Table of Contents

Object Hierarchy

```
GObject
    GVolObject
        GVolContainer
            GVolContainer1D
                GVolArray1Dgfloat
                GVolContainer1Dgfloat
            GVolContainer3D
                GVolArray3Dguint16
                GVolContainer3DRawRGBAgfloat
                    GVolArray3DRawRGBAgfloat
                GVolContainer3DVector3Dgfloat
                    GVolArray3DVector3Dgfloat
                GVolContainer3Dguint16
                GVolCubed2x4x4guint16
        GVolCamera
            GVolCameraFull
        GVolColor
            GVolColorPtr
                GVolColorPtrRGBAgfloat
            GVolColorRGBAgfloat
        GVolConvFilter3Dguint16
            GVolConvMatrix3Dguint16
        GVolInterp
            GVolInterpGSL
                GVolInterpGSLLinear
        GVolFunc
            GVolFunc_gfloat__gint32
                GVolLUT_gfloat__gint32
        GVolRenderingCtx
            GVolRenderingCtxRawRGBguchar
        GVolVolume
            GVolVolumeguint16
            GVolVolumeRGBAgfloat
GInterface
```

<< GVol Reference Manual GVol API reference >>

GVolArray1Dgfloat

GVolArray1Dgfloat —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolArray1Dgfloat;
GVolContainer* gvol_array1d_gfloat_new      (guint32 x_dim);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolContainer
               +----GVolContainer1D
                     +----GVolArray1Dgfloat
```

Description

Details

struct GVolArray1Dgfloat

```
struct GVolArray1Dgfloat;
```

---

gvol_array1d_gfloat_new ()

```
GVolContainer* gvol_array1d_gfloat_new      (guint32 x_dim);
```

Create a new GVolArray1Dgfloat with the specified dimension.

x_dim :

Returns : a new GVolArray1Dgfloat.

GVolArray3Dguint16

GVolArray3Dguint16 —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolArray3Dguint16;
GVolContainer* gvol_array3d_guint16_new     (guint32 x_dim,
                                             guint32 y_dim,
                                             guint32 z_dim);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolContainer
               +----GVolContainer3D
                     +----GVolArray3Dguint16
```

Description

Details

struct GVolArray3Dguint16

```
struct GVolArray3Dguint16;
```

---

gvol_array3d_guint16_new ()

```
GVolContainer* gvol_array3d_guint16_new     (guint32 x_dim,
                                             guint32 y_dim,
                                             guint32 z_dim);
```

Create a new GVolArray3Dguint16 with the specified dimension.

x_dim : X dimension of the new array

y_dim : Y dimension of the new array

z_dim : Z dimension of the new array

Returns : a new GVolArray3Dguint16.

GVolArray3DRawRGBAgfloat

GVolArray3DRawRGBAgfloat —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolArray3DRawRGBAgfloat;
GVolContainer* gvol_array3d_raw_rgba_gfloat_new
                                            (guint32 x_dim,
                                             guint32 y_dim,
                                             guint32 z_dim);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolContainer
               +----GVolContainer3D
                     +----GVolContainer3DRawRGBAgfloat
                           +----GVolArray3DRawRGBAgfloat
```

Description

Details

struct GVolArray3DRawRGBAgfloat

```
struct GVolArray3DRawRGBAgfloat;
```

---

gvol_array3d_raw_rgba_gfloat_new ()

```
GVolContainer* gvol_array3d_raw_rgba_gfloat_new
                                            (guint32 x_dim,
                                             guint32 y_dim,
                                             guint32 z_dim);
```

Create a new GVolArray3DRawRGBAgfloat with the specified dimension.

x_dim : X dimension of the new array

y_dim : Y dimension of the new array

z_dim : Z dimension of the new array

Returns : a new GVolArray3DRawRGBAgfloat.

<< **GVolArray3Dguint16** **GVolArray3DVector3Dgfloat** >> [p 81]

GVolArray3DVector3Dgfloat

GVolArray3DVector3Dgfloat —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolArray3DVector3Dgfloat;
GVolContainer* gvol_array3d_vector3d_gfloat_new
                                            (guint32 x_dim,
                                             guint32 y_dim,
                                             guint32 z_dim);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolContainer
               +----GVolContainer3D
                     +----GVolContainer3DVector3Dgfloat
                           +----GVolArray3DVector3Dgfloat
```

Description

Details

struct GVolArray3DVector3Dgfloat

```
struct GVolArray3DVector3Dgfloat;
```

---

gvol_array3d_vector3d_gfloat_new ()

```
GVolContainer* gvol_array3d_vector3d_gfloat_new
                                            (guint32 x_dim,
                                             guint32 y_dim,
                                             guint32 z_dim);
```

Create a new GVolArray3DVector3Dgfloat with the specified dimension.

x_dim : X dimension of the new array

y_dim : Y dimension of the new array

z_dim : Z dimension of the new array

Returns : a new GVolArray3DVector3Dgfloat.

<< **GVolArray3DRawRGBAgfloat** **GVolCamera** >> [p 85]

GVolCameraFull

GVolCameraFull —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolCameraFull;
GVolCamera* gvol_camera_full_new            (void);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolCamera
               +----GVolCameraFull
```

Properties

```
  "volume"               GVolVolume           : Read / Write
```

Description

Details

struct GVolCameraFull

```
struct GVolCameraFull;
```

---

gvol_camera_full_new ()

```
GVolCamera* gvol_camera_full_new            (void);
```

Create a new GVolCameraFull.

Returns : a new GVolCameraFull, placed in (0, 0, 0), looking along the (0, 0, -1) direction and with a view up vector parallel to (0, 1, 0).

Properties

"`volume`" (GVolVolume [p 135] : Read / Write) The volume to be rendered by the camera.

<< **GVolCamera** [p 85] **GVolColor** >> [p 92]

GVolCamera

GVolCamera —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolCamera;
gdouble     gvol_camera_get_back_clip_plane_dist
                                        (const GVolCamera *camera);
void        gvol_camera_get_eye_pos     (const GVolCamera *camera,
                                         GVolPoint3Dgdouble *pos);
gdouble     gvol_camera_get_front_clip_plane_dist
                                        (const GVolCamera *camera);
void        gvol_camera_get_view_dir    (const GVolCamera *camera,
                                         GVolVector3Dgdouble *dir);
void        gvol_camera_get_view_up     (const GVolCamera *camera,
                                         GVolVector3Dgdouble *vector);
void        gvol_camera_move_along_view_dir (GVolCamera *camera,
                                         gdouble modulo);
void        gvol_camera_place           (GVolCamera *camera,
                                         const GVolPoint3Dgdouble *eye_pos,
                                         const GVolVector3Dgdouble *view_dir,
                                         const GVolVector3Dgdouble *view_up);
void        gvol_camera_render          (GVolCamera *camera);
void        gvol_camera_rotate          (GVolCamera *camera,
                                         const GVolVector3Dgdouble *vector,
                                         const GVolPoint3Dgdouble *point,
                                         gdouble angle);
void        gvol_camera_rotate_around_vrp   (GVolCamera *camera,
                                         gdouble x_angle,
                                         gdouble y_angle);
void        gvol_camera_set_back_clip_plane_dist
                                        (GVolCamera *camera,
                                         gdouble dist);
void        gvol_camera_set_eye_pos     (GVolCamera *camera,
                                         const GVolPoint3Dgdouble *pos);
void        gvol_camera_set_front_clip_plane_dist
                                        (GVolCamera *camera,
                                         gdouble dist);
void        gvol_camera_set_rendering_ctx   (GVolCamera *camera,
                                         GVolRenderingCtx *ctx);
void        gvol_camera_set_view_dir    (GVolCamera *camera,
                                         const GVolVector3Dgdouble *direction);
void        gvol_camera_set_view_up     (GVolCamera *camera,
                                         const GVolVector3Dgdouble *vector);
```

Object Hierarchy

```
GObject
 +----GVolObject
       +----GVolCamera
```

Description

Details

struct GVolCamera

```
struct GVolCamera;
```

---

gvol_camera_get_back_clip_plane_dist ()

```
gdouble     gvol_camera_get_back_clip_plane_dist
                                        (const GVolCamera *camera);
```

Get the distance between the eye position and the back clipping plane.

camera : a GVolCamera

Returns : the distance between the eye position and the back clipping plane.

---

gvol_camera_get_eye_pos ()

```
void        gvol_camera_get_eye_pos         (const GVolCamera *camera,
                                             GVolPoint3Dgdouble *pos);
```

Get the camera eye position, storing it in *pos*.

camera : a GVolCamera

pos : a GVolVector3Dgdouble [p 186] in which the camera position will be stored

---

gvol_camera_get_front_clip_plane_dist ()

```
gdouble     gvol_camera_get_front_clip_plane_dist
                                        (const GVolCamera *camera);
```

Get the distance between the eye position and the front clipping plane.

camera : a GVolCamera

Returns : the distance between the eye position and the front clipping plane.

---

gvol_camera_get_view_dir ()

```
void        gvol_camera_get_view_dir        (const GVolCamera *camera,
                                             GVolVector3Dgdouble *dir);
```

Get the camera viewing direction, storing it in *dir*.

camera : a GVolCamera

dir : a GVolVector3Dgdouble [p 186] in which the camera viewing direction will be stored

---

gvol_camera_get_view_up ()

```
void        gvol_camera_get_view_up         (const GVolCamera *camera,
                                             GVolVector3Dgdouble *vector);
```

Get the camera view up vector, storing it in *vector*.

camera : a GVolCamera

vector : a GVolVector3Dgdouble [p 186] in which the camera view up vector will be stored

---

gvol_camera_move_along_view_dir ()

```
void        gvol_camera_move_along_view_dir (GVolCamera *camera,
                                             gdouble modulo);
```

Move the camera in along the viewing direction, with a length of *modulo*.

Note: if *modulo* is negative, then the movement will be backwards.

camera : a GVolCamera

modulo : the length of the movement

---

gvol_camera_place ()

```
void        gvol_camera_place               (GVolCamera *camera,
                                             const GVolPoint3Dgdouble *eye_pos,
                                             const GVolVector3Dgdouble *view_dir,
                                             const GVolVector3Dgdouble *view_up);
```

Place the camera in the three-dimensional space, setting all its positional parameters.

camera : a GVolCamera

eye_pos : the new eye position

view_dir : the viewing direction

view_up : the view up vector

---

gvol_camera_render ()

```
void        gvol_camera_render              (GVolCamera *camera);
```

Render an image of the camera, writing it in the GVolRenderingCtx [p 128] set with gvol_camera_set_rendering_ctx [p 90] ().

camera : a GVolCamera

---

gvol_camera_rotate ()

```
void        gvol_camera_rotate              (GVolCamera *camera,
                                             const GVolVector3Dgdouble *vector,
                                             const GVolPoint3Dgdouble *point,
                                             gdouble angle);
```

Rotate the camera by *angle* radiants, around a rotation axis parallel to *vector*, and passing through *point*.

camera : a GVolCamera

vector : the direction of the rotation axis

point : the point on which *vector* is applied

angle : the rotation angle (in radiants)

---

gvol_camera_rotate_around_vrp ()

```
void        gvol_camera_rotate_around_vrp   (GVolCamera *camera,
                                             gdouble x_angle,
                                             gdouble y_angle);
```

Rotate the camera spherically, around the view reference point. *x_angle* and *y_angle* are referred to the X and Y axis built perpendicular to the viewing direction, forming an orthonormal reference system when seen from the view reference point.

camera : a GVolCamera

x_angle : the rotation angle on the X axis

y_angle : the rotation angle on the Y axis

---

gvol_camera_set_back_clip_plane_dist ()

```
void        gvol_camera_set_back_clip_plane_dist
                                            (GVolCamera *camera,
                                             gdouble dist);
```

Set the distance between the eye position and the back clipping plane.

camera : a GVolCamera

dist : the new back clipping plane distance

---

gvol_camera_set_eye_pos ()

```
void        gvol_camera_set_eye_pos         (GVolCamera *camera,
                                             const GVolPoint3Dgdouble *pos);
```

Set the camera eye position to *pos*.

Note: The *pos* vector will be copied, and may be safely deallocated after the function call.

camera : a GVolCamera

pos : the new camera position

---

gvol_camera_set_front_clip_plane_dist ()

```
void        gvol_camera_set_front_clip_plane_dist
                                        (GVolCamera *camera,
                                         gdouble dist);
```

Set the distance between the eye position and the front clipping plane.

camera : a GVolCamera

dist : the new front clipping plane distance

---

gvol_camera_set_rendering_ctx ()

```
void        gvol_camera_set_rendering_ctx   (GVolCamera *camera,
                                             GVolRenderingCtx *ctx);
```

Set *ctx* as the rendering context that will be used to draw images rendered by *camera*.

camera : a GVolCamera

ctx : a GVolRenderingCtx [p 128] to be used for drawing

---

gvol_camera_set_view_dir ()

```
void        gvol_camera_set_view_dir        (GVolCamera *camera,
                                             const GVolVector3Dgdouble *direction);
```

Set the camera viewing direction.

Note: The *dir* vector will be copied, and may be safely deallocated after the function call.

camera : a GVolCamera

direction :

---

gvol_camera_set_view_up ()

```
void        gvol_camera_set_view_up         (GVolCamera *camera,
                                             const GVolVector3Dgdouble *vector);
```

Set the camera view up vector..

