ゲームグラフィックス特論

構築: Doxygen 1.8.6

2018 年 10 月 17 日 (水) 08 時 13 分 24 秒

Contents

Chapter 1

名前空間索引

1.1 名前空間一覧

全名前空間の一覧です。

Chapter 2

階層索引

2.1 クラス階層

クラス階層一覧です。大雑把に文字符号順で並べられています。

Chapter 3

クラス索引

3.1 クラス一覧

クラス・構造体・共用体・インターフェースの一覧です。

Chapter 4

ファイル索引

4.1 ファイル一覧

ファイル一覧です。

Chapter 5

名前空間詳解

5.1 gg 名前空間

ゲームグラフィックス特論の宿題用補助プログラムの名前空間

クラス

- class GgMatrix
  変換行列.
- class GgQuaternion
  四元数.
- class GgTrackball
  簡易トラックボール処理.
- class GgTexture
  テクスチャ.
- class GgColorTexture
  カラーマップ.
- class GgNormalTexture
  法線マップ.
- class GgBuffer
  バッファオブジェクト.
- class GgUniformBuffer
  ユニフォームバッファオブジェクト.
- class GgShape
  形状データの基底クラス.
- class GgPoints
  点.
- struct GgVertex
  三角形の頂点データ.
- class GgTriangles
  三角形で表した形状データ *(Arrays* **形式***)*.
- class GgElements
  三角形で表した形状データ *(Elements* **形式***)*.
- class GgShader
  シェーダの基底クラス.
- class GgPointShader
  点のシェーダ.

- class GgSimpleShader

  三角形に単純な陰影付けを行うシェーダ.
- class GgSimpleObj

  Wavefront OBJ **形式のファイル** *(Arrays* **形式**).

型定義

- using GgVector = std::array< GLfloat, 4 >

  4 **要素の単精度実数の配列**

関数

- void ggInit ()

  ゲームグラフィックス特論の都合にもとづく初期化を行う.
- void _ggError (const char *name=nullptr, unsigned int line=0)

  OpenGL **のエラーをチェックする**.
- void _ggFBOError (const char *name=nullptr, unsigned int line=0)

  FBO **のエラーをチェックする**.
- bool ggSaveTga (const char *name, const void *buffer, unsigned int width, unsigned int height, unsigned int depth)

  配列の内容を *TGA* **ファイルに保存する**.
- bool ggSaveColor (const char *name)

  カラーバッファの内容を *TGA* **ファイルに保存する**.
- bool ggSaveDepth (const char *name)

  デプスバッファの内容を *TGA* **ファイルに保存する**.
- bool ggReadImage (const char *name, std::vector< GLubyte > &image, GLsizei *pWidth, GLsizei *pHeight, GLenum *pFormat)

  TGA **ファイル** *(8/16/24/32bit)* **をメモリに読み込む**.
- GLuint ggLoadTexture (const GLvoid *image, GLsizei width, GLsizei height, GLenum format=GL_BGR, GLenum type=GL_UNSIGNED_BYTE, GLenum internal=GL_RGB, GLenum wrap=GL_CLAMP_TO_EDGE)

  テクスチャメモリを確保して画像データをテクスチャとして読み込む.
- GLuint ggLoadImage (const char *name, GLsizei *pWidth=nullptr, GLsizei *pHeight=nullptr, GLenum internal=0, GLenum wrap=GL_CLAMP_TO_EDGE)

  テクスチャメモリを確保して *TGA* **画像ファイルを読み込む**.
- void ggCreateNormalMap (const GLubyte *hmap, GLsizei width, GLsizei height, GLenum format, GLfloat nz, GLenum internal, std::vector< GgVector > &nmap)

  グレースケール画像 *(8bit)* **から法線マップのデータを作成する**.
- GLuint ggLoadHeight (const char *name, float nz, GLsizei *pWidth=nullptr, GLsizei *pHeight=nullptr, GLenum internal=GL_RGBA)

  テクスチャメモリを確保して *TGA* **画像ファイルを読み込み法線マップを作成する**.
- GLuint ggCreateShader (const char *vsrc, const char *fsrc=nullptr, const char *gsrc=nullptr, GLint nvarying=0, const char *const varyings[]=nullptr, const char *vtext="vertex shader", const char *ftext="fragment shader", const char *gtext="geometry shader")

  シェーダのソースプログラムの文字列を読み込んでプログラムオブジェクトを作成する.
- GLuint ggLoadShader (const char *vert, const char *frag=nullptr, const char *geom=nullptr, GLint nvarying=0, const char *const varyings[]=nullptr)

  シェーダのソースファイルを読み込んでプログラムオブジェクトを作成する.
- GLuint ggCreateComputeShader (const char *csrc, const char *ctext="compute shader")

  コンピュートシェーダのソースプログラムの文字列を読み込んでプログラムオブジェクトを作成する.
- GLuint ggLoadComputeShader (const char *comp)

  コンピュートシェーダのソースファイルを読み込んでプログラムオブジェクトを作成する.

- GLfloat ggLength3 (const GLfloat *a)
  3 **要素の長さ**.
- void ggNormalize3 (GLfloat *a)
  3 **要素の正規化**.
- GLfloat ggDot3 (const GLfloat *a, const GLfloat *b)
  3 **要素の内積**.
- void ggCross (GLfloat *c, const GLfloat *a, const GLfloat *b)
  3 **要素の外積**.
- GLfloat ggLength4 (const GLfloat *a)
  4 **要素の長さ**.
- GLfloat ggLength4 (const GgVector &a)
  GgVector **型の長さ**.
- void ggNormalize4 (GLfloat *a)
  4 **要素の正規化**.
- void ggNormalize4 (GgVector &a)
  GgVector **型の正規化**.
- GLfloat ggDot4 (const GLfloat *a, const GLfloat *b)
  4 **要素の内積**
- GLfloat ggDot4 (const GgVector &a, const GgVector &b)
  GgVector **型の内積**
- GgMatrix ggIdentity ()
  単位行列を返す.
- GgMatrix ggTranslate (GLfloat x, GLfloat y, GLfloat z, GLfloat w=1.0f)
  平行移動の変換行列を返す.
- GgMatrix ggTranslate (const GLfloat *t)
  平行移動の変換行列を返す.
- GgMatrix ggTranslate (const GgVector &t)
  平行移動の変換行列を返す.
- GgMatrix ggScale (GLfloat x, GLfloat y, GLfloat z, GLfloat w=1.0f)
  拡大縮小の変換行列を返す.
- GgMatrix ggScale (const GLfloat *s)
  拡大縮小の変換行列を返す.
- GgMatrix ggScale (const GgVector &s)
  拡大縮小の変換行列を返す.
- GgMatrix ggRotateX (GLfloat a)
  x **軸中心の回転の変換行列を返す**.
- GgMatrix ggRotateY (GLfloat a)
  y **軸中心の回転の変換行列を返す**.
- GgMatrix ggRotateZ (GLfloat a)
  z **軸中心の回転の変換行列を返す**.
- GgMatrix ggRotate (GLfloat x, GLfloat y, GLfloat z, GLfloat a)
  (x, y, z) **方向のベクトルを軸とする回転の変換行列を乗じた結果を返す**.
- GgMatrix ggRotate (const GLfloat *r, GLfloat a)
  r **方向のベクトルを軸とする回転の変換行列を乗じた結果を返す**.
- GgMatrix ggRotate (const GgVector &r, GLfloat a)
  r **方向のベクトルを軸とする回転の変換行列を乗じた結果を返す**.
- GgMatrix ggRotate (const GLfloat *r)
  r **方向のベクトルを軸とする回転の変換行列を乗じた結果を返す**.
- GgMatrix ggRotate (const GgVector &r)
  r **方向のベクトルを軸とする回転の変換行列を乗じた結果を返す**.

- GgMatrix ggLookat (GLfloat ex, GLfloat ey, GLfloat ez, GLfloat tx, GLfloat ty, GLfloat tz, GLfloat ux, GLfloat uy, GLfloat uz)

  ビュー変換行列を返す.
- GgMatrix ggLookat (const GLfloat *e, const GLfloat *t, const GLfloat *u)

  ビュー変換行列を返す.
- GgMatrix ggLookat (const GgVector &e, const GgVector &t, const GgVector &u)

  ビュー変換行列を返す.
- GgMatrix ggOrthogonal (GLfloat left, GLfloat right, GLfloat bottom, GLfloat top, GLfloat zNear, GLfloat zFar)

  直交投影変換行列を返す.
- GgMatrix ggFrustum (GLfloat left, GLfloat right, GLfloat bottom, GLfloat top, GLfloat zNear, GLfloat zFar)

  透視透視投影変換行列を返す.
- GgMatrix ggPerspective (GLfloat fovy, GLfloat aspect, GLfloat zNear, GLfloat zFar)

  画角を指定して透視投影変換行列を返す.
- GgMatrix ggTranspose (const GgMatrix &m)

  転置行列を返す.
- GgMatrix ggInvert (const GgMatrix &m)

  逆行列を返す.
- GgMatrix ggNormal (const GgMatrix &m)

  法線変換行列を返す.
- GgQuaternion ggQuaternion (GLfloat x, GLfloat y, GLfloat z, GLfloat w)

  四元数を返す
- GgQuaternion ggQuaternion (const GLfloat *a)

  四元数を返す
- GgQuaternion ggIdentityQuaternion ()

  単位四元数を返す
- GgQuaternion ggMatrixQuaternion (const GLfloat *a)

  回転の変換行列 *m* **を表す四元数を返す**.
- GgQuaternion ggMatrixQuaternion (const GgMatrix &m)

  回転の変換行列 *m* **を表す四元数を返す**.
- GgMatrix ggQuaternionMatrix (const GgQuaternion &q)

  四元数 *q* **の回転の変換行列を返す**.
- GgMatrix ggQuaternionTransposeMatrix (const GgQuaternion &q)

  四元数 *q* **の回転の転置した変換行列を返す**.
- GgQuaternion ggRotateQuaternion (GLfloat x, GLfloat y, GLfloat z, GLfloat a)

  (x, y, z) **を軸として角度** *a* **回転する四元数を返す**.
- GgQuaternion ggRotateQuaternion (const GLfloat *v, GLfloat a)

  (v[0], v[1], v[2]) **を軸として角度** *a* **回転する四元数を返す**.
- GgQuaternion ggRotateQuaternion (const GLfloat *v)

  (v[0], v[1], v[2]) **を軸として角度** *v[3]* **回転する四元数を返す**.
- GgQuaternion ggEulerQuaternion (GLfloat heading, GLfloat pitch, GLfloat roll)

  オイラー角 *(heading, pitch, roll)* **で与えられた回転を表す四元数を返す**.
- GgQuaternion ggEulerQuaternion (const GLfloat *e)

  オイラー角 *(e[0], e[1], e[2])* **で与えられた回転を表す四元数を返す**.
- GgQuaternion ggSlerp (const GLfloat *a, const GLfloat *b, GLfloat t)

  二つの四元数の球面線形補間の結果を返す.
- GgQuaternion ggSlerp (const GgQuaternion &q, const GgQuaternion &r, GLfloat t)

  二つの四元数の球面線形補間の結果を返す.
- GgQuaternion ggSlerp (const GgQuaternion &q, const GLfloat *a, GLfloat t)

  二つの四元数の球面線形補間の結果を返す.
- GgQuaternion ggSlerp (const GLfloat *a, const GgQuaternion &q, GLfloat t)

  二つの四元数の球面線形補間の結果を返す.

- GLfloat ggNorm (const GgQuaternion &q)

  四元数のノルムを返す.
- GgQuaternion ggNormalize (const GgQuaternion &q)

  正規化した四元数を返す.
- GgQuaternion ggConjugate (const GgQuaternion &q)

  共役四元数を返す.
- GgQuaternion ggInvert (const GgQuaternion &q)

  四元数の逆元を求める.
- GgPoints * ggPointsCube (GLsizei countv, GLfloat length=1.0f, GLfloat cx=0.0f, GLfloat cy=0.0f, GLfloat cz=0.0f)

  点群を立方体状に生成する.
- GgPoints * ggPointsSphere (GLsizei countv, GLfloat radius=0.5f, GLfloat cx=0.0f, GLfloat cy=0.0f, GLfloat cz=0.0f)

  点群を球状に生成する.
- GgTriangles * ggRectangle (GLfloat width=1.0f, GLfloat height=1.0f)

  矩形状に *2* **枚の三角形を生成する**.
- GgTriangles * ggEllipse (GLfloat width=1.0f, GLfloat height=1.0f, GLuint slices=16)

  楕円状に三角形を生成する.
- GgTriangles * ggArraysObj (const char *name, bool normalize=false)

  Wavefront OBJ **ファイルを読み込む** *(Arrays* **形式***)*
- GgElements * ggElementsObj (const char *name, bool normalize=false)

  Wavefront OBJ **ファイル を読み込む** *(Elements* **形式***).*
- GgElements * ggElementsMesh (GLuint slices, GLuint stacks, const GLfloat(*pos)[3], const GLfloat(*norm)[3]=nullptr)

  メッシュ形状を作成する *(Elements* **形式***).*
- GgElements * ggElementsSphere (GLfloat radius=1.0f, int slices=16, int stacks=8)
- bool ggLoadSimpleObj (const char *name, std::vector< std::array< GLuint, 3 > > &group, std::vector< GgSimpleShader::Material > &material, std::vector< GgVertex > &vert, bool normalize=false)

  三角形分割された *OBJ* **ファイルと** *MTL* **ファイルを読み込む** *(Arrays* **形式***)*
- bool ggLoadSimpleObj (const char *name, std::vector< std::array< GLuint, 3 > > &group, std::vector< GgSimpleShader::Material > &material, std::vector< GgVertex > &vert, std::vector< GLuint > &face, bool normalize=false)

  三角形分割された *OBJ* **ファイルを読み込む** *(Elements* **形式***).*

変数

- GLint ggBufferAlignment

  使用している *GPU* **のバッファオブジェクトのアライメント**

5.1.1 詳解

ゲームグラフィックス特論の宿題用補助プログラムの名前空間

5.1.2 型定義詳解

5.1.2.1 using gg::GgVector = typedef std::array<GLfloat, 4>

4 要素の単精度実数の配列

gg.h の 1304 行目に定義があります。

5.1.3 関数詳解

5.1.3.1 void gg::_ggError (const char * *name* = nullptr, unsigned int *line* = 0)

OpenGL のエラーをチェックする.

OpenGL の API を呼び出し直後に実行すればエラーのあるときにメッセージを表示する.

引数

msg	エラー発生時に標準エラー出力に出力する文字列. nullptr なら何も出力しない.

gg.cpp の 2569 行目に定義があります。

5.1.3.2 void gg::_ggFBOError (const char * *name* = nullptr, unsigned int *line* = 0)

FBO のエラーをチェックする.

FBO の API を呼び出し直後に実行すればエラーのあるときにメッセージを表示する.

引数

msg	エラー発生時に標準エラー出力に出力する文字列. nullptr なら何も出力しない.

gg.cpp の 2613 行目に定義があります。

5.1.3.3 gg::GgTriangles * gg::ggArraysObj (const char * *name*, bool *normalize* = false)

Wavefront OBJ ファイルを読み込む (Arrays 形式)

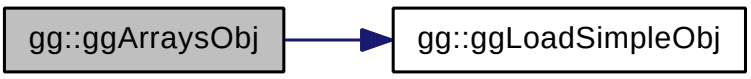
三角形分割された Wavefront OBJ ファイルを読み込んで GgArrays 形式の三角形データを生成する.

引数

name	ファイル名.
normalize	true なら大きさを正規化.

gg.cpp の 5090 行目に定義があります。

呼び出し関係図:



5.1.3.4 GgQuaternion gg::ggConjugate (const GgQuaternion & *q*) [inline]

共役四元数を返す.

引数

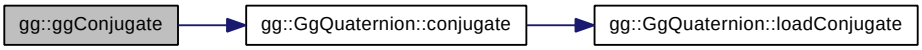
q	GgQuaternion 型の四元数.

戻り値

四元数 q の共役四元数.

gg.h の 3664 行目に定義があります。

呼び出し関係図:



5.1.3.5 GLuint gg::ggCreateComputeShader (const char * *csrc*, const char * *ctext* = "compute shader")

コンピュートシェーダのソースプログラムの文字列を読み込んでプログラムオブジェクトを作成する.

引数

csrc	コンピュートシェーダのソースプログラムの文字列.
ctext	コンピュートシェーダのコンパイル時のメッセージに追加する文字列.

戻り値

プログラムオブジェクトのプログラム名 (作成できなければ 0).

gg.cpp の 4043 行目に定義があります。

被呼び出し関係図:

5.1.3.6 void gg::ggCreateNormalMap (const GLubyte * *hmap*, GLsizei *width*, GLsizei *height*, GLenum *format*, GLfloat *nz*, GLenum *internal*, std::vector< GgVector > & *nmap*)

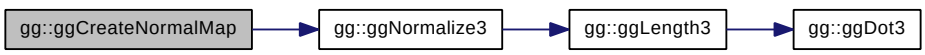
グレースケール画像 (8bit) から法線マップのデータを作成する.

引数

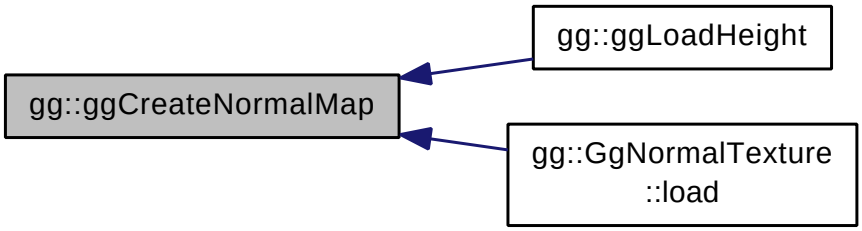
hmap	グレースケール画像のデータ.
width	高さマップのグレースケール画像 hmap の横の画素数.
height	高さマップのグレースケール画像 hmap の縦の画素数.
stride	データの間隔.
nz	法線の z 成分の割合.
法線マップを格納するテクスチャの内部フォーマット.	
nmap	法線マップを格納する vector.

gg.cpp の 3001 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:



5.1.3.7 GLuint gg::ggCreateShader (const char * *vsrc*, const char * *fsrc* = nullptr, const char * *gsrc* = nullptr, GLint *nvarying* = 0, const char *const *varyings[]* = nullptr, const char * *vtext* = "vertex shader", const char * *ftext* = "fragment shader", const char * *gtext* = "geometry shader")

シェーダのソースプログラムの文字列を読み込んでプログラムオブジェクトを作成する.

引数

vsrc	バーテックスシェーダのソースプログラムの文字列.
fsrc	フラグメントシェーダのソースプログラムの文字列 (nullptr なら不使用).
gsrc	ジオメトリシェーダのソースプログラムの文字列 (nullptr なら不使用).
nvarying	フィードバックする varying 変数の数 (0 なら不使用).
varyings	フィードバックする varying 変数のリスト (nullptr なら不使用).
vtext	バーテックスシェーダのコンパイル時のメッセージに追加する文字列.
ftext	フラグメントシェーダのコンパイル時のメッセージに追加する文字列.
gtext	ジオメトリシェーダのコンパイル時のメッセージに追加する文字列.

戻り値

プログラムオブジェクトのプログラム名 (作成できなければ 0).

gg.cpp の 3970 行目に定義があります。

被呼び出し関係図:

5.1.3.8 void gg::ggCross (GLfloat * *c*, const GLfloat * *a*, const GLfloat * *b*) [inline]

3 要素の外積.

引数

a	GLfloat 型の 3 要素の配列変数.
b	GLfloat 型の 3 要素の配列変数.
c	結果を格納する GLfloat 型の 3 要素の配列変数.

gg.h の 1528 行目に定義があります。

5.1.3.9 GLfloat gg::ggDot3 (const GLfloat * *a*, const GLfloat * *b*) [inline]

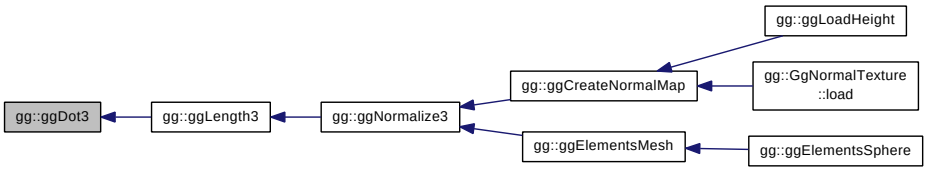
3 要素の内積.

引数

a	GLfloat 型の 3 要素の配列変数.
b	GLfloat 型の 3 要素の配列変数.

gg.h の 1516 行目に定義があります。

被呼び出し関係図:



5.1.3.10 GLfloat gg::ggDot4 (const GLfloat * *a*, const GLfloat * *b*) [inline]

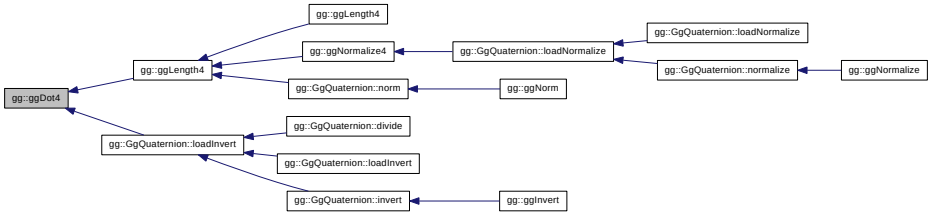
4 要素の内積

引数

a	GLfloat 型の 4 要素の配列変数.
b	GLfloat 型の 4 要素の配列変数.

gg.h の 1592 行目に定義があります。

被呼び出し関係図:



5.1.3.11 GLfloat gg::ggDot4 (const GgVector & *a*, const GgVector & *b*) [inline]

GgVector 型の内積

引数

a	GgVector 型の変数.
b	GgVector 型の変数.

gg.h の 1603 行目に定義があります。

5.1.3.12 gg::GgElements * gg::ggElementsMesh (GLuint *slices*, GLuint *stacks*, const GLfloat(*) *pos[3]*, const GLfloat(*) *norm[3]* = nullptr)

メッシュ形状を作成する (Elements 形式).

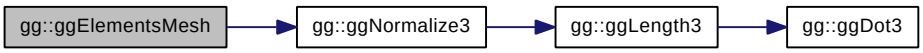
メッシュ状に GgElements 形式の三角形データを生成する.

引数

slices	メッシュの横方向の分割数.
stacks	メッシュの縦方向の分割数.
pos	メッシュの頂点の位置.
norm	メッシュの頂点の法線.

gg.cpp の 5124 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:

5.1.3.13 gg::GgElements * gg::ggElementsObj (const char * *name*, bool *normalize* = false)

Wavefront OBJ ファイル を読み込む (Elements 形式).

