MOUSE_UP event callbacks.
- typedef _Evas_Event_Mouse_In Evas_Event_Mouse_In
  Event structure for EVAS_CALLBACK_MOUSE_IN event callbacks.
- typedef _Evas_Event_Mouse_Out Evas_Event_Mouse_Out
  Event structure for EVAS_CALLBACK_MOUSE_OUT event callbacks.
- typedef _Evas_Event_Mouse_Move Evas_Event_Mouse_Move
  Event structure for EVAS_CALLBACK_MOUSE_MOVE event callbacks.

- typedef _Evas_Event_Mouse_Wheel Evas_Event_Mouse_Wheel

  Event structure for EVAS_CALLBACK_MOUSE_WHEEL event callbacks.

- typedef _Evas_Event_Key_Down Evas_Event_Key_Down

  Event structure for EVAS_CALLBACK_KEY_DOWN event callbacks.

- typedef _Evas_Event_Key_Up Evas_Event_Key_Up

  Event structure for EVAS_CALLBACK_KEY_UP event callbacks.

Enumerations

- enum _Evas_Callback_Type {
  EVAS_CALLBACK_MOUSE_IN,
  EVAS_CALLBACK_MOUSE_OUT,
  EVAS_CALLBACK_MOUSE_DOWN,
  EVAS_CALLBACK_MOUSE_UP,
  EVAS_CALLBACK_MOUSE_MOVE,
  EVAS_CALLBACK_MOUSE_WHEEL,
  EVAS_CALLBACK_FREE,
  EVAS_CALLBACK_KEY_DOWN,
  EVAS_CALLBACK_KEY_UP,
  EVAS_CALLBACK_FOCUS_IN,
  EVAS_CALLBACK_FOCUS_OUT,
  EVAS_CALLBACK_SHOW,
  EVAS_CALLBACK_HIDE,
  EVAS_CALLBACK_MOVE,
  EVAS_CALLBACK_RESIZE,
  EVAS_CALLBACK_RESTACK }
- enum _Evas_Button_Flags {
  EVAS_BUTTON_NONE = 0,
  EVAS_BUTTON_DOUBLE_CLICK = (1 << 0),
  EVAS_BUTTON_TRIPLE_CLICK = (1 << 1) }
- enum _Evas_Font_Hinting_Flags {
  EVAS_FONT_HINTING_NONE,
  EVAS_FONT_HINTING_AUTO,
  EVAS_FONT_HINTING_BYTECODE }
- enum _Evas_Colorspace {
  EVAS_COLORSPACE_ARGB8888,
  EVAS_COLORSPACE_YCBCR422P601_PL,
  EVAS_COLORSPACE_YCBCR422P709_PL }

- enum _Evas_Render_Op {
  EVAS_RENDER_BLEND = 0,
  EVAS_RENDER_BLEND_REL = 1,
  EVAS_RENDER_COPY = 2,
  EVAS_RENDER_COPY_REL = 3,
  EVAS_RENDER_ADD = 4,
  EVAS_RENDER_ADD_REL = 5,
  EVAS_RENDER_SUB = 6,
  EVAS_RENDER_SUB_REL = 7,
  EVAS_RENDER_TINT = 8,
  EVAS_RENDER_TINT_REL = 9,
  EVAS_RENDER_MASK = 10,
  EVAS_RENDER_MUL = 11 }

Functions

- EAPI Evas_List * evas_list_append (Evas_List *list, const void *data)
  Appends the given data to the given linked list.
- EAPI Evas_List * evas_list_prepend (Evas_List *list, const void *data)
  Prepends the given data to the given linked list.
- EAPI Evas_List * evas_list_append_relative (Evas_List *list, const void *data, const void *relative)
  Inserts the given data into the given linked list after the specified data.
- EAPI Evas_List * evas_list_prepend_relative (Evas_List *list, const void *data, const void *relative)
  Prepend a data pointer to a linked list before the memeber specified.
- EAPI Evas_List * evas_list_remove (Evas_List *list, const void *data)
  Removes the first instance of the specified data from the given list.
- EAPI Evas_List * evas_list_remove_list (Evas_List *list, Evas_List *remove_list)
  Removes the specified data.
- EAPI Evas_List * evas_list_promote_list (Evas_List *list, Evas_List *move_list)
  Moves the specified data to the head of the list.
- EAPI void * evas_list_find (Evas_List *list, const void *data)
  Find a member of a list and return the member.
- EAPI Evas_List * evas_list_find_list (Evas_List *list, const void *data)
  Find a member of a list and return the list node containing that member.
- EAPI Evas_List * evas_list_free (Evas_List *list)
  Free an entire list and all the nodes, ignoring the data contained.

- EAPI Evas_List * evas_list_last (Evas_List *list)
  Get the last list node in the list.
- EAPI Evas_List * evas_list_next (Evas_List *list)
  Get the next list node after the specified list node.
- EAPI Evas_List * evas_list_prev (Evas_List *list)
  Get the previous list node before the specified list node.
- EAPI void * evas_list_data (Evas_List *list)
  Get the list node data member.
- EAPI int evas_list_count (Evas_List *list)
  Get the count of the number of items in a list.
- EAPI void * evas_list_nth (Evas_List *list, int n)
  Get the nth member's data pointer in a list.
- EAPI Evas_List * evas_list_nth_list (Evas_List *list, int n)
  Get the nth member's list node in a list.
- EAPI Evas_List * evas_list_reverse (Evas_List *list)
  Reverse all the elements in the list.
- EAPI Evas_List * evas_list_sort (Evas_List *list, int size, int(*func)(void *, void *))
  Sort a list according to the ordering func will return.
- EAPI int evas_list_alloc_error (void)
  Return the memory allocation failure flag after any operation needin allocation.
- EAPI Evas_Hash * evas_hash_add (Evas_Hash *hash, const char *key, const void *data)
  Adds an entry to the given hash table.
- EAPI Evas_Hash * evas_hash_direct_add (Evas_Hash *hash, const char *key, const void *data)
  Adds an entry to the given hash table and does not duplicate the string key.
- EAPI Evas_Hash * evas_hash_del (Evas_Hash *hash, const char *key, const void *data)
  Removes the entry identified by `key` *or* `data` *from the given hash table.*
- EAPI void * evas_hash_find (Evas_Hash *hash, const char *key)
  Retrieves a specific entry in the given hash table.
- EAPI void * evas_hash_modify (Evas_Hash *hash, const char *key, const void *data)
  Modifies the entry pointer at the specified key and returns the old entry.
- EAPI int evas_hash_size (Evas_Hash *hash)
  Retrieves the number of buckets available in the given hash table.

- EAPI void evas_hash_free (Evas_Hash *hash)

  Free an entire hash table.

- EAPI void evas_hash_foreach (Evas_Hash *hash, Evas_Bool(*func)(Evas_Hash *hash, const char *key, void *data, void *fdata), const void *fdata)

  Call a function on every member stored in the hash table.

- EAPI int evas_hash_alloc_error (void)

  Return memory allocation failure flag after an function requiring allocation.

- EAPI int evas_alloc_error (void)

  Return if any allocation errors have occured during the prior function.

- EAPI Evas * evas_new (void)

  Creates a new empty evas.

- EAPI void evas_free (Evas *e)

  Frees the given evas and any objects created on it.

- EAPI int evas_render_method_lookup (const char *name)

  Look up a numeric ID from a string name of a rendering engine.

- EAPI Evas_List * evas_render_method_list (void)

  List all the rendering engines compiled into the copy of the Evas library.

- EAPI void evas_render_method_list_free (Evas_List *list)

  This function should be called to free a list of engine names.

- EAPI void evas_output_method_set (Evas *e, int render_method)

  Sets the output engine for the given evas.

- EAPI int evas_output_method_get (Evas *e)

  Retrieves the number of the output engine used for the given evas.

- EAPI Evas_Engine_Info * evas_engine_info_get (Evas *e)

  Retrieves the current render engine info struct from the given evas.

- EAPI void evas_engine_info_set (Evas *e, Evas_Engine_Info *info)

  Applies the engine settings for the given evas from the given `Evas_Engine_Info` *structure.*

- EAPI void evas_output_size_set (Evas *e, int w, int h)

  Sets the output size of the render engine of the given evas.

- EAPI void evas_output_size_get (Evas *e, int *w, int *h)

  Retrieve the output size of the render engine of the given evas.

- EAPI void evas_output_viewport_set (Evas *e, Evas_Coord x, Evas_Coord y, Evas_Coord w, Evas_Coord h)

  Sets the output viewport of the given evas in evas units.

- EAPI void evas_output_viewport_get (Evas *e, Evas_Coord *x, Evas_Coord *y, Evas_Coord *w, Evas_Coord *h)

  Get the render engine's output viewport co-ordinates in canvas units.

- EAPI Evas_Coord evas_coord_screen_x_to_world (Evas *e, int x)

  Convert/scale an ouput screen co-ordinate into canvas co-ordinates.

- EAPI Evas_Coord evas_coord_screen_y_to_world (Evas *e, int y)

  Convert/scale an ouput screen co-ordinate into canvas co-ordinates.

- EAPI int evas_coord_world_x_to_screen (Evas *e, Evas_Coord x)

  Convert/scale a canvas co-ordinate into output screen co-ordinates.