Note: *vector* will be copied, and may be safely deallocated after the function call.

camera : a GVolCamera

vector : the new view up vector

<< **GVolArray3DVector3Dgfloat** **GVolCameraFull** >>

GVolColor

GVolColor —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolColor;
void        gvol_color_clamp                (GVolColor *color);
void        gvol_color_clamp_alpha          (GVolColor *color);
void        gvol_color_get_rgba_guchar      (const GVolColor *color,
                                             GVolRawRGBAguchar *rgba);
void        gvol_color_get_rgb_guchar       (const GVolColor *color,
                                             GVolRawRGBguchar *rgba);
gboolean    gvol_color_has_alpha            (const GVolColor *color);
void        gvol_color_interp               (GVolColor *color,
                                             gdouble xv[],
                                             const GVolColor *const  colors[],
                                             guint num_colors,
                                             gdouble x,
                                             const GVolInterp *interp);
void        gvol_color_scale                (GVolColor *color,
                                             const GVolColor *color1,
                                             gdouble scalar);
void        gvol_color_set_rgba_guchar      (GVolColor *color,
                                             const GVolRawRGBAguchar *rgba);
void        gvol_color_set_rgb_guchar       (GVolColor *color,
                                             const GVolRawRGBguchar *rgba);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolColor
```

Description

Details

struct GVolColor

```
struct GVolColor;
```

gvol_color_clamp ()

```
void        gvol_color_clamp                (GVolColor *color);
```

Clamp *color* if it is outside of the allowed range of values.

Note: the alpha channel of *color* (if any) won't be clamped. Use gvol_color_clamp_alpha [p 93] ().

color : a GVolColor

gvol_color_clamp_alpha ()

```
void        gvol_color_clamp_alpha          (GVolColor *color);
```

Clamp the alpha channel (if any) of *color*, if it is outside of the allowed range of values.

Note: if you need to clamp the other components of *color*, see gvol_color_clamp [p 93] ().

color : a GVolColor

gvol_color_get_rgba_guchar ()

```
void        gvol_color_get_rgba_guchar      (const GVolColor *color,
                                             GVolRawRGBAguchar *rgba);
```

Store in *rgba* the RGBA value associated with *color*.

Note: if *color* doesn't have an alpha channel, the returned alpha component will be set to 1.0.

color : a GVolColor

rgba :

gvol_color_get_rgb_guchar ()

```
void        gvol_color_get_rgb_guchar       (const GVolColor *color,
                                             GVolRawRGBguchar *rgba);
```

Store in *rgb* the RGB value associated with *color*.

Note: the alpha component of *color*, if any, will be used to scale the resulting R, G and B values.

color : a GVolColor

rgba :

---

gvol_color_has_alpha ()

```
gboolean    gvol_color_has_alpha            (const GVolColor *color);
```

Say whether *color* has an alpha channel.

Return values: TRUE if *color* has an alpha channel, FALSE otherwise.

color : a GVolColor

Returns :

---

gvol_color_interp ()

```
void        gvol_color_interp               (GVolColor *color,
                                             gdouble xv[],
                                             const GVolColor *const  colors[],
                                             guint num_colors,
                                             gdouble x,
                                             const GVolInterp *interp);
```

Make *color* the interpolation between the *num_colors* GVolColor's pointed by *colors*, using *interp* as interpolator.

NOTE: the alpha channels of *colors* (if any) will be interpolated, too.

color : a GVolColor

xv : values on the X axis of the collocation points

colors : an array of pointers to GVolColor's to be interpolated

num_colors : number of colorso in *colors*

x : the X coordinate in which the interpolation will be evaluated

interp : the GVolInterp [p 124] to be used for interpolations

---

gvol_color_scale ()

```
void        gvol_color_scale                (GVolColor *color,
                                             const GVolColor *color1,
                                             gdouble scalar);
```

Multiply *color1* by *scalar* (e. g. to simulate fog, distance or opacity attenuation), storing the result in *color*.

Note: the alpha component of *color1*, if any, won't be scaled.

color : a GVolColor

color1 : a GVolColor to be scaled

scalar : a scalar for which *color1* will be multiplied (usually between 0.0 and 1.0)

---

gvol_color_set_rgba_guchar ()

```
void        gvol_color_set_rgba_guchar      (GVolColor *color,
                                             const GVolRawRGBAguchar *rgba);
```

color :

rgba :

---

gvol_color_set_rgb_guchar ()

```
void        gvol_color_set_rgb_guchar       (GVolColor *color,
                                             const GVolRawRGBguchar *rgba);
```

Set the RGB value of *color* with the values contained in *rgb*.

Note: if *color* doesn't have an alpha channel, the alpha component or *rgba* will be used to scale the R, G, and B values that will be read from *rgba* itself.

color : a GVolColor

rgba : the GVolRawRGBAguchar [p 176] from which the RGBA value will be read

<< **GVolCameraFull** **GVolColorPtr** >> [p 96]

GVolColorPtr

GVolColorPtr —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolColorPtr;
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolColor
               +----GVolColorPtr
```

Description

Details

struct GVolColorPtr

```
struct GVolColorPtr;
```

<< **GVolColor** **GVolColorPtrRGBAgfloat** >> [p 97]

GVolColorPtrRGBAgfloat

GVolColorPtrRGBAgfloat —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolColorPtrRGBAgfloat;
GVolColorPtrRGBAgfloat* gvol_color_ptr_rgba_gfloat_new
                                            (GVolRawRGBAgfloat *ptr);
void        gvol_color_ptr_set_ptr          (GVolColorPtr *color,
                                             gpointer ptr);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolColor
               +----GVolColorPtr
                     +----GVolColorPtrRGBAgfloat
```

Description

Details

struct GVolColorPtrRGBAgfloat

```
struct GVolColorPtrRGBAgfloat;
```

---

gvol_color_ptr_rgba_gfloat_new ()

```
GVolColorPtrRGBAgfloat* gvol_color_ptr_rgba_gfloat_new
                                            (GVolRawRGBAgfloat *ptr);
```

Create a new GVolColorPtrRGBAgfloat.

ptr :	a pointer to a RGBA gfloat vector
Returns :	a new GVolColorPtrRGBAgfloat object.

---

gvol_color_ptr_set_ptr ()

```
void        gvol_color_ptr_set_ptr          (GVolColorPtr *color,
                                             gpointer ptr);
```

Set the pointer to the actual color data.

Note: the type of data pointed by *ptr* will depend on the subclass of GVolColorPtr you are using.

color : a GVolColorPtr

ptr : a pointer to actual color data

<< **GVolColorPtr** **GVolColorRGBAgfloat** >> [p 99]

GVolColorRGBAgfloat

GVolColorRGBAgfloat —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolColorRGBAgfloat;
GVolColorRGBAgfloat* gvol_color_rgba_gfloat_new
                                            (gfloat red,
                                             gfloat green,
                                             gfloat blue,
                                             gfloat alpha);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolColor
               +----GVolColorRGBAgfloat
```

Description

Details

struct GVolColorRGBAgfloat

```
struct GVolColorRGBAgfloat;
```

---

gvol_color_rgba_gfloat_new ()

```
GVolColorRGBAgfloat* gvol_color_rgba_gfloat_new
                                            (gfloat red,
                                             gfloat green,
                                             gfloat blue,
                                             gfloat alpha);
```

Create a new GVolColorRGBAgfloat.

`red` : red color component

`green` : green color component

`blue` : blue color component

`alpha` : alpha channel

Returns : a new GVolColorRGBAgfloat object.

<< **GVolColorPtrRGBAgfloat** **GVolContainer** >> [p 114]

GVolContainer1Dgfloat

GVolContainer1Dgfloat —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolContainer1Dgfloat;
gfloat      gvol_container1d_gfloat_get_value
                                        (const GVolContainer1Dgfloat *container,
                                         guint32 x);
void        gvol_container1d_gfloat_set_value
                                        (GVolContainer1Dgfloat *container,
                                         guint32 x,
                                         gfloat value);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolContainer
               +----GVolContainer1D
                     +----GVolContainer1Dgfloat
```

Description

Details

struct GVolContainer1Dgfloat

```
struct GVolContainer1Dgfloat;
```

---

gvol_container1d_gfloat_get_value ()

```
gfloat      gvol_container1d_gfloat_get_value
                                        (const GVolContainer1Dgfloat *container,
                                         guint32 x);
```

Get the value at the *x* coordinate of *container*.

container : a GVolContainer1Dgfloat

x : the coordinate of the element to retrieve

Returns : the value read at the *x* position.

---

gvol_container1d_gfloat_set_value ()

```
void        gvol_container1d_gfloat_set_value
                                            (GVolContainer1Dgfloat *container,
                                             guint32 x,
                                             gfloat value);
```

Set the value at the *x* coordinate of *container*.

container : a GVolContainer1Dgfloat

x : the coordinate of the element to set

value : the value to be set.

<< **GVolContainer1D** [p 103] **GVolContainer3D** >> [p 107]

GVolContainer1D

GVolContainer1D —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolContainer1D;
guint32     gvol_container1d_get_dimension  (const GVolContainer1D *container);
void        gvol_container1d_get_gvalue     (const GVolContainer1D *container,
                                              guint32 x,
                                              GValue *value);
gpointer    gvol_container1d_get_ptr        (const GVolContainer1D *container,
                                              guint32 x);
void        gvol_container1d_set_gvalue     (GVolContainer1D *container,
                                              guint32 x,
                                              const GValue *value);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolContainer
               +----GVolContainer1D
```

Description

Details

struct GVolContainer1D

```
struct GVolContainer1D;
```

---

gvol_container1d_get_dimension ()

```
guint32     gvol_container1d_get_dimension  (const GVolContainer1D *container);
```

Get the dimension of *container*.

container : a GVolContainer1D

Returns : the dimension of *container*.

gvol_container1d_get_gvalue ()

```
void        gvol_container1d_get_gvalue     (const GVolContainer1D *container,
                                             guint32 x,
                                             GValue *value);
```

container :

x :

value :

gvol_container1d_get_ptr ()

```
gpointer    gvol_container1d_get_ptr        (const GVolContainer1D *container,
                                             guint32 x);
```

Get a pointer to the element at the *x* coordinate of *container*.

container : a GVolContainer1D

x : the coordinate of the element to retrieve

Returns : a pointer to the element at the *x* position.

gvol_container1d_set_gvalue ()

```
void        gvol_container1d_set_gvalue     (GVolContainer1D *container,
                                             guint32 x,
                                             const GValue *value);
```

container :

x :

value :

<< **GVolContainer** [p 114] **GVolContainer1Dgfloat** >>

GVolContainer3Dguint16

GVolContainer3Dguint16 —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolContainer3Dguint16;
guint16     gvol_container3d_guint16_get_value
                                        (const GVolContainer3Dguint16 *container,
                                         guint32 x,
                                         guint32 y,
                                         guint32 z);
void        gvol_container3d_guint16_set_value
                                        (GVolContainer3Dguint16 *container,
                                         guint32 x,
                                         guint32 y,
                                         guint32 z,
                                         guint16 value);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolContainer
               +----GVolContainer3D
                     +----GVolContainer3Dguint16
```

Description

Details

struct GVolContainer3Dguint16

```
struct GVolContainer3Dguint16;
```

---

gvol_container3d_guint16_get_value ()

```
guint16     gvol_container3d_guint16_get_value
                                        (const GVolContainer3Dguint16 *container,
                                         guint32 x,
                                         guint32 y,
                                         guint32 z);
```

Get the value at the (*x*, *y*, *z*) coordinate of *container*.

container :	a GVolContainer3Dguint16
x :	The X coordinate of the element to retrieve
y :	The Y coordinate of the element to retrieve
z :	The Z coordinate of the element to retrieve
Returns :	the value read at (*x*, *y*, *z*).

---

gvol_container3d_guint16_set_value ()

```
void        gvol_container3d_guint16_set_value
                                        (GVolContainer3Dguint16 *container,
                                         guint32 x,
                                         guint32 y,
                                         guint32 z,
                                         guint16 value);
```

Set the value at the (*x*, *y*, *z*) coordinate of *container*.

container :	a GVolContainer3Dguint16
x :	the X coordinate of the element to set
y :	the Y coordinate of the element to set
z :	the Z coordinate of the element to set
value :	the value to be set.

<< **GVolContainer3D** [p 107] **GVolContainer3DRawRGBAgfloat** >> [p 110]

GVolContainer3D

GVolContainer3D —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolContainer3D;
void        gvol_container3d_get_dimensions (const GVolContainer3D *container,
                                             guint32 *x_dim,
                                             guint32 *y_dim,
                                             guint32 *z_dim);
GValue*     gvol_container3d_get_gvalue     (const GVolContainer3D *container,
                                             guint32 x,
                                             guint32 y,
                                             guint32 z);
gpointer    gvol_container3d_get_ptr        (const GVolContainer3D *container,
                                             guint32 x,
                                             guint32 y,
                                             guint32 z);
void        gvol_container3d_set_gvalue     (GVolContainer3D *container,
                                             guint32 x,
                                             guint32 y,
                                             guint32 z,
                                             GValue *value);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolContainer
               +----GVolContainer3D
```

Description

Details

struct GVolContainer3D

```
struct GVolContainer3D;
```

gvol_container3d_get_dimensions ()

```
void        gvol_container3d_get_dimensions (const GVolContainer3D *container,
                                             guint32 *x_dim,
                                             guint32 *y_dim,
                                             guint32 *z_dim);
```

Get the X, Y and/or Z dimensions of *container*, storing them in *x_dim*, *y_dim* and/or *z_dim*.

NOTE: if *x_dim*, *y_dim* or *z_dim* are NULL, the corresponding dimension won't be retrieved.

container : a GVolContainer3D

x_dim : a guint32 pointer used to store the X dimension of *container*

y_dim : a guint32 pointer used to store the Y dimension of *container*

z_dim : a guint32 pointer used to store the Z dimension of *container*

---

gvol_container3d_get_gvalue ()

```
GValue*     gvol_container3d_get_gvalue     (const GVolContainer3D *container,
                                             guint32 x,
                                             guint32 y,
                                             guint32 z);
```

container :

x :

y :

z :

Returns :

---

gvol_container3d_get_ptr ()

```
gpointer    gvol_container3d_get_ptr        (const GVolContainer3D *container,
                                             guint32 x,
                                             guint32 y,
                                             guint32 z);
```

Get a pointer to the element at the (*x*, *y*, *z*) coordinate of *container*.

container : a GVolContainer3D

x : the X coordinate of the element to retrieve

y : the Y coordinate of the element to retrieve

z : the Z coordinate of the element to retrieve

Returns : a pointer to the element at position (*x*, *y*, *z*).