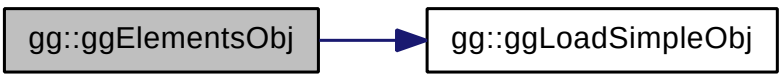
三角形分割された Wavefront OBJ ファイル を読み込んで GgElements 形式の三角形データを生成する.

引数

name	ファイル名.
normalize	true なら大きさを正規化.

gg.cpp の 5106 行目に定義があります。

呼び出し関係図:



5.1.3.14 gg::GgElements * gg::ggElementsSphere (GLfloat *radius* = 1.0f, int *slices* = 16, int *stacks* = 8)

球状に三角形データを生成する (Elements 形式).

球状に GgElements 形式の三角形データを生成する.

引数

radius	球の半径.
slices	球の経度方向の分割数.
stacks	球の緯度方向の分割数.

gg.cpp の 5220 行目に定義があります。

呼び出し関係図:

5.1.3.15 gg::GgTriangles * gg::ggEllipse (GLfloat *width* = 1.0f, GLfloat *height* = 1.0f, GLuint *slices* = 16)

楕円状に三角形を生成する.

引数

width	楕円の横幅.
height	楕円の高さ.
slices	楕円の分割数.

gg.cpp の 5064 行目に定義があります。

5.1.3.16 GgQuaternion gg::ggEulerQuaternion (GLfloat *heading*, GLfloat *pitch*, GLfloat *roll*) [inline]

オイラー角 (heading, pitch, roll) で与えられた回転を表す四元数を返す.

引数

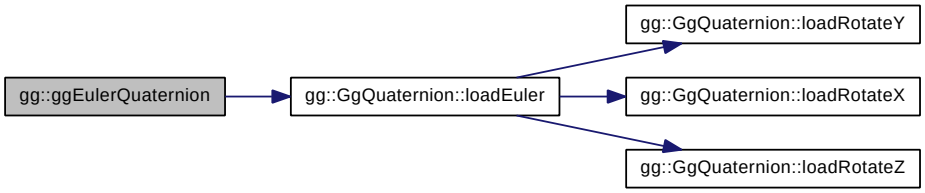
heading	y 軸中心の回転角.
pitch	x 軸中心の回転角.
roll	z 軸中心の回転角.

戻り値

回転を表す四元数.

gg.h の 3590 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:

5.1.3.17 GgQuaternion gg::ggEulerQuaternion (const GLfloat * *e*) [inline]

オイラー角 (e[0], e[1], e[2]) で与えられた回転を表す四元数を返す.

引数

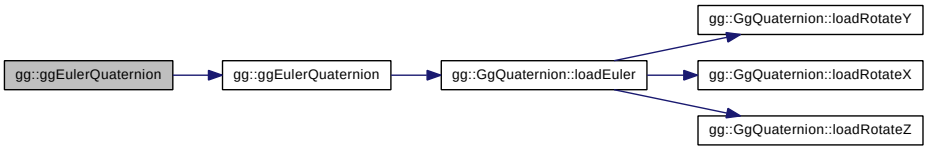
e	オイラー角を表す GLfloat 型の 3 要素の配列変数 (heading, pitch, roll).

戻り値

回転を表す四元数.

gg.h の 3599 行目に定義があります。

呼び出し関係図:



5.1.3.18 GgMatrix gg::ggFrustum (GLfloat *left*, GLfloat *right*, GLfloat *bottom*, GLfloat *top*, GLfloat *zNear*, GLfloat *zFar*) [inline]

透視透視投影変換行列を返す.

引数

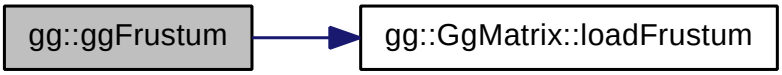
left	ウィンドウの左端の位置.
right	ウィンドウの右端の位置.
bottom	ウィンドウの下端の位置.
top	ウィンドウの上端の位置.
zNear	視点から前方面までの位置.
zFar	視点から後方面までの位置.

戻り値

求めた透視投影変換行列.

gg.h の 2611 行目に定義があります。

呼び出し関係図:



5.1.3.19 GgMatrix gg::ggIdentity ( ) [inline]

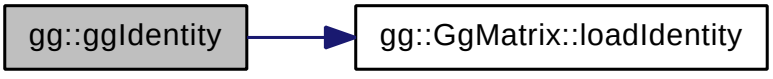
単位行列を返す.

戻り値

単位行列

gg.h の 2393 行目に定義があります。

呼び出し関係図:



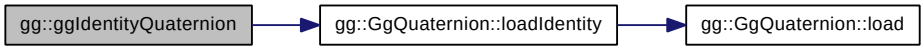
5.1.3.20 GgQuaternion gg::ggIdentityQuaternion ( ) [inline]

単位四元数を返す

戻り値

単位四元数.

gg.h の 3511 行目に定義があります。
呼び出し関係図:



5.1.3.21 void gg::ggInit ( )

ゲームグラフィックス特論の都合にもとづく初期化を行う.
Windows で OpenGL 1.2 以降の API を有効化する.
gg.cpp の 1314 行目に定義があります。

5.1.3.22 GgMatrix gg::ggInvert (const GgMatrix & *m*) [inline]

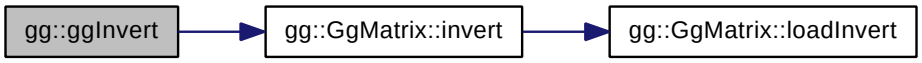
逆行列を返す.
引数

m	元の変換行列.

戻り値

m の逆行列.

gg.h の 2643 行目に定義があります。
呼び出し関係図:



5.1.3.23 GgQuaternion gg::ggInvert (const GgQuaternion & *q*) [inline]

四元数の逆元を求める.
引数

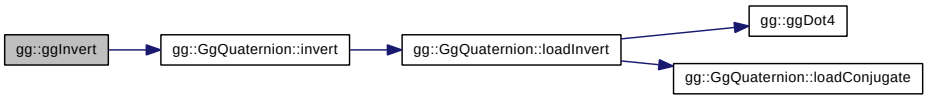
q	GgQuaternion 型の四元数.

戻り値

四元数 q の逆元.

gg.h の 3672 行目に定義があります。

呼び出し関係図:



5.1.3.24 GLfloat gg::ggLength3 (const GLfloat * a)

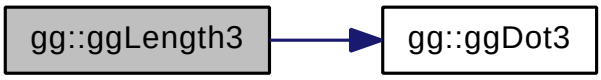
3 要素の長さ.

引数

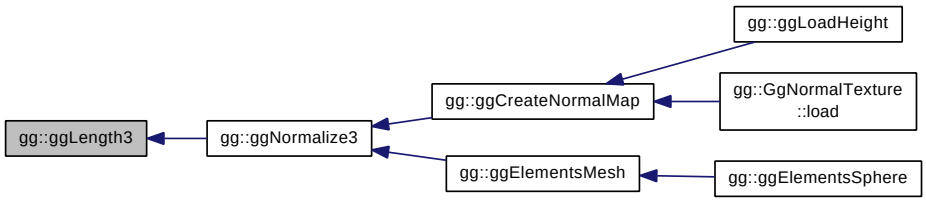
a	GLfloat 型の 3 要素の配列変数.

gg.cpp の 4177 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:



5.1.3.25 GLfloat gg::ggLength4 (const GLfloat * a)

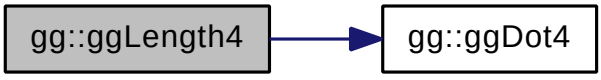
4 要素の長さ.

引数

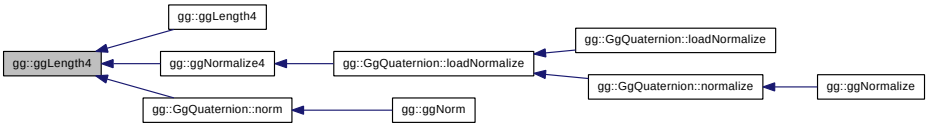
a	GLfloat 型の 4 要素の配列変数.

gg.cpp の 4187 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:



5.1.3.26 GLfloat gg::ggLength4 (const GgVector & *a*) [inline]

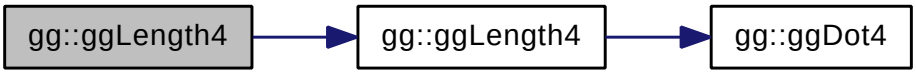
GgVector 型の長さ.

引数

a	GgVector 型の変数.

gg.h の 1547 行目に定義があります。

呼び出し関係図:



5.1.3.27 GLuint gg::ggLoadComputeShader (const char * *comp*)

コンピュートシェーダのソースファイルを読み込んでプログラムオブジェクトを作成する.

引数

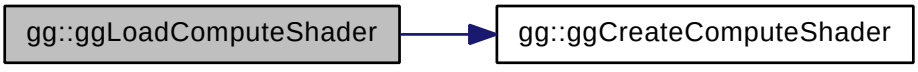
vert	コンピュートシェーダのソースファイル名.

戻り値

プログラムオブジェクトのプログラム名 (作成できなければ 0).

gg.cpp の 4158 行目に定義があります。

呼び出し関係図:



5.1.3.28 GLuint gg::ggLoadHeight (const char * *name*, float *nz*, GLsizei * *pWidth* = nullptr, GLsizei * *pHeight* = nullptr, GLenum *internal* = GL_RGBA)

テクスチャメモリを確保して TGA 画像ファイルを読み込み法線マップを作成する.

引数

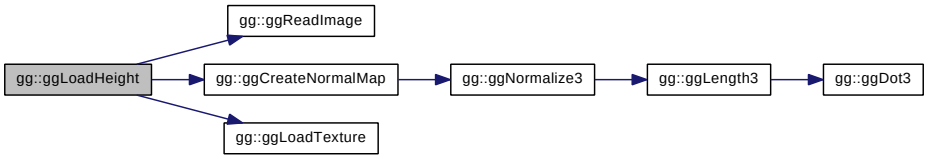
name	読み込むファイル名.
nz	法線の z 成分の割合.
pWidth	読みだした画像ファイルの横の画素数の格納先のポインタ (nullptr なら格納しない). ++
pHeight	読みだした画像ファイルの縦の画素数の格納先のポインタ (nullptr なら格納しない).
internal	glTexImage2D() に指定するテクスチャの内部フォーマット.

戻り値

テクスチャの作成に成功すればテクスチャ名, 失敗すれば 0.

gg.cpp の 3079 行目に定義があります。

呼び出し関係図:



5.1.3.29 GLuint gg::ggLoadImage (const char * *name*, GLsizei * *pWidth* = nullptr, GLsizei * *pHeight* = nullptr, GLenum *internal* = 0, GLenum *wrap* = GL_CLAMP_TO_EDGE)

テクスチャメモリを確保して TGA 画像ファイルを読み込む.

引数

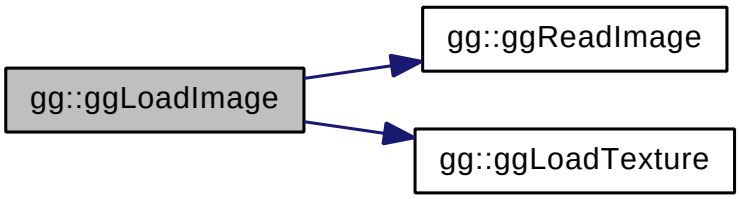
name	読み込むファイル名.
pWidth	読みだした画像ファイルの横の画素数の格納先のポインタ (nullptr なら格納しない). ++
pHeight	読みだした画像ファイルの縦の画素数の格納先のポインタ (nullptr なら格納しない).
internal	glTexImage2D() に指定するテクスチャの内部フォーマット. 0 なら外部フォーマットに合わせる.
wrap	テクスチャのラッピングモード, デフォルトは GL_CLAMP_TO_EDGE.

戻り値

テクスチャの作成に成功すればテクスチャ名, 失敗すれば 0.

gg.cpp の 2945 行目に定義があります。

呼び出し関係図:



5.1.3.30 GLuint gg::ggLoadShader (const char * *vert*, const char * *frag* = nullptr, const char * *geom* = nullptr, GLint *nvarying* = 0, const char *const *varyings[]* = nullptr)

シェーダのソースファイルを読み込んでプログラムオブジェクトを作成する.

引数

vert	バーテックスシェーダのソースファイル名.
frag	フラグメントシェーダのソースファイル名 (nullptr なら不使用).
geom	ジオメトリシェーダのソースファイル名 (nullptr なら不使用).
nvarying	フィードバックする varying 変数の数 (0 なら不使用).
varyings	フィードバックする varying 変数のリスト (nullptr なら不使用).

戻り値

プログラムオブジェクトのプログラム名 (作成できなければ 0).

gg.cpp の 4137 行目に定義があります。

呼び出し関係図:

5.1.3.31 bool gg::ggLoadSimpleObj (const char * *name*, std::vector< std::array< GLuint, 3 > > & *group*, std::vector< GgSimpleShader::Material > & *material*, std::vector< GgVertex > & *vert*, bool *normalize* = `false`)

三角形分割された OBJ ファイルと MTL ファイルを読み込む (Arrays 形式)

引数

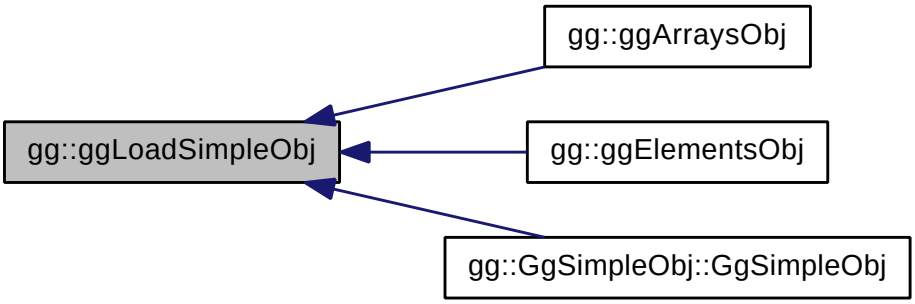
name	読み込む Wavefront OBJ ファイル名.
group	読み込んだデータのポリゴングループごとの最初の三角形の番号と三角形数・材質番号.
material	読み込んだデータのポリゴングループごとの GgSimpleShader::Material 型の材質.
vert	読み込んだデータの頂点属性.
normalize	true なら読み込んだデータの大きさを正規化する.

戻り値

ファイルの読み込みに成功したら true.

gg.cpp の 3715 行目に定義があります。

被呼び出し関係図:



5.1.3.32 bool gg::ggLoadSimpleObj (const char * *name*, std::vector< std::array< GLuint, 3 > > & *group*, std::vector< GgSimpleShader::Material > & *material*, std::vector< GgVertex > & *vert*, std::vector< GLuint > & *face*, bool *normalize* = `false`)

三角形分割された OBJ ファイルを読み込む (Elements 形式).

引数

name	読み込む Wavefront OBJ ファイル名.
group	読み込んだデータのポリゴングループごとの最初の三角形の番号と三角形数・材質番号.
material	読み込んだデータのポリゴングループごとの材質.
vert	読み込んだデータの頂点属性.
face	読み込んだデータの三角形の頂点インデックス.
normalize	true なら読み込んだデータの大きさを正規化する.

戻り値

ファイルの読み込みに成功したら true.

gg.cpp の 3804 行目に定義があります。

5.1.3.33 GLuint gg::ggLoadTexture (const GLvoid * *image*, GLsizei *width*, GLsizei *height*, GLenum *format* = GL_BGR, GLenum *type* = GL_UNSIGNED_BYTE, GLenum *internal* = GL_RGB, GLenum *wrap* = GL_CLAMP_TO_EDGE)

テクスチャメモリを確保して画像データをテクスチャとして読み込む.

引数

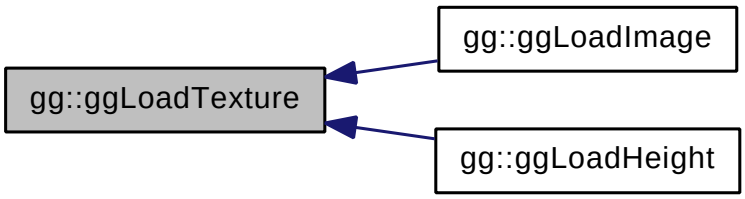
image	テクスチャとして読み込むデータ. nullptr ならテクスチャメモリの確保のみを行う.
width	テクスチャとして読み込むデータ image の横の画素数.
height	テクスチャとして読み込むデータ image の縦の画素数.
format	image のフォーマット.
type	image のデータ型.
internal	テクスチャの内部フォーマット.
wrap	テクスチャのラッピングモード, デフォルトは GL_CLAMP_TO_EDGE.

戻り値

テクスチャの作成に成功すればテクスチャ名, 失敗すれば 0.

gg.cpp の 2911 行目に定義があります。

被呼び出し関係図:



5.1.3.34 GgMatrix gg::ggLookat (GLfloat *ex*, GLfloat *ey*, GLfloat *ez*, GLfloat *tx*, GLfloat *ty*, GLfloat *tz*, GLfloat *ux*, GLfloat *uy*, GLfloat *uz*) [inline]

ビュー変換行列を返す.

引数

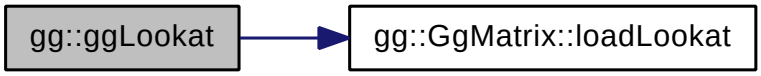
ex	視点の位置の x 座標値.
ey	視点の位置の y 座標値.
ez	視点の位置の z 座標値.
tx	目標点の位置の x 座標値.
ty	目標点の位置の y 座標値.
tz	目標点の位置の z 座標値.
ux	上方向のベクトルの x 成分.
uy	上方向のベクトルの y 成分.
uz	上方向のベクトルの z 成分.

戻り値

求めたビュー変換行列.

gg.h の 2547 行目に定義があります。

呼び出し関係図:



5.1.3.35 GgMatrix gg::ggLookat (const GLfloat * *e*, const GLfloat * *t*, const GLfloat * *u*) [inline]

ビュー変換行列を返す.

引数

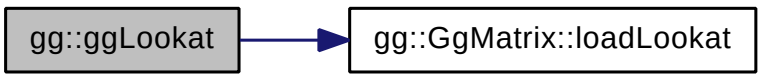
e	視点の位置を格納した GLfloat 型の 3 要素の配列変数.
t	目標点の位置を格納した GLfloat 型の 3 要素の配列変数.
u	上方向のベクトルを格納した GLfloat 型の 3 要素の配列変数.

戻り値

求めたビュー変換行列.

gg.h の 2562 行目に定義があります。

呼び出し関係図:



5.1.3.36 GgMatrix gg::ggLookat (const GgVector & *e*, const GgVector & *t*, const GgVector & *u*) [inline]

ビュー変換行列を返す.

引数

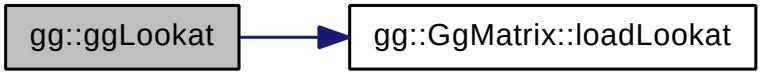
e	視点の位置を格納した GgVector 型の変数.
t	目標点の位置を格納した GgVector 型の変数.
u	上方向のベクトルを格納した GgVector 型の変数.

戻り値

求めたビュー変換行列.

gg.h の 2577 行目に定義があります。

呼び出し関係図:



5.1.3.37 GgQuaternion gg::ggMatrixQuaternion (const GLfloat * *a*) [inline]

回転の変換行列 m を表す四元数を返す.

引数

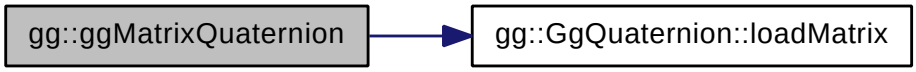
m	GLfloat 型の 16 要素の配列変数.

戻り値

m による回転の変換に相当する四元数.

gg.h の 3520 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:

5.1.3.38 GgQuaternion gg::ggMatrixQuaternion (const GgMatrix & *m*) [inline]

回転の変換行列 m を表す四元数を返す.

引数

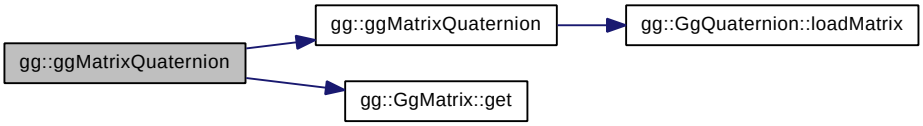
m	GgMatrix 型の変換行列.

戻り値

m による回転の変換に相当する四元数.

gg.h の 3529 行目に定義があります。

呼び出し関係図:



5.1.3.39 GLfloat gg::ggNorm (const GgQuaternion & *q*) [inline]

四元数のノルムを返す.

引数

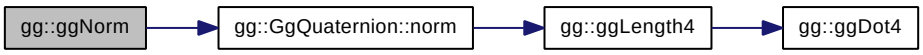
q	GgQuaternion 型の四元数.

戻り値

四元数 q のノルム.

gg.h の 3648 行目に定義があります。

呼び出し関係図:



5.1.3.40 GgMatrix gg::ggNormal (const GgMatrix & *m*) [inline]

法線変換行列を返す.

引数

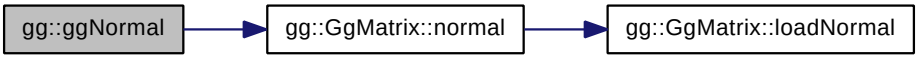
m	元の変換行列.

戻り値

m の法線変換行列.

gg.h の 2651 行目に定義があります。

呼び出し関係図:



5.1.3.41 GgQuaternion gg::ggNormalize (const GgQuaternion & *q*) [inline]

正規化した四元数を返す.

引数

q	GgQuaternion 型の四元数.

戻り値

四元数 q を正規化した四元数.

gg.h の 3656 行目に定義があります。

呼び出し関係図:

5.1.3.42 void gg::ggNormalize3 (GLfloat * *a*) [inline]

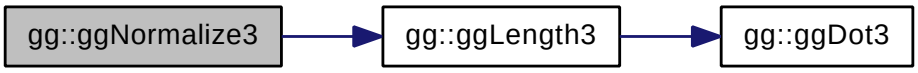
3 要素の正規化.

引数

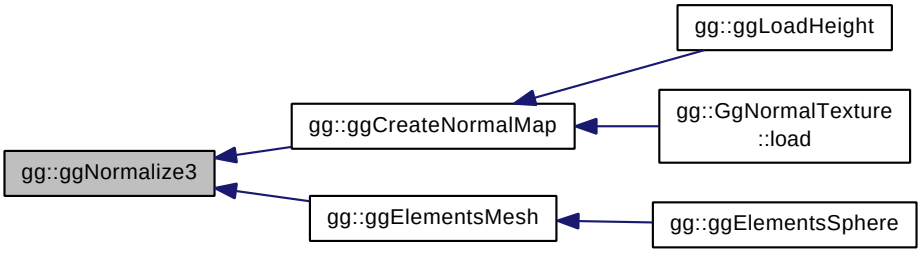
a	GLfloat 型の 3 要素の配列変数.

gg.h の 1499 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:



5.1.3.43 void gg::ggNormalize4 (GLfloat * *a*) [inline]

4 要素の正規化.