- EAPI int evas_coord_world_y_to_screen (Evas *e, Evas_Coord y)

  Convert/scale a canvas co-ordinate into output screen co-ordinates.

- EAPI void evas_pointer_output_xy_get (Evas *e, int *x, int *y)

  This function returns the current known pointer co-ordinates.

- EAPI void evas_pointer_canvas_xy_get (Evas *e, Evas_Coord *x, Evas_Coord *y)

  This function returns the current known pointer co-ordinates.

- EAPI int evas_pointer_button_down_mask_get (Evas *e)

  Returns a bitmask with the mouse buttons currently pressed, set to 1.

- EAPI Evas_Bool evas_pointer_inside_get (Evas *e)

  Returns whether the mouse pointer is logically inside the canvas.

- EAPI void evas_data_attach_set (Evas *e, void *data)

  Attaches a specific pointer to the evas for fetching later.

- EAPI void * evas_data_attach_get (Evas *e)

  Returns the pointer attached by evas_data_attach_set().

- EAPI void evas_damage_rectangle_add (Evas *e, int x, int y, int w, int h)

  To be documented.

- EAPI void evas_obscured_rectangle_add (Evas *e, int x, int y, int w, int h)

  To be documented.

- EAPI void evas_obscured_clear (Evas *e)

  To be documented.

- EAPI Evas_List * evas_render_updates (Evas *e)

  To be documented.

- EAPI void evas_render_updates_free (Evas_List *updates)

  To be documented.

- EAPI void evas_render (Evas *e)

To be documented.

- EAPI void evas_norender (Evas *e)

  To be documented.

- EAPI Evas_Object * evas_object_rectangle_add (Evas *e)

  Adds a rectangle to the given evas.

- EAPI Evas_Object * evas_object_line_add (Evas *e)

  Adds a new evas line object to the given evas.

- EAPI void evas_object_line_xy_set (Evas_Object *obj, Evas_Coord x1, Evas_Coord y1, Evas_Coord x2, Evas_Coord y2)

  Sets the coordinates of the end points of the given evas line object.

- EAPI void evas_object_line_xy_get (Evas_Object *obj, Evas_Coord *x1, Evas_Coord *y1, Evas_Coord *x2, Evas_Coord *y2)

  Retrieves the coordinates of the end points of the given evas line object.

- EAPI Evas_Object * evas_object_gradient_add (Evas *e)

  Adds a gradient object to the given evas.

- EAPI void evas_object_gradient_color_stop_add (Evas_Object *obj, int r, int g, int b, int a, int delta)

  Adds a color stop to the given evas gradient object.

- EAPI void evas_object_gradient_alpha_stop_add (Evas_Object *obj, int a, int delta)

  Adds an alpha stop to the given evas gradient object.

- EAPI void evas_object_gradient_color_data_set (Evas_Object *obj, void *color_data, int len, Evas_Bool has_alpha)

  Sets color data for the given evas gradient object.

- EAPI void evas_object_gradient_alpha_data_set (Evas_Object *obj, void *alpha_data, int len)

  Sets alpha data for the given evas gradient object.

- EAPI void evas_object_gradient_clear (Evas_Object *obj)

  Deletes all stops set for the given evas gradient object or any set data.

- EAPI void evas_object_gradient_type_set (Evas_Object *obj, const char *type, const char *instance_params)

  Sets the geometric type displayed by the given gradient object.

- EAPI void evas_object_gradient_type_get (Evas_Object *obj, char **type, char **instance_params)

  Retrieves the type name and params of the given gradient object.

- EAPI void evas_object_gradient_fill_set (Evas_Object *obj, Evas_Coord x, Evas_Coord y, Evas_Coord w, Evas_Coord h)

Sets the rectangle on the gradient object that the gradient will be drawn to.

- EAPI void evas_object_gradient_fill_get (Evas_Object *obj, Evas_Coord *x, Evas_Coord *y, Evas_Coord *w, Evas_Coord *h)

  Retrieves the dimensions of the rectangle on the gradient object that the gradient will use as its fill rect.

- EAPI void evas_object_gradient_fill_angle_set (Evas_Object *obj, Evas_Angle angle)

  Sets the angle at which the given evas gradient object's fill sits clockwise from vertical.

- EAPI Evas_Angle evas_object_gradient_fill_angle_get (Evas_Object *obj)

  Retrieves the angle at which the given evas gradient object's fill sits clockwise from the vertical.

- EAPI void evas_object_gradient_fill_spread_set (Evas_Object *obj, int tile_mode)

  Sets the tiling mode for the given evas gradient object's fill.

- EAPI int evas_object_gradient_fill_spread_get (Evas_Object *obj)

  Retrieves the spread (tiling mode) for the given gradient object's fill.

- EAPI void evas_object_gradient_angle_set (Evas_Object *obj, Evas_Angle angle)

  Sets the angle at which the given evas gradient sits, relative to whatever intrinsic orientation of the grad type.

- EAPI Evas_Angle evas_object_gradient_angle_get (Evas_Object *obj)

  Retrieves the angle at which the given evas gradient object sits rel to its intrinsic orientation.

- EAPI void evas_object_gradient_direction_set (Evas_Object *obj, int direction)

  Sets the direction of the given evas gradient object's spectrum.

- EAPI int evas_object_gradient_direction_get (Evas_Object *obj)

  Retrieves the evas gradient object's spectrum direction.

- EAPI void evas_object_gradient_offset_set (Evas_Object *obj, float offset)

  Sets the offset of the given evas gradient object's spectrum.

- EAPI float evas_object_gradient_offset_get (Evas_Object *obj)

  Retrieves the spectrum's offset.

- EAPI Evas_Object * evas_object_polygon_add (Evas *e)

  Adds a new evas polygon object to the given evas.

- EAPI void evas_object_polygon_point_add (Evas_Object *obj, Evas_Coord x, Evas_Coord y)

  Adds the given point to the given evas polygon object.

- EAPI void evas_object_polygon_points_clear (Evas_Object *obj)

  Removes all of the points from the given evas polygon object.

- EAPI Evas_Object * evas_object_image_add (Evas *e)

  Creates a new image object on the given evas.

- EAPI void evas_object_image_file_set (Evas_Object *obj, const char *file, const char *key)

  Sets the image displayed by the given image object.

- EAPI void evas_object_image_file_get (Evas_Object *obj, const char **file, const char **key)

  Retrieves the filename and key of the given image object.

- EAPI void evas_object_image_border_set (Evas_Object *obj, int l, int r, int t, int b)

  Sets how much of each border of the given evas image object is not to be scaled.

- EAPI void evas_object_image_border_get (Evas_Object *obj, int *l, int *r, int *t, int *b)

  Retrieves how much of each border of the given evas image is not to be scaled.

- EAPI void evas_object_image_border_center_fill_set (Evas_Object *obj, Evas_Bool fill)

  Sets if the center part of an image (not the border) should be drawn.

- EAPI Evas_Bool evas_object_image_border_center_fill_get (Evas_Object *obj)

  Retrieves If the center of an image object is to be filled or not.

- EAPI void evas_object_image_fill_set (Evas_Object *obj, Evas_Coord x, Evas_Coord y, Evas_Coord w, Evas_Coord h)

  Sets the rectangle on the image object that the image will be drawn to.

- EAPI void evas_object_image_fill_get (Evas_Object *obj, Evas_Coord *x, Evas_Coord *y, Evas_Coord *w, Evas_Coord *h)

  Retrieves the dimensions of the rectangle on the image object that the image will be drawn to.

- EAPI void evas_object_image_size_set (Evas_Object *obj, int w, int h)

  Sets the size of the image to be display by the given image object.

- EAPI void evas_object_image_size_get (Evas_Object *obj, int *w, int *h)

  Retrieves the size of the image displayed by the given image object.

- EAPI int evas_object_image_load_error_get (Evas_Object *obj)

  Retrieves a number representing any error that occurred during the last load for the given image object.

- EAPI void evas_object_image_data_set (Evas_Object *obj, void *data)

  To be documented.

- EAPI void * evas_object_image_data_get (Evas_Object *obj, Evas_Bool for_writing)

  To be documented.

- EAPI void evas_object_image_data_copy_set (Evas_Object *obj, void *data)

  To be documented.

- EAPI void evas_object_image_data_update_add (Evas_Object *obj, int x, int y, int w, int h)

  To be documented.

- EAPI void evas_object_image_alpha_set (Evas_Object *obj, Evas_Bool has_alpha)

  To be documented.

- EAPI Evas_Bool evas_object_image_alpha_get (Evas_Object *obj)

  To be documented.

- EAPI void evas_object_image_smooth_scale_set (Evas_Object *obj, Evas_Bool smooth_scale)

  To be documented.

- EAPI Evas_Bool evas_object_image_smooth_scale_get (Evas_Object *obj)

  To be documented.

- EAPI void evas_object_image_reload (Evas_Object *obj)

  To be documented.

- EAPI Evas_Bool evas_object_image_save (Evas_Object *obj, const char *file, const char *key, const char *flags)

  To be documented.

- EAPI Evas_Bool evas_object_image_pixels_import (Evas_Object *obj, Evas_Pixel_Import_Source *pixels)

  To be documented.