---

gvol_container3d_set_gvalue ()

```
void        gvol_container3d_set_gvalue     (GVolContainer3D *container,
                                              guint32 x,
                                              guint32 y,
                                              guint32 z,
                                              GValue *value);
```

container :

x :

y :

z :

value :

<< **GVolContainer1Dgfloat** **GVolContainer3Dguint16** >>

GVolContainer3DRawRGBAgfloat

GVolContainer3DRawRGBAgfloat —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolContainer3DRawRGBAgfloat;
const GVolRawRGBAgfloat* gvol_container3d_raw_rgba_gfloat_get_value
                                            (const GVolContainer3DRawRGBAgfloat *container,
                                             guint32 x,
                                             guint32 y,
                                             guint32 z);
void        gvol_container3d_raw_rgba_gfloat_set_value
                                            (GVolContainer3DRawRGBAgfloat *container,
                                             guint32 x,
                                             guint32 y,
                                             guint32 z,
                                             GVolRawRGBAgfloat *value);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolContainer
               +----GVolContainer3D
                     +----GVolContainer3DRawRGBAgfloat
```

Description

Details

struct GVolContainer3DRawRGBAgfloat

```
struct GVolContainer3DRawRGBAgfloat;
```

---

gvol_container3d_raw_rgba_gfloat_get_value ()

```
const GVolRawRGBAgfloat* gvol_container3d_raw_rgba_gfloat_get_value
                                            (const GVolContainer3DRawRGBAgfloat *container,
                                             guint32 x,
                                             guint32 y,
                                             guint32 z);
```

Get the GVolRawRGBAgfloat [p 172] at the (*x*, *y*, *z*) coordinate of *container*.

container :	a GVolContainer3DRawRGBAgfloat
x :	The X coordinate of the element to retrieve
y :	The Y coordinate of the element to retrieve
z :	The Z coordinate of the element to retrieve
Returns :	the GVolRawRGBAgfloat [p 172] at position (*x*, *y*, *z*).

---

gvol_container3d_raw_rgba_gfloat_set_value ()

```
void        gvol_container3d_raw_rgba_gfloat_set_value
                                        (GVolContainer3DRawRGBAgfloat *container,
                                         guint32 x,
                                         guint32 y,
                                         guint32 z,
                                         GVolRawRGBAgfloat *value);
```

Set the value at the (*x*, *y*, *z*) coordinate of *container*, copying it from the location pointed by *value*.

container :	a GVolContainer3DRawRGBAgfloat
x :	the X coordinate of the element to set
y :	the Y coordinate of the element to set
z :	the Z coordinate of the element to set
value :	the value to be set.

<< **GVolContainer3Dguint16** **GVolContainer3DVector3Dgfloat** >> [p 112]

GVolContainer3DVector3Dgfloat

GVolContainer3DVector3Dgfloat —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolContainer3DVector3Dgfloat;
const GVolVector3Dgfloat* gvol_container3d_vector3d_gfloat_get_value
                                            (const GVolContainer3DVector3Dgfloat *container,
                                             guint32 x,
                                             guint32 y,
                                             guint32 z);
void        gvol_container3d_vector3d_gfloat_set_value
                                            (GVolContainer3DVector3Dgfloat *container,
                                             guint32 x,
                                             guint32 y,
                                             guint32 z,
                                             GVolVector3Dgfloat *value);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolContainer
               +----GVolContainer3D
                     +----GVolContainer3DVector3Dgfloat
```

Description

Details

struct GVolContainer3DVector3Dgfloat

```
struct GVolContainer3DVector3Dgfloat;
```

---

gvol_container3d_vector3d_gfloat_get_value ()

```
const GVolVector3Dgfloat* gvol_container3d_vector3d_gfloat_get_value
                                            (const GVolContainer3DVector3Dgfloat *container,
                                             guint32 x,
                                             guint32 y,
                                             guint32 z);
```

Get the GVolVector3Dgfloat [p 192] at the (*x*, *y*, *z*) coordinate of *container*.

container :	a GVolContainer3DVector3Dgfloat
x :	The X coordinate of the element to retrieve
y :	The Y coordinate of the element to retrieve
z :	The Z coordinate of the element to retrieve
Returns :	the GVolVector3Dgfloat [p 192] at position (*x*, *y*, *z*).

---

gvol_container3d_vector3d_gfloat_set_value ()

```
void        gvol_container3d_vector3d_gfloat_set_value
                                          (GVolContainer3DVector3Dgfloat *container,
                                           guint32 x,
                                           guint32 y,
                                           guint32 z,
                                           GVolVector3Dgfloat *value);
```

Set the value at the (*x*, *y*, *z*) coordinate of *container*, copying it from the location pointed by *value*.

container :	a GVolContainer3DVector3Dgfloat
x :	the X coordinate of the element to set
y :	the Y coordinate of the element to set
z :	the Z coordinate of the element to set
value :	the value to be set.

<< **GVolContainer3DRawRGBAgfloat** **GVolConvFilter3Dguint16** >> [p 116]

GVolContainer

GVolContainer —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolContainer;
gboolean    gvol_container_check_elements_type
                                        (const GVolContainer *container,
                                         GType type);
guint32     gvol_container_get_elements_length
                                        (const GVolContainer *container);
GIterator*  gvol_container_iterate      (const GVolContainer *container);
GValueIterator* gvol_container_iterate_gvalue
                                        (const GVolContainer *container);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolContainer
```

Description

Details

struct GVolContainer

```
struct GVolContainer;
```

---

gvol_container_check_elements_type ()

```
gboolean    gvol_container_check_elements_type
                                        (const GVolContainer *container,
                                         GType type);
```

Check whether *container* supports elements of type *type*.

container : a GVolContainer

type : a GType

Returns : TRUE if *container* could contain elements of type *type*, FALSE otherwise.

---

gvol_container_get_elements_length ()

```
guint32     gvol_container_get_elements_length
                                        (const GVolContainer *container);
```

container :

Returns :

---

gvol_container_iterate ()

```
GIterator*  gvol_container_iterate          (const GVolContainer *container);
```

Get a GIterator that can be used to access the *container* elements. Of course, the size of the element pointed by the iterator will depend on the type of GVolContainer that has been instantiated.

container : a GVolContainer

Returns : a GIterator.

---

gvol_container_iterate_gvalue ()

```
GValueIterator* gvol_container_iterate_gvalue
                                        (const GVolContainer *container);
```

Get a GValueIterator that can be used to access the *container* elements.

container : a GVolContainer

Returns : a GValueIterator.

<< **GVolColorRGBAgfloat** **GVolContainer1D** >>

GVolConvFilter3Dguint16

GVolConvFilter3Dguint16 —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolConvFilter3Dguint16;
void        gvol_convfilter3d_guint16_eval  (const GVolConvFilter3Dguint16 *filter,
                                             const GVolContainer3Dguint16 *container,
                                             guint32 x,
                                             guint32 y,
                                             guint32 z,
                                             GVolVector3Dgdouble *result);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolConvFilter3Dguint16
```

Description

Details

struct GVolConvFilter3Dguint16

```
struct GVolConvFilter3Dguint16;
```

---

gvol_convfilter3d_guint16_eval ()

```
void        gvol_convfilter3d_guint16_eval  (const GVolConvFilter3Dguint16 *filter,
                                             const GVolContainer3Dguint16 *container,
                                             guint32 x,
                                             guint32 y,
                                             guint32 z,
                                             GVolVector3Dgdouble *result);
```

Evaluate the result of *filter* applied on the value at the (*x*, *y*, *z*) position of *container*. The resulting vector will be stored inside *result*.

filter :	a GVolConvFilter3Dguint16
container :	a GVolContainer3Dguint16
x :	the X coordinate of *container* in which the filter will be applied
y :	the Y coordinate of *container* in which the filter will be applied
z :	the Z coordinate of *container* in which the filter will be applied
result :	a GVolVector3Dgdouble [p 186] to store the filter result

<< **GVolContainer3DVector3Dgfloat** **GVolConvMatrix3Dguint16** >> [p 118]

GVolConvMatrix3Dguint16

GVolConvMatrix3Dguint16 —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolConvMatrix3Dguint16;
GVolConvFilter3Dguint16* gvol_convmatrix3d_guint16_new
                                            (void);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolConvFilter3Dguint16
               +----GVolConvMatrix3Dguint16
```

Description

Details

struct GVolConvMatrix3Dguint16

```
struct GVolConvMatrix3Dguint16;
```

---

gvol_convmatrix3d_guint16_new ()

```
GVolConvFilter3Dguint16* gvol_convmatrix3d_guint16_new
                                            (void);
```

Create a new GVolConvMatrix3Dguint16.

Returns :	a new GVolConvMatrix3Dguint16. NOTE: the returned convolution matrix will be a zero filter (i. e. it will return zero for every dimension of application).

<< GVolConvFilter3Dguint16 **GVolCubed2x4x4guint16 >>** [p 119]

GVolCubed2x4x4guint16

GVolCubed2x4x4guint16 —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolCubed2x4x4guint16;
GVolContainer* gvol_cubed2x4x4_guint16_new  (guint32 x_dim,
                                             guint32 y_dim,
                                             guint32 z_dim);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolContainer
               +----GVolContainer3D
                     +----GVolCubed2x4x4guint16
```

Description

Details

struct GVolCubed2x4x4guint16

```
struct GVolCubed2x4x4guint16;
```

---

gvol_cubed2x4x4_guint16_new ()

```
GVolContainer* gvol_cubed2x4x4_guint16_new  (guint32 x_dim,
                                             guint32 y_dim,
                                             guint32 z_dim);
```

Create a new GVolCubed2x4x4guint16 with the specified dimension.

x_dim : X dimension of the new array

y_dim : Y dimension of the new array

z_dim : Z dimension of the new array

Returns : a new GVolCubed2x4x4guint16.

GVolInterpGSL

GVolInterpGSL —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolInterpGSL;
void        gvol_interp_gsl_free_accel      (GVolInterpGSL *interp);
void        gvol_interp_gsl_init_accel      (GVolInterpGSL *interp);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolInterp
               +----GVolInterpGSL
```

Description

Details

struct GVolInterpGSL

```
struct GVolInterpGSL;
```

---

gvol_interp_gsl_free_accel ()

```
void        gvol_interp_gsl_free_accel      (GVolInterpGSL *interp);
```

Free the interpolation accelerator used by *interp*.

interp : a GVolInterpGSL

---

gvol_interp_gsl_init_accel ()

```
void        gvol_interp_gsl_init_accel      (GVolInterpGSL *interp);
```

Initialize the interpolation accelerator used by *interp*.

Note: the interpolator accelerator should be used only as long as you interpolate on the same application points. If you change the application points, you should also free and re-init the accelerator.

interp : a GVolInterpGSL

<< **GVolInterp** [p 124] **GVolInterpGSLLinear** >> [p 123]

GVolInterpGSLLinear

GVolInterpGSLLinear —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolInterpGSLLinear;
GVolInterp* gvol_interp_gsl_linear_new      (void);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolInterp
               +----GVolInterpGSL
                     +----GVolInterpGSLLinear
```

Description

Details

struct GVolInterpGSLLinear

```
struct GVolInterpGSLLinear;
```

---

gvol_interp_gsl_linear_new ()

```
GVolInterp* gvol_interp_gsl_linear_new      (void);
```

Create a new GVolInterpGSLLinear.

Returns : a new GVolInterp [p 124] object.

GVolInterp

GVolInterp —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolInterp;
gdouble     gvol_interp_eval                (const GVolInterp *interp,
                                             const gdouble xv[],
                                             const gdouble yv[],
                                             guint num_points,
                                             gdouble x);
guint       gvol_interp_get_min_points      (GVolInterp *interp);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolInterp
```

Description

Details

struct GVolInterp

```
struct GVolInterp;
```

---

gvol_interp_eval ()

```
gdouble     gvol_interp_eval                (const GVolInterp *interp,
                                             const gdouble xv[],
                                             const gdouble yv[],
                                             guint num_points,
                                             gdouble x);
```

Return the interpolated value of y for a given point *x*.

interp :	a GVolInterp
xv :	values on the X axis of the collocation points
yv :	values on the Y axis of the collocation points
num_points :	number of collocation points (i. e. size of *xv* and *yv*)
x :	the X coordinate in which the interpolation will be evaluated
Returns :	the interpolated value of y for *x*.

---

gvol_interp_get_min_points ()

```
guint       gvol_interp_get_min_points      (GVolInterp *interp);
```

Get the miniumun number of collocation points required by *interp*.

interp :	a GVolInterp
Returns :	the miniumun number of collocation points required by *interp*.

<< **GVolCubed2x4x4guint16** **GVolInterpGSL** >>

GVolObject

GVolObject —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolObject;
```

Object Hierarchy

```
  GObject
   +----GVolObject
```

Properties

```
  "name"                 gchararray           : Read / Write
```

Signal Prototypes

```
"progress"  void        user_function      (GVolObject *gvolobject,
                                            gchar *arg1,
                                            gint arg2,
                                            gpointer user_data);
```

Description

Details

struct GVolObject

```
struct GVolObject;
```

Properties

"name" (gchararray : Read / Write) The name of the object.

Signals

The "progress" signal

```
void        user_function                  (GVolObject *gvolobject,
                                            gchar *arg1,
                                            gint arg2,
                                            gpointer user_data);
```

gvolobject : the object which received the signal.

arg1 :

arg2 :

user_data : user data set when the signal handler was connected.

<< **GVolMatrix4x4gfloat** [p 160] **GVolPoint3Dgdouble** >> [p 165]

GVolRenderingCtx

GVolRenderingCtx —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolRenderingCtx;
gboolean    gvol_rendering_ctx_check_rectangle
                                        (const GVolRenderingCtx *ctx,
                                         const GVolRectanglegint32 *rect);
GType       gvol_rendering_ctx_get_elements_type
                                        (const GVolRenderingCtx *ctx);
void        gvol_rendering_ctx_get_extent   (const GVolRenderingCtx *ctx,
                                         GVolRectanglegint32 *rect);
void        gvol_rendering_ctx_get_rendering_rgn
                                        (GVolRenderingCtx *ctx,
                                         const GVolRectanglegint32 *rect,
                                         GVolRenderingRgn *rgn);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolRenderingCtx
```

Description

Details

struct GVolRenderingCtx

```
struct GVolRenderingCtx;
```

---

gvol_rendering_ctx_check_rectangle ()

```
gboolean    gvol_rendering_ctx_check_rectangle
                                        (const GVolRenderingCtx *ctx,
                                         const GVolRectanglegint32 *rect);
```

Check whether *ctx* could contain a region with the dimension of *rect*.

ctx : a GVolRenderingCtx

rect : a GVolRectanglegint32 [p 182]

Returns : TRUE if *ctx* could contain *rect*, FALSE otherwhise.

---

gvol_rendering_ctx_get_elements_type ()

```
GType        gvol_rendering_ctx_get_elements_type
                                        (const GVolRenderingCtx *ctx);
```

Get the type of rendering items supported by *ctx*.

ctx : a GVolRenderingCtx

Returns : the type of rendering items supported by *ctx*.

---

gvol_rendering_ctx_get_extent ()

```
void         gvol_rendering_ctx_get_extent   (const GVolRenderingCtx *ctx,
                                              GVolRectanglegint32 *rect);
```

Get the extent of *ctx* (i. e. the dimension of the largest GVolRenderingRgn [p 184] that could be extracted from *ctx* using gvol_rendering_ctx_get_rendering_rgn [p 129] ()), storing them in *rect*.

ctx : a GVolRenderingCtx

rect : a GVolRectanglegint32 [p 182] used to store the dimensions

---

gvol_rendering_ctx_get_rendering_rgn ()

```
void         gvol_rendering_ctx_get_rendering_rgn
                                        (GVolRenderingCtx *ctx,
                                         const GVolRectanglegint32 *rect,
                                         GVolRenderingRgn *rgn);
```

Get a rendering region from *ctx*, storing it in *rgn*.

ctx : a GVolRenderingCtx

rect :

rgn : a GVolRenderingRgn [p 184] to store the required region

<< **GVolRectanglegint32** [p 182] **GVolRenderingCtxRawRGBguchar** >> [p 131]

GVolRenderingCtxRawRGBguchar

GVolRenderingCtxRawRGBguchar —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolRenderingCtxRawRGBguchar;
GVolRenderingCtx* gvol_rendering_ctx_raw_rgb_guchar_new
                                            (GVolRawRGBguchar *buf,
                                             guint32 x_dim,
                                             guint32 y_dim);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolRenderingCtx
               +----GVolRenderingCtxRawRGBguchar
```

Description

Details

struct GVolRenderingCtxRawRGBguchar

```
struct GVolRenderingCtxRawRGBguchar;
```

---

gvol_rendering_ctx_raw_rgb_guchar_new ()

```
GVolRenderingCtx* gvol_rendering_ctx_raw_rgb_guchar_new
                                            (GVolRawRGBguchar *buf,
                                             guint32 x_dim,
                                             guint32 y_dim);
```

Create a new GVolRenderingCtxRawRGBguchar that draws on *buf*.

buf : a bidimensional array of GVolRawRGBguchar [p 179] 's for rendering

x_dim : X dimension of the new rendering context

y_dim : Y dimension of the new rendering context

Returns : a new GVolRenderingCtxRawRGBguchar.