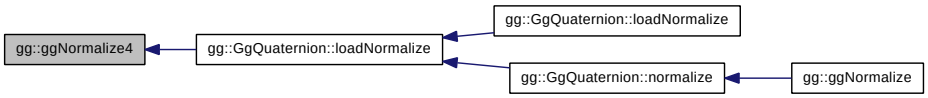
引数

a	GLfloat 型の 4 要素の配列変数.

gg.h の 1557 行目に定義があります。

呼び出し関係図:

被呼び出し関係図:



5.1.3.44 void gg::ggNormalize4 (GgVector & *a*) [inline]

GgVector 型の正規化.

引数

a	GgVector 型の変数

gg.h の 1574 行目に定義があります。

呼び出し関係図:

5.1.3.45 GgMatrix gg::ggOrthogonal (GLfloat *left*, GLfloat *right*, GLfloat *bottom*, GLfloat *top*, GLfloat *zNear*, GLfloat *zFar*) [inline]

直交投影変換行列を返す.

引数

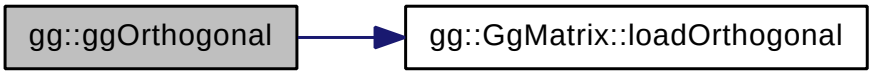
left	ウィンドウの左端の位置.
right	ウィンドウの右端の位置.
bottom	ウィンドウの下端の位置.
top	ウィンドウの上端の位置.
zNear	視点から前方面までの位置.
zFar	視点から後方面までの位置.

戻り値

求めた直交投影変換行列.

gg.h の 2595 行目に定義があります。

呼び出し関係図:



5.1.3.46 GgMatrix gg::ggPerspective (GLfloat *fovy*, GLfloat *aspect*, GLfloat *zNear*, GLfloat *zFar*) [inline]

画角を指定して透視投影変換行列を返す.

引数

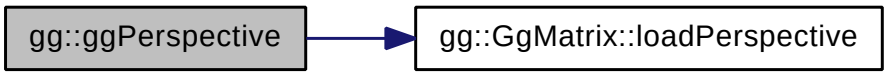
fovy	y 方向の画角.
aspect	縦横比.
zNear	視点から前方面までの位置.
zFar	視点から後方面までの位置.

戻り値

求めた透視投影変換行列.

gg.h の 2625 行目に定義があります。

呼び出し関係図:



5.1.3.47 gg::GgPoints * gg::ggPointsCube (GLsizei *countv*, GLfloat *length* = 1.0f, GLfloat *cx* = 0.0f, GLfloat *cy* = 0.0f, GLfloat *cz* = 0.0f)

点群を立方体状に生成する.

引数

countv	生成する点の数.
length	点群を生成する立方体の一辺の長さ.
cx	点群の中心の x 座標.
cy	点群の中心の y 座標.
cz	点群の中心の z 座標.

gg.cpp の 4981 行目に定義があります。

5.1.3.48 gg::GgPoints * gg::ggPointsSphere (GLsizei *countv*, GLfloat *radius* = 0.5f, GLfloat *cx* = 0.0f, GLfloat *cy* = 0.0f, GLfloat *cz* = 0.0f)

点群を球状に生成する.

引数

countv	生成する点の数.
radius	点群を生成する半径.
cx	点群の中心の x 座標.
cy	点群の中心の y 座標.
cz	点群の中心の z 座標.

gg.cpp の 5011 行目に定義があります。

5.1.3.49 GgQuaternion gg::ggQuaternion (GLfloat *x*, GLfloat *y*, GLfloat *z*, GLfloat *w*) [inline]

四元数を返す

引数

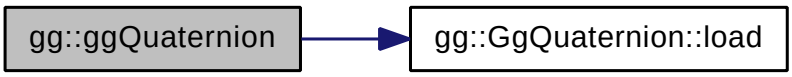
x	四元数の x 要素.
y	四元数の y 要素.
z	四元数の z 要素.
w	四元数の w 要素.

戻り値

四元数.

gg.h の 3495 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:

5.1.3.50 GgQuaternion gg::ggQuaternion (const GLfloat * *a*) [inline]

四元数を返す

引数

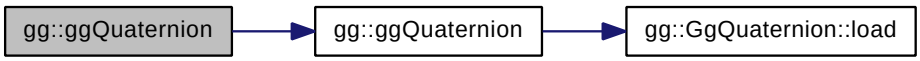
a	GLfloat 型の GLfloat 型の 4 要素の配列変数に格納した四元数.

戻り値

四元数.

gg.h の 3504 行目に定義があります。

呼び出し関係図:



5.1.3.51 GgMatrix gg::ggQuaternionMatrix (const GgQuaternion & *q*) [inline]

四元数 q の回転の変換行列を返す.

引数

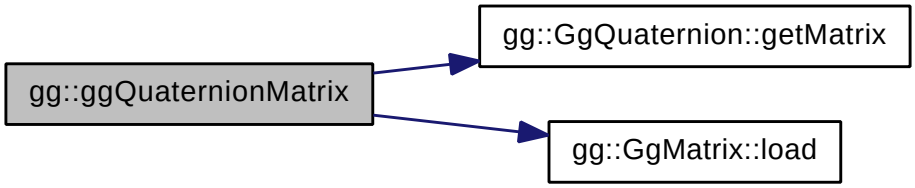
q	元の四元数.

戻り値

四元数 q が表す回転に相当する GgMatrix 型の変換行列.

gg.h の 3537 行目に定義があります。

呼び出し関係図:



5.1.3.52 GgMatrix gg::ggQuaternionTransposeMatrix (const GgQuaternion & *q*) [inline]

四元数 q の回転の転置した変換行列を返す.

引数

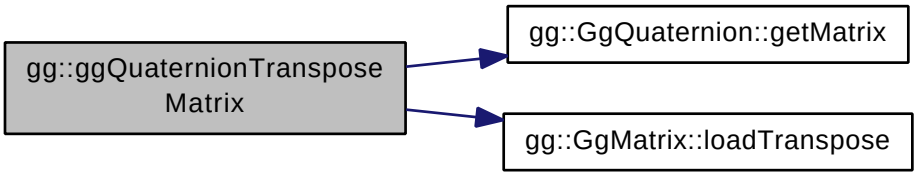
q	元の四元数.

戻り値

四元数 q が表す回転に相当する転置した GgMatrix 型の変換行列.

gg.h の 3548 行目に定義があります。

呼び出し関係図:



5.1.3.53 bool gg::ggReadImage (const char * *name*, std::vector< GLubyte > & *image*, GLsizei * *pWidth*, GLsizei * *pHeight*, GLenum * *pFormat*)

TGA ファイル (8/16/24/32bit) をメモリに読み込む.

引数

name	読み込むファイル名.
image	読み込んだデータを格納する vector.

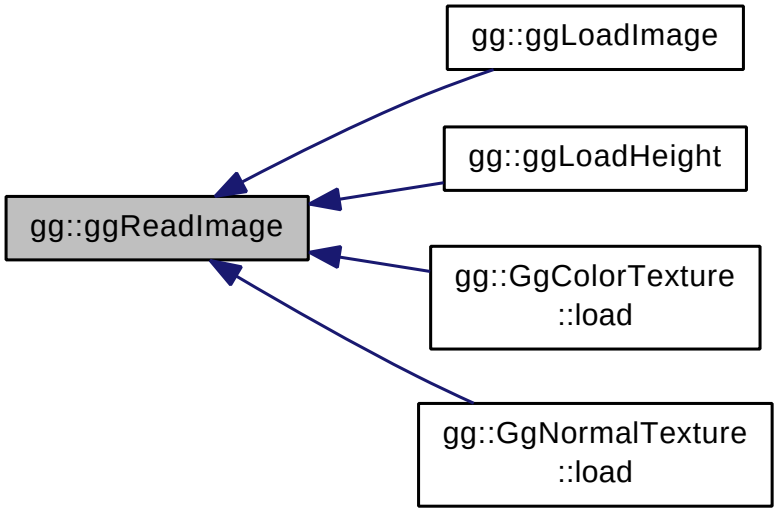
pWidth	読み込んだ画像の横の画素数の格納先のポインタ. nullptr なら格納しない.
pHeight	読み込んだ画像の縦の画素数の格納先のポインタ. nullptr なら格納しない.
pFormat	読み込んだファイルの書式 (GL_RED, G_RG, GL_BGR, G_BGRA) の格納先のポインタ. nullptr なら格納しない.

戻り値

読み込みに成功すれば true, 失敗すれば false.

gg.cpp の 2799 行目に定義があります。

被呼び出し関係図:



5.1.3.54 gg::GgTriangles * gg::ggRectangle (GLfloat *width* = 1.0f, GLfloat *height* = 1.0f)

矩形状に 2 枚の三角形を生成する.

引数

width	矩形の横幅.
height	矩形の高さ.

gg.cpp の 5043 行目に定義があります。

5.1.3.55 GgMatrix gg::ggRotate (GLfloat *x*, GLfloat *y*, GLfloat *z*, GLfloat *a*) [inline]

(x, y, z) 方向のベクトルを軸とする回転の変換行列を乗じた結果を返す.

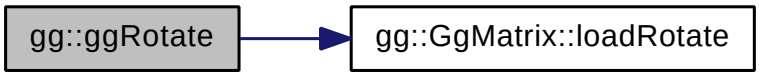
引数

x	回転軸の x 成分.
y	回転軸の y 成分.
z	回転軸の z 成分.
a	回転角.

戻り値

(x, y, z) を軸にさらに a 回転する変換行列.

gg.h の 2492 行目に定義があります。
呼び出し関係図:



5.1.3.56 GgMatrix gg::ggRotate (const GLfloat * r, GLfloat a) [inline]

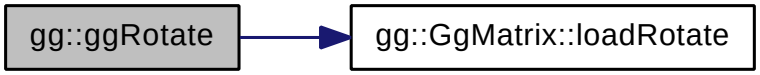
r 方向のベクトルを軸とする回転の変換行列を乗じた結果を返す.
引数

r	回転軸のベクトルを表す GLfloat 型の 3 要素の配列変数.
a	回転角.

戻り値

r を軸に a だけ回転する変換行列.

gg.h の 2502 行目に定義があります。
呼び出し関係図:



5.1.3.57 GgMatrix gg::ggRotate (const GgVector & r, GLfloat a) [inline]

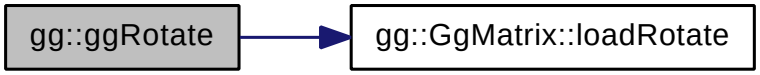
r 方向のベクトルを軸とする回転の変換行列を乗じた結果を返す.
引数

r	回転軸のベクトルを表す GgVector 型の変数.
a	回転角.

戻り値

r を軸に a だけ回転する変換行列.

gg.h の 2512 行目に定義があります。
呼び出し関係図:



5.1.3.58 GgMatrix gg::ggRotate (const GLfloat * r) [inline]

r 方向のベクトルを軸とする回転の変換行列を乗じた結果を返す.

引数

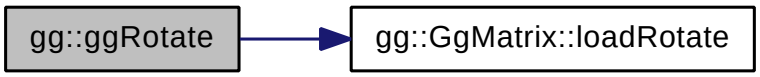
r	回転軸のベクトルと回転角を表す GLfloat 型の 4 要素の配列変数.

戻り値

(r[0], r[1], r[2]) を軸に r[3] だけ回転する変換行列.

gg.h の 2521 行目に定義があります。

呼び出し関係図:



5.1.3.59 GgMatrix gg::ggRotate (const GgVector & *r*) [inline]

r 方向のベクトルを軸とする回転の変換行列を乗じた結果を返す.

引数

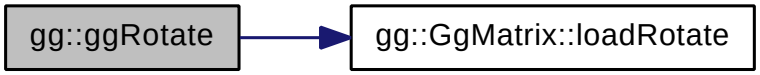
r	回転軸のベクトルと回転角を表す GgVector 型の変数.

戻り値

(r[0], r[1], r[2]) を軸に r[3] だけ回転する変換行列.

gg.h の 2530 行目に定義があります。

呼び出し関係図:



5.1.3.60 GgQuaternion gg::ggRotateQuaternion (GLfloat *x*, GLfloat *y*, GLfloat *z*, GLfloat *a*) [inline]

(x, y, z) を軸として角度 a 回転する四元数を返す.

引数

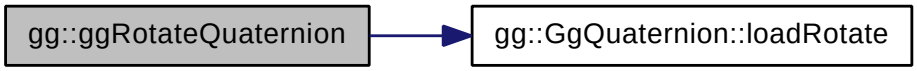
x	軸ベクトルの x 成分.
y	軸ベクトルの y 成分.
z	軸ベクトルの z 成分.
a	回転角.

戻り値

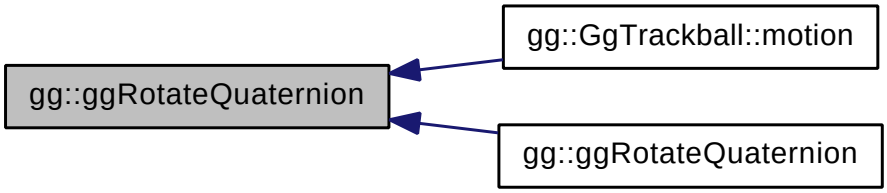
回転を表す四元数.

gg.h の 3562 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:



5.1.3.61 GgQuaternion gg::ggRotateQuaternion (const GLfloat * *v*, GLfloat *a*) [inline]

(v[0], v[1], v[2]) を軸として角度 a 回転する四元数を返す.

引数

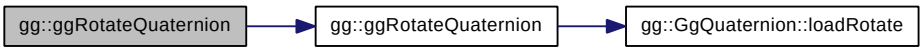
v	軸ベクトルを表す GLfloat 型の 3 要素の配列変数.
a	回転角.

戻り値

回転を表す四元数.

gg.h の 3572 行目に定義があります。

呼び出し関係図:



5.1.3.62 GgQuaternion gg::ggRotateQuaternion (const GLfloat * *v*) [inline]

(v[0], v[1], v[2]) を軸として角度 v[3] 回転する四元数を返す.

引数

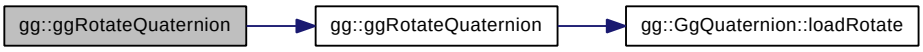
v	軸ベクトルを表す GLfloat 型の 4 要素の配列変数.

戻り値

回転を表す四元数.

gg.h の 3580 行目に定義があります。

呼び出し関係図:



5.1.3.63 GgMatrix gg::ggRotateX (GLfloat *a*) [inline]

x 軸中心の回転の変換行列を返す.

引数

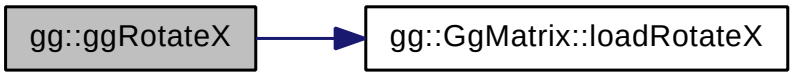
a	回転角.

戻り値

x 軸中心に a だけ回転する変換行列.

gg.h の 2462 行目に定義があります。

呼び出し関係図:



5.1.3.64 GgMatrix gg::ggRotateY (GLfloat *a*) [inline]

y 軸中心の回転の変換行列を返す.

引数

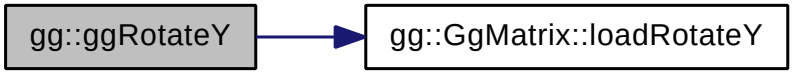
a	回転角.

戻り値

y 軸中心に a だけ回転する変換行列.

gg.h の 2471 行目に定義があります。

呼び出し関係図:



5.1.3.65 GgMatrix gg::ggRotateZ (GLfloat *a*) [inline]

z 軸中心の回転の変換行列を返す.

引数

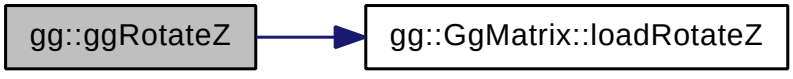
a	回転角.

戻り値

z 軸中心に a だけ回転する変換行列.

gg.h の 2480 行目に定義があります。

呼び出し関係図:



5.1.3.66 bool gg::ggSaveColor (const char * *name*)

カラーバッファの内容を TGA ファイルに保存する.

引数

name	保存するファイル名.

戻り値

保存に成功すれば true, 失敗すれば false.

gg.cpp の 2743 行目に定義があります。

呼び出し関係図:

5.1.3.67 bool gg::ggSaveDepth (const char * *name*)

デプスバッファの内容を TGA ファイルに保存する.

引数

name	保存するファイル名.

戻り値

保存に成功すれば true, 失敗すれば false.

gg.cpp の 2769 行目に定義があります。

呼び出し関係図:

5.1.3.68 bool gg::ggSaveTga (const char * *name*, const void * *buffer*, unsigned int *width*, unsigned int *height*, unsigned int *depth*)

配列の内容を TGA ファイルに保存する.

引数

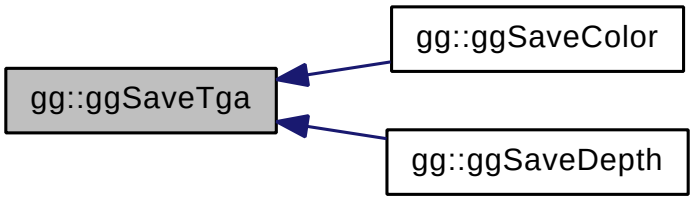
name	保存するファイル名.
buffer	画像データを格納した配列.
width	画像の横の画素数.
height	画像の縦の画素数.
depth	1 画素のバイト数.

戻り値

保存に成功すれば true, 失敗すれば false.

gg.cpp の 2660 行目に定義があります。

被呼び出し関係図:



5.1.3.69 GgMatrix gg::ggScale (GLfloat *x*, GLfloat *y*, GLfloat *z*, GLfloat *w* = 1.0f) [inline]

拡大縮小の変換行列を返す.

引数

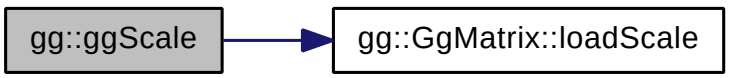
x	x 方向の拡大率.
y	y 方向の拡大率.
z	z 方向の拡大率.
w	拡大率のスケールファクタ (= 1.0f).

戻り値

拡大縮小の変換行列.

gg.h の 2435 行目に定義があります。

呼び出し関係図:



5.1.3.70 GgMatrix gg::ggScale (const GLfloat * *s*) [inline]

拡大縮小の変換行列を返す.

引数

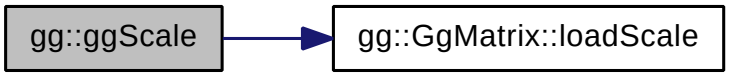
s	拡大率の GLfloat 型の 3 要素の配列変数 (x, y, z).

戻り値

拡大縮小の変換行列.

gg.h の 2444 行目に定義があります。

呼び出し関係図:



5.1.3.71 GgMatrix gg::ggScale (const GgVector & *s*) [inline]

拡大縮小の変換行列を返す.

引数

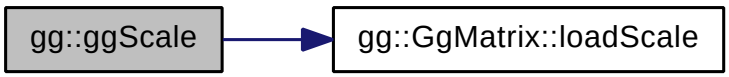
s	拡大率の GgVector 型の変数.

戻り値

拡大縮小の変換行列.

gg.h の 2453 行目に定義があります。

呼び出し関係図:



5.1.3.72 GgQuaternion gg::ggSlerp (const GLfloat * *a*, const GLfloat * *b*, GLfloat *t*) [inline]

二つの四元数の球面線形補間の結果を返す.

引数

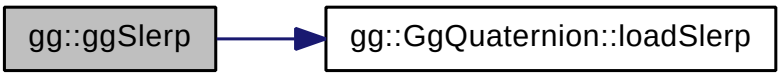
a	四元数を格納した GLfloat 型の 4 要素の配列変数.
b	四元数を格納した GLfloat 型の 4 要素の配列変数.
t	補間パラメータ.

戻り値

a, b を t で内分した四元数.

gg.h の 3609 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:

5.1.3.73 GgQuaternion gg::ggSlerp (const GgQuaternion & *q*, const GgQuaternion & *r*, GLfloat *t*) [inline]

二つの四元数の球面線形補間の結果を返す.

引数

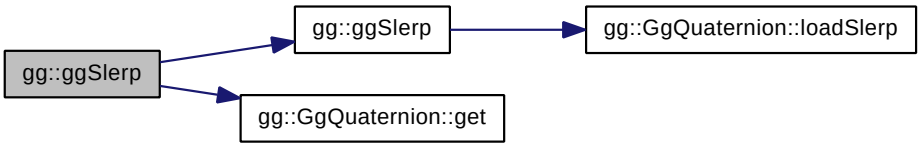
q	GgQuaternion 型の四元数.
r	GgQuaternion 型の四元数.
t	補間パラメータ.

戻り値

q, r を t で内分した四元数.

gg.h の 3620 行目に定義があります。

呼び出し関係図:



5.1.3.74 GgQuaternion gg::ggSlerp (const GgQuaternion & *q*, const GLfloat * *a*, GLfloat *t*) [inline]

二つの四元数の球面線形補間の結果を返す.

引数

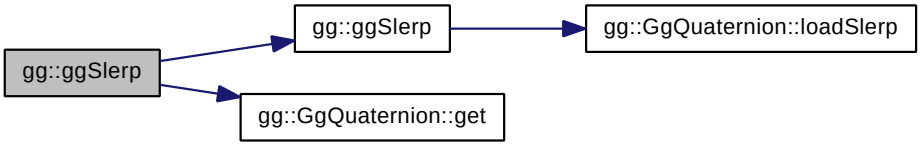
q	GgQuaternion 型の四元数.
a	四元数を格納した GLfloat 型の 4 要素の配列変数.
t	補間パラメータ.

戻り値

q, a を t で内分した四元数.

gg.h の 3630 行目に定義があります。

呼び出し関係図:



5.1.3.75 GgQuaternion gg::ggSlerp (const GLfloat * *a*, const GgQuaternion & *q*, GLfloat *t*) [inline]

二つの四元数の球面線形補間の結果を返す.

引数

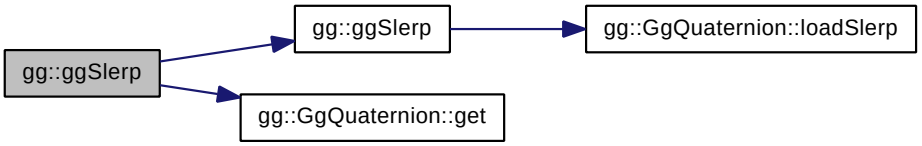
a	四元数を格納した GLfloat 型の 4 要素の配列変数.
q	GgQuaternion 型の四元数.
t	補間パラメータ.