- EAPI void evas_object_image_pixels_get_callback_set (Evas_Object *obj, void(*func)(void *data, Evas_Object *o), void *data)

  To be documented.

- EAPI void evas_object_image_pixels_dirty_set (Evas_Object *obj, Evas_Bool dirty)

  To be documented.

- EAPI Evas_Bool evas_object_image_pixels_dirty_get (Evas_Object *obj)

  To be documented.

- EAPI void evas_object_image_load_dpi_set (Evas_Object *obj, double dpi)

  To be documented.

- EAPI double evas_object_image_load_dpi_get (Evas_Object *obj)

  To be documented.

- EAPI void evas_object_image_load_size_set (Evas_Object *obj, int w, int h)

  To be documented.

- EAPI void evas_object_image_load_size_get (Evas_Object *obj, int *w, int *h)

  To be documented.

- EAPI void evas_object_image_load_scale_down_set (Evas_Object *obj, int scale_down)

  To be documented.

- EAPI int evas_object_image_load_scale_down_get (Evas_Object *obj)

  To be documented.

- EAPI void evas_object_image_colorspace_set (Evas_Object *obj, Evas_Colorspace cspace)

  To be documented.

- EAPI Evas_Colorspace evas_object_image_colorspace_get (Evas_Object *obj)

  To be documented.

- EAPI void evas_object_image_native_surface_set (Evas_Object *obj, Evas_Native_Surface *surf)

  To be documented.

- EAPI Evas_Native_Surface * evas_object_image_native_surface_get (Evas_Object *obj)

  To be documented.

- EAPI void evas_image_cache_flush (Evas *e)

  To be documented.

- EAPI void evas_image_cache_reload (Evas *e)

  To be documented.

- EAPI void evas_image_cache_set (Evas *e, int size)

  To be documented.

- EAPI int evas_image_cache_get (Evas *e)

  To be documented.

- EAPI Evas_Object * evas_object_text_add (Evas *e)

  To be documented.

- EAPI void evas_object_text_font_source_set (Evas_Object *obj, const char *font)

  To be documented.

- EAPI const char * evas_object_text_font_source_get (Evas_Object *obj)

  To be documented.

- EAPI void evas_object_text_font_set (Evas_Object *obj, const char *font, Evas_Font_Size size)

  To be documented.

- EAPI void evas_object_text_font_get (Evas_Object *obj, const char **font, Evas_Font_Size *size)

  To be documented.

- EAPI void evas_object_text_text_set (Evas_Object *obj, const char *text)

  Sets the text to be displayed by the given evas text object.

- EAPI const char * evas_object_text_text_get (Evas_Object *obj)

  Retrieves the text currently being displayed by the given evas text object.

- EAPI Evas_Coord evas_object_text_ascent_get (Evas_Object *obj)

  To be documented.

- EAPI Evas_Coord evas_object_text_descent_get (Evas_Object *obj)

  To be documented.

- EAPI Evas_Coord evas_object_text_max_ascent_get (Evas_Object *obj)

  To be documented.

- EAPI Evas_Coord evas_object_text_max_descent_get (Evas_Object *obj)

  To be documented.

- EAPI Evas_Coord evas_object_text_horiz_advance_get (Evas_Object *obj)

  To be documented.

- EAPI Evas_Coord evas_object_text_vert_advance_get (Evas_Object *obj)

  To be documented.

- EAPI Evas_Coord evas_object_text_inset_get (Evas_Object *obj)

  To be documented.

- EAPI int evas_object_text_char_pos_get (Evas_Object *obj, int pos, Evas_Coord *cx, Evas_Coord *cy, Evas_Coord *cw, Evas_Coord *ch)

  To be documented.

- EAPI int evas_object_text_char_coords_get (Evas_Object *obj, Evas_Coord x, Evas_Coord y, Evas_Coord *cx, Evas_Coord *cy, Evas_Coord *cw, Evas_Coord *ch)

  To be documented.

- EAPI Evas_Text_Style_Type evas_object_text_style_get (Evas_Object *obj)

  To be documented.

- EAPI void evas_object_text_style_set (Evas_Object *obj, Evas_Text_Style_Type type)

  To be documented.

- EAPI void evas_object_text_shadow_color_set (Evas_Object *obj, int r, int g, int b, int a)

  To be documented.

- EAPI void evas_object_text_shadow_color_get (Evas_Object *obj, int *r, int *g, int *b, int *a)

  To be documented.

- EAPI void evas_object_text_glow_color_set (Evas_Object *obj, int r, int g, int b, int a)

  To be documented.

- EAPI void evas_object_text_glow_color_get (Evas_Object *obj, int *r, int *g, int *b, int *a)

  To be documented.

- EAPI void evas_object_text_glow2_color_set (Evas_Object *obj, int r, int g, int b, int a)

  To be documented.

- EAPI void evas_object_text_glow2_color_get (Evas_Object *obj, int *r, int *g, int *b, int *a)

  To be documented.

- EAPI void evas_object_text_outline_color_set (Evas_Object *obj, int r, int g, int b, int a)

  To be documented.

- EAPI void evas_object_text_outline_color_get (Evas_Object *obj, int *r, int *g, int *b, int *a)

  To be documented.

- EAPI void evas_object_text_style_pad_get (Evas_Object *obj, int *l, int *r, int *t, int *b)

  To be documented.

- EAPI int evas_string_char_next_get (const char *str, int pos, int *decoded)

  To be documented.

- EAPI int evas_string_char_prev_get (const char *str, int pos, int *decoded)

  To be documented.

- EAPI void evas_font_path_clear (Evas *e)

  Removes all font paths loaded into memory for the given evas.

- EAPI void evas_font_path_append (Evas *e, const char *path)

  Appends a font path to the list of font paths used by the given evas.

- EAPI void evas_font_path_prepend (Evas *e, const char *path)

  Prepends a font path to the list of font paths used by the given evas.

- EAPI const Evas_List * evas_font_path_list (Evas *e)

  Retrieves the list of font paths used by the given evas.

- EAPI void evas_font_cache_flush (Evas *e)

  To be documented.

- EAPI void evas_font_cache_set (Evas *e, int size)
  To be documented.
- EAPI int evas_font_cache_get (Evas *e)
  To be documented.
- EAPI Evas_List * evas_font_available_list (Evas *e)
  To be documented.
- EAPI void evas_font_available_list_free (Evas *e, Evas_List *available)
  To be documented.
- EAPI Evas_Object * evas_object_textblock_add (Evas *e)
  Adds a textblock to the given evas.
- EAPI void evas_object_del (Evas_Object *obj)
  Deletes the given evas object and frees its memory.
- EAPI const char * evas_object_type_get (Evas_Object *obj)
  Retrieves the name of the type of the given evas object.
- EAPI void evas_object_layer_set (Evas_Object *obj, int l)
  Sets the layer of the evas that the given object will be part of.
- EAPI int evas_object_layer_get (Evas_Object *obj)
  Retrieves the layer of the evas that the given object is part of.
- EAPI void evas_object_raise (Evas_Object *obj)
  Raise `obj` *to the top of its layer.*
- EAPI void evas_object_lower (Evas_Object *obj)
  Lower `obj` *to the bottom of its layer.*
- EAPI void evas_object_stack_above (Evas_Object *obj, Evas_Object *above)
  Stack `obj` *immediately above* `above`.
- EAPI void evas_object_stack_below (Evas_Object *obj, Evas_Object *below)
  Stack `obj` *immediately below* `below`.
- EAPI Evas_Object * evas_object_above_get (Evas_Object *obj)
  Get the evas object above `obj`.
- EAPI Evas_Object * evas_object_below_get (Evas_Object *obj)
  Get the evas object below `obj`.
- EAPI Evas_Object * evas_object_bottom_get (Evas *e)
  Get the lowest evas object on the Evas `e`.
- EAPI Evas_Object * evas_object_top_get (Evas *e)
  Get the highest evas object on the Evas `e`.

- EAPI void evas_object_move (Evas_Object *obj, Evas_Coord x, Evas_Coord y)
  Moves the given evas object to the given location.
- EAPI void evas_object_resize (Evas_Object *obj, Evas_Coord w, Evas_Coord h)
  Changes the size of the given evas object.
- EAPI void evas_object_geometry_get (Evas_Object *obj, Evas_Coord *x, Evas_Coord *y, Evas_Coord *w, Evas_Coord *h)
  Retrieves the position and rectangular size of the given evas object.
- EAPI void evas_object_show (Evas_Object *obj)
  Makes the given evas object visible.
- EAPI void evas_object_hide (Evas_Object *obj)
  Makes the given evas object invisible.
- EAPI Evas_Bool evas_object_visible_get (Evas_Object *obj)
  Retrieves whether or not the given evas object is visible.
- EAPI void evas_object_render_op_set (Evas_Object *obj, Evas_Render_Op op)
  Sets the render_op to be used for rendering the evas object.
- EAPI Evas_Render_Op evas_object_render_op_get (Evas_Object *obj)
  Retrieves the current value of the operation used for rendering the evas object.
- EAPI void evas_object_anti_alias_set (Evas_Object *obj, Evas_Bool antialias)
  Sets whether or not the given evas object is to be drawn anti_aliased.
- EAPI Evas_Bool evas_object_anti_alias_get (Evas_Object *obj)
  Retrieves whether or not the given evas object is to be drawn anti_aliased.
- EAPI void evas_object_color_set (Evas_Object *obj, int r, int g, int b, int a)
  Sets the general colour of the given evas object to the given colour.
- EAPI void evas_object_color_get (Evas_Object *obj, int *r, int *g, int *b, int *a)
  Retrieves the general colour of the given evas object.
- EAPI void evas_object_color_interpolation_set (Evas_Object *obj, int color_space)
  Sets the color_space to be used for linear interpolation of colors.
- EAPI int evas_object_color_interpolation_get (Evas_Object *obj)
  Retrieves the current value of the color space used for linear interpolation.
- EAPI void evas_object_clip_set (Evas_Object *obj, Evas_Object *clip)
  Clip one object to another.
- EAPI Evas_Object * evas_object_clip_get (Evas_Object *obj)
  Get the object clipping this one (if any).