<< **GVolRenderingCtx** **GVolRenderingRgn** >> [p 184]

GVolVolumeguint16

GVolVolumeguint16 —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolVolumeguint16;
GVolVolume* gvol_volume_guint16_new         (void);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolVolume
               +----GVolVolumeguint16
```

Properties

```
  "dataset"              GVolContainer3D      : Read / Write
  "gradient-filter"      GVolConvFilter3Dguint16 : Read / Write
  "gradients-container"  GVolContainer3DVector3Dgfloat : Read / Write
  "interpolator"         GVolInterp           : Read / Write
  "opacity-tfunc"        GVolFunc_gfloat__gint32 : Read / Write
  "z-sampling"           gdouble              : Read / Write
```

Description

Details

struct GVolVolumeguint16

```
struct GVolVolumeguint16;
```

---

gvol_volume_guint16_new ()

```
GVolVolume* gvol_volume_guint16_new         (void);
```

Create a new GVolVolumeguint16.

Returns : a new GVolVolumeguint16 Note: the new volume will be empty (i. e. no voxels inside), will be centered at the origin of the world reference system, and its bounding box will extend from (-1, -1, -1) to (1, 1, 1).

Properties

"`dataset`" (GVolContainer3D : Read / Write)	The dataset associated with the volume.
"`gradient-filter`" (GVolConvFilter3Dguint16 : Read / Write)	The convolution filter used to calculate gradients.
"`gradients-container`" (GVolContainer3DVector3Dgfloat : Read / Write)	The container used to store gradients.
"`interpolator`" (GVolInterp : Read / Write)	The interpolator to be used with colors.
"`opacity-tfunc`" (GVolFunc_gfloat__gint32 [p 145] : Read / Write)	The transfer function used to get opacity from voxel values.
"`z-sampling`" (gdouble : Read / Write)	Depth oversampling factor.

<< **GVolVolume** [p 135] **GVolVolumeRGBAgfloat** >> [p 139]

GVolVolume

GVolVolume —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolVolume;
void        gvol_volume_attach_dataset      (GVolVolume *volume,
                                             const GVolContainer3D *dataset);
void        gvol_volume_cast_ray            (const GVolVolume *volume,
                                             const GVolPoint3Dgdouble *eye_pos,
                                             const GVolVector3Dgdouble *dir,
                                             const GVolPoint3Dgdouble *ray_begin,
                                             const GVolPoint3Dgdouble *ray_end,
                                             GVolRenderingRgn *rgn,
                                             guint32 rgn_x,
                                             guint32 rgn_y);
void        gvol_volume_cast_ray_local      (const GVolVolume *volume,
                                             const GVolPoint3Dgdouble *eye_pos,
                                             const GVolVector3Dgdouble *dir,
                                             const GVolPoint3Dgdouble *ray_begin,
                                             const GVolPoint3Dgdouble *ray_end,
                                             GVolRenderingRgn *rgn,
                                             guint32 rgn_x,
                                             guint32 rgn_y);
void        gvol_volume_get_coord_sys       (const GVolVolume *volume,
                                             GVolPoint3Dgdouble *o,
                                             GVolVector3Dgdouble *x,
                                             GVolVector3Dgdouble *y,
                                             GVolVector3Dgdouble *z);
void        gvol_volume_get_dimensions      (const GVolVolume *volume,
                                             guint32 *x_dim,
                                             guint32 *y_dim,
                                             guint32 *z_dim);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolVolume
```

Description

Details

struct GVolVolume

```
struct GVolVolume;
```

---

gvol_volume_attach_dataset ()

```
void        gvol_volume_attach_dataset      (GVolVolume *volume,
                                             const GVolContainer3D *dataset);
```

volume :

dataset :

---

gvol_volume_cast_ray ()

```
void        gvol_volume_cast_ray            (const GVolVolume *volume,
                                             const GVolPoint3Dgdouble *eye_pos,
                                             const GVolVector3Dgdouble *dir,
                                             const GVolPoint3Dgdouble *ray_begin,
                                             const GVolPoint3Dgdouble *ray_end,
                                             GVolRenderingRgn *rgn,
                                             guint32 rgn_x,
                                             guint32 rgn_y);
```

Cast a ray through *volume*, storing the result in the specified coordinate of *rgn*.

Note: for better accuracy, *ray_begin* and *ray_end* should be as near as possible to the line determined by *eye_pos* and *dir*.

volume : a GVolVolume

eye_pos : the eye position

dir : the ray direction

ray_begin : a GVolPoint3Dgdouble [p 165] from which the rendering will begin

ray_end : a GVolPoint3Dgdouble [p 165] that the ray must not overtake

rgn : the GVolRenderingRgn [p 184] to store the ray traversing result

rgn_x : the X coordinate of *rgn* to be used

rgn_y : the Y coordinate of *rgn* to be used

gvol_volume_cast_ray_local ()

```
void        gvol_volume_cast_ray_local      (const GVolVolume *volume,
                                             const GVolPoint3Dgdouble *eye_pos,
                                             const GVolVector3Dgdouble *dir,
                                             const GVolPoint3Dgdouble *ray_begin,
                                             const GVolPoint3Dgdouble *ray_end,
                                             GVolRenderingRgn *rgn,
                                             guint32 rgn_x,
                                             guint32 rgn_y);
```

Cast a ray through *volume*, storing the result in the specified coordinate of *rgn*. This function assumes that *eye_pos*, *dir* and *ray_end* are expressed in local volume coordinates.

Note: for better accuracy, *ray_begin* and *ray_end* should be as near as possible to the line determined by *eye_pos* and *dir*.

volume :	a GVolVolume
eye_pos :	the eye position (voxels at the back of it won't be rendered)
dir :	the ray direction
ray_begin :	a GVolPoint3Dgdouble [p 165] from which the rendering will begin
ray_end :	a GVolPoint3Dgdouble [p 165] that the ray must not overtake
rgn :	the GVolRenderingRgn [p 184] to store the ray traversing result
rgn_x :	the X coordinate of *rgn* to be used
rgn_y :	the Y coordinate of *rgn* to be used

gvol_volume_get_coord_sys ()

```
void        gvol_volume_get_coord_sys       (const GVolVolume *volume,
                                             GVolPoint3Dgdouble *o,
                                             GVolVector3Dgdouble *x,
                                             GVolVector3Dgdouble *y,
                                             GVolVector3Dgdouble *z);
```

Get the volume local coordinate system.

Note: *o*, *x*, *y* and *z* may be `NULL` if you don't want to retrieve them.

volume : a GVolVolume;

o : a GVolPoint3D to store the origin of the coordinate system

x : a GVolVector3Dgdouble [p 186] to store the volume X axis

y : a GVolVector3Dgdouble [p 186] to store the volume Y axis

z : a GVolVector3Dgdouble [p 186] to store the volume Z axis

---

gvol_volume_get_dimensions ()

```
void        gvol_volume_get_dimensions      (const GVolVolume *volume,
                                              guint32 *x_dim,
                                              guint32 *y_dim,
                                              guint32 *z_dim);
```

Get the X, Y and/or Z dimensions of *volume* (i. e. the number of voxels), storing them in *x_dim*, *y_dim* and/or *z_dim*.

NOTE: if *x_dim*, *y_dim* or *z_dim* are `NULL`, the corresponding number of voxels won't be retrieved.

volume : a GVolVolume

x_dim : a guint32 pointer used to store the X dimension of *volume*

y_dim : a guint32 pointer used to store the Y dimension of *volume*

z_dim : a guint32 pointer used to store the Z dimension of *volume*

<< **GVolVector3Dgfloat** [p 191] **GVolVolumeguint16** >>

GVolVolumeRGBAgfloat

GVolVolumeRGBAgfloat —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolVolumeRGBAgfloat;
GVolVolume* gvol_volume_rgba_gfloat_new     (void);
void        gvol_volume_set_coord_sys       (GVolVolume *volume,
                                             const GVolPoint3Dgdouble *o,
                                             const GVolVector3Dgdouble *x,
                                             const GVolVector3Dgdouble *y,
                                             const GVolVector3Dgdouble *z);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolVolume
               +----GVolVolumeRGBAgfloat
```

Properties

```
  "dataset"              GVolContainer3DRawRGBAgfloat : Read / Write
  "interpolator"         GVolInterp           : Read / Write
  "z-sampling"           gdouble              : Read / Write
```

Description

Details

struct GVolVolumeRGBAgfloat

```
struct GVolVolumeRGBAgfloat;
```

---

gvol_volume_rgba_gfloat_new ()

```
GVolVolume* gvol_volume_rgba_gfloat_new     (void);
```

Create a new GVolVolumeRGBAgfloat.

Returns : a new GVolVolumeRGBAgfloat Note: the new volume will be empty (i. e. no voxels inside), will be centered at the origin of the world reference system, and its bounding box will extend from (-1, -1, -1) to (1, 1, 1).

---

gvol_volume_set_coord_sys ()

```
void        gvol_volume_set_coord_sys       (GVolVolume *volume,
                                             const GVolPoint3Dgdouble *o,
                                             const GVolVector3Dgdouble *x,
                                             const GVolVector3Dgdouble *y,
                                             const GVolVector3Dgdouble *z);
```

Set the new volume local coordinate system.

volume : a GVolVolume;

o : a GVolPoint3D with the new origin of the coordinate system

x : a GVolVector3Dgdouble [p 186] with the new X axis

y : a GVolVector3Dgdouble [p 186] with the new Y axis

z : a GVolVector3Dgdouble [p 186] with the new Z axis

Properties

"dataset" (GVolContainer3DRawRGBAgfloat : Read / Write)	The dataset associated with the volume.
"interpolator" (GVolInterp : Read / Write)	The interpolator to be used with colors.
"z-sampling" (gdouble : Read / Write)	Depth oversampling factor.

<< GVolVolumeguint16

GVoFunc

GVoFunc —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolFunc;
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolFunc
```

Description

Details

struct GVolFunc

```
struct GVolFunc;
```

GVolAABBgdouble

GVolAABBgdouble —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolAABBgdouble;
GVolAABBgdouble* gvol_aabb_gdouble_copy     (const GVolAABBgdouble *bbox);
void        gvol_aabb_gdouble_copy2         (GVolAABBgdouble *dest,
                                             const GVolAABBgdouble *bbox);
void        gvol_aabb_gdouble_free          (GVolAABBgdouble *bbox);
gboolean    gvol_aabb_gdouble_intersect     (const GVolAABBgdouble *bbox,
                                             const GVolPoint3Dgdouble *point,
                                             const GVolVector3Dgdouble *vector,
                                             GVolPoint3Dgdouble *intr);
gboolean    gvol_aabb_gdouble_is_inside     (const GVolAABBgdouble *bbox,
                                             const GVolPoint3Dgdouble *point);
```

Description

Details

struct GVolAABBgdouble

```
struct GVolAABBgdouble {

  GVolPoint3Dgdouble min;
  GVolPoint3Dgdouble max;
};
```

---

gvol_aabb_gdouble_copy ()

```
GVolAABBgdouble* gvol_aabb_gdouble_copy     (const GVolAABBgdouble *bbox);
```

Copy a GVolAABBgdouble [p 142] .

bbox :	a bounding box to be copied.
Returns :	a copy of *bbox*.

---

gvol_aabb_gdouble_copy2 ()

```
void        gvol_aabb_gdouble_copy2         (GVolAABBgdouble *dest,
                                             const GVolAABBgdouble *bbox);
```

Copy the content of *bbox* to *dest* (that will be overwritten).

dest : a destination GVolAABBgdouble [p 142]

bbox : the GVolAABBgdouble [p 142] to be copied

---

gvol_aabb_gdouble_free ()

```
void        gvol_aabb_gdouble_free          (GVolAABBgdouble *bbox);
```

Free a GVolAABBgdouble [p 142] .

bbox : a bounding box to be freed.

---

gvol_aabb_gdouble_intersect ()

```
gboolean    gvol_aabb_gdouble_intersect     (const GVolAABBgdouble *bbox,
                                             const GVolPoint3Dgdouble *point,
                                             const GVolVector3Dgdouble *vector,
                                             GVolPoint3Dgdouble *intr);
```

bbox :

point :

vector :

intr :

Returns :

---

gvol_aabb_gdouble_is_inside ()

```
gboolean    gvol_aabb_gdouble_is_inside     (const GVolAABBgdouble *bbox,
                                             const GVolPoint3Dgdouble *point);
```

Check wether *point* is inside *bbox*.

bbox : a GVolAABBgdouble [p 142]

point : a GVolPoint3Dgdouble

Returns : TRUE if *point* is inside *bbox* (or on one of its faces), FALSE otherwise.

<< **GVol API reference** **GVolArray1Dgfloat** >>

GVolFunc_gfloat_gint32

GVolFunc_gfloat_gint32 —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolFunc_gfloat__gint32;
gfloat      gvol_func_gfloat__gint32_eval   (GVolFunc_gfloat__gint32 *func,
                                             gint32 x);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolFunc
               +----GVolFunc_gfloat__gint32
```

Description

Details

struct GVolFunc_gfloat__gint32

```
struct GVolFunc_gfloat__gint32;
```

---

gvol_func_gfloat__gint32_eval ()

```
gfloat      gvol_func_gfloat__gint32_eval   (GVolFunc_gfloat__gint32 *func,
                                             gint32 x);
```

Apply *func* to *x*, getting the result.

func :	a GVolFunc_gfloat__gint32 [p 145]
x :	the the argument for *func*
Returns :	the result of applync *func* to *x*.