戻り値

a, q を t で内分した四元数.

gg.h の 3640 行目に定義があります。

呼び出し関係図:



5.1.3.76 GgMatrix gg::ggTranslate (GLfloat *x*, GLfloat *y*, GLfloat *z*, GLfloat *w* = 1.0f) [inline]

平行移動の変換行列を返す.

引数

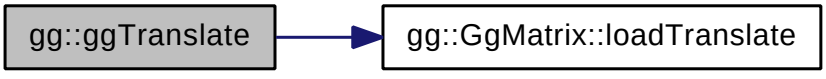
x	x 方向の移動量.
y	y 方向の移動量.
z	z 方向の移動量.
w	移動量のスケールファクタ (= 1.0f).

戻り値

平行移動の変換行列.

gg.h の 2405 行目に定義があります。

呼び出し関係図:



5.1.3.77 GgMatrix gg::ggTranslate (const GLfloat * *t*) [inline]

平行移動の変換行列を返す.

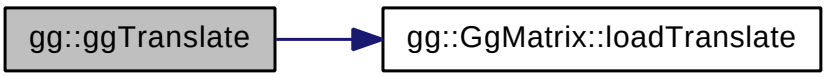
引数

t	移動量の GLfloat 型の 3 要素の配列変数 (x, y, z).

戻り値

平行移動の変換行列

gg.h の 2414 行目に定義があります。
呼び出し関係図:



5.1.3.78 GgMatrix gg::ggTranslate (const GgVector & *t*) [inline]

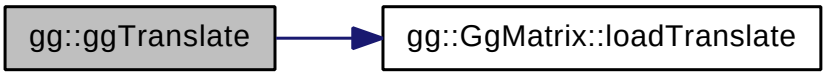
平行移動の変換行列を返す.
引数

t	移動量の GgVector 型の変数.

戻り値

平行移動の変換行列

gg.h の 2423 行目に定義があります。
呼び出し関係図:



5.1.3.79 GgMatrix gg::ggTranspose (const GgMatrix & *m*) [inline]

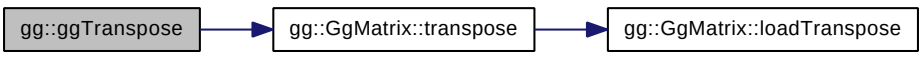
転置行列を返す.
引数

m	元の変換行列.

戻り値

m の転置行列.

gg.h の 2635 行目に定義があります。
呼び出し関係図:



5.1.4 変数詳解

5.1.4.1 GLint gg::ggBufferAlignment

使用している GPU のバッファオブジェクトのアライメント
使用している GPU のバッファアライメント

Chapter 6

クラス詳解

6.1 gg::GgBuffer< T > クラステンプレート

バッファオブジェクト.

`#include <gg.h>`

公開メンバ関数

- virtual ~GgBuffer ()
  デストラクタ.
- GgBuffer (GLenum target, const T *data, GLsizei stride, GLsizei count, GLenum usage)
  コンストラクタ.
- GLuint getTarget () const
  バッファオブジェクトのターゲットを取り出す.
- GLsizeiptr getStride () const
  バッファオブジェクトのアライメントを考慮したデータの間隔を取り出す.
- GLsizei getCount () const
  バッファオブジェクトが保持するデータの数を取り出す.
- GLuint getBuffer () const
  バッファオブジェクト名を取り出す.
- void bind () const
  バッファオブジェクトを結合する.
- void unbind () const
  バッファオブジェクトを解放する.
- void * map () const
  バッファオブジェクトをマップする.
- void * map (GLint first, GLsizei count) const
  バッファオブジェクトの指定した範囲をマップする.
- void unmap () const
  バッファオブジェクトをアンマップする.
- void send (const T *data, GLint first, GLsizei count) const
  すでに確保したバッファオブジェクトにデータを転送する.
- void read (T *data, GLint first, GLsizei count) const
  バッファオブジェクトのデータから抽出する.
- void copy (GLuint src_buffer, GLint src_first=0, GLint dst_first=0, GLsizei count=0) const
  別のバッファオブジェクトからデータを複写する.

6.1.1 詳解

template<typename T>class gg::GgBuffer< T >

バッファオブジェクト.

頂点属性／頂点インデックス／ユニフォーム変数を格納するバッファオブジェクトの基底クラス.

gg.h の 4037 行目に定義があります。

6.1.2 構築子と解体子

6.1.2.1 template<typename T> virtual gg::GgBuffer< T >::~GgBuffer ( ) [inline], [virtual]

デストラクタ.

gg.h の 4060 行目に定義があります。

6.1.2.2 template<typename T> gg::GgBuffer< T >::GgBuffer (GLenum *target*, const T * *data*, GLsizei *stride*, GLsizei *count*, GLenum *usage*) [inline]

コンストラクタ.

引数

target	バッファオブジェクトのターゲット.
data	データが格納されている領域の先頭のポインタ (nullptr ならデータを転送しない).
count	データの数.
stride	データの間隔.
usage	バッファオブジェクトの使い方.

gg.h の 4073 行目に定義があります。

呼び出し関係図:

6.1.3 関数詳解

6.1.3.1 template<typename T> void gg::GgBuffer< T >::bind ( ) const [inline]

バッファオブジェクトを結合する.

gg.h の 4113 行目に定義があります。

6.1.3.2 template<typename T> void gg::GgBuffer< T >::copy (GLuint *src_buffer*, GLint *src_first* = 0, GLint *dst_first* = 0, GLsizei *count* = 0) const [inline]

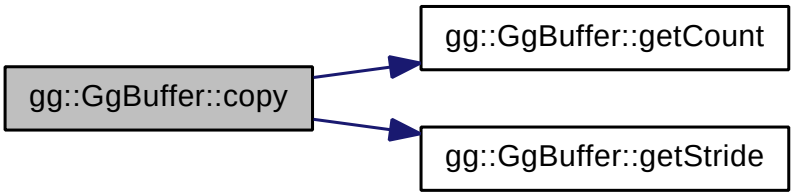
別のバッファオブジェクトからデータを複写する.

引数

src_buffer	複写元のバッファオブジェクト名.
src_first	複写元 (src_buffer) の先頭のデータの位置.
dst_first	複写先 (getBuffer()) の先頭のデータの位置.
count	複写するデータの数 (0 ならバッファオブジェクト全体).

gg.h の 4187 行目に定義があります。

呼び出し関係図:



6.1.3.3 template<typename T> GLuint gg::GgBuffer< T >::getBuffer ( ) const [inline]

バッファオブジェクト名を取り出す.

戻り値

このバッファオブジェクト名.

gg.h の 4107 行目に定義があります。

6.1.3.4 template<typename T> GLsizei gg::GgBuffer< T >::getCount ( ) const [inline]

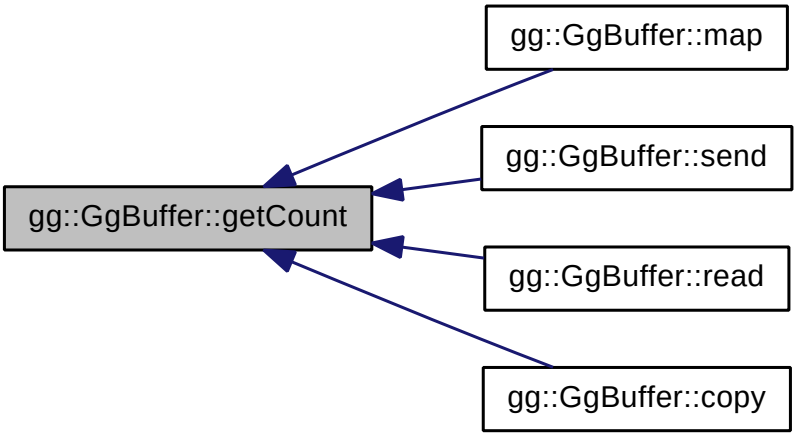
バッファオブジェクトが保持するデータの数を取り出す.

戻り値

このバッファオブジェクトが保持するデータの数.

gg.h の 4100 行目に定義があります。

被呼び出し関係図:



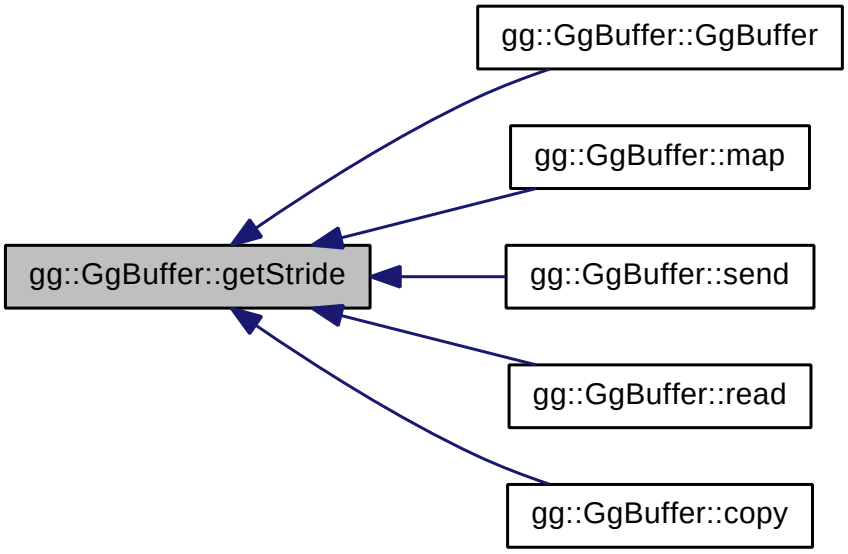
6.1.3.5 template<typename T> GLsizeiptr gg::GgBuffer< T >::getStride ( ) const [inline]

バッファオブジェクトのアライメントを考慮したデータの間隔を取り出す.

戻り値

このバッファオブジェクトのデータの間隔.

gg.h の 4093 行目に定義があります。
被呼び出し関係図:



6.1.3.6 template<typename T> GLuint gg::GgBuffer< T >::getTarget ( ) const [inline]

バッファオブジェクトのターゲットを取り出す.

戻り値

このバッファオブジェクトのターゲット.

gg.h の 4086 行目に定義があります。

6.1.3.7 template<typename T> void* gg::GgBuffer< T >::map ( ) const [inline]

バッファオブジェクトをマップする.

戻り値

マップしたメモリの先頭のポインタ.

gg.h の 4126 行目に定義があります。

6.1.3.8 template<typename T> void* gg::GgBuffer< T >::map (GLint *first*, GLsizei *count*) const [inline]

バッファオブジェクトの指定した範囲をマップする.

引数

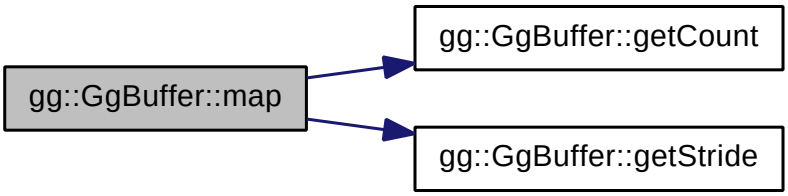
first	マップする範囲のバッファオブジェクトの先頭からの位置.
count	マップするデータの数 (0 ならバッファオブジェクト全体).

戻り値

マップしたメモリの先頭のポインタ.

gg.h の 4136 行目に定義があります。

呼び出し関係図:



6.1.3.9 template<typename T> void gg::GgBuffer< T >::read (T * *data*, GLint *first*, GLsizei *count*) const [inline]

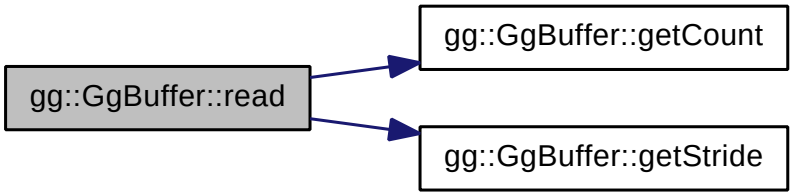
バッファオブジェクトのデータから抽出する.

引数

data	抽出先の領域の先頭のポインタ.
first	抽出元のバッファオブジェクトの取り出すデータの領域の先頭の要素番号.
count	抽出するデータの数 (0 ならバッファオブジェクト全体).

gg.h の 4171 行目に定義があります。

呼び出し関係図:



6.1.3.10 template<typename T> void gg::GgBuffer< T >::send (const T * *data*, GLint *first*, GLsizei *count*) const [inline]

すでに確保したバッファオブジェクトにデータを転送する.

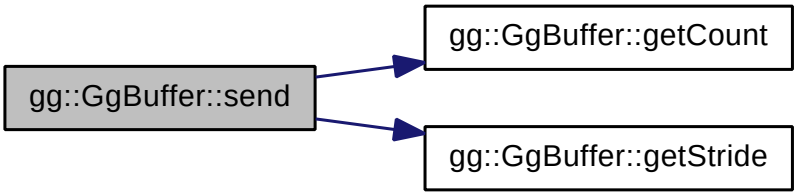
引数

data	転送元のデータが格納されてている領域の先頭のポインタ.
first	転送先のバッファオブジェクトの先頭の要素番号.

count	転送するデータの数 (0 ならバッファオブジェクト全体).

gg.h の 4156 行目に定義があります。

呼び出し関係図:



6.1.3.11 template<typename T> void gg::GgBuffer< T >::unbind ( ) const [inline]

バッファオブジェクトを解放する.

gg.h の 4119 行目に定義があります。

6.1.3.12 template<typename T> void gg::GgBuffer< T >::unmap ( ) const [inline]

バッファオブジェクトをアンマップする.

gg.h の 4147 行目に定義があります。

このクラス詳解は次のファイルから抽出されました:

- gg.h

6.2 gg::GgColorTexture クラス

カラーマップ.

`#include <gg.h>`

公開メンバ関数

- virtual ~GgColorTexture ()
  デストラクタ.
- GgColorTexture ()
  コンストラクタ.
- GgColorTexture (const GLvoid *image, GLsizei width, GLsizei height, GLenum format=GL_BGR, GLenum type=GL_UNSIGNED_BYTE, GLenum internal=GL_RGB, GLenum wrap=GL_CLAMP_TO_EDGE)
  メモリ上のデータからテクスチャを作成するコンストラクタ.
- GgColorTexture (const char *name, GLenum internal=0, GLenum wrap=GL_CLAMP_TO_EDGE)
  ファイルからデータを読み込んでテクスチャを作成するコンストラクタ.
- void load (const GLvoid *image, GLsizei width, GLsizei height, GLenum format=GL_BGR, GLenum type=GL_UNSIGNED_BYTE, GLenum internal=GL_RGB, GLenum wrap=GL_CLAMP_TO_EDGE)
  テクスチャを作成してメモリ上のデータを読み込む.
- void load (const char *name, GLenum internal=0, GLenum wrap=GL_CLAMP_TO_EDGE)
  テクスチャを作成してファイルからデータを読み込む.

6.2.1 詳解

カラーマップ.

カラー画像を読み込んでテクスチャを作成する.

gg.h の 3903 行目に定義があります。

6.2.2 構築子と解体子

6.2.2.1 virtual gg::GgColorTexture::~GgColorTexture ( ) [inline], [virtual]

デストラクタ.

gg.h の 3911 行目に定義があります。

6.2.2.2 gg::GgColorTexture::GgColorTexture ( ) [inline]

コンストラクタ.

gg.h の 3914 行目に定義があります。

6.2.2.3 gg::GgColorTexture::GgColorTexture (const GLvoid * *image*, GLsizei *width*, GLsizei *height*, GLenum *format* = GL_BGR, GLenum *type* = GL_UNSIGNED_BYTE, GLenum *internal* = GL_RGB, GLenum *wrap* = GL_CLAMP_TO_EDGE) [inline]

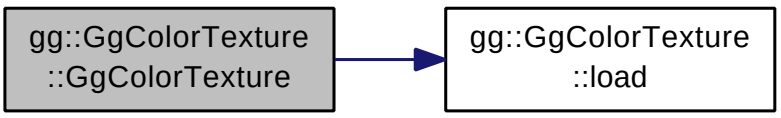
メモリ上のデータからテクスチャを作成するコンストラクタ.

引数

image	テクスチャとして用いる画像データ, nullptr ならデータを読み込まない.
width	読み込む画像の横の画素数.
height	読み込む画像の縦の画素数.
format	読み込む画像のフォーマット.
type	読み込む画像のデータ型.
internal	テクスチャの内部フォーマット.
wrap	テクスチャのラッピングモード.

gg.h の 3924 行目に定義があります。

呼び出し関係図:



6.2.2.4 gg::GgColorTexture::GgColorTexture (const char * *name*, GLenum *internal* = 0, GLenum *wrap* = GL_CLAMP_TO_EDGE) [inline]

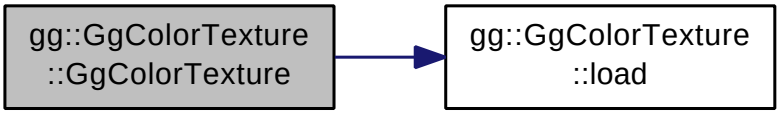
ファイルからデータを読み込んでテクスチャを作成するコンストラクタ.

引数

name	読み込むファイル名.
internal	glTexImage2D() に指定するテクスチャの内部フォーマット. 0 なら外部フォーマットに合わせる.

gg.h の 3934 行目に定義があります。

呼び出し関係図:



6.2.3 関数詳解

6.2.3.1 void gg::GgColorTexture::load (const GLvoid * *image*, GLsizei *width*, GLsizei *height*, GLenum *format* = GL_BGR, GLenum *type* = GL_UNSIGNED_BYTE, GLenum *internal* = GL_RGB, GLenum *wrap* = GL_CLAMP_TO_EDGE) [inline]

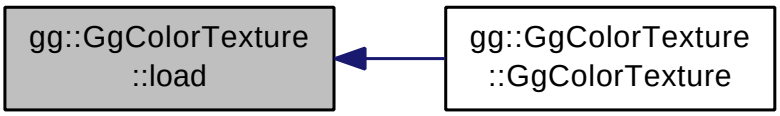
テクスチャを作成してメモリ上のデータを読み込む.

引数

image	テクスチャとして用いる画像データ, nullptr ならデータを読み込まない.
width	テクスチャの横の画素数.
height	テクスチャの縦の画素数.
format	読み込む画像のフォーマット.
type	読み込む画像のデータ型.
internal	テクスチャの内部フォーマット.
wrap	テクスチャのラッピングモード.

gg.h の 3947 行目に定義があります。

被呼び出し関係図:



6.2.3.2 void gg::GgColorTexture::load (const char * *name*, GLenum *internal* = 0, GLenum *wrap* = GL_CLAMP_TO_EDGE)

テクスチャを作成してファイルからデータを読み込む.

引数

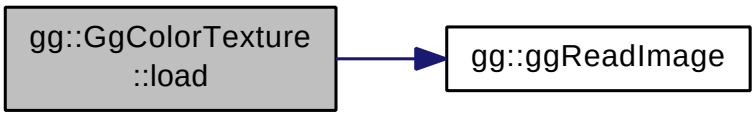
name	読み込むファイル名.
internal	glTexImage2D() に指定するテクスチャの内部フォーマット. 0 なら外部フォーマットに合わせる.

戻り値

テクスチャの作成に成功すれば true, 失敗すれば false.

gg.cpp の 3117 行目に定義があります。

呼び出し関係図:



このクラス詳解は次のファイルから抽出されました:

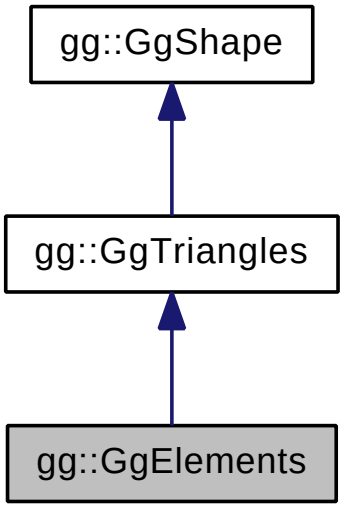
- gg.h
- gg.cpp

6.3 gg::GgElements クラス

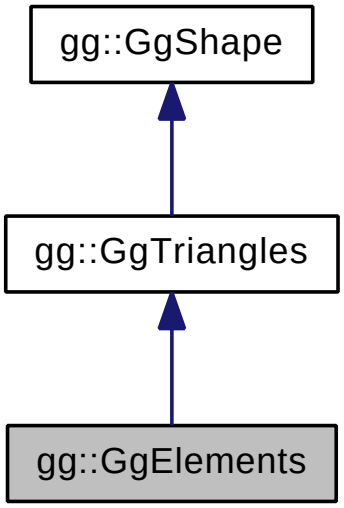
三角形で表した形状データ (Elements 形式).

`#include <gg.h>`

gg::GgElements の継承関係図



gg::GgElements 連携図



公開メンバ関数

- virtual ~GgElements ()
  デストラクタ.
- GgElements (GLenum mode=GL_TRIANGLES)
  コンストラクタ.
- GgElements (const GgVertex *vert, GLsizei countv, const GLuint *face, GLsizei countf, GLenum mode=GL_TRIANGLES, GLenum usage=GL_STATIC_DRAW)
  コンストラクタ.
- GLsizei getIndexCount () const
  データの数を取り出す.
- GLuint getIndexBuffer () const
  三角形の頂点インデックスデータを格納した頂点バッファオブジェクト名を取り出す.
- void send (const GgVertex *vert, GLuint firstv, GLsizei countv, const GLuint *face=nullptr, GLuint firstf=0, GLsizei countf=0) const
  既存のバッファオブジェクトに頂点属性と三角形の頂点インデックスデータを転送する.
- void load (const GgVertex *vert, GLsizei countv, const GLuint *face, GLsizei countf, GLenum usage=GL_STATIC_DRAW)
  バッファオブジェクトを確保して頂点属性と三角形の頂点インデックスデータを格納する.
- virtual void draw (GLint first=0, GLsizei count=0) const
  インデックスを使った三角形の描画.

6.3.1 詳解

三角形で表した形状データ (Elements 形式).

gg.h の 4725 行目に定義があります。

6.3.2 構築子と解体子

6.3.2.1 virtual gg::GgElements::~GgElements ( ) [inline], [virtual]

デストラクタ.

gg.h の 4734 行目に定義があります。

6.3.2.2 gg::GgElements::GgElements (GLenum *mode* = GL_TRIANGLES) [inline]

コンストラクタ.

引数

mode	描画する基本図形の種類.

gg.h の 4738 行目に定義があります。

6.3.2.3 gg::GgElements::GgElements (const GgVertex * *vert*, GLsizei *countv*, const GLuint * *face*, GLsizei *countf*, GLenum *mode* = GL_TRIANGLES, GLenum *usage* = GL_STATIC_DRAW) [inline]

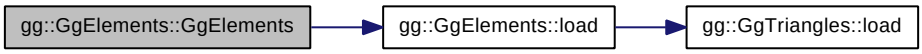
コンストラクタ.