- EAPI void evas_object_clip_unset (Evas_Object *obj)

  Disable clipping for an object.

- EAPI const Evas_List * evas_object_clipees_get (Evas_Object *obj)

  Return a list of objects currently clipped by a specific object.

- EAPI void evas_object_data_set (Evas_Object *obj, const char *key, const void *data)

  Set an attached data pointer to an object with a given string key.

- EAPI void * evas_object_data_get (Evas_Object *obj, const char *key)

  Return an attached data pointer by its given string key.

- EAPI void * evas_object_data_del (Evas_Object *obj, const char *key)

  Delete at attached data pointer from an object.

- EAPI void evas_object_name_set (Evas_Object *obj, const char *name)

  Sets the name of the given evas object to the given name.

- EAPI const char * evas_object_name_get (Evas_Object *obj)

  Retrieves the name of the given evas object.

- EAPI Evas_Object * evas_object_name_find (Evas *e, const char *name)

  Retrieves the object on the given evas with the given name.

- EAPI Evas * evas_object_evas_get (Evas_Object *obj)

  Retrieves the evas that the given evas object is on.

- EAPI Evas_Object * evas_object_top_at_xy_get (Evas *e, Evas_Coord x, Evas_Coord y, Evas_Bool include_pass_events_objects, Evas_Bool include_hidden_objects)

  To be documented.

- EAPI Evas_Object * evas_object_top_at_pointer_get (Evas *e)

  To be documented.

- EAPI Evas_Object * evas_object_top_in_rectangle_get (Evas *e, Evas_Coord x, Evas_Coord y, Evas_Coord w, Evas_Coord h, Evas_Bool include_pass_events_objects, Evas_Bool include_hidden_objects)

  To be documented.

- EAPI Evas_List * evas_objects_at_xy_get (Evas *e, Evas_Coord x, Evas_Coord y, Evas_Bool include_pass_events_objects, Evas_Bool include_hidden_objects)

  To be documented.

- EAPI Evas_List * evas_objects_in_rectangle_get (Evas *e, Evas_Coord x, Evas_Coord y, Evas_Coord w, Evas_Coord h, Evas_Bool include_pass_events_objects, Evas_Bool include_hidden_objects)

  To be documented.

- EAPI void evas_smart_free (Evas_Smart *s)

  Free an Evas_Smart.

- EAPI Evas_Smart * evas_smart_class_new (Evas_Smart_Class *sc)

  Creates an Evas_Smart from an Evas_Smart_Class.

- EAPI Evas_Smart_Class * evas_smart_class_get (Evas_Smart *s)

  Get the Evas_Smart_Class of an Evas_Smart.

- EAPI void * evas_smart_data_get (Evas_Smart *s)

  Get the data pointer set on an Evas_Smart.

- EAPI Evas_Object * evas_object_smart_add (Evas *e, Evas_Smart *s)

  Instantiates a new smart object described by `s`.

- EAPI void evas_object_smart_member_add (Evas_Object *obj, Evas_Object *smart_obj)

  Set an evas object as a member of a smart object.

- EAPI void evas_object_smart_member_del (Evas_Object *obj)

  Removes a member object from a smart object.

- EAPI Evas_Object * evas_object_smart_parent_get (Evas_Object *obj)

  Gets the smart parent of an Evas_Object.

- EAPI Evas_List * evas_object_smart_members_get (Evas_Object *obj)

  Gets the list of the member objects of an Evas_Object.

- EAPI Evas_Smart * evas_object_smart_smart_get (Evas_Object *obj)

  Get the Evas_Smart from which `obj` *was created.*

- EAPI void * evas_object_smart_data_get (Evas_Object *obj)

  Retrieve user data stored on a smart object.

- EAPI void evas_object_smart_data_set (Evas_Object *obj, void *data)

  Store a pointer to user data for a smart object.

- EAPI void evas_object_smart_callback_add (Evas_Object *obj, const char *event, void(*func)(void *data, Evas_Object *obj, void *event_info), const void *data)

  Add a callback for the smart event specified by `event`.

- EAPI void * evas_object_smart_callback_del (Evas_Object *obj, const char *event, void(*func)(void *data, Evas_Object *obj, void *event_info))

  Remove a smart callback.

- EAPI void evas_object_smart_callback_call (Evas_Object *obj, const char *event, void *event_info)

  Call any smart callbacks on `obj` *for* `event`.

- EAPI void evas_event_freeze (Evas *e)

  Freeze all event processing.

- EAPI void evas_event_thaw (Evas *e)

  Thaw a canvas out after freezing.

- EAPI int evas_event_freeze_get (Evas *e)

  Return the freeze count of a given canvas.

- EAPI void evas_event_feed_mouse_down (Evas *e, int b, Evas_Button_Flags flags, unsigned int timestamp, const void *data)

  To be documented.

- EAPI void evas_event_feed_mouse_up (Evas *e, int b, Evas_Button_Flags flags, unsigned int timestamp, const void *data)

  To be documented.

- EAPI void evas_event_feed_mouse_move (Evas *e, int x, int y, unsigned int timestamp, const void *data)

  To be documented.

- EAPI void evas_event_feed_mouse_in (Evas *e, unsigned int timestamp, const void *data)

  To be documented.

- EAPI void evas_event_feed_mouse_out (Evas *e, unsigned int timestamp, const void *data)

  To be documented.

- EAPI void evas_event_feed_mouse_wheel (Evas *e, int direction, int z, unsigned int timestamp, const void *data)

  To be documented.

- EAPI void evas_event_feed_key_down (Evas *e, const char *keyname, const char *key, const char *string, const char *compose, unsigned int timestamp, const void *data)

  To be documented.

- EAPI void evas_event_feed_key_up (Evas *e, const char *keyname, const char *key, const char *string, const char *compose, unsigned int timestamp, const void *data)

  To be documented.

- EAPI void evas_object_focus_set (Evas_Object *obj, Evas_Bool focus)

  To be documented.

- EAPI Evas_Bool evas_object_focus_get (Evas_Object *obj)

  To be documented.

- EAPI Evas_Object * evas_focus_get (Evas *e)

  To be documented.

- EAPI Evas_Modifier * evas_key_modifier_get (Evas *e)

  To be documented.

- EAPI Evas_Lock * evas_key_lock_get (Evas *e)

  To be documented.

- EAPI Evas_Bool evas_key_modifier_is_set (Evas_Modifier *m, const char *keyname)

  To be documented.

- EAPI Evas_Bool evas_key_lock_is_set (Evas_Lock *l, const char *keyname)

  To be documented.

- EAPI void evas_key_modifier_add (Evas *e, const char *keyname)

  To be documented.

- EAPI void evas_key_modifier_del (Evas *e, const char *keyname)

  To be documented.

- EAPI void evas_key_lock_add (Evas *e, const char *keyname)

  To be documented.

- EAPI void evas_key_lock_del (Evas *e, const char *keyname)

  To be documented.

- EAPI void evas_key_modifier_on (Evas *e, const char *keyname)

  To be documented.

- EAPI void evas_key_modifier_off (Evas *e, const char *keyname)

  To be documented.

- EAPI void evas_key_lock_on (Evas *e, const char *keyname)

  To be documented.

- EAPI void evas_key_lock_off (Evas *e, const char *keyname)

  To be documented.

- EAPI Evas_Modifier_Mask evas_key_modifier_mask_get (Evas *e, const char *keyname)

  To be documented.

- EAPI Evas_Bool evas_object_key_grab (Evas_Object *obj, const char *keyname, Evas_Modifier_Mask modifiers, Evas_Modifier_Mask not_modifiers, Evas_Bool exclusive)

  To be documented.

- EAPI void evas_object_key_ungrab (Evas_Object *obj, const char *keyname, Evas_Modifier_Mask modifiers, Evas_Modifier_Mask not_modifiers)

  To be documented.

- EAPI void evas_object_pass_events_set (Evas_Object *obj, Evas_Bool pass)

  Set an object's pass events state.

- EAPI Evas_Bool evas_object_pass_events_get (Evas_Object *obj)

Determine whether an object is set to pass events.

- EAPI void evas_object_repeat_events_set (Evas_Object *obj, Evas_Bool repeat)

  Set an object's repeat events state.