GVolLUT_gfloat_gint32

GVolLUT_gfloat_gint32 —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolLUT_gfloat__gint32;
guint32     gvol_lut_gfloat__gint32_get_dimension
                                        (const GVolLUT_gfloat__gint32 *l);
gint32      gvol_lut_gfloat__gint32_get_min_index
                                        (const GVolLUT_gfloat__gint32 *l);
void        gvol_lut_gfloat__gint32_get_value_range
                                        (const GVolLUT_gfloat__gint32 *l,
                                         gfloat *val_min,
                                         gfloat *val_max);
GVolContainer1Dgfloat* gvol_lut_gfloat__gint32_get_values_container
                                        (const GVolLUT_gfloat__gint32 *l);
GVolFunc*   gvol_lut_gfloat__gint32_new     (GVolContainer1Dgfloat *values);
void        gvol_lut_gfloat__gint32_set_min_index
                                        (GVolLUT_gfloat__gint32 *l,
                                         gint32 min_index);
void        gvol_lut_gfloat__gint32_set_value_range
                                        (GVolLUT_gfloat__gint32 *l,
                                         gfloat val_min,
                                         gfloat val_max);
```

Object Hierarchy

```
  GObject
   +----GVolObject
         +----GVolFunc
               +----GVolFunc_gfloat__gint32
                     +----GVolLUT_gfloat__gint32
```

Description

Details

struct GVolLUT_gfloat__gint32

```
struct GVolLUT_gfloat__gint32;
```

gvol_lut_gfloat__gint32_get_dimension ()

```
guint32     gvol_lut_gfloat__gint32_get_dimension
                                        (const GVolLUT_gfloat__gint32 *l);
```

Get the dimension of *lut*.

l :

Returns : the dimension of *lut*.

gvol_lut_gfloat__gint32_get_min_index ()

```
gint32      gvol_lut_gfloat__gint32_get_min_index
                                        (const GVolLUT_gfloat__gint32 *l);
```

Get the minimum index of *lut* --- i. e. the value that, when passed to the transfer function corresponding to *lut*, will retrieve the first element.

l :

Returns : the minimum index of *lut*

gvol_lut_gfloat__gint32_get_value_range ()

```
void        gvol_lut_gfloat__gint32_get_value_range
                                        (const GVolLUT_gfloat__gint32 *l,
                                         gfloat *val_min,
                                         gfloat *val_max);
```

Get the minimum and maximum value that could be carried by *lut*. For more information, see gvol_lut_gfloat__gint32_set_value_range [p 149] ().

Note: *val_min* and *val_max* could be NULL, if you are not interested in the correspoinding value.

l :

val_min : a pointer to store the minimum value that could be carried by *lut*

val_max :

gvol_lut_gfloat__gint32_get_values_container ()

```
GVolContainer1Dgfloat* gvol_lut_gfloat__gint32_get_values_container
                                            (const GVolLUT_gfloat__gint32 *l);
```

Get the container used by lut to store its values.

l :

Returns : the GVolContainer1Dgfloat used to store the *lut* values

---

gvol_lut_gfloat__gint32_new ()

```
GVolFunc*   gvol_lut_gfloat__gint32_new     (GVolContainer1Dgfloat *values);
```

Create a new GVolLUT_gfloat__gint32 [p 146] with the specified dimension.

values : the container for the values of *lut*.

Returns : a new GVolLUT_gfloat__gint32 [p 146] (casted to GVolFunc).

---

gvol_lut_gfloat__gint32_set_min_index ()

```
void        gvol_lut_gfloat__gint32_set_min_index
                                            (GVolLUT_gfloat__gint32 *l,
                                             gint32 min_index);
```

Set the minimum index of *lut* --- i. e. the value that, when passed to the transfer function corresponding to *lut*, will retrieve the first element.

l :

min_index : the minimum index of *lut*

---

gvol_lut_gfloat__gint32_set_value_range ()

```
void        gvol_lut_gfloat__gint32_set_value_range
                                            (GVolLUT_gfloat__gint32 *l,
                                             gfloat val_min,
                                             gfloat val_max);
```

Set the minimum and maximum value that could be carried by *lut*. After calling this function, elements retrieved from lut will be clamped inside the (*val_min*, *val_max*) interval.

l :

val_min : the minimum value that could be carried by *lut*

val_max :

<< **GVolInterpGSLLinear** **GVolMatrix3x3x3gdouble** >> [p 151]

GVolMatrix3x3x3gdouble

GVolMatrix3x3x3gdouble —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolMatrix3x3x3gdouble;
GVolMatrix3x3x3gdouble* gvol_matrix3x3x3_gdouble_copy
                                          (const GVolMatrix3x3x3gdouble *m);
void        gvol_matrix3x3x3_gdouble_copy2  (GVolMatrix3x3x3gdouble *dest,
                                           const GVolMatrix3x3x3gdouble *m);
void        gvol_matrix3x3x3_gdouble_free   (GVolMatrix3x3x3gdouble *m);
void        gvol_matrix3x3x3_gdouble_zero   (GVolMatrix3x3x3gdouble *m);
```

Description

Details

struct GVolMatrix3x3x3gdouble

```
struct GVolMatrix3x3x3gdouble {

  gdouble el[3][3][3]; /* Matrix elements */
};
```

---

gvol_matrix3x3x3_gdouble_copy ()

```
GVolMatrix3x3x3gdouble* gvol_matrix3x3x3_gdouble_copy
                                          (const GVolMatrix3x3x3gdouble *m);
```

Copy a GVolMatrix3x3x3gdouble [p 150] .

m :	a matrix to be copied.
Returns :	a copy of *m*.

---

gvol_matrix3x3x3_gdouble_copy2 ()

```
void        gvol_matrix3x3x3_gdouble_copy2  (GVolMatrix3x3x3gdouble *dest,
                                             const GVolMatrix3x3x3gdouble *m);
```

Copy the content of *m* to *dest* (that will be overwritten).

dest : a destination matrix

m : the matrix to be copied

---

gvol_matrix3x3x3_gdouble_free ()

```
void        gvol_matrix3x3x3_gdouble_free   (GVolMatrix3x3x3gdouble *m);
```

Free a GVolMatrix3x3x3gdouble [p 150] .

m : a matrix to be freed.

---

gvol_matrix3x3x3_gdouble_zero ()

```
void        gvol_matrix3x3x3_gdouble_zero   (GVolMatrix3x3x3gdouble *m);
```

Set all the elements of *m* to 0.0.

m : the matrix to be zero'ed

<< **GVolLUT_gfloat_gint32** **GVolMatrix3x3x3gfloat** >> [p 153]

GVolMatrix3x3x3gfloat

GVolMatrix3x3x3gfloat —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolMatrix3x3x3gfloat;
GVolMatrix3x3x3gfloat* gvol_matrix3x3x3_gfloat_copy
                                            (const GVolMatrix3x3x3gfloat *m);
void        gvol_matrix3x3x3_gfloat_copy2   (GVolMatrix3x3x3gfloat *dest,
                                             const GVolMatrix3x3x3gfloat *m);
void        gvol_matrix3x3x3_gfloat_free    (GVolMatrix3x3x3gfloat *m);
void        gvol_matrix3x3x3_gfloat_zero    (GVolMatrix3x3x3gfloat *m);
```

Description

Details

struct GVolMatrix3x3x3gfloat

```
struct GVolMatrix3x3x3gfloat {

  gfloat el[3][3][3]; /* Matrix elements */
};
```

---

gvol_matrix3x3x3_gfloat_copy ()

```
GVolMatrix3x3x3gfloat* gvol_matrix3x3x3_gfloat_copy
                                            (const GVolMatrix3x3x3gfloat *m);
```

Copy a GVolMatrix3x3x3gfloat [p 152] .

m :	a matrix to be copied.
Returns :	a copy of *m*.

---

gvol_matrix3x3x3_gfloat_copy2 ()

```
void        gvol_matrix3x3x3_gfloat_copy2   (GVolMatrix3x3x3gfloat *dest,
                                             const GVolMatrix3x3x3gfloat *m);
```

Copy the content of *m* to *dest* (that will be overwritten).

dest : a destination matrix

m : the matrix to be copied

---

gvol_matrix3x3x3_gfloat_free ()

```
void        gvol_matrix3x3x3_gfloat_free    (GVolMatrix3x3x3gfloat *m);
```

Free a GVolMatrix3x3x3gfloat [p 152] .

m : a matrix to be freed.

---

gvol_matrix3x3x3_gfloat_zero ()

```
void        gvol_matrix3x3x3_gfloat_zero    (GVolMatrix3x3x3gfloat *m);
```

Set all the elements of *m* to 0.0.

m : the matrix to be zero'ed

<< **GVolMatrix3x3x3gdouble** **GVolMatrix4x4gdouble** >> [p 155]

GVolMatrix4x4gdouble

GVolMatrix4x4gdouble —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolMatrix4x4gdouble;
GVolMatrix4x4gdouble* gvol_matrix4x4_gdouble_copy
                                        (const GVolMatrix4x4gdouble *m);
void        gvol_matrix4x4_gdouble_copy2    (GVolMatrix4x4gdouble *dest,
                                         const GVolMatrix4x4gdouble *m);
void        gvol_matrix4x4_gdouble_free     (GVolMatrix4x4gdouble *m);
void        gvol_matrix4x4_gdouble_gen_coord_transf
                                        (GVolMatrix4x4gdouble *transf,
                                         GVolMatrix4x4gdouble *inv_transf,
                                         const GVolPoint3Dgdouble *o,
                                         const GVolVector3Dgdouble *x,
                                         const GVolVector3Dgdouble *y,
                                         const GVolVector3Dgdouble *z);
void        gvol_matrix4x4_gdouble_gen_rotation
                                        (GVolMatrix4x4gdouble *m,
                                         const GVolVector3Dgdouble *v,
                                         const GVolPoint3Dgdouble *point,
                                         gdouble cos_angle,
                                         gdouble sin_angle);
void        gvol_matrix4x4_gdouble_identity (GVolMatrix4x4gdouble *m);
void        gvol_matrix4x4_gdouble_invert   (GVolMatrix4x4gdouble *inv,
                                         const GVolMatrix4x4gdouble *mat);
void        gvol_matrix4x4_gdouble_mult     (GVolMatrix4x4gdouble *dest,
                                         const GVolMatrix4x4gdouble *m1,
                                         const GVolMatrix4x4gdouble *m2);
void        gvol_matrix4x4_gdouble_scale    (GVolMatrix4x4gdouble *m,
                                         gdouble scalar);
void        gvol_matrix4x4_gdouble_transpose
                                        (GVolMatrix4x4gdouble *transp,
                                         const GVolMatrix4x4gdouble *mat);
void        gvol_matrix4x4_gdouble_zero     (GVolMatrix4x4gdouble *m);
```

Description

Details

struct GVolMatrix4x4gdouble

```
struct GVolMatrix4x4gdouble {

  gdouble el[4][4]; /* Matrix elements */
};
```

---

gvol_matrix4x4_gdouble_copy ()

```
GVolMatrix4x4gdouble* gvol_matrix4x4_gdouble_copy
                                          (const GVolMatrix4x4gdouble *m);
```

Copy a GVolMatrix4x4gdouble [p 155] .

m : a matrix to be copied.

Returns : a copy of *m*.

---

gvol_matrix4x4_gdouble_copy2 ()

```
void        gvol_matrix4x4_gdouble_copy2    (GVolMatrix4x4gdouble *dest,
                                             const GVolMatrix4x4gdouble *m);
```

Copy the content of *m* to *dest* (that will be overwritten).

dest : a destination matrix

m : the matrix to be copied

---

gvol_matrix4x4_gdouble_free ()

```
void        gvol_matrix4x4_gdouble_free     (GVolMatrix4x4gdouble *m);
```

Free a GVolMatrix4x4gdouble [p 155] .

m : a matrix to be freed.

---

gvol_matrix4x4_gdouble_gen_coord_transf ()

```
void        gvol_matrix4x4_gdouble_gen_coord_transf
                                          (GVolMatrix4x4gdouble *transf,
                                           GVolMatrix4x4gdouble *inv_transf,
```

```
                                                const GVolPoint3Dgdouble *o,
                                                const GVolVector3Dgdouble *x,
                                                const GVolVector3Dgdouble *y,
                                                const GVolVector3Dgdouble *z);
```

Generate the direct and inverse transformation matrix from one coordinate system to a new one.

transf :	the transformation matrix that will be computed
inv_transf :	the inverse transformation matrix that will be computed
o :	the origin of the new reference system
x :	the X axis of the new reference system
y :	the Y axis of the new reference system
z :	the Z axis of thie new reference system

---

gvol_matrix4x4_gdouble_gen_rotation ()

```
void        gvol_matrix4x4_gdouble_gen_rotation
                                              (GVolMatrix4x4gdouble *m,
                                               const GVolVector3Dgdouble *v,
                                               const GVolPoint3Dgdouble *point,
                                               gdouble cos_angle,
                                               gdouble sin_angle);
```

Make *m* a transformation matrix which performs a rotation around *v* (with *v* applied on *point*).

m :	the target matrix
v :	the rotation vector (should be normalized);
point :	the point in which *v* is applied
cos_angle :	the rotation angle cosine
sin_angle :	the rotation angle sine

---

gvol_matrix4x4_gdouble_identity ()

```
void        gvol_matrix4x4_gdouble_identity (GVolMatrix4x4gdouble *m);
```

Make *m* an identity matrix.

m :	the matrix to set to identity

gvol_matrix4x4_gdouble_invert ()

```
void        gvol_matrix4x4_gdouble_invert   (GVolMatrix4x4gdouble *inv,
                                             const GVolMatrix4x4gdouble *mat);
```

Compute the inverse of *mat*, storing the result in *inv*.

inv : the inverse matrix that will be computed

mat :

gvol_matrix4x4_gdouble_mult ()

```
void        gvol_matrix4x4_gdouble_mult     (GVolMatrix4x4gdouble *dest,
                                             const GVolMatrix4x4gdouble *m1,
                                             const GVolMatrix4x4gdouble *m2);
```

Multiply *m1* by *m2*, storing the result in *dest*.

dest : a GVolMatrix4x4gdouble [p 155] to store the multiplication result

m1 : a GVolMatrix4x4gdouble [p 155]

m2 : a GVolMatrix4x4gdouble [p 155]

gvol_matrix4x4_gdouble_scale ()

```
void        gvol_matrix4x4_gdouble_scale    (GVolMatrix4x4gdouble *m,
                                             gdouble scalar);
```

Multiply every element of *m* by *scalar*.

m : a GVolMatrix4x4gdouble [p 155]

scalar : a scalar value

gvol_matrix4x4_gdouble_transpose ()

```
void        gvol_matrix4x4_gdouble_transpose
                                            (GVolMatrix4x4gdouble *transp,
                                             const GVolMatrix4x4gdouble *mat);
```

Transpose *mat*, storing the result in *transp*.

transp : the transposed matrix that will be computed

mat : the GVolMatrix4x4gdouble [p 155] matrix to be transposed

---

gvol_matrix4x4_gdouble_zero ()

```
void        gvol_matrix4x4_gdouble_zero     (GVolMatrix4x4gdouble *m);
```

Set all the elements of *m* to 0.0.

m : the matrix to be zero'ed

<< **GVolMatrix3x3x3gfloat** **GVolMatrix4x4gfloat** >> [p 160]