引数

vert	この図形の頂点属性の配列 (nullptr ならデータを転送しない).
countv	頂点数.
face	三角形の頂点インデックス.
countf	三角形の頂点数.
mode	描画する基本図形の種類.
usage	バッファオブジェクトの使い方.

gg.h の 4749 行目に定義があります。

呼び出し関係図:



6.3.3 関数詳解

6.3.3.1 void gg::GgElements::draw (GLint *first* = 0, GLsizei *count* = 0) const [virtual]

インデックスを使った三角形の描画.

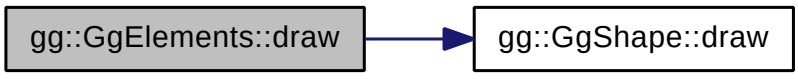
引数

first	描画を開始する最初の三角形番号.
count	描画する三角形数, 0 なら全部の三角形を描く.

gg::GgTrianglesを再実装しています。

gg.cpp の 4968 行目に定義があります。

呼び出し関係図:



6.3.3.2 GLuint gg::GgElements::getIndexBuffer ( ) const [inline]

三角形の頂点インデックスデータを格納した頂点バッファオブジェクト名を取り出す.

戻り値

この図形の三角形の頂点インデックスデータを格納した頂点バッファオブジェクト名.

gg.h の 4765 行目に定義があります。

6.3.3.3 GLsizei gg::GgElements::getIndexCount ( ) const [inline]

データの数を取り出す.

戻り値

この図形の三角形数.

gg.h の 4758 行目に定義があります。

6.3.3.4 void gg::GgElements::load (const GgVertex * *vert*, GLsizei *countv*, const GLuint * *face*, GLsizei *countf*, GLenum *usage* = GL_STATIC_DRAW) [inline]

バッファオブジェクトを確保して頂点属性と三角形の頂点インデックスデータを格納する.

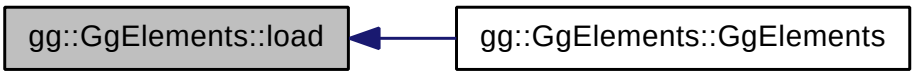
引数

vertex	頂点属性が格納されてている領域の先頭のポインタ.
countv	頂点のデータの数 (頂点数).
face	三角形の頂点インデックスデータ.
countf	三角形の頂点数.
usage	バッファオブジェクトの使い方.

gg.h の 4790 行目に定義があります。

呼び出し関係図:

被呼び出し関係図:



6.3.3.5 void gg::GgElements::send (const GgVertex * *vert*, GLuint *firstv*, GLsizei *countv*, const GLuint * *face* = nullptr, GLuint *firstf* = 0, GLsizei *countf* = 0) const [inline]

既存のバッファオブジェクトに頂点属性と三角形の頂点インデックスデータを転送する.

引数

vertex	頂点属性が格納されてている領域の先頭のポインタ.

firstv	頂点属性の転送先のバッファオブジェクトの先頭の要素番号.
countv	頂点のデータの数 (頂点数).
face	三角形の頂点インデックスデータ.
firstf	インデックスの転送先のバッファオブジェクトの先頭の要素番号.
countf	三角形の頂点数.

gg.h の 4777 行目に定義があります。

呼び出し関係図:

このクラス詳解は次のファイルから抽出されました:

- gg.h
- gg.cpp

6.4 gg::GgMatrix クラス

変換行列.

`#include <gg.h>`

公開メンバ関数

- ~GgMatrix ()
  デストラクタ.
- GgMatrix ()
  コンストラクタ.
- GgMatrix (const GLfloat *a)
  コンストラクタ.
- GgMatrix (const GgMatrix &m)
  コピーコンストラクタ.
- GgMatrix & load (const GLfloat *a)
  配列変数の値を格納する.
- GgMatrix & load (const GgMatrix &m)
  別の変換行列の値を格納する.
- GgMatrix & loadAdd (const GLfloat *a)
  変換行列に配列に格納した変換行列を加算した結果を格納する.
- GgMatrix & loadAdd (const GgMatrix &m)
  変換行列に別の変換行列を加算した結果を格納する.
- GgMatrix & loadSubtract (const GLfloat *a)
  変換行列から配列に格納した変換行列を減算した結果を格納する.
- GgMatrix & loadSubtract (const GgMatrix &m)
  変換行列から別の変換行列を減算した結果を格納する.
- GgMatrix & loadMultiply (const GLfloat *a)
  変換行列に配列に格納した変換行列を乗算した結果を格納する.
- GgMatrix & loadMultiply (const GgMatrix &m)
  変換行列に別の変換行列を乗算した結果を格納する.
- GgMatrix & loadDivide (const GLfloat *a)

変換行列を配列に格納した変換行列で除算した結果を格納する.

- GgMatrix & loadDivide (const GgMatrix &m)

  変換行列を別の変換行列で除算した結果を格納する.
- GgMatrix add (const GLfloat *a) const

  変換行列に配列に格納した変換行列を加算した値を返す.
- GgMatrix add (const GgMatrix &m) const

  変換行列に別の変換行列を加算した値を返す.
- GgMatrix subtract (const GLfloat *a) const

  変換行列から配列に格納した変換行列を減算した値を返す.
- GgMatrix subtract (const GgMatrix &m) const

  変換行列から別の変換行列を減算した値を返す.
- GgMatrix multiply (const GLfloat *a) const

  変換行列に配列に格納した変換行列を乗算した値を返す.
- GgMatrix multiply (const GgMatrix &m) const

  変換行列に別の変換行列を乗算した値を返す.
- GgMatrix divide (const GLfloat *a) const

  変換行列を配列に格納した変換行列で除算した値を返す.
- GgMatrix divide (const GgMatrix &m) const

  変換行列を配列に格納した変換行列で除算した値を返す.
- GgMatrix & operator= (const GLfloat *a)
- GgMatrix & operator= (const GgMatrix &m)
- GgMatrix & operator+= (const GLfloat *a)
- GgMatrix & operator+= (const GgMatrix &m)
- GgMatrix & operator-= (const GLfloat *a)
- GgMatrix & operator-= (const GgMatrix &m)
- GgMatrix & operator*= (const GLfloat *a)
- GgMatrix & operator*= (const GgMatrix &m)
- GgMatrix & operator/= (const GLfloat *a)
- GgMatrix & operator/= (const GgMatrix &m)
- GgMatrix operator+ (const GLfloat *a) const
- GgMatrix operator+ (const GgMatrix &m) const
- GgMatrix operator- (const GLfloat *a) const
- GgMatrix operator- (const GgMatrix &m) const
- GgMatrix operator* (const GLfloat *a) const
- GgMatrix operator* (const GgMatrix &m) const
- GgMatrix operator/ (const GLfloat *a) const
- GgMatrix operator/ (const GgMatrix &m) const
- GgMatrix & loadIdentity ()

  単位行列を格納する.
- GgMatrix & loadTranslate (GLfloat x, GLfloat y, GLfloat z, GLfloat w=1.0f)

  平行移動の変換行列を格納する.
- GgMatrix & loadTranslate (const GLfloat *t)

  平行移動の変換行列を格納する.
- GgMatrix & loadTranslate (const GgVector &t)

  平行移動の変換行列を格納する.
- GgMatrix & loadScale (GLfloat x, GLfloat y, GLfloat z, GLfloat w=1.0f)

  拡大縮小の変換行列を格納する.
- GgMatrix & loadScale (const GLfloat *s)

  拡大縮小の変換行列を格納する.
- GgMatrix & loadScale (const GgVector &s)

  拡大縮小の変換行列を格納する.
- GgMatrix & loadRotateX (GLfloat a)

x **軸中心の回転の変換行列を格納する**.

- GgMatrix & loadRotateY (GLfloat a)

  y **軸中心の回転の変換行列を格納する**.
- GgMatrix & loadRotateZ (GLfloat a)

  z **軸中心の回転の変換行列を格納する**.
- GgMatrix & loadRotate (GLfloat x, GLfloat y, GLfloat z, GLfloat a)

  (x, y, z) **方向のベクトルを軸とする回転の変換行列を格納する**.
- GgMatrix & loadRotate (const GLfloat ∗r, GLfloat a)

  r **方向のベクトルを軸とする回転の変換行列を格納する**.
- GgMatrix & loadRotate (const GgVector &r, GLfloat a)

  r **方向のベクトルを軸とする回転の変換行列を格納する**.
- GgMatrix & loadRotate (const GLfloat ∗r)

  r **方向のベクトルを軸とする回転の変換行列を格納する**.
- GgMatrix & loadRotate (const GgVector &r)

  r **方向のベクトルを軸とする回転の変換行列を格納する**.
- GgMatrix & loadLookat (GLfloat ex, GLfloat ey, GLfloat ez, GLfloat tx, GLfloat ty, GLfloat tz, GLfloat ux, GLfloat uy, GLfloat uz)

  ビュー変換行列を格納する.
- GgMatrix & loadLookat (const GLfloat ∗e, const GLfloat ∗t, const GLfloat ∗u)

  ビュー変換行列を格納する.
- GgMatrix & loadLookat (const GgVector &e, const GgVector &t, const GgVector &u)

  ビュー変換行列を格納する.
- GgMatrix & loadOrthogonal (GLfloat left, GLfloat right, GLfloat bottom, GLfloat top, GLfloat zNear, GLfloat zFar)

  直交投影変換行列を格納する.
- GgMatrix & loadFrustum (GLfloat left, GLfloat right, GLfloat bottom, GLfloat top, GLfloat zNear, GLfloat zFar)

  透視透視投影変換行列を格納する.
- GgMatrix & loadPerspective (GLfloat fovy, GLfloat aspect, GLfloat zNear, GLfloat zFar)

  画角を指定して透視投影変換行列を格納する.
- GgMatrix & loadTranspose (const GLfloat ∗a)

  転置行列を格納する.
- GgMatrix & loadTranspose (const GgMatrix &m)

  転置行列を格納する.
- GgMatrix & loadInvert (const GLfloat ∗a)

  逆行列を格納する.
- GgMatrix & loadInvert (const GgMatrix &m)

  逆行列を格納する.
- GgMatrix & loadNormal (const GLfloat ∗a)

  法線変換行列を格納する.
- GgMatrix & loadNormal (const GgMatrix &m)

  法線変換行列を格納する.
- GgMatrix translate (GLfloat x, GLfloat y, GLfloat z, GLfloat w=1.0f) const

  平行移動変換を乗じた結果を返す.
- GgMatrix translate (const GLfloat ∗t) const

  平行移動変換を乗じた結果を返す.
- GgMatrix translate (const GgVector &t) const

  平行移動変換を乗じた結果を返す.
- GgMatrix scale (GLfloat x, GLfloat y, GLfloat z, GLfloat w=1.0f) const

  拡大縮小変換を乗じた結果を返す.
- GgMatrix scale (const GLfloat ∗s) const

  拡大縮小変換を乗じた結果を返す.

- GgMatrix scale (const GgVector &s) const
  拡大縮小変換を乗じた結果を返す.
- GgMatrix rotateX (GLfloat a) const
  x **軸中心の回転変換を乗じた結果を返す**.
- GgMatrix rotateY (GLfloat a) const
  y **軸中心の回転変換を乗じた結果を返す**.
- GgMatrix rotateZ (GLfloat a) const
  z **軸中心の回転変換を乗じた結果を返す**.
- GgMatrix rotate (GLfloat x, GLfloat y, GLfloat z, GLfloat a) const
  (x, y, z) **方向のベクトルを軸とする回転変換を乗じた結果を返す**.
- GgMatrix rotate (const GLfloat *r, GLfloat a) const
  r **方向のベクトルを軸とする回転変換を乗じた結果を返す**.
- GgMatrix rotate (const GgVector &r, GLfloat a) const
  r **方向のベクトルを軸とする回転変換を乗じた結果を返す**.
- GgMatrix rotate (const GLfloat *r) const
  r **方向のベクトルを軸とする回転の変換行列を乗じた結果を返す**.
- GgMatrix rotate (const GgVector &r) const
  r **方向のベクトルを軸とする回転の変換行列を乗じた結果を返す**.
- GgMatrix lookat (GLfloat ex, GLfloat ey, GLfloat ez, GLfloat tx, GLfloat ty, GLfloat tz, GLfloat ux, GLfloat uy, GLfloat uz) const
  ビュー変換を乗じた結果を返す.
- GgMatrix lookat (const GLfloat *e, const GLfloat *t, const GLfloat *u) const
  ビュー変換を乗じた結果を返す.
- GgMatrix lookat (const GgVector &e, const GgVector &t, const GgVector &u) const
  ビュー変換を乗じた結果を返す.
- GgMatrix orthogonal (GLfloat left, GLfloat right, GLfloat bottom, GLfloat top, GLfloat zNear, GLfloat zFar) const
  直交投影変換を乗じた結果を返す.
- GgMatrix frustum (GLfloat left, GLfloat right, GLfloat bottom, GLfloat top, GLfloat zNear, GLfloat zFar) const
  透視投影変換を乗じた結果を返す.
- GgMatrix perspective (GLfloat fovy, GLfloat aspect, GLfloat zNear, GLfloat zFar) const
  画角を指定して透視投影変換を乗じた結果を返す.
- GgMatrix transpose () const
  転置行列を返す.
- GgMatrix invert () const
  逆行列を返す.
- GgMatrix normal () const
  法線変換行列を返す.
- void projection (GLfloat *c, const GLfloat *v) const
  ベクトルに対して投影変換を行う.
- void projection (GLfloat *c, const GgVector &v) const
  ベクトルに対して投影変換を行う.
- void projection (GgVector &c, const GLfloat *v) const
  ベクトルに対して投影変換を行う.
- void projection (GgVector &c, const GgVector &v) const
  ベクトルに対して投影変換を行う.
- GgVector operator* (const GgVector &v) const
  ベクトルに対して投影変換を行う.
- const GLfloat * get () const
  変換行列を取り出す.
- void get (GLfloat *a) const
  変換行列を取り出す.
- const GLfloat get (int i) const
  変換行列の要素を取り出す.

フレンド

- class GgQuaternion

6.4.1 詳解

変換行列.

gg.h の 1611 行目に定義があります。

6.4.2 構築子と解体子

6.4.2.1 gg::GgMatrix::~GgMatrix ( ) [inline]

デストラクタ.

gg.h の 1628 行目に定義があります。

6.4.2.2 gg::GgMatrix::GgMatrix ( ) [inline]

コンストラクタ.

gg.h の 1631 行目に定義があります。

6.4.2.3 gg::GgMatrix::GgMatrix (const GLfloat * a) [inline]

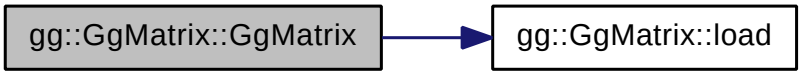
コンストラクタ.

引数

a	GLfloat 型の 16 要素の配列変数.

gg.h の 1635 行目に定義があります。

呼び出し関係図:



6.4.2.4 gg::GgMatrix::GgMatrix (const GgMatrix & m) [inline]

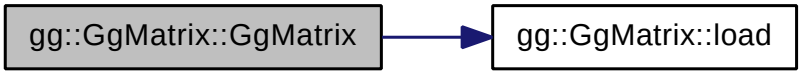
コピーコンストラクタ.

引数

m	GgMatrix 型の変数.

gg.h の 1642 行目に定義があります。

呼び出し関係図:



6.4.3 関数詳解

6.4.3.1 GgMatrix gg::GgMatrix::add (const GLfloat * *a*) const [inline]

変換行列に配列に格納した変換行列を加算した値を返す.

引数

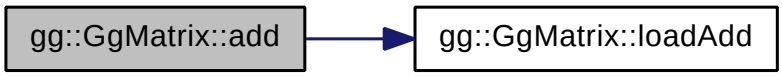
a	GLfloat 型の 16 要素の配列変数.

戻り値

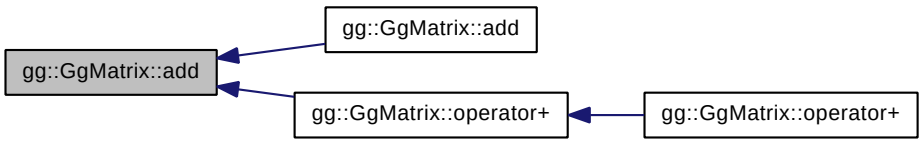
変換行列に a を加えた GgMatrix 型の値.

gg.h の 1733 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:



6.4.3.2 GgMatrix gg::GgMatrix::add (const GgMatrix & *m*) const [inline]

変換行列に別の変換行列を加算した値を返す.

引数

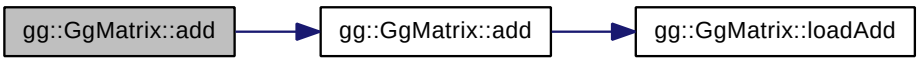
m	GgMatrix 型の変数.

戻り値

変換行列に m を加えた GgMatrix 型の値.

gg.h の 1742 行目に定義があります。

呼び出し関係図:



6.4.3.3 GgMatrix gg::GgMatrix::divide (const GLfloat * *a*) const [inline]

変換行列を配列に格納した変換行列で除算した値を返す.

引数

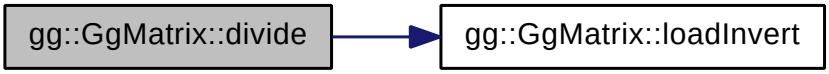
a	GLfloat 型の 16 要素の配列変数.

戻り値

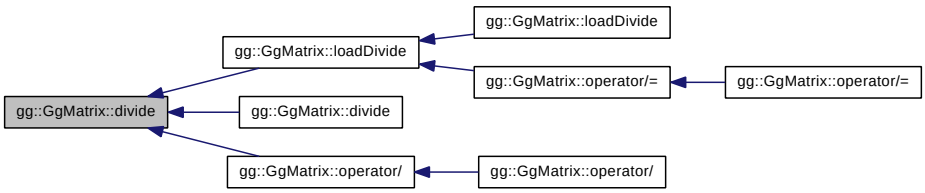
変換行列を a で割った GgMatrix 型の値.

gg.h の 1785 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:



6.4.3.4 GgMatrix gg::GgMatrix::divide (const GgMatrix & *m*) const [inline]

変換行列を配列に格納した変換行列で除算した値を返す.

引数

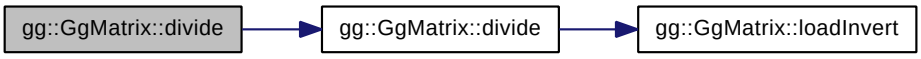
m	GgMatrix 型の変数.

戻り値

変換行列を m で割った GgMatrix 型の値.

gg.h の 1796 行目に定義があります。

呼び出し関係図:



6.4.3.5 GgMatrix gg::GgMatrix::frustum (GLfloat *left*, GLfloat *right*, GLfloat *bottom*, GLfloat *top*, GLfloat *zNear*, GLfloat *zFar*) const [inline]

透視投影変換を乗じた結果を返す.

引数

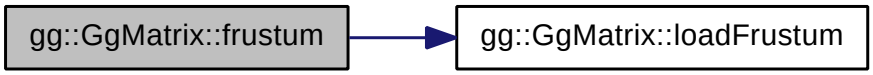
left	ウィンドウの左端の位置.
right	ウィンドウの右端の位置.
bottom	ウィンドウの下端の位置.
top	ウィンドウの上端の位置.
zNear	視点から前方面までの位置.
zFar	視点から後方面までの位置.

戻り値

透視投影変換行列を乗じた変換行列.

gg.h の 2282 行目に定義があります。

呼び出し関係図:



6.4.3.6 const GLfloat* gg::GgMatrix::get ( ) const [inline]

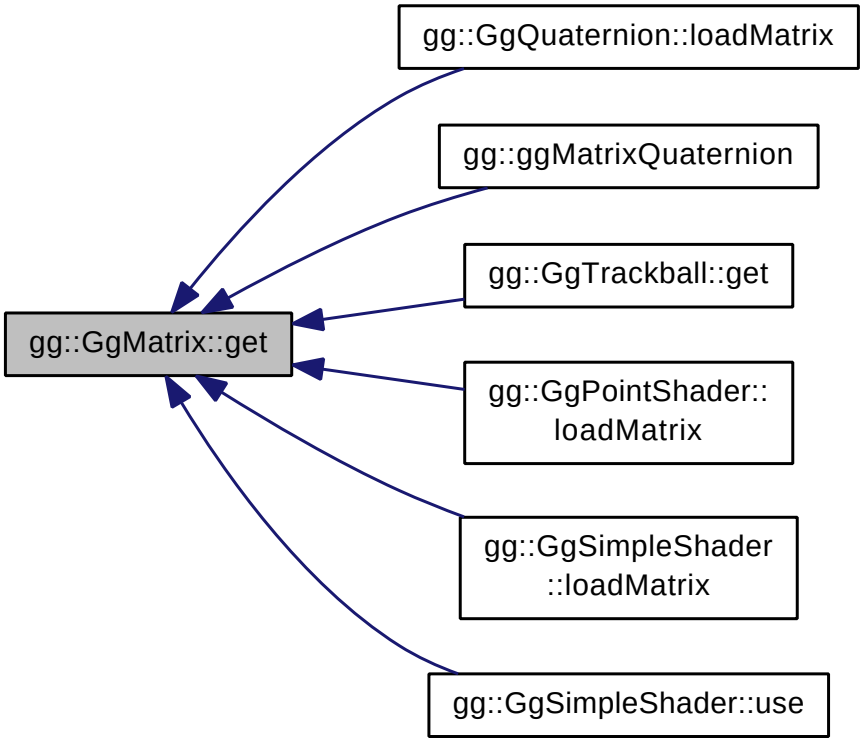
変換行列を取り出す.

戻り値

変換行列を格納した GLfloat 型の 16 要素の配列変数.

gg.h の 2371 行目に定義があります。

被呼び出し関係図:



6.4.3.7 void gg::GgMatrix::get (GLfloat * a) const [inline]

変換行列を取り出す.

引数

a	変換行列を格納する GLfloat 型の 16 要素の配列変数.

gg.h の 2378 行目に定義があります。

6.4.3.8 const GLfloat gg::GgMatrix::get (int *i*) const [inline]

変換行列の要素を取り出す.

戻り値

変換行列を格納した GLfloat 型の 16 要素の配列変数.

gg.h の 2385 行目に定義があります。

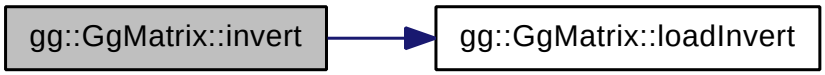
6.4.3.9 GgMatrix gg::GgMatrix::invert ( ) const [inline]

逆行列を返す.