- EAPI Evas_Bool evas_object_repeat_events_get (Evas_Object *obj)

  Determine whether an object is set to repeat events.

- EAPI void evas_object_propagate_events_set (Evas_Object *obj, Evas_Bool prop)

  Set whether events on a smart member object should propagate to its parent.

- EAPI Evas_Bool evas_object_propagate_events_get (Evas_Object *obj)

  Determine whether an object is set to propagate events.

- EAPI void evas_object_event_callback_add (Evas_Object *obj, Evas_Callback_Type type, void(*func)(void *data, Evas *e, Evas_Object *obj, void *event_info), const void *data)

  Add a callback function to an object.

- EAPI void * evas_object_event_callback_del (Evas_Object *obj, Evas_Callback_Type type, void(*func)(void *data, Evas *e, Evas_Object *obj, void *event_info))

  Delete a callback function from an object.

- EAPI void evas_object_intercept_show_callback_add (Evas_Object *obj, void(*func)(void *data, Evas_Object *obj), const void *data)

  To be documented.

- EAPI void * evas_object_intercept_show_callback_del (Evas_Object *obj, void(*func)(void *data, Evas_Object *obj))

  To be documented.

- EAPI void evas_object_intercept_hide_callback_add (Evas_Object *obj, void(*func)(void *data, Evas_Object *obj), const void *data)

  To be documented.

- EAPI void * evas_object_intercept_hide_callback_del (Evas_Object *obj, void(*func)(void *data, Evas_Object *obj))

  To be documented.

- EAPI void evas_object_intercept_move_callback_add (Evas_Object *obj, void(*func)(void *data, Evas_Object *obj, Evas_Coord x, Evas_Coord y), const void *data)

  To be documented.

- EAPI void * evas_object_intercept_move_callback_del (Evas_Object *obj, void(*func)(void *data, Evas_Object *obj, Evas_Coord x, Evas_Coord y))

  To be documented.

- EAPI void evas_object_intercept_resize_callback_add (Evas_Object *obj, void(*func)(void *data, Evas_Object *obj, Evas_Coord w, Evas_Coord h), const void *data)

To be documented.

- EAPI void * evas_object_intercept_resize_callback_del (Evas_Object *obj, void(*func)(void *data, Evas_Object *obj, Evas_Coord w, Evas_Coord h))

  To be documented.

- EAPI void evas_object_intercept_raise_callback_add (Evas_Object *obj, void(*func)(void *data, Evas_Object *obj), const void *data)

  To be documented.

- EAPI void * evas_object_intercept_raise_callback_del (Evas_Object *obj, void(*func)(void *data, Evas_Object *obj))

  To be documented.

- EAPI void evas_object_intercept_lower_callback_add (Evas_Object *obj, void(*func)(void *data, Evas_Object *obj), const void *data)

  To be documented.

- EAPI void * evas_object_intercept_lower_callback_del (Evas_Object *obj, void(*func)(void *data, Evas_Object *obj))

  To be documented.

- EAPI void evas_object_intercept_stack_above_callback_add (Evas_Object *obj, void(*func)(void *data, Evas_Object *obj, Evas_Object *above), const void *data)

  To be documented.

- EAPI void * evas_object_intercept_stack_above_callback_del (Evas_Object *obj, void(*func)(void *data, Evas_Object *obj, Evas_Object *above))

  To be documented.

- EAPI void evas_object_intercept_stack_below_callback_add (Evas_Object *obj, void(*func)(void *data, Evas_Object *obj, Evas_Object *below), const void *data)

  To be documented.

- EAPI void * evas_object_intercept_stack_below_callback_del (Evas_Object *obj, void(*func)(void *data, Evas_Object *obj, Evas_Object *below))

  To be documented.

- EAPI void evas_object_intercept_layer_set_callback_add (Evas_Object *obj, void(*func)(void *data, Evas_Object *obj, int l), const void *data)

  To be documented.

- EAPI void * evas_object_intercept_layer_set_callback_del (Evas_Object *obj, void(*func)(void *data, Evas_Object *obj, int l))

  To be documented.

- EAPI void evas_color_hsv_to_rgb (float h, float s, float v, int *r, int *g, int *b)

  FIXME: .

- EAPI void evas_color_rgb_to_hsv (int r, int g, int b, float *h, float *s, float *v)

  FIXME: .

- EAPI void evas_color_argb_premul (int a, int *r, int *g, int *b)
  FIXME: .
- EAPI void evas_color_argb_unpremul (int a, int *r, int *g, int *b)
  FIXME: .
- EAPI void evas_data_argb_premul (unsigned int *data, unsigned int len)
  FIXME: .
- EAPI void evas_data_argb_unpremul (unsigned int *data, unsigned int len)
  FIXME: .

7.1.1 Detailed Description

These routines are used for Evas library interaction.

7.1.2 Enumeration Type Documentation

7.1.2.1 enum _Evas_Button_Flags

Enumerator:

EVAS_BUTTON_NONE No extra mouse button data.

EVAS_BUTTON_DOUBLE_CLICK This mouse button press was the 2nd press of a double click.

EVAS_BUTTON_TRIPLE_CLICK This mouse button press was the 3rd press of a triple click.

7.1.2.2 enum _Evas_Callback_Type

Enumerator:

EVAS_CALLBACK_MOUSE_IN Mouse In Event.

EVAS_CALLBACK_MOUSE_OUT Mouse Out Event.

EVAS_CALLBACK_MOUSE_DOWN Mouse Button Down Event.

EVAS_CALLBACK_MOUSE_UP Mouse Button Up Event.

EVAS_CALLBACK_MOUSE_MOVE Mouse Move Event.

EVAS_CALLBACK_MOUSE_WHEEL Mouse Wheel Event.

EVAS_CALLBACK_FREE Object Being Freed.

EVAS_CALLBACK_KEY_DOWN Key Press Event.

EVAS_CALLBACK_KEY_UP Key Release Event.

EVAS_CALLBACK_FOCUS_IN Focus In Event.

EVAS_CALLBACK_FOCUS_OUT Focus Out Event.

EVAS_CALLBACK_SHOW Show Event.

EVAS_CALLBACK_HIDE Hide Event.

EVAS_CALLBACK_MOVE Move Event.

EVAS_CALLBACK_RESIZE Resize Event.

EVAS_CALLBACK_RESTACK Restack Event.

7.1.2.3 enum _Evas_Colorspace

Enumerator:

EVAS_COLORSPACE_ARGB8888 ARGB 32 bits per pixel, high-byte is Alpha, accessed 1 32bit word at a time.

EVAS_COLORSPACE_YCBCR422P601_PL YCbCr 4:2:2 Planar, ITU.BT-601 specifications.

The data poitned to is just an array of row pointer, pointing to the Y rows, then the Cb, then Cr rows

EVAS_COLORSPACE_YCBCR422P709_PL YCbCr 4:2:2 Planar, ITU.BT-709 specifications.

The data poitned to is just an array of row pointer, pointing to the Y rows, then the Cb, then Cr rows

7.1.2.4 enum _Evas_Font_Hinting_Flags

Enumerator:

EVAS_FONT_HINTING_NONE No font hinting.

EVAS_FONT_HINTING_AUTO Automatic font hinting.

EVAS_FONT_HINTING_BYTECODE Bytecode font hinting.

7.1.2.5 enum _Evas_Render_Op

Enumerator:

EVAS_RENDER_BLEND default op: d = d∗(1-sa) + s

EVAS_RENDER_BLEND_REL d = d∗(1 - sa) + s∗da

EVAS_RENDER_COPY d = s

EVAS_RENDER_COPY_REL d = s∗da

EVAS_RENDER_ADD d = d + s

EVAS_RENDER_ADD_REL d = d + s∗da

EVAS_RENDER_SUB d = d - s
EVAS_RENDER_SUB_REL d = d - s∗da
EVAS_RENDER_TINT d = d∗s + d∗(1 - sa) + s∗(1 - da)
EVAS_RENDER_TINT_REL d = d∗(1 - sa + s)
EVAS_RENDER_MASK d = d∗sa
EVAS_RENDER_MUL d = d∗s

7.1.3 Function Documentation

7.1.3.1 EAPI int evas_alloc_error (void)

Return if any allocation errors have occured during the prior function.

Returns:

The allocation error flag

This function will return if any memory allocation errors occured during, and what kind they were. The return value will be one of EVAS_ALLOC_ERROR_NONE, EVAS_ALLOC_ERROR_-FATAL or EVAS_ALLOC_ERROR_RECOVERED with each meaning something different.

EVAS_ALLOC_ERROR_NONE means that no errors occured at all and the function worked as expected.

EVAS_ALLOC_ERROR_FATAL means the function was completely unable to perform its job and will have exited as cleanly as possible. The programmer should consider this as a sign of very low memory and should try and safely recover from the prior functions failure (or try free up memory elsewhere and try again after more memory is freed).

EVAS_ALLOC_ERROR_RECOVERED means that an allocation error occured, but was recovered from by evas finding memory of its own it has allocated and freeing what it sees as not really usefully allocated memory. What is freed may vary. Evas may reduce the resolution of images, free cached images or fonts, trhow out pre-rendered data, reduce the complexity of change lists etc. Evas and the program will function as per normal after this, but this is a sign of low memory, and it is suggested that the program try and identify memory it doesn't need, and free it.