GVolMatrix4x4gfloat

GVolMatrix4x4gfloat —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolMatrix4x4gfloat;
GVolMatrix4x4gfloat* gvol_matrix4x4_gfloat_copy
                                        (const GVolMatrix4x4gfloat *m);
void        gvol_matrix4x4_gfloat_copy2     (GVolMatrix4x4gfloat *dest,
                                         const GVolMatrix4x4gfloat *m);
void        gvol_matrix4x4_gfloat_free      (GVolMatrix4x4gfloat *m);
void        gvol_matrix4x4_gfloat_gen_coord_transf
                                        (GVolMatrix4x4gfloat *transf,
                                         GVolMatrix4x4gfloat *inv_transf,
                                         const GVolPoint3Dgfloat *o,
                                         const GVolVector3Dgfloat *x,
                                         const GVolVector3Dgfloat *y,
                                         const GVolVector3Dgfloat *z);
void        gvol_matrix4x4_gfloat_gen_rotation
                                        (GVolMatrix4x4gfloat *m,
                                         const GVolVector3Dgfloat *v,
                                         const GVolPoint3Dgfloat *point,
                                         gfloat cos_angle,
                                         gfloat sin_angle);
void        gvol_matrix4x4_gfloat_identity  (GVolMatrix4x4gfloat *m);
void        gvol_matrix4x4_gfloat_invert    (GVolMatrix4x4gfloat *inv,
                                         const GVolMatrix4x4gfloat *mat);
void        gvol_matrix4x4_gfloat_mult      (GVolMatrix4x4gfloat *dest,
                                         const GVolMatrix4x4gfloat *m1,
                                         const GVolMatrix4x4gfloat *m2);
void        gvol_matrix4x4_gfloat_scale     (GVolMatrix4x4gfloat *m,
                                         gfloat scalar);
void        gvol_matrix4x4_gfloat_transpose (GVolMatrix4x4gfloat *transp,
                                         const GVolMatrix4x4gfloat *mat);
void        gvol_matrix4x4_gfloat_zero      (GVolMatrix4x4gfloat *m);
```

Description

Details

struct GVolMatrix4x4gfloat

```
struct GVolMatrix4x4gfloat {

  gfloat el[4][4]; /* Matrix elements */
};
```

---

gvol_matrix4x4_gfloat_copy ()

```
GVolMatrix4x4gfloat* gvol_matrix4x4_gfloat_copy
                                          (const GVolMatrix4x4gfloat *m);
```

Copy a GVolMatrix4x4gfloat [p 160] .

m : a matrix to be copied.

Returns : a copy of *m*.

---

gvol_matrix4x4_gfloat_copy2 ()

```
void        gvol_matrix4x4_gfloat_copy2     (GVolMatrix4x4gfloat *dest,
                                             const GVolMatrix4x4gfloat *m);
```

Copy the content of *m* to *dest* (that will be overwritten).

dest : a destination matrix

m : the matrix to be copied

---

gvol_matrix4x4_gfloat_free ()

```
void        gvol_matrix4x4_gfloat_free      (GVolMatrix4x4gfloat *m);
```

Free a GVolMatrix4x4gfloat [p 160] .

m : a matrix to be freed.

---

gvol_matrix4x4_gfloat_gen_coord_transf ()

```
void        gvol_matrix4x4_gfloat_gen_coord_transf
                                          (GVolMatrix4x4gfloat *transf,
                                           GVolMatrix4x4gfloat *inv_transf,
```

```
                                                const GVolPoint3Dgfloat *o,
                                                const GVolVector3Dgfloat *x,
                                                const GVolVector3Dgfloat *y,
                                                const GVolVector3Dgfloat *z);
```

Generate the direct and inverse transformation matrix from one coordinate system to a new one.

transf :	the transformation matrix that will be computed
inv_transf :	the inverse transformation matrix that will be computed
o :	the origin of the new reference system
x :	the X axis of the new reference system
y :	the Y axis of the new reference system
z :	the Z axis of thie new reference system

---

gvol_matrix4x4_gfloat_gen_rotation ()

```
void        gvol_matrix4x4_gfloat_gen_rotation
                                              (GVolMatrix4x4gfloat *m,
                                               const GVolVector3Dgfloat *v,
                                               const GVolPoint3Dgfloat *point,
                                               gfloat cos_angle,
                                               gfloat sin_angle);
```

Make *m* a transformation matrix which performs a rotation around *v* (with *v* applied on *point*).

m :	the target matrix
v :	the rotation vector (should be normalized);
point :	the point in which *v* is applied
cos_angle :	the rotation angle cosine
sin_angle :	the rotation angle sine

---

gvol_matrix4x4_gfloat_identity ()

```
void        gvol_matrix4x4_gfloat_identity  (GVolMatrix4x4gfloat *m);
```

Make *m* an identity matrix.

m :	the matrix to set to identity

gvol_matrix4x4_gfloat_invert ()

```
void        gvol_matrix4x4_gfloat_invert    (GVolMatrix4x4gfloat *inv,
                                             const GVolMatrix4x4gfloat *mat);
```

Compute the inverse of *mat*, storing the result in *inv*.

inv : the inverse matrix that will be computed

mat :

gvol_matrix4x4_gfloat_mult ()

```
void        gvol_matrix4x4_gfloat_mult      (GVolMatrix4x4gfloat *dest,
                                             const GVolMatrix4x4gfloat *m1,
                                             const GVolMatrix4x4gfloat *m2);
```

Multiply *m1* by *m2*, storing the result in *dest*.

dest : a GVolMatrix4x4gfloat [p 160] to store the multiplication result

m1 : a GVolMatrix4x4gfloat [p 160]

m2 : a GVolMatrix4x4gfloat [p 160]

gvol_matrix4x4_gfloat_scale ()

```
void        gvol_matrix4x4_gfloat_scale     (GVolMatrix4x4gfloat *m,
                                             gfloat scalar);
```

Multiply every element of *m* by *scalar*.

m : a GVolMatrix4x4gfloat [p 160]

scalar : a scalar value

gvol_matrix4x4_gfloat_transpose ()

```
void        gvol_matrix4x4_gfloat_transpose (GVolMatrix4x4gfloat *transp,
                                             const GVolMatrix4x4gfloat *mat);
```

Transpose *mat*, storing the result in *transp*.

transp : the transposed matrix that will be computed

mat : the GVolMatrix4x4gfloat [p 160] matrix to be transposed

---

gvol_matrix4x4_gfloat_zero ()

```
void        gvol_matrix4x4_gfloat_zero      (GVolMatrix4x4gfloat *m);
```

Set all the elements of *m* to 0.0.

m : the matrix to be zero'ed

<< **GVolMatrix4x4gdouble** **GVolObject** >>

GVolPoint3Dgdouble

GVolPoint3Dgdouble —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolPoint3Dgdouble;
void        gvol_point3d_gdouble_add        (GVolPoint3Dgdouble *res,
                                             const GVolPoint3Dgdouble *p,
                                             const GVolVector3Dgdouble *v);
void        gvol_point3d_gdouble_add_self   (GVolPoint3Dgdouble *p,
                                             const GVolVector3Dgdouble *v);
GVolPoint3Dgdouble* gvol_point3d_gdouble_copy
                                            (const GVolPoint3Dgdouble *p);
void        gvol_point3d_gdouble_copy2      (GVolPoint3Dgdouble *dest,
                                             const GVolPoint3Dgdouble *p);
void        gvol_point3d_gdouble_free       (GVolPoint3Dgdouble *v);
gdouble     gvol_point3d_gdouble_get_2_dist (const GVolPoint3Dgdouble *p1,
                                             const GVolPoint3Dgdouble *p2);
void        gvol_point3d_gdouble_mult       (GVolPoint3Dgdouble *res,
                                             const GVolMatrix4x4gdouble *m,
                                             const GVolPoint3Dgdouble *v);
```

Description

Details

struct GVolPoint3Dgdouble

```
struct GVolPoint3Dgdouble {

  gdouble x;
  gdouble y;
  gdouble z;
  gdouble w;
};
```

---

gvol_point3d_gdouble_add ()

```
void        gvol_point3d_gdouble_add        (GVolPoint3Dgdouble *res,
                                             const GVolPoint3Dgdouble *p,
                                             const GVolVector3Dgdouble *v);
```

Add *v* to *p*, storing the result in *res*.

res : a GVolPoint3Dgdouble [p 164] in which the addition result will be stored

p : a GVolPoint3Dgdouble [p 164]

v : a GVolVector3Dgdouble [p 186] to be added to *p*

---

gvol_point3d_gdouble_add_self ()

```
void        gvol_point3d_gdouble_add_self   (GVolPoint3Dgdouble *p,
                                              const GVolVector3Dgdouble *v);
```

Add *v* to *p*, storing the result in *p* itself.

p : a GVolPoint3Dgdouble [p 164]

v : a GVolVector3Dgdouble [p 186] to be added to *p*

---

gvol_point3d_gdouble_copy ()

```
GVolPoint3Dgdouble* gvol_point3d_gdouble_copy
                                            (const GVolPoint3Dgdouble *p);
```

Copy a GVolPoint3Dgdouble [p 164] .

p : a point to be copied.

Returns : a copy of *v*.

---

gvol_point3d_gdouble_copy2 ()

```
void        gvol_point3d_gdouble_copy2      (GVolPoint3Dgdouble *dest,
                                              const GVolPoint3Dgdouble *p);
```

Copy the content of *v* to *dest* (that will be overwritten).

dest : a destination point

p : the point to be copied

---

gvol_point3d_gdouble_free ()

```
void        gvol_point3d_gdouble_free       (GVolPoint3Dgdouble *v);
```

Free a GVolPoint3Dgdouble [p 164] .

v :

---

gvol_point3d_gdouble_get_2_dist ()

```
gdouble     gvol_point3d_gdouble_get_2_dist (const GVolPoint3Dgdouble *p1,
                                             const GVolPoint3Dgdouble *p2);
```

Get the distance between *p1* and *p2*, using the 2-norm (euclidean norm).

p1 : a GVolPoint3Dgdouble [p 164]

p2 : a GVolPoint3Dgdouble [p 164]

Returns : the distance between *p1* and *p2*.

---

gvol_point3d_gdouble_mult ()

```
void        gvol_point3d_gdouble_mult       (GVolPoint3Dgdouble *res,
                                             const GVolMatrix4x4gdouble *m,
                                             const GVolPoint3Dgdouble *v);
```

Multiply the matrix *m* by the point *v*, storing the result in *res*.

res : a GVolPoint3Dgdouble [p 164] to store the result

m : the multiplication matrix

v :

<< **GVolObject** **GVolPoint3Dgfloat** >> [p 168]

GVolPoint3Dgfloat

GVolPoint3Dgfloat —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolPoint3Dgfloat;
void        gvol_point3d_gfloat_add         (GVolPoint3Dgfloat *res,
                                             const GVolPoint3Dgfloat *p,
                                             const GVolVector3Dgfloat *v);
void        gvol_point3d_gfloat_add_self    (GVolPoint3Dgfloat *p,
                                             const GVolVector3Dgfloat *v);
GVolPoint3Dgfloat* gvol_point3d_gfloat_copy (const GVolPoint3Dgfloat *p);
void        gvol_point3d_gfloat_copy2       (GVolPoint3Dgfloat *dest,
                                             const GVolPoint3Dgfloat *p);
void        gvol_point3d_gfloat_free        (GVolPoint3Dgfloat *v);
gfloat      gvol_point3d_gfloat_get_2_dist  (const GVolPoint3Dgfloat *p1,
                                             const GVolPoint3Dgfloat *p2);
void        gvol_point3d_gfloat_mult        (GVolPoint3Dgfloat *res,
                                             const GVolMatrix4x4gfloat *m,
                                             const GVolPoint3Dgfloat *v);
```

Description

Details

struct GVolPoint3Dgfloat

```
struct GVolPoint3Dgfloat {

  gfloat x;
  gfloat y;
  gfloat z;
  gfloat w;
};
```

---

gvol_point3d_gfloat_add ()

```
void        gvol_point3d_gfloat_add         (GVolPoint3Dgfloat *res,
                                             const GVolPoint3Dgfloat *p,
                                             const GVolVector3Dgfloat *v);
```

Add *v* to *p*, storing the result in `res`.

`res` : a GVolPoint3Dgfloat [p 167] in which the addition result will be stored

p : a GVolPoint3Dgfloat [p 167]

v : a GVolVector3Dgfloat [p 192] to be added to *p*

---

gvol_point3d_gfloat_add_self ()

```
void        gvol_point3d_gfloat_add_self    (GVolPoint3Dgfloat *p,
                                             const GVolVector3Dgfloat *v);
```

Add *v* to *p*, storing the result in *p* itself.

p : a GVolPoint3Dgfloat [p 167]

v : a GVolVector3Dgfloat [p 192] to be added to *p*

---

gvol_point3d_gfloat_copy ()

```
GVolPoint3Dgfloat* gvol_point3d_gfloat_copy (const GVolPoint3Dgfloat *p);
```

Copy a GVolPoint3Dgfloat [p 167] .

p : a point to be copied.

Returns : a copy of *v*.

---

gvol_point3d_gfloat_copy2 ()

```
void        gvol_point3d_gfloat_copy2       (GVolPoint3Dgfloat *dest,
                                             const GVolPoint3Dgfloat *p);
```

Copy the content of *v* to `dest` (that will be overwritten).

`dest` : a destination point

p : the point to be copied

---

gvol_point3d_gfloat_free ()

```
void        gvol_point3d_gfloat_free        (GVolPoint3Dgfloat *v);
```

Free a GVolPoint3Dgfloat [p 167] .

v :

---

gvol_point3d_gfloat_get_2_dist ()

```
gfloat      gvol_point3d_gfloat_get_2_dist  (const GVolPoint3Dgfloat *p1,
                                             const GVolPoint3Dgfloat *p2);
```

Get the distance between *p1* and *p2*, using the 2-norm (euclidean norm).

p1 : a GVolPoint3Dgfloat [p 167]

p2 : a GVolPoint3Dgfloat [p 167]

Returns : the distance between *p1* and *p2*.

---

gvol_point3d_gfloat_mult ()

```
void        gvol_point3d_gfloat_mult        (GVolPoint3Dgfloat *res,
                                             const GVolMatrix4x4gfloat *m,
                                             const GVolPoint3Dgfloat *v);
```

Multiply the matrix *m* by the point *v*, storing the result in *res*.

res : a GVolPoint3Dgfloat [p 167] to store the result

m : the multiplication matrix

v :

<< **GVolPoint3Dgdouble** **GVolRawRGBAgfloat** >> [p 171]

GVolRawRGBAgfloat

GVolRawRGBAgfloat —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolRawRGBAgfloat;
void        gvol_raw_rgba_gfloat_clamp      (GVolRawRGBAgfloat *color);
GVolRawRGBAgfloat* gvol_raw_rgba_gfloat_copy
                                            (const GVolRawRGBAgfloat *c);
void        gvol_raw_rgba_gfloat_copy2      (GVolRawRGBAgfloat *dest,
                                             const GVolRawRGBAgfloat *c);
void        gvol_raw_rgba_gfloat_free       (GVolRawRGBAgfloat *c);
void        gvol_raw_rgba_gfloat_get_rgba_guchar
                                            (const GVolRawRGBAgfloat *color,
                                             GVolRawRGBAguchar *rgb);
void        gvol_raw_rgba_gfloat_get_rgb_guchar
                                            (const GVolRawRGBAgfloat *color,
                                             GVolRawRGBguchar *rgb);
void        gvol_raw_rgba_gfloat_interp     (GVolRawRGBAgfloat *color,
                                             gdouble xv[],
                                             const GVolRawRGBAgfloat *const  colors[],
                                             guint num_colors,
                                             gdouble x,
                                             const GVolInterp *interp);
gboolean    gvol_raw_rgba_gfloat_is_valid   (GVolRawRGBAgfloat *rgba);
void        gvol_raw_rgba_gfloat_scale      (GVolRawRGBAgfloat *color,
                                             const GVolRawRGBAgfloat *color1,
                                             gdouble scalar);
void        gvol_raw_rgba_gfloat_set_rgba_guchar
                                            (GVolRawRGBAgfloat *color,
                                             const GVolRawRGBAguchar *rgba);
void        gvol_raw_rgba_gfloat_set_rgb_guchar
                                            (GVolRawRGBAgfloat *color,
                                             const GVolRawRGBguchar *rgb);
```

Description

Details

struct GVolRawRGBAgfloat

```
struct GVolRawRGBAgfloat {

  gfloat r;
  gfloat g;
  gfloat b;
  gfloat a;
};
```

---

gvol_raw_rgba_gfloat_clamp ()

```
void        gvol_raw_rgba_gfloat_clamp      (GVolRawRGBAgfloat *color);
```

color :

---

gvol_raw_rgba_gfloat_copy ()

```
GVolRawRGBAgfloat* gvol_raw_rgba_gfloat_copy
                                            (const GVolRawRGBAgfloat *c);
```

Copy a GVolRawRGBAgfloat [p 171] .

c : a color to be copied.