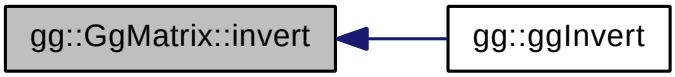
戻り値

逆行列.

gg.h の 2313 行目に定義があります。
呼び出し関係図:



被呼び出し関係図:



6.4.3.10 GgMatrix& gg::GgMatrix::load (const GLfloat * a) [inline]

配列変数の値を格納する.

引数

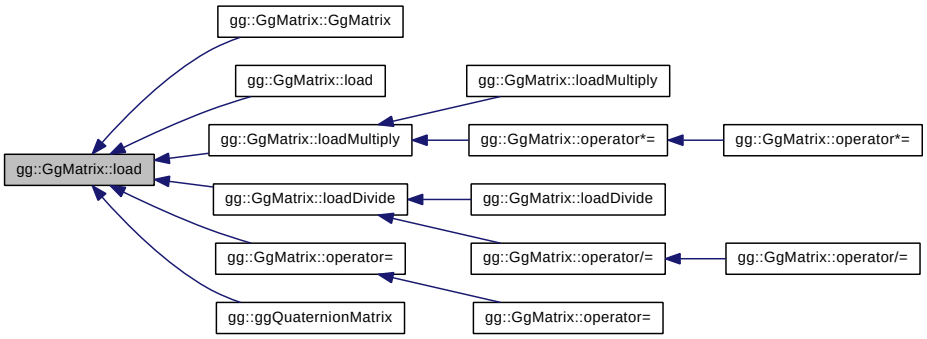
a	GLfloat 型の 16 要素の配列変数.

戻り値

a を代入した GgMatrix 型の値.

gg.h の 1650 行目に定義があります。

被呼び出し関係図:



6.4.3.11 GgMatrix& gg::GgMatrix::load (const GgMatrix & *m*) [inline]

別の変換行列の値を格納する.

引数

m	GgMatrix 型の変数.

戻り値

m を代入した GgMatrix 型の値.

gg.h の 1659 行目に定義があります。

呼び出し関係図:

6.4.3.12 GgMatrix& gg::GgMatrix::loadAdd (const GLfloat * *a*) [inline]

変換行列に配列に格納した変換行列を加算した結果を格納する.

引数

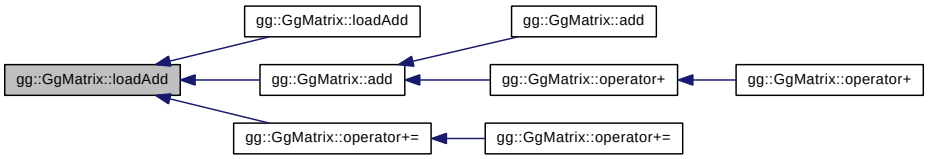
a	GLfloat 型の 16 要素の配列変数.

戻り値

変換行列に a を加えた GgMatrix 型の値.

gg.h の 1667 行目に定義があります。

被呼び出し関係図:



6.4.3.13 GgMatrix& gg::GgMatrix::loadAdd (const GgMatrix & *m*) [inline]

変換行列に別の変換行列を加算した結果を格納する.

引数

m	GgMatrix 型の変数.

戻り値

変換行列に m を加えた GgMatrix 型の値.

gg.h の 1676 行目に定義があります。

呼び出し関係図:

6.4.3.14 GgMatrix& gg::GgMatrix::loadDivide (const GLfloat * *a*) [inline]

変換行列を配列に格納した変換行列で除算した結果を格納する.

引数

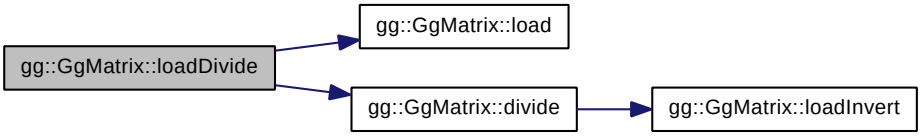
a	GLfloat 型の 16 要素の配列変数.

戻り値

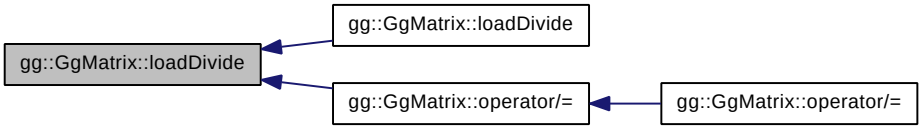
変換行列に a を乗じた GgMatrix 型の値.

gg.h の 1717 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:



6.4.3.15 GgMatrix& gg::GgMatrix::loadDivide (const GgMatrix & *m*) [inline]

変換行列を別の変換行列で除算した結果を格納する.

引数

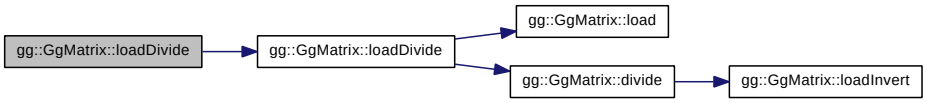
m	GgMatrix 型の変数.

戻り値

変換行列に m を乗じた GgMatrix 型の値.

gg.h の 1725 行目に定義があります。

呼び出し関係図:



6.4.3.16 gg::GgMatrix & gg::GgMatrix::loadFrustum (GLfloat *left*, GLfloat *right*, GLfloat *bottom*, GLfloat *top*, GLfloat *zNear*, GLfloat *zFar*)

透視透視投影変換行列を格納する.

引数

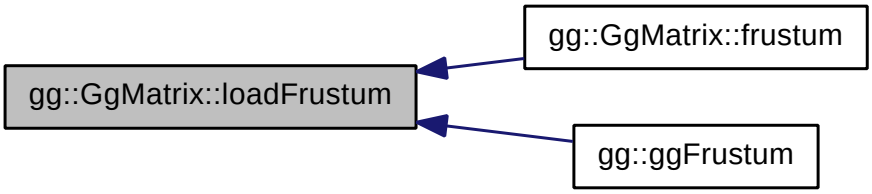
left	ウィンドウの左端の位置.
right	ウィンドウの右端の位置.
bottom	ウィンドウの下端の位置.
top	ウィンドウの上端の位置.
zNear	視点から前方面までの位置.
zFar	視点から後方面までの位置.

戻り値

設定した透視投影変換行列.

gg.cpp の 4561 行目に定義があります。

被呼び出し関係図:



6.4.3.17 gg::GgMatrix & gg::GgMatrix::loadIdentity ( )

単位行列を格納する.

gg.cpp の 4219 行目に定義があります。

被呼び出し関係図:

6.4.3.18 gg::GgMatrix & gg::GgMatrix::loadInvert (const GLfloat * a)

逆行列を格納する.

引数

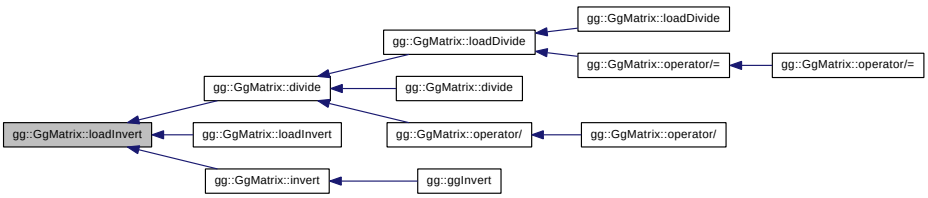
a	GLfloat 型の 16 要素の変換行列.

戻り値

設定した a の逆行列.

gg.cpp の 4376 行目に定義があります。

被呼び出し関係図:



6.4.3.19 GgMatrix& gg::GgMatrix::loadInvert (const GgMatrix & *m*) [inline]

逆行列を格納する.

引数

m	GgMatrix 型の変換行列.

戻り値

設定した m の逆行列.

gg.h の 2072 行目に定義があります。

呼び出し関係図:

6.4.3.20 gg::GgMatrix & gg::GgMatrix::loadLookat (GLfloat *ex*, GLfloat *ey*, GLfloat *ez*, GLfloat *tx*, GLfloat *ty*, GLfloat *tz*, GLfloat *ux*, GLfloat *uy*, GLfloat *uz*)

ビュー変換行列を格納する.

引数

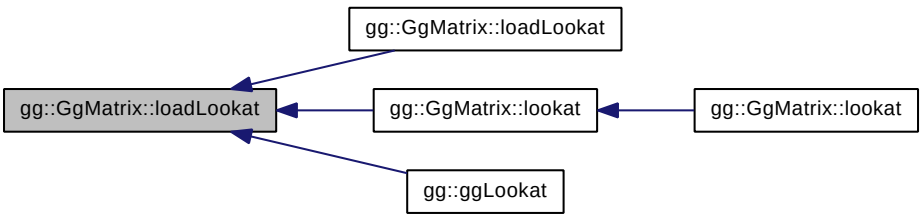
ex	視点の位置の x 座標値.
ey	視点の位置の y 座標値.
ez	視点の位置の z 座標値.
tx	目標点の位置の x 座標値.
ty	目標点の位置の y 座標値.
tz	目標点の位置の z 座標値.
ux	上方向のベクトルの x 成分.
uy	上方向のベクトルの y 成分.
uz	上方向のベクトルの z 成分.

戻り値

設定したビュー変換行列.

gg.cpp の 4478 行目に定義があります。

被呼び出し関係図:



6.4.3.21 GgMatrix& gg::GgMatrix::loadLookat (const GLfloat * *e*, const GLfloat * *t*, const GLfloat * *u*) [inline]

ビュー変換行列を格納する.

引数

e	視点の位置の配列変数.
t	目標点の位置の配列変数.
u	上方向のベクトルの配列変数.

戻り値

設定したビュー変換行列.

gg.h の 2003 行目に定義があります。

呼び出し関係図:

6.4.3.22 GgMatrix& gg::GgMatrix::loadLookat (const GgVector & *e*, const GgVector & *t*, const GgVector & *u*) [inline]

ビュー変換行列を格納する.

引数

e	視点の位置の GgVector 型の変数.
t	目標点の位置の GgVector 型の変数.
u	上方向のベクトルの GgVector 型の変数.

戻り値

設定したビュー変換行列.

gg.h の 2013 行目に定義があります。

呼び出し関係図:

6.4.3.23 GgMatrix& gg::GgMatrix::loadMultiply (const GLfloat * *a*) [inline]

変換行列に配列に格納した変換行列を乗算した結果を格納する.

引数

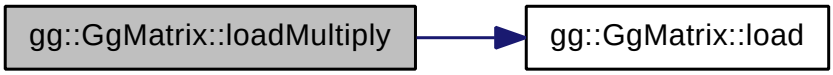
a	GLfloat 型の 16 要素の配列変数.

戻り値

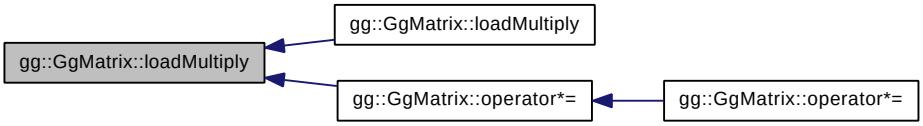
変換行列に a を掛けた GgMatrix 型の値.

gg.h の 1701 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:



6.4.3.24 GgMatrix& gg::GgMatrix::loadMultiply (const GgMatrix & *m*) [inline]

変換行列に別の変換行列を乗算した結果を格納する.

引数

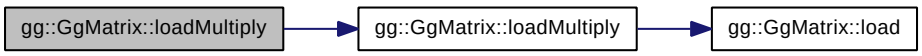
m	GgMatrix 型の変数.

戻り値

変換行列に m を掛けた GgMatrix 型の値.

gg.h の 1709 行目に定義があります。

呼び出し関係図:



6.4.3.25 gg::GgMatrix & gg::GgMatrix::loadNormal (const GLfloat * *a*)

法線変換行列を格納する.

引数

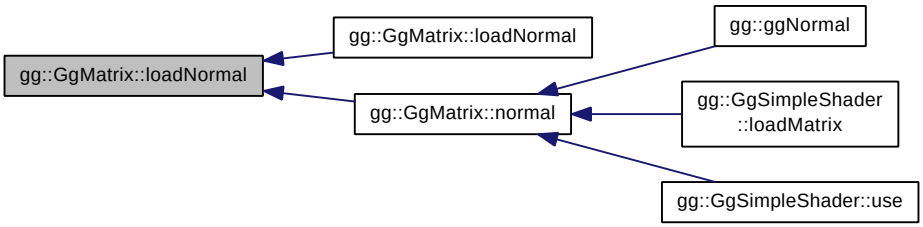
a	GLfloat 型の 16 要素の変換行列.

戻り値

設定した m の法線変換行列.

gg.cpp の 4458 行目に定義があります。

被呼び出し関係図:



6.4.3.26 GgMatrix& gg::GgMatrix::loadNormal (const GgMatrix & *m*) [inline]

法線変換行列を格納する.

引数

GgMatrix	型の変換行列.

戻り値

設定した m の法線変換行列.

gg.h の 2085 行目に定義があります。

呼び出し関係図:

6.4.3.27 gg::GgMatrix & gg::GgMatrix::loadOrthogonal (GLfloat *left*, GLfloat *right*, GLfloat *bottom*, GLfloat *top*, GLfloat *zNear*, GLfloat *zFar*)

直交投影変換行列を格納する.

引数

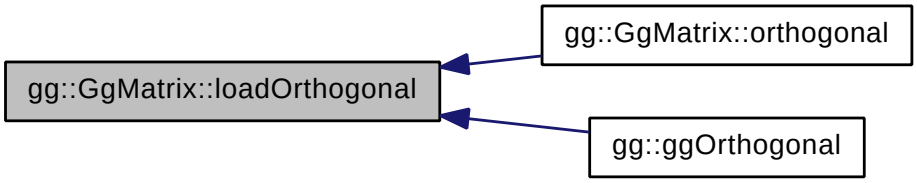
left	ウィンドウの左端の位置.
right	ウィンドウの右端の位置.
bottom	ウィンドウの下端の位置.
top	ウィンドウの上端の位置.
zNear	視点から前方面までの位置.
zFar	視点から後方面までの位置.

戻り値

設定した直交投影変換行列.

gg.cpp の 4534 行目に定義があります。

被呼び出し関係図:



6.4.3.28 gg::GgMatrix & gg::GgMatrix::loadPerspective (GLfloat *fovy*, GLfloat *aspect*, GLfloat *zNear*, GLfloat *zFar*)

画角を指定して透視投影変換行列を格納する.

引数

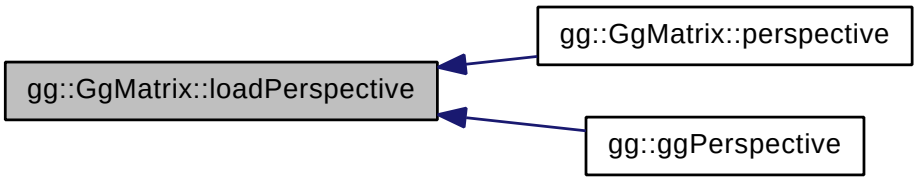
fovy	y 方向の画角.
aspect	縦横比.
zNear	視点から前方面までの位置.
zFar	視点から後方面までの位置.

戻り値

設定した透視投影変換行列.

gg.cpp の 4588 行目に定義があります。

被呼び出し関係図:



6.4.3.29 gg::GgMatrix & gg::GgMatrix::loadRotate (GLfloat *x*, GLfloat *y*, GLfloat *z*, GLfloat *a*)

(x, y, z) 方向のベクトルを軸とする回転の変換行列を格納する.

引数

x	回転軸の x 成分.
y	回転軸の y 成分.
z	回転軸の z 成分.

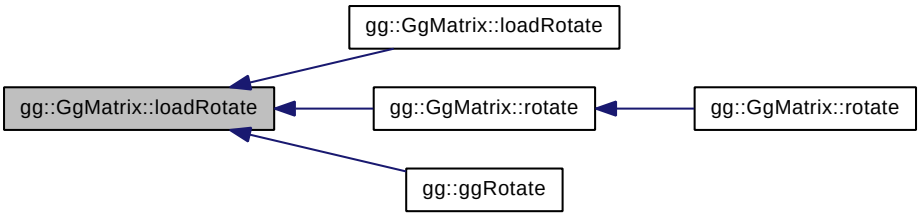
a	回転角.

戻り値

設定した変換行列.

gg.cpp の 4312 行目に定義があります。

被呼び出し関係図:



6.4.3.30 GgMatrix& gg::GgMatrix::loadRotate (const GLfloat * *r*, GLfloat *a*) [inline]

r 方向のベクトルを軸とする回転の変換行列を格納する.

引数

r	回転軸の方向ベクトルを格納した GLfloat 型の 3 要素の配列変数 (x, y, z).
a	回転角.

戻り値

設定した変換行列.

gg.h の 1953 行目に定義があります。

呼び出し関係図:

6.4.3.31 GgMatrix& gg::GgMatrix::loadRotate (const GgVector & *r*, GLfloat *a*) [inline]

r 方向のベクトルを軸とする回転の変換行列を格納する.

引数

r	回転軸の方向ベクトルを格納した GgVector 型の変数.
a	回転角.

戻り値

設定した変換行列.

gg.h の 1962 行目に定義があります。

呼び出し関係図:

6.4.3.32 GgMatrix& gg::GgMatrix::loadRotate (const GLfloat * r) [inline]

r 方向のベクトルを軸とする回転の変換行列を格納する.

引数

r	回転軸の方向ベクトルと回転角を格納した GLfloat 型の 4 要素の配列変数 (x, y, z, a).

戻り値

設定した変換行列.

gg.h の 1970 行目に定義があります。

呼び出し関係図:

6.4.3.33 GgMatrix& gg::GgMatrix::loadRotate (const GgVector & r) [inline]

r 方向のベクトルを軸とする回転の変換行列を格納する.

引数

r	回転軸の方向ベクトルと回転角を格納した GgVector 型の変数.

戻り値

設定した変換行列.

gg.h の 1978 行目に定義があります。

呼び出し関係図:

6.4.3.34 gg::GgMatrix & gg::GgMatrix::loadRotateX (GLfloat *a*)

x 軸中心の回転の変換行列を格納する.

引数

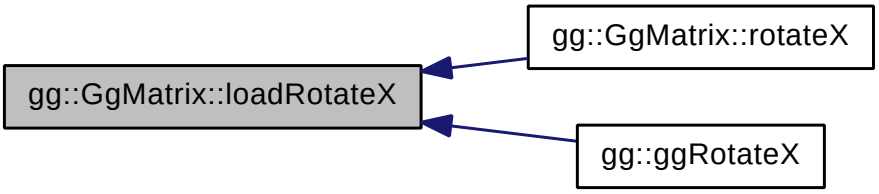
a	回転角.

戻り値

設定した変換行列.

gg.cpp の 4264 行目に定義があります。

被呼び出し関係図:



6.4.3.35 gg::GgMatrix & gg::GgMatrix::loadRotateY (GLfloat *a*)

y 軸中心の回転の変換行列を格納する.

引数

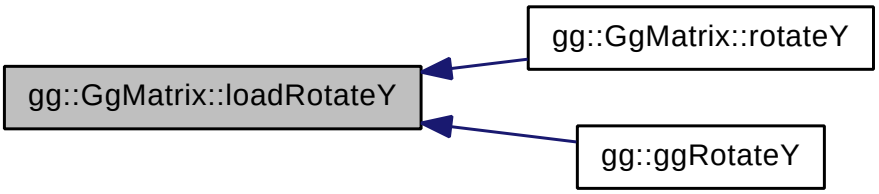
a	回転角.

戻り値

設定した変換行列.

gg.cpp の 4280 行目に定義があります。

被呼び出し関係図:



6.4.3.36 gg::GgMatrix & gg::GgMatrix::loadRotateZ (GLfloat *a*)

z 軸中心の回転の変換行列を格納する.

引数

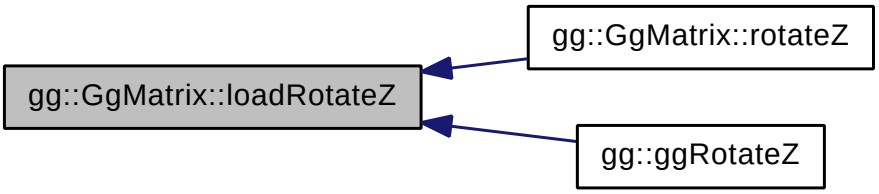
a	回転角.

戻り値

設定した変換行列.

gg.cpp の 4296 行目に定義があります。

被呼び出し関係図:



6.4.3.37 gg::GgMatrix & gg::GgMatrix::loadScale (GLfloat *x*, GLfloat *y*, GLfloat *z*, GLfloat *w* = 1.0f)

拡大縮小の変換行列を格納する.

引数

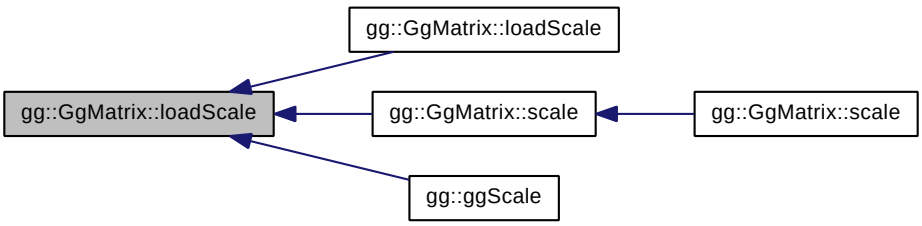
x	x 方向の拡大率.
y	y 方向の拡大率.
z	z 方向の拡大率.
w	w 拡大率のスケールファクタ (= 1.0f).

戻り値

設定した変換行列.

gg.cpp の 4248 行目に定義があります。

被呼び出し関係図:



6.4.3.38 GgMatrix& gg::GgMatrix::loadScale (const GLfloat * *s*) [inline]

拡大縮小の変換行列を格納する.

引数

s	拡大率の GLfloat 型の配列 (x, y, z).

戻り値

設定した変換行列.

gg.h の 1913 行目に定義があります。
呼び出し関係図:

6.4.3.39 GgMatrix& gg::GgMatrix::loadScale (const GgVector & *s*) [inline]

拡大縮小の変換行列を格納する.
引数

s	拡大率の GgVector 型の変数.

戻り値

設定した変換行列.

gg.h の 1921 行目に定義があります。
呼び出し関係図:

6.4.3.40 GgMatrix& gg::GgMatrix::loadSubtract (const GLfloat * *a*) [inline]

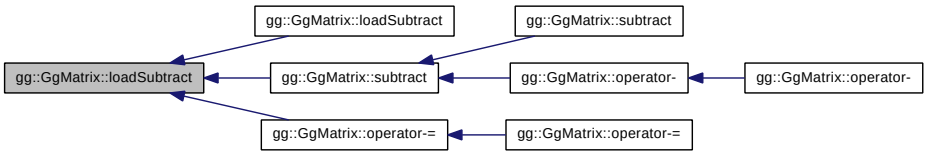
変換行列から配列に格納した変換行列を減算した結果を格納する.
引数

a	GLfloat 型の 16 要素の配列変数.