Example:

```
extern Evas_Object *object;
void callback (void *data, Evas *e, Evas_Object *obj, void *event_info);

evas_object_event_callback_add(object, EVAS_CALLBACK_MOUSE_DOWN, callback, NULL);
if (evas_alloc_error() == EVAS_ALLOC_ERROR_FATAL)
  {
    fprintf(stderr, "ERROR: Completely unable to attach callabck. Must\n");
    fprintf(stderr, "       destroy object now as it cannot be used.\n");
    evas_object_del(object);
    object = NULL;
    fprintf(stderr, "WARNING: Memory is really low. Cleaning out RAM.\n");
    my_memory_cleanup();
  }
if (evas_alloc_error() == EVAS_ALLOC_ERROR_RECOVERED)
  {
    fprintf(stderr, "WARNING: Memory is really low. Cleaning out RAM.\n");
    my_memory_cleanup();
  }
```

7.1.3.2 EAPI void evas_color_argb_premul (int *a*, int * *r*, int * *g*, int * *b*)

FIXME: .

..

7.1.3.3 EAPI void evas_color_argb_unpremul (int *a*, int * *r*, int * *g*, int * *b*)

FIXME: .

..

7.1.3.4 EAPI void evas_color_hsv_to_rgb (float *h*, float *s*, float *v*, int * *r*, int * *g*, int * *b*)

FIXME: .

..

7.1.3.5 EAPI void evas_color_rgb_to_hsv (int *r*, int *g*, int *b*, float * *h*, float * *s*, float * *v*)

FIXME: .

..

7.1.3.6 EAPI void evas_damage_rectangle_add (Evas * *e*, int *x*, int *y*, int *w*, int *h*)

To be documented.

FIXME: To be fixed.

7.1.3.7 EAPI void evas_data_argb_premul (unsigned int * *data*, unsigned int *len*)

FIXME: .

..

7.1.3.8 EAPI void evas_data_argb_unpremul (unsigned int * *data*, unsigned int *len*)

FIXME: .

..

7.1.3.9 EAPI void* evas_data_attach_get (Evas * *e*)

Returns the pointer attached by evas_data_attach_set().

Parameters:

e The canvas to attach the pointer to

Returns:

The pointer attached

7.1.3.10 EAPI void evas_data_attach_set (Evas * *e*, void * *data*)

Attaches a specific pointer to the evas for fetching later.

Parameters:

e The canvas to attach the pointer to

data The pointer to attach

7.1.3.11 EAPI void evas_event_feed_key_down (Evas * *e*, const char * *keyname*, const char * *key*, const char * *string*, const char * *compose*, unsigned int *timestamp*, const void * *data*)

To be documented.

FIXME: To be fixed.

7.1.3.12 EAPI void evas_event_feed_key_up (Evas * *e*, const char * *keyname*, const char * *key*, const char * *string*, const char * *compose*, unsigned int *timestamp*, const void * *data*)

To be documented.

FIXME: To be fixed.

7.1.3.13 EAPI void evas_event_feed_mouse_down (Evas * *e*, int *b*, Evas_Button_Flags *flags*, unsigned int *timestamp*, const void * *data*)

To be documented.

FIXME: To be fixed.

7.1.3.14 EAPI void evas_event_feed_mouse_in (Evas * *e*, unsigned int *timestamp*, const void * *data*)

To be documented.

FIXME: To be fixed.

7.1.3.15 EAPI void evas_event_feed_mouse_move (Evas * e, int x, int y, unsigned int timestamp, const void * data)

To be documented.

FIXME: To be fixed.

7.1.3.16 EAPI void evas_event_feed_mouse_out (Evas * e, unsigned int timestamp, const void * data)

To be documented.

FIXME: To be fixed.

7.1.3.17 EAPI void evas_event_feed_mouse_up (Evas * e, int b, Evas_Button_Flags flags, unsigned int timestamp, const void * data)

To be documented.

FIXME: To be fixed.

7.1.3.18 EAPI void evas_event_feed_mouse_wheel (Evas * e, int direction, int z, unsigned int timestamp, const void * data)

To be documented.

FIXME: To be fixed.

7.1.3.19 EAPI Evas_Object* evas_focus_get (Evas * e)

To be documented.

FIXME: To be fixed.

7.1.3.20 EAPI Evas_List* evas_font_available_list (Evas * e)

To be documented.

FIXME: To be fixed.

7.1.3.21 EAPI void evas_font_available_list_free (Evas * e, Evas_List * available)

To be documented.

FIXME: To be fixed.

7.1.3.22 EAPI void evas_font_cache_flush (Evas * e)

To be documented.

FIXME: To be fixed.

7.1.3.23 EAPI int evas_font_cache_get (Evas * e)

To be documented.

FIXME: To be fixed.

7.1.3.24 EAPI void evas_font_cache_set (Evas * e, int size)

To be documented.

FIXME: To be fixed.

7.1.3.25 EAPI void evas_image_cache_flush (Evas * e)

To be documented.

FIXME: To be fixed.

7.1.3.26 EAPI int evas_image_cache_get (Evas * e)

To be documented.

FIXME: To be fixed.

7.1.3.27 EAPI void evas_image_cache_reload (Evas * e)

To be documented.

FIXME: To be fixed.

7.1.3.28 EAPI void evas_image_cache_set (Evas * e, int size)

To be documented.

FIXME: To be fixed.

7.1.3.29 EAPI void evas_key_lock_add (Evas * e, const char * keyname)

To be documented.

FIXME: To be fixed.

7.1.3.30 EAPI void evas_key_lock_del (Evas * e, const char * keyname)

To be documented.

FIXME: To be fixed.

7.1.3.31 EAPI Evas_Lock* evas_key_lock_get (Evas * e)

To be documented.

FIXME: To be fixed.

7.1.3.32 EAPI Evas_Bool evas_key_lock_is_set (Evas_Lock * *l*, const char * *keyname*)

To be documented.

FIXME: To be fixed.

7.1.3.33 EAPI void evas_key_lock_off (Evas * *e*, const char * *keyname*)

To be documented.

FIXME: To be fixed.

7.1.3.34 EAPI void evas_key_lock_on (Evas * *e*, const char * *keyname*)

To be documented.

FIXME: To be fixed.

7.1.3.35 EAPI void evas_key_modifier_add (Evas * *e*, const char * *keyname*)

To be documented.

FIXME: To be fixed.

7.1.3.36 EAPI void evas_key_modifier_del (Evas * *e*, const char * *keyname*)

To be documented.

FIXME: To be fixed.

7.1.3.37 EAPI Evas_Modifier* evas_key_modifier_get (Evas * *e*)

To be documented.

FIXME: To be fixed.

7.1.3.38 EAPI Evas_Bool evas_key_modifier_is_set (Evas_Modifier * *m*, const char * *keyname*)

To be documented.

FIXME: To be fixed.

7.1.3.39 EAPI Evas_Modifier_Mask evas_key_modifier_mask_get (Evas * *e*, const char * *keyname*)

To be documented.

FIXME: To be fixed.

7.1.3.40 EAPI void evas_key_modifier_off (Evas * e, const char * keyname)

To be documented.

FIXME: To be fixed.

7.1.3.41 EAPI void evas_key_modifier_on (Evas * e, const char * keyname)

To be documented.

FIXME: To be fixed.

7.1.3.42 EAPI Evas_List* evas_list_promote_list (Evas_List * list, Evas_List * move_list)

Moves the specified data to the head of the list.

Move a specified member to the head of the list

Parameters:

list The list handle to move `inside`

move_list The list node which is to be moved

Returns:

A new list handle to replace the old one

Calling this function takes the list node `move_list` and moves it to the front of the `list`.

Example:

```
extern Evas_List *list;
Evas_List *l;
extern void *my_data;

for (l = list; l; l= l->next)
  {
    if (l->data == my_data)
      {
        list = evas_list_promote_list(list, l);
        break;
      }
  }
```

7.1.3.43 EAPI void evas_norender (Evas * e)

To be documented.

FIXME: To be fixed.

7.1.3.44 EAPI Evas_Object* evas_object_above_get (Evas_Object * obj)

Get the evas object above `obj`.

Parameters:

obj an Evas_Object

Returns:

the Evas_Object directly above

7.1.3.45 EAPI Evas_Object* evas_object_below_get (Evas_Object * *obj*)

Get the evas object below `obj`.

Parameters:

obj an Evas_Object

Returns:

the Evas_Object directly below

7.1.3.46 EAPI Evas_Object* evas_object_bottom_get (Evas * *e*)

Get the lowest evas object on the Evas `e`.

Parameters:

e an Evas

Returns:

the lowest object

7.1.3.47 EAPI Evas_Bool evas_object_focus_get (Evas_Object * *obj*)

To be documented.

FIXME: To be fixed.

7.1.3.48 EAPI void evas_object_focus_set (Evas_Object * *obj*, Evas_Bool *focus*)

To be documented.

FIXME: To be fixed.

7.1.3.49 EAPI Evas_Bool evas_object_image_alpha_get (Evas_Object * *obj*)

To be documented.