Returns : a copy of *c*.

---

gvol_raw_rgba_gfloat_copy2 ()

```
void        gvol_raw_rgba_gfloat_copy2      (GVolRawRGBAgfloat *dest,
                                             const GVolRawRGBAgfloat *c);
```

Copy the content of *c* to *dest* (that will be overwritten).

dest : a destination color

c : the color to be copied

---

gvol_raw_rgba_gfloat_free ()

```
void        gvol_raw_rgba_gfloat_free       (GVolRawRGBAgfloat *c);
```

Free a GVolRawRGBAgfloat [p 171] .

c : a color to be freed.

---

gvol_raw_rgba_gfloat_get_rgba_guchar ()

```
void        gvol_raw_rgba_gfloat_get_rgba_guchar
                                        (const GVolRawRGBAgfloat *color,
                                         GVolRawRGBAguchar *rgb);
```

Store in *rgba* the RGBA value associated with *color*.

Note: if *color* doesn't have an alpha channel, the returned alpha component will be set to 1.0.

color : a GVolRawRGBAgfloat [p 171]

rgb :

---

gvol_raw_rgba_gfloat_get_rgb_guchar ()

```
void        gvol_raw_rgba_gfloat_get_rgb_guchar
                                        (const GVolRawRGBAgfloat *color,
                                         GVolRawRGBguchar *rgb);
```

Store in *rgb* the RGB value associated with *color*.

Note: the alpha component of *color*, if any, will be used to scale the resulting R, G and B values.

color : a GVolRawRGBAgfloat [p 171]

rgb : the GVolRawRGBguchar [p 179] in which the value will be stored

---

gvol_raw_rgba_gfloat_interp ()

```
void        gvol_raw_rgba_gfloat_interp     (GVolRawRGBAgfloat *color,
                                             gdouble xv[],
                                             const GVolRawRGBAgfloat *const  colors[],
                                             guint num_colors,
                                             gdouble x,
                                             const GVolInterp *interp);
```

Make *color* the interpolation between the *num_colors* GVolRawRGBAgfloat [p 171] 's pointed by *colors*, using *interp* as interpolator.

color : a GVolRawRGBAgfloat [p 171]

xv : values on the X axis of the collocation points

colors : an array of pointers to GVolRawRGBAgfloat [p 171] 's to be interpolated

num_colors : number of colors in *colors*

x : the X coordinate in which the interpolation will be evaluated

interp : the GVolInterp to be used for interpolations

---

gvol_raw_rgba_gfloat_is_valid ()

```
gboolean    gvol_raw_rgba_gfloat_is_valid   (GVolRawRGBAgfloat *rgba);
```

Check whether *ptr* is a valid pointer to a GVolRawRGBAgfloat [p 171] structure (e. g. whether its elements are between 0.0 and 1.0).

Return values: TRUE if *ptr* seems to be valid, otherwise FALSE.

rgba : a GVolRawRGBAgfloat [p 171]

Returns :

---

gvol_raw_rgba_gfloat_scale ()

```
void        gvol_raw_rgba_gfloat_scale      (GVolRawRGBAgfloat *color,
                                             const GVolRawRGBAgfloat *color1,
                                             gdouble scalar);
```

Multiply *color1* by *scalar* (e. g. to simulate fog, distance or opacity attenuation), storing the result in *color*.

Note: the alpha component of *color* won't be scaled.

color : a GVolRawRGBAgfloat [p 171]

color1 : a GVolRawRGBAgfloat [p 171] to be scaled

scalar : a scalar for which *color1* will be multiplied (usually between 0.0 and 1.0)

---

gvol_raw_rgba_gfloat_set_rgba_guchar ()

```
void        gvol_raw_rgba_gfloat_set_rgba_guchar
                                        (GVolRawRGBAgfloat *color,
                                         const GVolRawRGBAguchar *rgba);
```

color :

rgba :

---

gvol_raw_rgba_gfloat_set_rgb_guchar ()

```
void        gvol_raw_rgba_gfloat_set_rgb_guchar
                                        (GVolRawRGBAgfloat *color,
                                         const GVolRawRGBguchar *rgb);
```

Set the RGB value of *color* with the values contained in *rgb*.

Note: if *color* doesn't have an alpha channel, the alpha component or *rgba* will be used to scale the R, G, and B values that will be read from *rgba* itself.

color : a GVolRawRGBAgfloat [p 171]

rgb :

<< **GVolPoint3Dgfloat** **GVolRawRGBAguchar** >> [p 176]

GVolRawRGBAguchar

GVolRawRGBAguchar —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolRawRGBAguchar;
GVolRawRGBAguchar* gvol_raw_rgba_guchar_copy
                                        (const GVolRawRGBAguchar *c);
void        gvol_raw_rgba_guchar_copy2      (GVolRawRGBAguchar *dest,
                                         const GVolRawRGBAguchar *c);
void        gvol_raw_rgba_guchar_free       (GVolRawRGBAguchar *c);
void        gvol_raw_rgba_guchar_interpolate
                                        (GVolRawRGBAguchar *color,
                                         const GVolRawRGBAguchar *color1,
                                         const GVolRawRGBAguchar *color2,
                                         gdouble color1_weight);
void        gvol_raw_rgba_guchar_scale      (GVolRawRGBAguchar *color,
                                         gdouble scalar);
```

Description

Details

struct GVolRawRGBAguchar

```
struct GVolRawRGBAguchar {

  guchar r;
  guchar g;
  guchar b;
  guchar a;
};
```

---

gvol_raw_rgba_guchar_copy ()

```
GVolRawRGBAguchar* gvol_raw_rgba_guchar_copy
                                        (const GVolRawRGBAguchar *c);
```

Copy a GVolRawRGBAguchar [p 175] .

c : a color to be copied.

Returns : a copy of *c*.

---

gvol_raw_rgba_guchar_copy2 ()

```
void        gvol_raw_rgba_guchar_copy2      (GVolRawRGBAguchar *dest,
                                             const GVolRawRGBAguchar *c);
```

Copy the content of *c* to *dest* (that will be overwritten).

dest : a destination color

c : the color to be copied

---

gvol_raw_rgba_guchar_free ()

```
void        gvol_raw_rgba_guchar_free       (GVolRawRGBAguchar *c);
```

Free a GVolRawRGBAguchar [p 175] .

c : a color to be freed.

---

gvol_raw_rgba_guchar_interpolate ()

```
void        gvol_raw_rgba_guchar_interpolate
                                            (GVolRawRGBAguchar *color,
                                             const GVolRawRGBAguchar *color1,
                                             const GVolRawRGBAguchar *color2,
                                             gdouble color1_weight);
```

color :

color1 :

color2 :

color1_weight :

---

gvol_raw_rgba_guchar_scale ()

```
void        gvol_raw_rgba_guchar_scale      (GVolRawRGBAguchar *color,
                                             gdouble scalar);
```

Multiply *color* by *scalar* (e. g. to simulate fog, distance or opacity attenuation).

Note: the alpha component of *color*, if any, won't be scaled.

color : a GVolRawRGBAguchar [p 175]

scalar : a scalar for which the color should me multiplied (usually between 0.0 and 1.0)

<< **GVolRawRGBAgfloat** **GVolRawRGBguchar** >> [p 179]

GVolRawRGBguchar

GVolRawRGBguchar —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolRawRGBguchar;
GVolRawRGBguchar* gvol_raw_rgb_guchar_copy  (const GVolRawRGBguchar *c);
void        gvol_raw_rgb_guchar_copy2       (GVolRawRGBguchar *dest,
                                             const GVolRawRGBguchar *c);
void        gvol_raw_rgb_guchar_free        (GVolRawRGBguchar *c);
void        gvol_raw_rgb_guchar_interpolate (GVolRawRGBguchar *color,
                                             const GVolRawRGBguchar *color1,
                                             const GVolRawRGBguchar *color2,
                                             gdouble color1_weight);
void        gvol_raw_rgb_guchar_scale       (GVolRawRGBguchar *color,
                                             gdouble scalar);
```

Description

Details

struct GVolRawRGBguchar

```
struct GVolRawRGBguchar {

  guchar r;
  guchar g;
  guchar b;
};
```

---

gvol_raw_rgb_guchar_copy ()

```
GVolRawRGBguchar* gvol_raw_rgb_guchar_copy  (const GVolRawRGBguchar *c);
```

Copy a GVolRawRGBguchar [p 178] .

c : a color to be copied.

Returns : a copy of *c*.

gvol_raw_rgb_guchar_copy2 ()

```
void        gvol_raw_rgb_guchar_copy2       (GVolRawRGBguchar *dest,
                                             const GVolRawRGBguchar *c);
```

Copy the content of *c* to *dest* (that will be overwritten).

dest : a destination color

c : the color to be copied

gvol_raw_rgb_guchar_free ()

```
void        gvol_raw_rgb_guchar_free        (GVolRawRGBguchar *c);
```

Free a GVolRawRGBguchar [p 178] .

c : a color to be freed.

gvol_raw_rgb_guchar_interpolate ()

```
void        gvol_raw_rgb_guchar_interpolate (GVolRawRGBguchar *color,
                                             const GVolRawRGBguchar *color1,
                                             const GVolRawRGBguchar *color2,
                                             gdouble color1_weight);
```

color :

color1 :

color2 :

color1_weight :

gvol_raw_rgb_guchar_scale ()

```
void        gvol_raw_rgb_guchar_scale       (GVolRawRGBguchar *color,
                                             gdouble scalar);
```

Multiply *color* by *scalar* (e. g. to simulate fog, distance or opacity attenuation).

Note: the alpha component of *color*, if any, won't be scaled.

color : a GVolRawRGBguchar [p 178]

scalar : a scalar for which the color should me multiplied (usually between 0.0 and 1.0)

<< **GVolRawRGBAguchar** **GVolRectanglegint32** >> [p 182]

GVolRectanglegint32

GVolRectanglegint32 —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolRectanglegint32;
GVolRectanglegint32* gvol_rectangle_gint32_copy
                                            (const GVolRectanglegint32 *rect);
void        gvol_rectangle_gint32_copy2     (GVolRectanglegint32 *dest,
                                             const GVolRectanglegint32 *rect);
void        gvol_rectangle_gint32_free      (GVolRectanglegint32 *rect);
gboolean    gvol_rectangle_gint32_is_inside (const GVolRectanglegint32 *rect,
                                             gint32 x,
                                             gint32 y);
```

Description

Details

struct GVolRectanglegint32

```
struct GVolRectanglegint32 {

  gint32 x; /* Coordinates of the upper-left corner */
  gint32 y;
  gint32 x_dim;
  gint32 y_dim;
};
```

---

gvol_rectangle_gint32_copy ()

```
GVolRectanglegint32* gvol_rectangle_gint32_copy
                                            (const GVolRectanglegint32 *rect);
```

Copy a GVolRectanglegint32 [p 181] .

rect : a rectangle to be copied.

Returns : a copy of *rect*.

gvol_rectangle_gint32_copy2 ()

```
void        gvol_rectangle_gint32_copy2     (GVolRectanglegint32 *dest,
                                             const GVolRectanglegint32 *rect);
```

Copy the content of *rect* to *dest* (that will be overwritten).

dest : a destination GVolRectanglegint32 [p 181]

rect : a GVolRectanglegint32 [p 181] to be copied

gvol_rectangle_gint32_free ()

```
void        gvol_rectangle_gint32_free      (GVolRectanglegint32 *rect);
```

Free a GVolRectanglegint32 [p 181] .

rect : a rectangle to be freed.

gvol_rectangle_gint32_is_inside ()

```
gboolean    gvol_rectangle_gint32_is_inside (const GVolRectanglegint32 *rect,
                                             gint32 x,
                                             gint32 y);
```

Check whether the point with coordinates (*x*, *y*) is inside *rect*.

rect : a GVolRectanglegint32 [p 181]

x : the X coordinate of the point to be checked

y : the Y coordinate of the point to be checked

Returns : TRUE if (*x*, *y*) is inside *rect*, FALSE otherwise.

<< **GVolRawRGBguchar** **GVolRenderingCtx** >>

GVolRenderingRgn

GVolRenderingRgn —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolRenderingRgn;
GVolRenderingRgn* gvol_rendering_rgn_copy   (const GVolRenderingRgn *rgn);
void        gvol_rendering_rgn_free         (GVolRenderingRgn *rgn);
gpointer    gvol_rendering_rgn_get_coord_ptr
                                            (GVolRenderingRgn *rgn,
                                             gint32 x,
                                             gint32 y);
void        gvol_rendering_rgn_release      (GVolRenderingRgn *rgn);
```

Description

Details

struct GVolRenderingRgn

```
struct GVolRenderingRgn {

  GVolRenderingCtx    *ctx; /* The rendering context that owns the region */
  GVolRectanglegint32  boundary; /* The boundary of the region */
  gpointer             ptr; /* The address of the region */

  /* The number of bytes to skip when incrementing the X or Y coordinate
     being accessed in the region */
  gint32               colstride;
  gint32               rowstride;
};
```

---

gvol_rendering_rgn_copy ()

```
GVolRenderingRgn* gvol_rendering_rgn_copy   (const GVolRenderingRgn *rgn);
```

Copy a GVolRenderingRgn [p 183] .

rgn : a rendering region information structure to be copied.