戻り値

変換行列に a を引いた GgMatrix 型の値.

gg.h の 1684 行目に定義があります。
被呼び出し関係図:



6.4.3.41 GgMatrix& gg::GgMatrix::loadSubtract (const GgMatrix & *m*) [inline]

変換行列から別の変換行列を減算した結果を格納する.

引数

m	GgMatrix 型の変数.

戻り値

変換行列に m を引いた GgMatrix 型の値.

gg.h の 1693 行目に定義があります。

呼び出し関係図:

6.4.3.42 gg::GgMatrix & gg::GgMatrix::loadTranslate (GLfloat *x*, GLfloat *y*, GLfloat *z*, GLfloat *w* = `1.0f`)

平行移動の変換行列を格納する.

引数

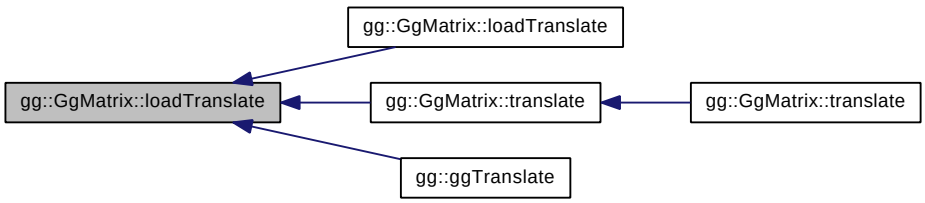
x	x 方向の移動量.
y	y 方向の移動量.
z	z 方向の移動量.
w	w 移動量のスケールファクタ (= 1.0f).

戻り値

設定した変換行列.

gg.cpp の 4232 行目に定義があります。

被呼び出し関係図:



6.4.3.43 GgMatrix& gg::GgMatrix::loadTranslate (const GLfloat * *t*) `[inline]`

平行移動の変換行列を格納する.

引数

t	移動量の GLfloat 型の配列 (x, y, z).

戻り値

設定した変換行列.

gg.h の 1889 行目に定義があります。

呼び出し関係図:

6.4.3.44 GgMatrix& gg::GgMatrix::loadTranslate (const GgVector & *t*) [inline]

平行移動の変換行列を格納する.

引数

t	移動量の GgVector 型の変数.

戻り値

設定した変換行列.

gg.h の 1897 行目に定義があります。

呼び出し関係図:

6.4.3.45 gg::GgMatrix & gg::GgMatrix::loadTranspose (const GLfloat * *a*)

転置行列を格納する.

引数

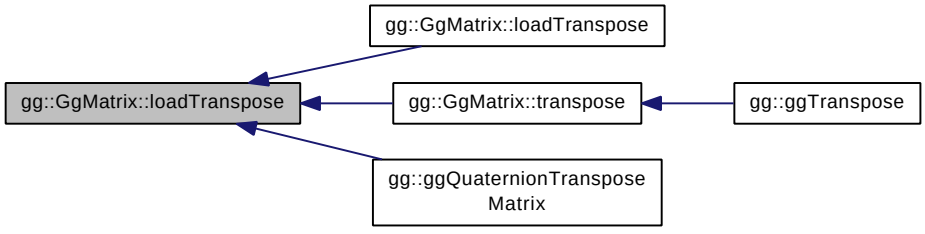
a	GLfloat 型の 16 要素の変換行列.

戻り値

設定した a の転置行列.

gg.cpp の 4351 行目に定義があります。

被呼び出し関係図:



6.4.3.46 GgMatrix& gg::GgMatrix::loadTranspose (const GgMatrix & *m*) [inline]

転置行列を格納する.

引数

m	GgMatrix 型の変換行列.

戻り値

設定した m の転置行列.

gg.h の 2059 行目に定義があります。

呼び出し関係図:

6.4.3.47 GgMatrix gg::GgMatrix::lookat (GLfloat *ex*, GLfloat *ey*, GLfloat *ez*, GLfloat *tx*, GLfloat *ty*, GLfloat *tz*, GLfloat *ux*, GLfloat *uy*, GLfloat *uz*) const [inline]

ビュー変換を乗じた結果を返す.

引数

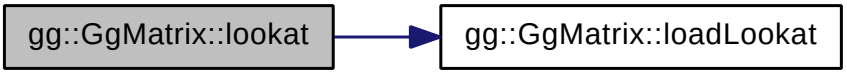
ex	視点の位置の x 座標値.
ey	視点の位置の y 座標値.
ez	視点の位置の z 座標値.
tx	目標点の位置の x 座標値.
ty	目標点の位置の y 座標値.
tz	目標点の位置の z 座標値.
ux	上方向のベクトルの x 成分.
uy	上方向のベクトルの y 成分.
uz	上方向のベクトルの z 成分.

戻り値

ビュー変換行列を乗じた変換行列.

gg.h の 2230 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:

6.4.3.48 GgMatrix gg::GgMatrix::lookat (const GLfloat * *e*, const GLfloat * *t*, const GLfloat * *u*) const [inline]

ビュー変換を乗じた結果を返す.

引数

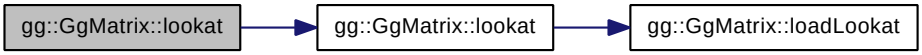
e	視点の位置を格納した GLfloat 型の 3 要素の配列変数.
t	目標点の位置を格納した GLfloat 型の 3 要素の配列変数.
u	上方向のベクトルを格納した GLfloat 型の 3 要素の配列変数.

戻り値

ビュー変換行列を乗じた変換行列.

gg.h の 2243 行目に定義があります。

呼び出し関係図:



6.4.3.49 GgMatrix gg::GgMatrix::lookat (const GgVector & *e*, const GgVector & *t*, const GgVector & *u*) const [inline]

ビュー変換を乗じた結果を返す.

引数

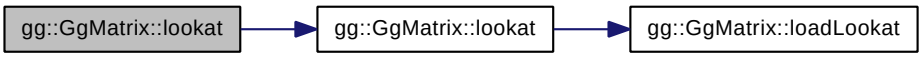
e	視点の位置を格納した GgVector 型の変数.
t	目標点の位置を格納した GgVector 型の変数.
u	上方向のベクトルを格納した GgVector 型の変数.

戻り値

ビュー変換行列を乗じた変換行列.

gg.h の 2253 行目に定義があります。

呼び出し関係図:



6.4.3.50 GgMatrix gg::GgMatrix::multiply (const GLfloat * *a*) const [inline]

変換行列に配列に格納した変換行列を乗算した値を返す.

引数

a	GLfloat 型の 16 要素の配列変数.

戻り値

変換行列に a を掛けた GgMatrix 型の値.

gg.h の 1767 行目に定義があります。

6.4.3.51 GgMatrix gg::GgMatrix::multiply (const GgMatrix & *m*) const [inline]

変換行列に別の変換行列を乗算した値を返す.

引数

m	GgMatrix 型の変数.

戻り値

変換行列に m を掛けた GgMatrix 型の値.

gg.h の 1777 行目に定義があります。

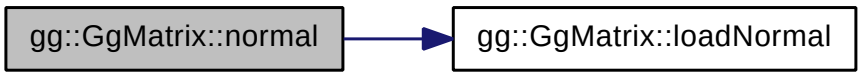
6.4.3.52 GgMatrix gg::GgMatrix::normal ( ) const [inline]

法線変換行列を返す.

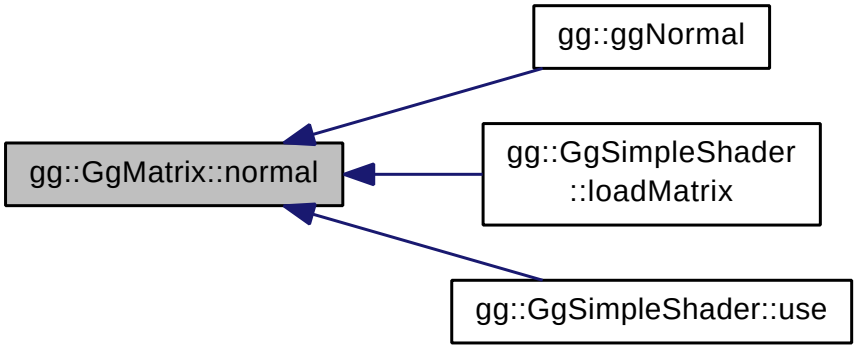
戻り値

法線変換行列.

gg.h の 2321 行目に定義があります。
呼び出し関係図:



被呼び出し関係図:



6.4.3.53 GgMatrix gg::GgMatrix::operator∗ (const GLfloat ∗ *a*) const [inline]

gg.h の 1858 行目に定義があります。
被呼び出し関係図:

6.4.3.54 GgMatrix gg::GgMatrix::operator∗ (const GgMatrix & *m*) const [inline]

gg.h の 1862 行目に定義があります。
呼び出し関係図:

6.4.3.55 GgVector gg::GgMatrix::operator∗ (const GgVector & *v*) const [inline]

ベクトルに対して投影変換を行う.
引数

v	元のベクトルの GgVector 型の変数.

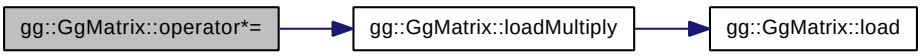
戻り値

c 変換結果の GgVector 型の値.

gg.h の 2362 行目に定義があります。

6.4.3.56 GgMatrix& gg::GgMatrix::operator∗= (const GLfloat ∗ *a*) [inline]

gg.h の 1826 行目に定義があります。
呼び出し関係図:



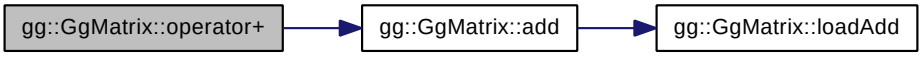
被呼び出し関係図:

6.4.3.57 GgMatrix& gg::GgMatrix::operator∗= (const GgMatrix & *m*) [inline]

gg.h の 1830 行目に定義があります。
呼び出し関係図:

6.4.3.58 GgMatrix gg::GgMatrix::operator+ (const GLfloat * a) const [inline]

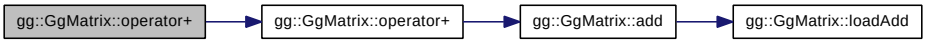
gg.h の 1842 行目に定義があります。
呼び出し関係図:



被呼び出し関係図:

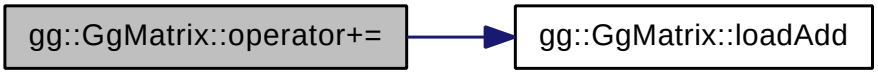
6.4.3.59 GgMatrix gg::GgMatrix::operator+ (const GgMatrix & m) const [inline]

gg.h の 1846 行目に定義があります。
呼び出し関係図:



6.4.3.60 GgMatrix& gg::GgMatrix::operator+= (const GLfloat * a) [inline]

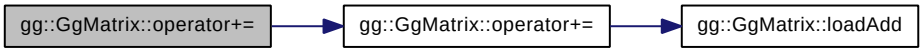
gg.h の 1810 行目に定義があります。
呼び出し関係図:



被呼び出し関係図:

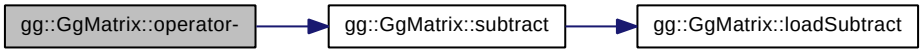
6.4.3.61 GgMatrix& gg::GgMatrix::operator+= (const GgMatrix & m) [inline]

gg.h の 1814 行目に定義があります。
呼び出し関係図:



6.4.3.62 GgMatrix gg::GgMatrix::operator- (const GLfloat * a) const [inline]

gg.h の 1850 行目に定義があります。
呼び出し関係図:



被呼び出し関係図:

6.4.3.63 GgMatrix gg::GgMatrix::operator- (const GgMatrix & m) const [inline]

gg.h の 1854 行目に定義があります。
呼び出し関係図:

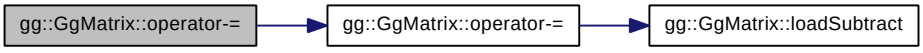
6.4.3.64 GgMatrix& gg::GgMatrix::operator-= (const GLfloat * a) [inline]

gg.h の 1818 行目に定義があります。
呼び出し関係図:

被呼び出し関係図:

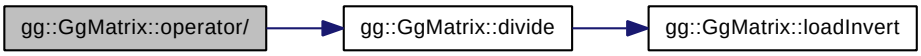
6.4.3.65 GgMatrix& gg::GgMatrix::operator-= (const GgMatrix & m) [inline]

gg.h の 1822 行目に定義があります。
呼び出し関係図:



6.4.3.66 GgMatrix gg::GgMatrix::operator/ (const GLfloat * *a*) const [inline]

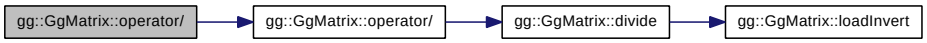
gg.h の 1866 行目に定義があります。
呼び出し関係図:



被呼び出し関係図:

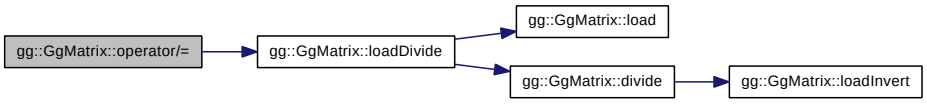
6.4.3.67 GgMatrix gg::GgMatrix::operator/ (const GgMatrix & *m*) const [inline]

gg.h の 1870 行目に定義があります。
呼び出し関係図:



6.4.3.68 GgMatrix& gg::GgMatrix::operator/= (const GLfloat * *a*) [inline]

gg.h の 1834 行目に定義があります。
呼び出し関係図:



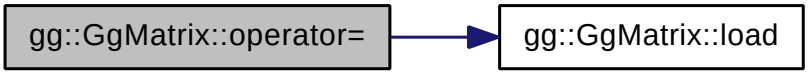
被呼び出し関係図:

6.4.3.69 GgMatrix& gg::GgMatrix::operator/= (const GgMatrix & *m*) [inline]

gg.h の 1838 行目に定義があります。
呼び出し関係図:

6.4.3.70 GgMatrix& gg::GgMatrix::operator= (const GLfloat * *a*) [inline]

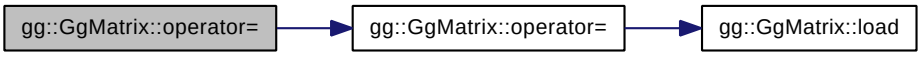
gg.h の 1802 行目に定義があります。
呼び出し関係図:



被呼び出し関係図:

6.4.3.71 GgMatrix& gg::GgMatrix::operator= (const GgMatrix & *m*) [inline]

gg.h の 1806 行目に定義があります。
呼び出し関係図:



6.4.3.72 GgMatrix gg::GgMatrix::orthogonal (GLfloat *left*, GLfloat *right*, GLfloat *bottom*, GLfloat *top*, GLfloat *zNear*, GLfloat *zFar*) const [inline]

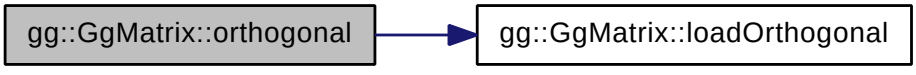
直交投影変換を乗じた結果を返す.
引数

left	ウィンドウの左端の位置.
right	ウィンドウの右端の位置.
bottom	ウィンドウの下端の位置.
top	ウィンドウの上端の位置.
zNear	視点から前方面までの位置.
zFar	視点から後方面までの位置.

戻り値

直交投影変換行列を乗じた変換行列.

gg.h の 2266 行目に定義があります。
呼び出し関係図:



6.4.3.73 GgMatrix gg::GgMatrix::perspective (GLfloat *fovy*, GLfloat *aspect*, GLfloat *zNear*, GLfloat *zFar*) const [inline]

画角を指定して透視投影変換を乗じた結果を返す.

引数

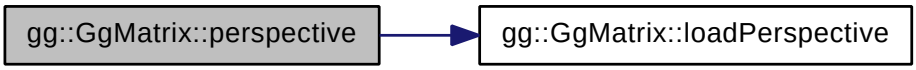
fovy	y 方向の画角.
aspect	縦横比.
zNear	視点から前方面までの位置.
zFar	視点から後方面までの位置.

戻り値

透視投影変換行列を乗じた変換行列.

gg.h の 2296 行目に定義があります。

呼び出し関係図:



6.4.3.74 void gg::GgMatrix::projection (GLfloat * *c*, const GLfloat * *v*) const [inline]

ベクトルに対して投影変換を行う.

引数

c	変換結果を格納する GLfloat 型の 4 要素の配列変数.
v	元のベクトルの GLfloat 型の 4 要素の配列変数.

gg.h の 2330 行目に定義があります。

6.4.3.75 void gg::GgMatrix::projection (GLfloat * *c*, const GgVector & *v*) const [inline]

ベクトルに対して投影変換を行う.

引数

c	変換結果を格納する GLfloat 型の 4 要素の配列変数.
v	元のベクトルの GgVector 型の変数.

gg.h の 2338 行目に定義があります。

6.4.3.76 void gg::GgMatrix::projection (GgVector & *c*, const GLfloat * *v*) const [inline]

ベクトルに対して投影変換を行う.

引数

c	変換結果を格納する GgVector 型の変数.
v	元のベクトルの GLfloat 型の 4 要素の配列変数.

gg.h の 2346 行目に定義があります。

6.4.3.77 void gg::GgMatrix::projection (GgVector & *c*, const GgVector & *v*) const [inline]

ベクトルに対して投影変換を行う.

引数

c	変換結果を格納する GgVector 型の変数.
v	元のベクトルの GgVector 型の変数.

gg.h の 2354 行目に定義があります。

6.4.3.78 GgMatrix gg::GgMatrix::rotate (GLfloat *x*, GLfloat *y*, GLfloat *z*, GLfloat *a*) const [inline]

(x, y, z) 方向のベクトルを軸とする回転変換を乗じた結果を返す.

引数

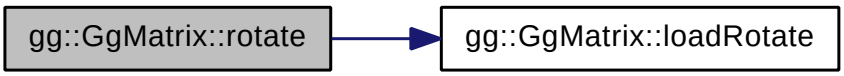
x	回転軸の x 成分.
y	回転軸の y 成分.
z	回転軸の z 成分.
a	回転角.

戻り値

(x, y, z) を軸にさらに a 回転した変換行列.

gg.h の 2179 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:

6.4.3.79 GgMatrix gg::GgMatrix::rotate (const GLfloat * *r*, GLfloat *a*) const [inline]

r 方向のベクトルを軸とする回転変換を乗じた結果を返す.

引数

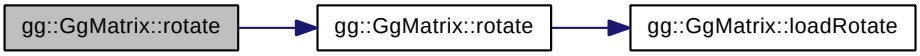
r	回転軸の方向ベクトルを格納した GLfloat 型の 3 要素の配列変数 (x, y, z).
a	回転角.

戻り値

(r[0], r[1], r[2]) を軸にさらに a 回転した変換行列.

gg.h の 2189 行目に定義があります。

呼び出し関係図:



6.4.3.80 GgMatrix gg::GgMatrix::rotate (const GgVector & *r*, GLfloat *a*) const [inline]

r 方向のベクトルを軸とする回転変換を乗じた結果を返す.

引数

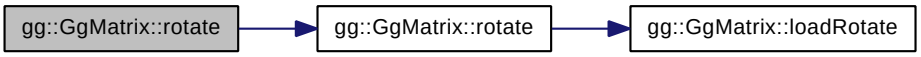
r	回転軸の方向ベクトルを格納した GgVector 型の変数.
a	回転角.

戻り値

(r[0], r[1], r[2]) を軸にさらに a 回転した変換行列.

gg.h の 2198 行目に定義があります。

呼び出し関係図:



6.4.3.81 GgMatrix gg::GgMatrix::rotate (const GLfloat * *r*) const [inline]

r 方向のベクトルを軸とする回転の変換行列を乗じた結果を返す.

引数

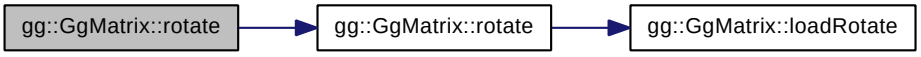
r	回転軸の方向ベクトルと回転角を格納した GLfloat 型の 4 要素の配列変数 (x, y, z, a).

戻り値

(r[0], r[1], r[2]) を軸にさらに r[3] 回転した変換行列.

gg.h の 2206 行目に定義があります。

呼び出し関係図:



6.4.3.82 GgMatrix gg::GgMatrix::rotate (const GgVector & *r*) const [inline]

r 方向のベクトルを軸とする回転の変換行列を乗じた結果を返す.

引数

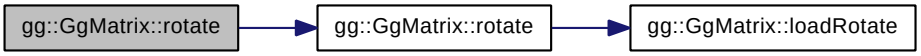
r	回転軸の方向ベクトルと回転角を格納した GgVector 型の変数).

戻り値

(r[0], r[1], r[2]) を軸にさらに r[3] 回転した変換行列.

gg.h の 2214 行目に定義があります。

呼び出し関係図:



6.4.3.83 GgMatrix gg::GgMatrix::rotateX (GLfloat *a*) const [inline]

x 軸中心の回転変換を乗じた結果を返す.

引数

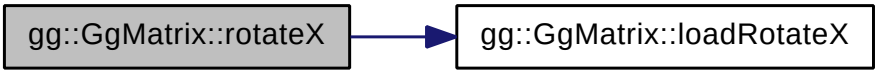
a	回転角.

戻り値

x 軸中心にさらに a 回転した変換行列.

gg.h の 2149 行目に定義があります。

呼び出し関係図:



6.4.3.84 GgMatrix gg::GgMatrix::rotateY (GLfloat *a*) const [inline]

y 軸中心の回転変換を乗じた結果を返す.

引数

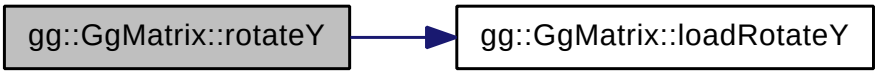
a	回転角.

戻り値

y 軸中心にさらに a 回転した変換行列.

gg.h の 2158 行目に定義があります。

呼び出し関係図:



6.4.3.85 GgMatrix gg::GgMatrix::rotateZ (GLfloat *a*) const [inline]

z 軸中心の回転変換を乗じた結果を返す.

引数

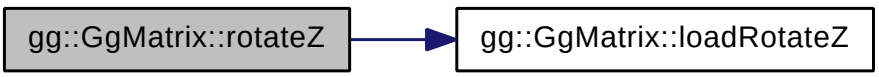
a	回転角.