FIXME: To be fixed.

7.1.3.50 EAPI void evas_object_image_alpha_set (Evas_Object * *obj*, Evas_Bool *has_alpha*)

To be documented.

FIXME: To be fixed.

7.1.3.51 EAPI Evas_Colorspace evas_object_image_colorspace_get (Evas_Object * *obj*)

To be documented.

FIXME: To be fixed.

7.1.3.52 EAPI void evas_object_image_colorspace_set (Evas_Object * *obj*, Evas_Colorspace *cspace*)

To be documented.

FIXME: To be fixed.

7.1.3.53 EAPI void evas_object_image_data_copy_set (Evas_Object * *obj*, void * *data*)

To be documented.

FIXME: To be fixed.

7.1.3.54 EAPI void* evas_object_image_data_get (Evas_Object * *obj*, Evas_Bool *for_writing*)

To be documented.

FIXME: To be fixed.

7.1.3.55 EAPI void evas_object_image_data_set (Evas_Object * *obj*, void * *data*)

To be documented.

FIXME: To be fixed.

7.1.3.56 EAPI void evas_object_image_data_update_add (Evas_Object * *obj*, int *x*, int *y*, int *w*, int *h*)

To be documented.

FIXME: To be fixed.

7.1.3.57 EAPI double evas_object_image_load_dpi_get (Evas_Object * *obj*)

To be documented.

FIXME: To be fixed.

7.1.3.58 EAPI void evas_object_image_load_dpi_set (Evas_Object * *obj*, double *dpi*)

To be documented.

FIXME: To be fixed.

7.1.3.59 EAPI int evas_object_image_load_scale_down_get (Evas_Object * *obj*)

To be documented.

FIXME: To be fixed.

7.1.3.60 EAPI void evas_object_image_load_scale_down_set (Evas_Object * *obj*, int *scale_down*)

To be documented.

FIXME: To be fixed.

7.1.3.61 EAPI void evas_object_image_load_size_get (Evas_Object * *obj*, int * *w*, int * *h*)

To be documented.

FIXME: To be fixed.

7.1.3.62 EAPI void evas_object_image_load_size_set (Evas_Object * *obj*, int *w*, int *h*)

To be documented.

FIXME: To be fixed.

7.1.3.63 EAPI Evas_Native_Surface* evas_object_image_native_surface_get (Evas_Object * *obj*)

To be documented.

FIXME: To be fixed.

7.1.3.64 EAPI void evas_object_image_native_surface_set (Evas_Object * *obj*, Evas_Native_Surface * *surf*)

To be documented.

FIXME: To be fixed.

7.1.3.65 EAPI Evas_Bool evas_object_image_pixels_dirty_get (Evas_Object * *obj*)

To be documented.

FIXME: To be fixed.

7.1.3.66 EAPI void evas_object_image_pixels_dirty_set (Evas_Object * *obj*, Evas_Bool *dirty*)

To be documented.

FIXME: To be fixed.

7.1.3.67 EAPI void evas_object_image_pixels_get_callback_set (Evas_Object * *obj*, void(*)(void *data, Evas_Object *o) *func*, void * *data*)

To be documented.

FIXME: To be fixed.

7.1.3.68 EAPI Evas_Bool evas_object_image_pixels_import (Evas_Object * *obj*, Evas_Pixel_Import_Source * *pixels*)

To be documented.

FIXME: To be fixed.

7.1.3.69 EAPI void evas_object_image_reload (Evas_Object * *obj*)

To be documented.

FIXME: To be fixed.

7.1.3.70 EAPI Evas_Bool evas_object_image_save (Evas_Object * *obj*, const char * *file*, const char * *key*, const char * *flags*)

To be documented.

FIXME: To be fixed.

7.1.3.71 EAPI Evas_Bool evas_object_image_smooth_scale_get (Evas_Object * *obj*)

To be documented.

FIXME: To be fixed.

7.1.3.72 EAPI void evas_object_image_smooth_scale_set (Evas_Object * *obj*, Evas_Bool *smooth_scale*)

To be documented.

FIXME: To be fixed.

7.1.3.73 EAPI void evas_object_intercept_hide_callback_add (Evas_Object * *obj*, void(*)(void *data, Evas_Object *obj) *func*, const void * *data*)

To be documented.

FIXME: To be fixed.

7.1.3.74 EAPI void* evas_object_intercept_hide_callback_del (Evas_Object * *obj*, void(*)(void *data, Evas_Object *obj) *func*)

To be documented.

FIXME: To be fixed.

7.1.3.75 EAPI void evas_object_intercept_layer_set_callback_add (Evas_Object * *obj*, void(*)(void *data, Evas_Object *obj, int l) *func*, const void * *data*)

To be documented.

FIXME: To be fixed.

7.1.3.76 EAPI void* evas_object_intercept_layer_set_callback_del (Evas_Object * *obj*, void(*)(void *data, Evas_Object *obj, int l) *func*)

To be documented.

FIXME: To be fixed.

7.1.3.77 EAPI void evas_object_intercept_lower_callback_add (Evas_Object * *obj*, void(*)(void *data, Evas_Object *obj) *func*, const void * *data*)

To be documented.

FIXME: To be fixed.

7.1.3.78 EAPI void* evas_object_intercept_lower_callback_del (Evas_Object * *obj*, void(*)(void *data, Evas_Object *obj) *func*)

To be documented.

FIXME: To be fixed.

7.1.3.79 EAPI void evas_object_intercept_move_callback_add (Evas_Object * *obj*, void(*)(void *data, Evas_Object *obj, Evas_Coord x, Evas_Coord y) *func*, const void * *data*)

To be documented.

FIXME: To be fixed.

7.1.3.80 EAPI void* evas_object_intercept_move_callback_del (Evas_Object * *obj*, void(*)(void *data, Evas_Object *obj, Evas_Coord x, Evas_Coord y) *func*)

To be documented.

FIXME: To be fixed.

7.1.3.81 EAPI void evas_object_intercept_raise_callback_add (Evas_Object * *obj*, void(*)(void *data, Evas_Object *obj) *func*, const void * *data*)

To be documented.

FIXME: To be fixed.

7.1.3.82 EAPI void* evas_object_intercept_raise_callback_del (Evas_Object * *obj*, void(*)(void *data, Evas_Object *obj) *func*)

To be documented.

FIXME: To be fixed.

7.1.3.83 EAPI void evas_object_intercept_resize_callback_add (Evas_Object * *obj*, void(*)(void *data, Evas_Object *obj, Evas_Coord w, Evas_Coord h) *func*, const void * *data*)

To be documented.

FIXME: To be fixed.

7.1.3.84 EAPI void* evas_object_intercept_resize_callback_del (Evas_Object * *obj*, void(*)(void *data, Evas_Object *obj, Evas_Coord w, Evas_Coord h) *func*)

To be documented.

FIXME: To be fixed.

7.1.3.85 EAPI void evas_object_intercept_show_callback_add (Evas_Object * *obj*, void(*)(void *data, Evas_Object *obj) *func*, const void * *data*)

To be documented.

FIXME: To be fixed.

7.1.3.86 EAPI void* evas_object_intercept_show_callback_del (Evas_Object * *obj*, void(*)(void *data, Evas_Object *obj) *func*)

To be documented.

FIXME: To be fixed.

7.1.3.87 EAPI void evas_object_intercept_stack_above_callback_add (Evas_Object * *obj*, void(*)(void *data, Evas_Object *obj, Evas_Object *above) *func*, const void * *data*)

To be documented.

FIXME: To be fixed.

7.1.3.88 EAPI void* evas_object_intercept_stack_above_callback_del (Evas_Object * *obj*, void(*)(void *data, Evas_Object *obj, Evas_Object *above) *func*)

To be documented.

FIXME: To be fixed.

7.1.3.89 EAPI void evas_object_intercept_stack_below_callback_add (Evas_Object * *obj*, void(*)(void *data, Evas_Object *obj, Evas_Object *below) *func*, const void * *data*)

To be documented.

FIXME: To be fixed.

7.1.3.90 EAPI void* evas_object_intercept_stack_below_callback_del (Evas_Object * *obj*, void(*)(void *data, Evas_Object *obj, Evas_Object *below) *func*)

To be documented.

FIXME: To be fixed.

7.1.3.91 EAPI Evas_Bool evas_object_key_grab (Evas_Object * *obj*, const char * *keyname*, Evas_Modifier_Mask *modifiers*, Evas_Modifier_Mask *not_modifiers*, Evas_Bool *exclusive*)

To be documented.

FIXME: To be fixed.

7.1.3.92 EAPI void evas_object_key_ungrab (Evas_Object * *obj*, const char * *keyname*, Evas_Modifier_Mask *modifiers*, Evas_Modifier_Mask *not_modifiers*)

To be documented.

FIXME: To be fixed.

7.1.3.93 EAPI void evas_object_lower (Evas_Object * *obj*)

Lower `obj` to the bottom of its layer.

Parameters:

obj the object to lower

7.1.3.94 EAPI void evas_object_raise (Evas_Object * *obj*)

Raise `obj` to the top of its layer.