Returns : a copy of *rgn*.

gvol_rendering_rgn_free ()

```
void        gvol_rendering_rgn_free         (GVolRenderingRgn *rgn);
```

Free a GVolRenderingRgn [p 183] .

rgn : a rendering region information structure to be freed.

gvol_rendering_rgn_get_coord_ptr ()

```
gpointer    gvol_rendering_rgn_get_coord_ptr
                                            (GVolRenderingRgn *rgn,
                                             gint32 x,
                                             gint32 y);
```

Get a pointer to the rendering element at the (*x*, *y*) coordinate of *rgn*.

rgn : a GVolRenderingRgn

x : the X coordinate of the pointer to be retrieved

y : the Y coordinate of the pointer to be retrieved

Returns : a pointer to the rendering element at position (*x*, *y*).

gvol_rendering_rgn_release ()

```
void        gvol_rendering_rgn_release      (GVolRenderingRgn *rgn);
```

Release the specified rendering region, breaking all the references with the originating GVolRenderingCtx.

Note: the memory occupied by *rgn* won't be freed. If you also need to free it, then you should call gvol_rendering_rgn() instead.

rgn : a GVolRenderingRgn [p 183]

<< **GVolRenderingCtxRawRGBguchar** **GVoFunc** >>

GVolVector3Dgdouble

GVolVector3Dgdouble —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolVector3Dgdouble;
void        gvol_vector3d_gdouble_2_normalize
                                        (GVolVector3Dgdouble *dest,
                                         const GVolVector3Dgdouble *v);
void        gvol_vector3d_gdouble_2_normalize_self
                                        (GVolVector3Dgdouble *v);
GVolVector3Dgdouble* gvol_vector3d_gdouble_copy
                                        (const GVolVector3Dgdouble *v);
void        gvol_vector3d_gdouble_copy2     (GVolVector3Dgdouble *dest,
                                         const GVolVector3Dgdouble *v);
void        gvol_vector3d_gdouble_crossprod (GVolVector3Dgdouble *crossv,
                                         const GVolVector3Dgdouble *v1,
                                         const GVolVector3Dgdouble *v2);
gdouble     gvol_vector3d_gdouble_dotprod   (const GVolVector3Dgdouble *v1,
                                         const GVolVector3Dgdouble *v2);
void        gvol_vector3d_gdouble_free      (GVolVector3Dgdouble *v);
gdouble     gvol_vector3d_gdouble_get_2_norm
                                        (const GVolVector3Dgdouble *v);
void        gvol_vector3d_gdouble_mult      (GVolVector3Dgdouble *res,
                                         const GVolMatrix4x4gdouble *m,
                                         const GVolVector3Dgdouble *v);
void        gvol_vector3d_gdouble_scale     (GVolVector3Dgdouble *dest,
                                         const GVolVector3Dgdouble *v,
                                         gdouble scalar);
void        gvol_vector3d_gdouble_scale_self
                                        (GVolVector3Dgdouble *v,
                                         gdouble scalar);
void        gvol_vector3d_gdouble_zero      (GVolVector3Dgdouble *v);
```

Description

Details

struct GVolVector3Dgdouble

```
struct GVolVector3Dgdouble {

  gdouble x;
```

```
  gdouble y;
  gdouble z;
  gdouble w;
};
```

gvol_vector3d_gdouble_2_normalize ()

```
void        gvol_vector3d_gdouble_2_normalize
                                        (GVolVector3Dgdouble *dest,
                                         const GVolVector3Dgdouble *v);
```

Normalize *v*, using the 2-norm (euclidean norm), storing the result in *dest*.

dest : a GVolVector3D in which the normalized vector will be stored

v : a GVolVector3D to normalize

gvol_vector3d_gdouble_2_normalize_self ()

```
void        gvol_vector3d_gdouble_2_normalize_self
                                        (GVolVector3Dgdouble *v);
```

Normalize *v*, using the 2-norm (euclidean norm), storing the result in *v* itself.

v : a GVolVector3D

gvol_vector3d_gdouble_copy ()

```
GVolVector3Dgdouble* gvol_vector3d_gdouble_copy
                                        (const GVolVector3Dgdouble *v);
```

Copy a GVolVector3Dgdouble [p 185] .

v : a vector to be copied.

Returns : a copy of *v*.

gvol_vector3d_gdouble_copy2 ()

```
void        gvol_vector3d_gdouble_copy2     (GVolVector3Dgdouble *dest,
                                             const GVolVector3Dgdouble *v);
```

Copy the content of *v* to *dest* (that will be overwritten).

dest : a destination vector

v : the vector to be copied

---

gvol_vector3d_gdouble_crossprod ()

```
void        gvol_vector3d_gdouble_crossprod (GVolVector3Dgdouble *crossv,
                                             const GVolVector3Dgdouble *v1,
                                             const GVolVector3Dgdouble *v2);
```

Compute the cross product between *v1* and *v2*, storing it in *crossv*.

crossv : a GVolVector3Dgdouble [p 185] to store the cross product

v1 : a GVolVector3Dgdouble [p 185]

v2 : a GVolVector3Dgdouble [p 185]

---

gvol_vector3d_gdouble_dotprod ()

```
gdouble     gvol_vector3d_gdouble_dotprod   (const GVolVector3Dgdouble *v1,
                                             const GVolVector3Dgdouble *v2);
```

Compute the dot product between *v1* and *v2*.

v1 : a GVolVector3Dgdouble [p 185]

v2 : a GVolVector3Dgdouble [p 185]

Returns : the dot product between *v1* and *v2*.

---

gvol_vector3d_gdouble_free ()

```
void        gvol_vector3d_gdouble_free      (GVolVector3Dgdouble *v);
```

Free a GVolVector3Dgdouble [p 185] .

v : a vector to be freed.

---

gvol_vector3d_gdouble_get_2_norm ()

```
gdouble     gvol_vector3d_gdouble_get_2_norm
                                        (const GVolVector3Dgdouble *v);
```

Get the 2-norm (euclidean norm) of *v*.

v : a GVolVector3D

Returns :

---

gvol_vector3d_gdouble_mult ()

```
void        gvol_vector3d_gdouble_mult      (GVolVector3Dgdouble *res,
                                             const GVolMatrix4x4gdouble *m,
                                             const GVolVector3Dgdouble *v);
```

Multiply the matrix *m* by the vector *v*, storing the result in *res*.

res : a GVolVector3Dgdouble [p 185] to store the result

m : the multiplication matrix

v : the vector to be multiplied

---

gvol_vector3d_gdouble_scale ()

```
void        gvol_vector3d_gdouble_scale     (GVolVector3Dgdouble *dest,
                                             const GVolVector3Dgdouble *v,
                                             gdouble scalar);
```

Multiply every element of *v* by *scalar*, storing the result in *res*.

dest :

v : a GVolVector3Dgdouble [p 185]

scalar : a scalar value

---

gvol_vector3d_gdouble_scale_self ()

```
void        gvol_vector3d_gdouble_scale_self
                                        (GVolVector3Dgdouble *v,
                                         gdouble scalar);
```

Multiply every element of *v* by *scalar*, storing the result in *v* itself.

v : a GVolVector3Dgdouble [p 185]

scalar : a scalar value

---

gvol_vector3d_gdouble_zero ()

```
void        gvol_vector3d_gdouble_zero      (GVolVector3Dgdouble *v);
```

Set all the elements ov *v* to 0.0.

v : a GVolVector3Dgdouble [p 185]

<< **GVolFunc_gfloat_gint32** **GVolVector3Dgfloat** >> [p 191]

GVolVector3Dgfloat

GVolVector3Dgfloat —

Synopsis

```
#include <gvol/gvol.h>


struct      GVolVector3Dgfloat;
void        gvol_vector3d_gfloat_2_normalize
                                        (GVolVector3Dgfloat *dest,
                                         const GVolVector3Dgfloat *v);
void        gvol_vector3d_gfloat_2_normalize_self
                                        (GVolVector3Dgfloat *v);
GVolVector3Dgfloat* gvol_vector3d_gfloat_copy
                                        (const GVolVector3Dgfloat *v);
void        gvol_vector3d_gfloat_copy2      (GVolVector3Dgfloat *dest,
                                         const GVolVector3Dgfloat *v);
void        gvol_vector3d_gfloat_crossprod  (GVolVector3Dgfloat *crossv,
                                         const GVolVector3Dgfloat *v1,
                                         const GVolVector3Dgfloat *v2);
gfloat      gvol_vector3d_gfloat_dotprod    (const GVolVector3Dgfloat *v1,
                                         const GVolVector3Dgfloat *v2);
void        gvol_vector3d_gfloat_free       (GVolVector3Dgfloat *v);
void        gvol_vector3d_gfloat_from_gdouble
                                        (GVolVector3Dgfloat *dest,
                                         const GVolVector3Dgdouble *src);
gfloat      gvol_vector3d_gfloat_get_2_norm (const GVolVector3Dgfloat *v);
void        gvol_vector3d_gfloat_mult       (GVolVector3Dgfloat *res,
                                         const GVolMatrix4x4gfloat *m,
                                         const GVolVector3Dgfloat *v);
void        gvol_vector3d_gfloat_scale      (GVolVector3Dgfloat *dest,
                                         const GVolVector3Dgfloat *v,
                                         gfloat scalar);
void        gvol_vector3d_gfloat_scale_self (GVolVector3Dgfloat *v,
                                         gfloat scalar);
void        gvol_vector3d_gfloat_zero       (GVolVector3Dgfloat *v);
```

Description

Details

struct GVolVector3Dgfloat

```
struct GVolVector3Dgfloat {

  gfloat x;
  gfloat y;
  gfloat z;
  gfloat w;
};
```

---

gvol_vector3d_gfloat_2_normalize ()

```
void        gvol_vector3d_gfloat_2_normalize
                                          (GVolVector3Dgfloat *dest,
                                           const GVolVector3Dgfloat *v);
```

Normalize *v*, using the 2-norm (euclidean norm), storing the result in *dest*.

dest : a GVolVector3D in which the normalized vector will be stored

v : a GVolVector3D to normalize

---

gvol_vector3d_gfloat_2_normalize_self ()

```
void        gvol_vector3d_gfloat_2_normalize_self
                                          (GVolVector3Dgfloat *v);
```

Normalize *v*, using the 2-norm (euclidean norm), storing the result in *v* itself.

v : a GVolVector3D

---

gvol_vector3d_gfloat_copy ()

```
GVolVector3Dgfloat* gvol_vector3d_gfloat_copy
                                          (const GVolVector3Dgfloat *v);
```

Copy a GVolVector3Dgfloat [p 191] .

v : a vector to be copied.

Returns : a copy of *v*.

---

gvol_vector3d_gfloat_copy2 ()

```
void        gvol_vector3d_gfloat_copy2      (GVolVector3Dgfloat *dest,
                                             const GVolVector3Dgfloat *v);
```

Copy the content of *v* to *dest* (that will be overwritten).

dest : a destination vector

v : the vector to be copied

---

gvol_vector3d_gfloat_crossprod ()

```
void        gvol_vector3d_gfloat_crossprod  (GVolVector3Dgfloat *crossv,
                                             const GVolVector3Dgfloat *v1,
                                             const GVolVector3Dgfloat *v2);
```

Compute the cross product between *v1* and *v2*, storing it in *crossv*.

crossv : a GVolVector3Dgfloat [p 191] to store the cross product

v1 : a GVolVector3Dgfloat [p 191]

v2 : a GVolVector3Dgfloat [p 191]

---

gvol_vector3d_gfloat_dotprod ()

```
gfloat      gvol_vector3d_gfloat_dotprod    (const GVolVector3Dgfloat *v1,
                                             const GVolVector3Dgfloat *v2);
```

Compute the dot product between *v1* and *v2*.

v1 : a GVolVector3Dgfloat [p 191]

v2 : a GVolVector3Dgfloat [p 191]

Returns : the dot product between *v1* and *v2*.

---

gvol_vector3d_gfloat_free ()

```
void        gvol_vector3d_gfloat_free       (GVolVector3Dgfloat *v);
```

Free a GVolVector3Dgfloat [p 191] .

v : a vector to be freed.

---

gvol_vector3d_gfloat_from_gdouble ()

```
void        gvol_vector3d_gfloat_from_gdouble
                                        (GVolVector3Dgfloat *dest,
                                         const GVolVector3Dgdouble *src);
```

Convert *src* to GVolVector3Dgfloat [p 191] type, storing the result in *dest*.

dest : the destination GVolVector3Dgfloat [p 191]

src : the source GVolVector3Dgdouble

---

gvol_vector3d_gfloat_get_2_norm ()

```
gfloat      gvol_vector3d_gfloat_get_2_norm (const GVolVector3Dgfloat *v);
```

Get the 2-norm (euclidean norm) of *v*.

v : a GVolVector3D

Returns :

---

gvol_vector3d_gfloat_mult ()

```
void        gvol_vector3d_gfloat_mult       (GVolVector3Dgfloat *res,
                                             const GVolMatrix4x4gfloat *m,
                                             const GVolVector3Dgfloat *v);
```

Multiply the matrix *m* by the vector *v*, storing the result in *res*.

res : a GVolVector3Dgfloat [p 191] to store the result

m : the multiplication matrix

v : the vector to be multiplied

---

gvol_vector3d_gfloat_scale ()

```
void        gvol_vector3d_gfloat_scale      (GVolVector3Dgfloat *dest,
                                             const GVolVector3Dgfloat *v,
                                             gfloat scalar);
```

Multiply every element of *v* by *scalar*, storing the result in *res*.

dest :

v : a GVolVector3Dgfloat [p 191]

scalar : a scalar value

---

gvol_vector3d_gfloat_scale_self ()

```
void        gvol_vector3d_gfloat_scale_self (GVolVector3Dgfloat *v,
                                             gfloat scalar);
```

Multiply every element of *v* by *scalar*, storing the result in *v* itself.

v : a GVolVector3Dgfloat [p 191]

scalar : a scalar value

---

gvol_vector3d_gfloat_zero ()

```
void        gvol_vector3d_gfloat_zero       (GVolVector3Dgfloat *v);
```

Set all the elements ov *v* to 0.0.

v : a GVolVector3Dgfloat [p 191]

<< **GVolVector3Dgdouble** **GVolVolume** >>