Parameters:

obj the object to raise

7.1.3.95 EAPI Evas_Object* evas_object_rectangle_add (Evas * *e*)

Adds a rectangle to the given evas.

Parameters:

e The given evas.

Returns:

The new rectangle object.

Todo

Find a documentation group to put this under.

7.1.3.96 EAPI Evas_List* evas_object_smart_members_get (Evas_Object * *obj*)

Gets the list of the member objects of an Evas_Object.

Parameters:

obj the Evas_Object you want to get the list of member objects

Returns:

Returns the list of the member objects of *obj*. The returned list should be freed with evas_list_free() when you no longer need it

7.1.3.97 EAPI void evas_object_stack_above (Evas_Object * *obj*, Evas_Object * *above*)

Stack `obj` immediately above `above`.

If `obj` is a member of a smart object, then `above` must also be a member of the same smart object.

Similarly, if `obj` is not a member of smart object, `above` may not either.

Parameters:

obj the object to stack

above the object above which to stack

7.1.3.98 EAPI void evas_object_stack_below (Evas_Object * *obj*, Evas_Object * *below*)

Stack `obj` immediately below `below`.

If `obj` is a member of a smart object, then `below` must also be a member of the same smart object.

Similarly, if `obj` is not a member of smart object, `below` may not either.

Parameters:

obj the object to stack

below the object below which to stack

7.1.3.99 EAPI Evas_Object* evas_object_text_add (Evas * *e*)

To be documented.

FIXME: To be fixed.

7.1.3.100 EAPI Evas_Coord evas_object_text_ascent_get (Evas_Object * *obj*)

To be documented.

FIXME: To be fixed.

7.1.3.101 EAPI int evas_object_text_char_coords_get (Evas_Object * *obj*, Evas_Coord *x*, Evas_Coord *y*, Evas_Coord * *cx*, Evas_Coord * *cy*, Evas_Coord * *cw*, Evas_Coord * *ch*)

To be documented.

FIXME: To be fixed.

7.1.3.102 EAPI int evas_object_text_char_pos_get (Evas_Object * *obj*, int *pos*, Evas_Coord * *cx*, Evas_Coord * *cy*, Evas_Coord * *cw*, Evas_Coord * *ch*)

To be documented.

FIXME: To be fixed.

7.1.3.103 EAPI Evas_Coord evas_object_text_descent_get (Evas_Object * *obj*)

To be documented.

FIXME: To be fixed.

7.1.3.104 EAPI void evas_object_text_font_get (Evas_Object * *obj*, const char ** *font*, Evas_Font_Size * *size*)

To be documented.

FIXME: To be fixed.

7.1.3.105 EAPI void evas_object_text_font_set (Evas_Object * *obj*, const char * *font*, Evas_Font_Size *size*)

To be documented.

FIXME: To be fixed.

7.1.3.106 EAPI const char* evas_object_text_font_source_get (Evas_Object * *obj*)

To be documented.

FIXME: To be fixed.

7.1.3.107 EAPI void evas_object_text_font_source_set (Evas_Object * *obj*, const char * *font_source*)

To be documented.

FIXME: To be fixed.

7.1.3.108 EAPI void evas_object_text_glow2_color_get (Evas_Object * *obj*, int * *r*, int * *g*, int * *b*, int * *a*)

To be documented.

FIXME: To be fixed.

7.1.3.109 EAPI void evas_object_text_glow2_color_set (Evas_Object * *obj*, int *r*, int *g*, int *b*, int *a*)

To be documented.

FIXME: To be fixed.

7.1.3.110 EAPI void evas_object_text_glow_color_get (Evas_Object * *obj*, int * *r*, int * *g*, int * *b*, int * *a*)

To be documented.

FIXME: To be fixed.

7.1.3.111 EAPI void evas_object_text_glow_color_set (Evas_Object * obj, int r, int g, int b, int a)

To be documented.

FIXME: To be fixed.

7.1.3.112 EAPI Evas_Coord evas_object_text_horiz_advance_get (Evas_Object * obj)

To be documented.

FIXME: To be fixed.

7.1.3.113 EAPI Evas_Coord evas_object_text_inset_get (Evas_Object * obj)

To be documented.

FIXME: To be fixed.

7.1.3.114 EAPI Evas_Coord evas_object_text_max_ascent_get (Evas_Object * obj)

To be documented.

FIXME: To be fixed.

7.1.3.115 EAPI Evas_Coord evas_object_text_max_descent_get (Evas_Object * obj)

To be documented.

FIXME: To be fixed.

7.1.3.116 EAPI void evas_object_text_outline_color_get (Evas_Object * obj, int * r, int * g, int * b, int * a)

To be documented.

FIXME: To be fixed.

7.1.3.117 EAPI void evas_object_text_outline_color_set (Evas_Object * obj, int r, int g, int b, int a)

To be documented.

FIXME: To be fixed.

7.1.3.118 EAPI void evas_object_text_shadow_color_get (Evas_Object * obj, int * r, int * g, int * b, int * a)

To be documented.

FIXME: To be fixed.

7.1.3.119 EAPI void evas_object_text_shadow_color_set (Evas_Object * *obj*, int *r*, int *g*, int *b*, int *a*)

To be documented.

FIXME: To be fixed.

7.1.3.120 EAPI Evas_Text_Style_Type evas_object_text_style_get (Evas_Object * *obj*)

To be documented.

FIXME: To be fixed.

7.1.3.121 EAPI void evas_object_text_style_pad_get (Evas_Object * *obj*, int * *l*, int * *r*, int * *t*, int * *b*)

To be documented.

FIXME: To be fixed.

7.1.3.122 EAPI void evas_object_text_style_set (Evas_Object * *obj*, Evas_Text_Style_Type *style*)

To be documented.

FIXME: To be fixed.

7.1.3.123 EAPI const char* evas_object_text_text_get (Evas_Object * *obj*)

Retrieves the text currently being displayed by the given evas text object.

Parameters:

obj The given evas text object.

Returns:

The text currently being displayed. Do not free it.

7.1.3.124 EAPI void evas_object_text_text_set (Evas_Object * *obj*, const char * *text*)

Sets the text to be displayed by the given evas text object.

Parameters:

obj Evas text object.

text Text to display.

7.1.3.125 EAPI Evas_Coord evas_object_text_vert_advance_get (Evas_Object * obj)

To be documented.

FIXME: To be fixed.

7.1.3.126 EAPI Evas_Object* evas_object_textblock_add (Evas * e)

Adds a textblock to the given evas.

Parameters:

e The given evas.

Returns:

The new textblock object.

Todo

Find a documentation group to put this under.

7.1.3.127 EAPI Evas_Object* evas_object_top_get (Evas * e)

Get the highest evas object on the Evas `e`.

Parameters:

e an Evas

Returns:

the highest object

7.1.3.128 EAPI void evas_obscured_clear (Evas * e)

To be documented.

FIXME: To be fixed.

7.1.3.129 EAPI void evas_obscured_rectangle_add (Evas * e, int x, int y, int w, int h)

To be documented.

FIXME: To be fixed.

7.1.3.130 EAPI void evas_render (Evas * e)

To be documented.

FIXME: To be fixed.

7.1.3.131 EAPI Evas_List* evas_render_updates (Evas * e)

To be documented.

FIXME: To be fixed.

7.1.3.132 EAPI void evas_render_updates_free (Evas_List * *updates*)

To be documented.

FIXME: To be fixed.

7.1.3.133 EAPI Evas_Smart_Class* evas_smart_class_get (Evas_Smart * *s*)

Get the Evas_Smart_Class of an Evas_Smart.

Parameters:

s the Evas_Smart

Returns:

the Evas_Smart_Class

7.1.3.134 EAPI Evas_Smart* evas_smart_class_new (Evas_Smart_Class * *sc*)

Creates an Evas_Smart from an Evas_Smart_Class.

Parameters:

Evas_Smart_Class the smart class definition

Returns:

an Evas_Smart

7.1.3.135 EAPI void* evas_smart_data_get (Evas_Smart * *s*)

Get the data pointer set on an Evas_Smart.

This data pointer is set either as the final parameter to evas_smart_new or as the data field in the Evas_Smart_Class passed in to evas_smart_class_new

Parameters:

Evas_Smart

7.1.3.136 EAPI void evas_smart_free (Evas_Smart * *s*)

Free an Evas_Smart.

If this smart was created using evas_smart_class_new(), the associated Evas_Smart_Class will not be freed.

Parameters:

s the Evas_Smart to free

7.1.3.137 EAPI int evas_string_char_next_get (const char * *str*, int *pos*, int * *decoded*)

To be documented.

FIXME: To be fixed.

7.1.3.138 EAPI int evas_string_char_prev_get (const char * *str*, int *pos*, int * *decoded*)

To be documented.

FIXME: To be fixed.

Chapter 8

Evas Page Documentation

8.1 Todo List

Global evas_object_rectangle_add Find a documentation group to put this under.

Global evas_object_textblock_add Find a documentation group to put this under.

Global evas_object_event_callback_add Move this next code example and most of the documentation for this next function into the group documentation.

Group Evas_Object_Layer_Group Document which way layers go.

Global evas_hash_free Complete polishing documentation for evas_hash.c. The functions' docs may be grouped, but they need some simplification.

Index