戻り値

z 軸中心にさらに a 回転した変換行列.

gg.h の 2167 行目に定義があります。

呼び出し関係図:



6.4.3.86 GgMatrix gg::GgMatrix::scale (GLfloat *x*, GLfloat *y*, GLfloat *z*, GLfloat *w* = 1.0f) const [inline]

拡大縮小変換を乗じた結果を返す.

引数

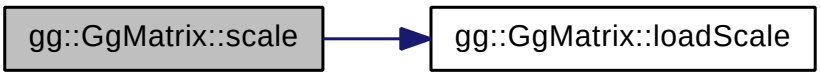
x	x 方向の拡大率.
y	y 方向の拡大率.
z	z 方向の拡大率.
w	w 移動量のスケールファクタ (= 1.0f).

戻り値

拡大縮小した結果の変換行列.

gg.h の 2124 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:

6.4.3.87 GgMatrix gg::GgMatrix::scale (const GLfloat * *s*) const [inline]

拡大縮小変換を乗じた結果を返す.

引数

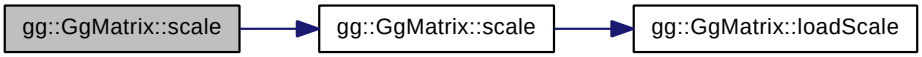
s	拡大率の GLfloat 型の 3 要素の配列変数 (x, y, z).

戻り値

拡大縮小した結果の変換行列.

gg.h の 2133 行目に定義があります。

呼び出し関係図:



6.4.3.88 GgMatrix gg::GgMatrix::scale (const GgVector & *s*) const [inline]

拡大縮小変換を乗じた結果を返す.

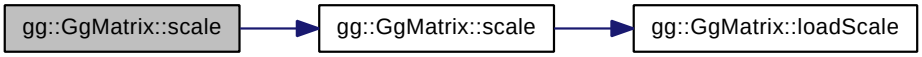
引数

s	拡大率の GgVector 型の変数.

戻り値

拡大縮小した結果の変換行列.

gg.h の 2141 行目に定義があります。
呼び出し関係図:



6.4.3.89 GgMatrix gg::GgMatrix::subtract (const GLfloat * *a*) const [inline]

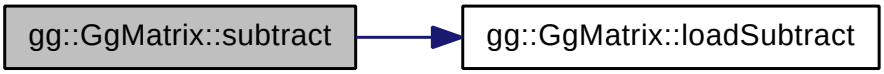
変換行列から配列に格納した変換行列を減算した値を返す.
引数

a	GLfloat 型の 16 要素の配列変数.

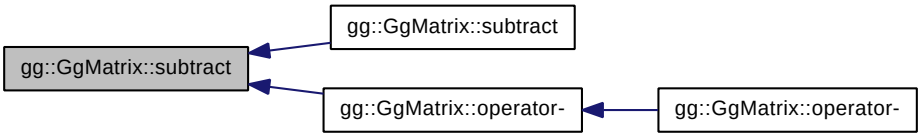
戻り値

変換行列に a を引いた GgMatrix 型の値.

gg.h の 1750 行目に定義があります。
呼び出し関係図:



被呼び出し関係図:



6.4.3.90 GgMatrix gg::GgMatrix::subtract (const GgMatrix & *m*) const [inline]

変換行列から別の変換行列を減算した値を返す.
引数

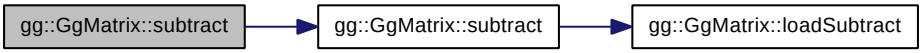
m	GgMatrix 型の変数.

戻り値

変換行列に m を引いた GgMatrix 型の値.

gg.h の 1759 行目に定義があります。

呼び出し関係図:



6.4.3.91 GgMatrix gg::GgMatrix::translate (GLfloat *x*, GLfloat *y*, GLfloat *z*, GLfloat *w* = `1.0f`) const [inline]

平行移動変換を乗じた結果を返す.

引数

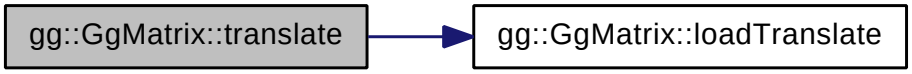
x	x 方向の移動量.
y	y 方向の移動量.
z	z 方向の移動量.
w	w 移動量のスケールファクタ (= 1.0f).

戻り値

平行移動した結果の変換行列.

gg.h の 2096 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:

6.4.3.92 GgMatrix gg::GgMatrix::translate (const GLfloat * *t*) const [inline]

平行移動変換を乗じた結果を返す.

引数

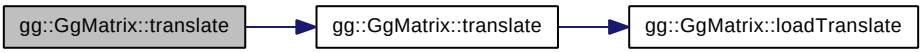
t	移動量の GLfloat 型の 3 要素の配列変数 (x, y, z).

戻り値

平行移動した結果の変換行列.

gg.h の 2105 行目に定義があります。

呼び出し関係図:



6.4.3.93 GgMatrix gg::GgMatrix::translate (const GgVector & *t*) const [inline]

平行移動変換を乗じた結果を返す.

引数

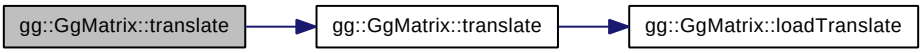
t	移動量の GgVector 型の変数.

戻り値

平行移動した結果の変換行列.

gg.h の 2113 行目に定義があります。

呼び出し関係図:



6.4.3.94 GgMatrix gg::GgMatrix::transpose ( ) const [inline]

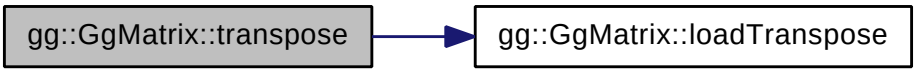
転置行列を返す.

戻り値

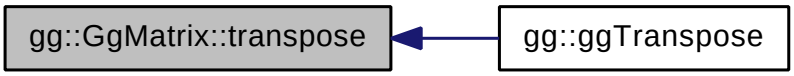
転置行列.

gg.h の 2305 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:



6.4.4 フレンドと関連関数の詳解

6.4.4.1 friend class GgQuaternion [friend]

gg.h の 1623 行目に定義があります。

このクラス詳解は次のファイルから抽出されました:

- gg.h
- gg.cpp

6.5 gg::GgNormalTexture クラス

法線マップ.

`#include <gg.h>`

公開メンバ関数

- virtual ~GgNormalTexture ()
  デストラクタ.
- GgNormalTexture ()
  コンストラクタ.
- GgNormalTexture (const GLubyte *image, GLsizei width, GLsizei height, GLenum format=GL_RED, float nz=1.0f, GLenum internal=GL_RGBA)
  メモリ上のデータから法線マップのテクスチャを作成するコンストラクタ.
- GgNormalTexture (const char *name, float nz=1.0f, GLenum internal=GL_RGBA)
  ファイルからデータを読み込んで法線マップのテクスチャを作成するコンストラクタ.
- void load (const GLubyte *hmap, GLsizei width, GLsizei height, GLenum format=GL_RED, float nz=1.0f, GLenum internal=GL_RGBA)
  メモリ上のデータから法線マップのテクスチャを作成する.
- void load (const char *name, float nz=1.0f, GLenum internal=GL_RGBA)
  ファイルからデータを読み込んで法線マップのテクスチャを作成する.

6.5.1 詳解

法線マップ.

高さマップ（グレイスケール画像）を読み込んで法線マップのテクスチャを作成する.

gg.h の 3967 行目に定義があります。

6.5.2 構築子と解体子

6.5.2.1 virtual gg::GgNormalTexture::~GgNormalTexture ( ) [inline], [virtual]

デストラクタ.

gg.h の 3975 行目に定義があります。

6.5.2.2 gg::GgNormalTexture::GgNormalTexture ( ) [inline]

コンストラクタ.

gg.h の 3978 行目に定義があります。

6.5.2.3 gg::GgNormalTexture::GgNormalTexture (const GLubyte * *image*, GLsizei *width*, GLsizei *height*, GLenum *format* = GL_RED, float *nz* = 1.0f, GLenum *internal* = GL_RGBA) [inline]

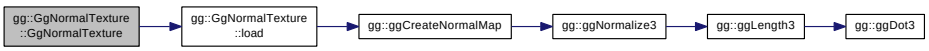
メモリ上のデータから法線マップのテクスチャを作成するコンストラクタ.

引数

image	テクスチャとして用いる画像データ, nullptr ならデータを読み込まない.
width	テクスチャとして用いる画像データの横幅.
height	テクスチャとして用いる画像データの高さ.
format	テクスチャとして用いる画像データのフォーマット (GL_RED, GL_RG, GL_RGB, GL_RGBA).
nz	法線マップの z 成分の値.
internal	テクスチャの内部フォーマット.

gg.h の 3987 行目に定義があります。

呼び出し関係図:



6.5.2.4 gg::GgNormalTexture::GgNormalTexture (const char * *name*, float *nz* = 1.0f, GLenum *internal* = GL_RGBA) [inline]

ファイルからデータを読み込んで法線マップのテクスチャを作成するコンストラクタ.

引数

name	画像ファイル名.
nz	法線マップの z 成分の値.
internal	テクスチャの内部フォーマット.

gg.h の 3998 行目に定義があります。

呼び出し関係図:

6.5.3 関数詳解

6.5.3.1 void gg::GgNormalTexture::load (const GLubyte * *hmap*, GLsizei *width*, GLsizei *height*, GLenum *format* = GL_RED, float *nz* = 1.0f, GLenum *internal* = GL_RGBA) [inline]

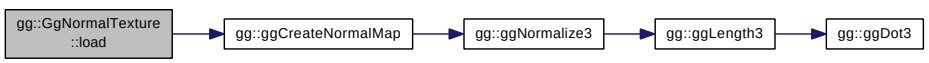
メモリ上のデータから法線マップのテクスチャを作成する.

引数

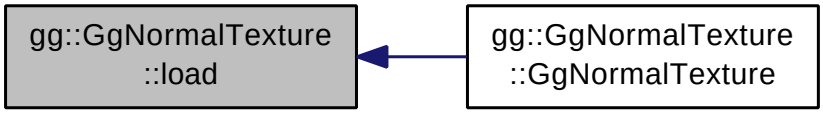
hmap	テクスチャとして用いる画像データ, nullptr ならデータを読み込まない.
width	テクスチャとして用いる画像データの横幅.
height	テクスチャとして用いる画像データの高さ.
format	テクスチャとして用いる画像データのフォーマット (GL_RED, GL_RG, GL_RGB, GL_RGBA).
nz	法線マップの z 成分の値.
internal	テクスチャの内部フォーマット.

gg.h の 4011 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:



6.5.3.2 void gg::GgNormalTexture::load (const char * *name*, float *nz* = 1.0f, GLenum *internal* = GL_RGBA)

ファイルからデータを読み込んで法線マップのテクスチャを作成する.

引数

name	画像ファイル名 (1 チャネルの TGA 画像).
nz	法線マップの z 成分の値.
internal	テクスチャの内部フォーマット.

gg.cpp の 3162 行目に定義があります。

呼び出し関係図:

このクラス詳解は次のファイルから抽出されました:

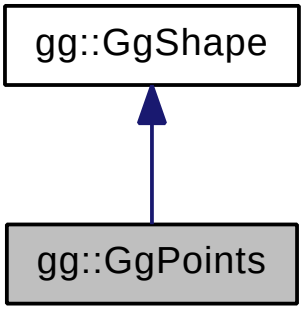
- gg.h
- gg.cpp

6.6 gg::GgPoints クラス

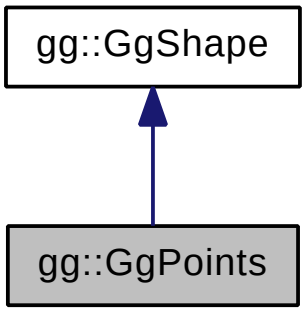
点.

`#include <gg.h>`

gg::GgPoints の継承関係図



gg::GgPoints 連携図



公開メンバ関数

- virtual ~GgPoints ()
  デストラクタ.
- GgPoints (GLenum mode=GL_POINTS)
  コンストラクタ.
- GgPoints (const GgVector *pos, GLsizei countv, GLenum mode=GL_POINTS, GLenum usage=GL_STATIC_DRAW)
  コンストラクタ.
- GLsizei getCount () const
  データの数を取り出す.
- GLuint getBuffer () const
  頂点の位置データを格納した頂点バッファオブジェクト名を取り出す.
- void send (const GgVector *pos, GLint first=0, GLsizei count=0) const
  既存のバッファオブジェクトに頂点の位置データを転送する.
- void load (const GgVector *pos, GLsizei count, GLenum usage=GL_STATIC_DRAW)
  バッファオブジェクトを確保して頂点の位置データを格納する.
- virtual void draw (GLint first=0, GLsizei count=0) const
  点の描画.

6.6.1 詳解

点.

gg.h の 4533 行目に定義があります。

6.6.2 構築子と解体子

6.6.2.1 virtual gg::GgPoints::~GgPoints ( ) [inline], [virtual]

デストラクタ.

gg.h の 4542 行目に定義があります。

6.6.2.2 gg::GgPoints::GgPoints (GLenum *mode* = GL_POINTS) [inline]

コンストラクタ.

gg.h の 4545 行目に定義があります。

6.6.2.3 gg::GgPoints::GgPoints (const GgVector * *pos*, GLsizei *countv*, GLenum *mode* = GL_POINTS, GLenum *usage* = GL_STATIC_DRAW) [inline]

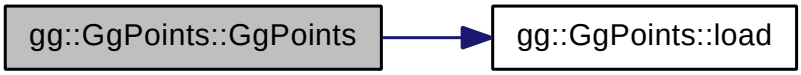
コンストラクタ.

引数

pos	この図形の頂点の位置のデータの配列 (nullptr ならデータを転送しない).
countv	頂点数.
mode	描画する基本図形の種類.
usage	バッファオブジェクトの使い方.

gg.h の 4554 行目に定義があります。

呼び出し関係図:



6.6.3 関数詳解

6.6.3.1 void gg::GgPoints::draw (GLint *first* = 0, GLsizei *count* = 0) const [virtual]

点の描画.

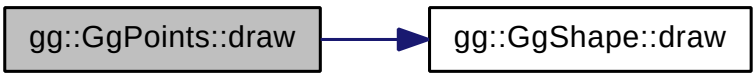
引数

first	描画を開始する最初の点の番号.
count	描画する点の数, 0 なら全部の点を描く.

gg::GgShapeを再実装しています。

gg.cpp の 4944 行目に定義があります。

呼び出し関係図:



6.6.3.2 GLuint gg::GgPoints::getBuffer ( ) const [inline]

頂点の位置データを格納した頂点バッファオブジェクト名を取り出す.

戻り値

この図形の頂点の位置データを格納した頂点バッファオブジェクト名.

gg.h の 4569 行目に定義があります。

6.6.3.3 GLsizei gg::GgPoints::getCount ( ) const [inline]

データの数を取り出す.

戻り値

この図形の頂点の位置データの数 (頂点数).

gg.h の 4562 行目に定義があります。

6.6.3.4 void gg::GgPoints::load (const GgVector * *pos*, GLsizei *count*, GLenum *usage* = GL_STATIC_DRAW) [inline]

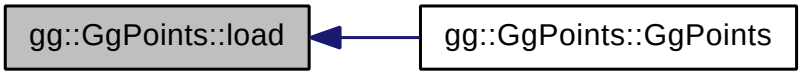
バッファオブジェクトを確保して頂点の位置データを格納する.

引数

pos	頂点の位置データが格納されてている領域の先頭のポインタ.
count	頂点のデータの数 (頂点数).
usage	バッファオブジェクトの使い方.

gg.h の 4587 行目に定義があります。

被呼び出し関係図:



6.6.3.5 void gg::GgPoints::send (const GgVector * *pos*, GLint *first* = 0, GLsizei *count* = 0) const [inline]

既存のバッファオブジェクトに頂点の位置データを転送する.

引数

pos	転送元の頂点の位置データが格納されてている領域の先頭のポインタ.
first	転送先のバッファオブジェクトの先頭の要素番号.
count	転送する頂点の位置データの数 (0 ならバッファオブジェクト全体).

gg.h の 4578 行目に定義があります。

このクラス詳解は次のファイルから抽出されました:

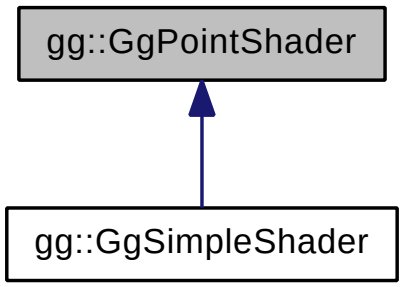
- gg.h
- gg.cpp

6.7 gg::GgPointShader クラス

点のシェーダ.

`#include <gg.h>`

gg::GgPointShader の継承関係図



公開メンバ関数

- virtual ~GgPointShader ()
  デストラクタ.
- GgPointShader ()
  コンストラクタ.

- GgPointShader (const char *vert, const char *frag=0, const char *geom=0, GLint nvarying=0, const char **varyings=0)
  コンストラクタ
- virtual void loadMatrix (const GLfloat *mp, const GLfloat *mv) const
  変換行列を設定する.
- virtual void loadMatrix (const GgMatrix &mp, const GgMatrix &mv) const
  変換行列を設定する.
- void use () const
  シェーダプログラムの使用を開始する.
- void unuse () const
  シェーダプログラムの使用を終了する.
- GLuint get () const
  シェーダのプログラム名を得る.

6.7.1 詳解

点のシェーダ.

gg.h の 4954 行目に定義があります。

6.7.2 構築子と解体子

6.7.2.1 virtual gg::GgPointShader::~GgPointShader ( ) [inline], [virtual]

デストラクタ.

gg.h の 4968 行目に定義があります。

6.7.2.2 gg::GgPointShader::GgPointShader ( ) [inline]

コンストラクタ.

gg.h の 4971 行目に定義があります。

6.7.2.3 gg::GgPointShader::GgPointShader (const char * *vert*, const char * *frag* = 0, const char * *geom* = 0, GLint *nvarying* = 0, const char ** *varyings* = 0) [inline]

コンストラクタ

引数

vert	バーテックスシェーダのソースファイル名.
frag	フラグメントシェーダのソースファイル名 (0 なら不使用).
geom	ジオメトリシェーダのソースファイル名 (0 なら不使用).
nvarying	フィードバックする varying 変数の数 (0 なら不使用).
varyings	フィードバックする varying 変数のリスト.

gg.h の 4979 行目に定義があります。

6.7.3 関数詳解

6.7.3.1 GLuint gg::GgPointShader::get ( ) const [inline]

シェーダのプログラム名を得る.

戻り値

シェーダのプログラム名.

gg.h の 5025 行目に定義があります。

6.7.3.2 virtual void gg::GgPointShader::loadMatrix (const GLfloat * *mp*, const GLfloat * *mv*) const [inline], [virtual]

変換行列を設定する.

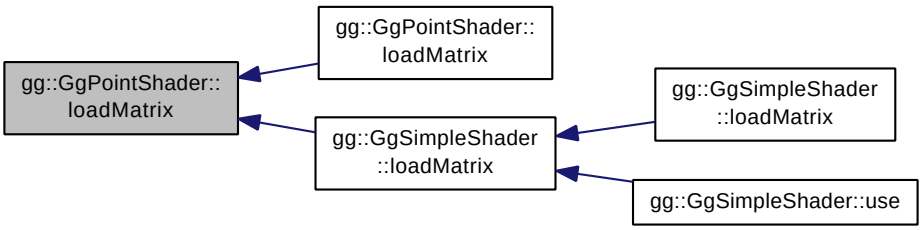
引数

mp	GLfloat 型の 16 要素の配列変数に格納された投影変換行列.
mv	GLfloat 型の 16 要素の配列変数に格納されたモデルビュー変換行列.

gg::GgSimpleShaderで再実装されています。

gg.h の 4996 行目に定義があります。

被呼び出し関係図:



6.7.3.3 virtual void gg::GgPointShader::loadMatrix (const GgMatrix & *mp*, const GgMatrix & *mv*) const [inline], [virtual]

変換行列を設定する.

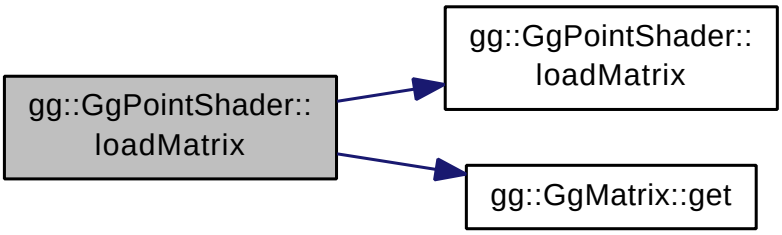
引数

mp	GgMatrix 型の投影変換行列.
mv	GgMatrix 型のモデルビュー変換行列.

gg::GgSimpleShaderで再実装されています。

gg.h の 5006 行目に定義があります。

呼び出し関係図:



6.7.3.4 void gg::GgPointShader::unuse ( ) const [inline]

シェーダプログラムの使用を終了する.

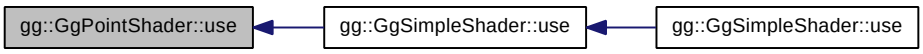
gg.h の 5018 行目に定義があります。

6.7.3.5 void gg::GgPointShader::use ( ) const [inline]

シェーダプログラムの使用を開始する.

gg.h の 5012 行目に定義があります。

被呼び出し関係図:



このクラス詳解は次のファイルから抽出されました:

- gg.h

6.8 gg::GgQuaternion クラス

四元数.

`#include <gg.h>`

公開メンバ関数

- ~GgQuaternion ()
  デストラクタ.
- GgQuaternion ()
  コンストラクタ.
- GgQuaternion (GLfloat x, GLfloat y, GLfloat z, GLfloat w)
  コンストラクタ.
- GgQuaternion (const GgVector &v)
  コンストラクタ.
- GgQuaternion (const GLfloat *a)
  コンストラクタ.
- GgQuaternion (const GgQuaternion &q)
  コピーコンストラクタ.
- GLfloat norm () const
  四元数のノルムを求める.
- GgQuaternion & load (GLfloat x, GLfloat y, GLfloat z, GLfloat w)
  四元数を格納する.
- GgQuaternion & load (const GgVector &v)
  四元数を格納する.
- GgQuaternion & load (const GLfloat *a)
  四元数を格納する.
- GgQuaternion & load (const GgQuaternion &q)
  四元数を格納する.
- GgQuaternion & loadAdd (GLfloat x, GLfloat y, GLfloat z, GLfloat w)

四元数に別の四元数を加算した結果を格納する.

- GgQuaternion & loadAdd (const GgVector &v)
  四元数に別の四元数を加算した結果を格納する.
- GgQuaternion & loadAdd (const GLfloat *a)
  四元数に別の四元数を加算した結果を格納する.
- GgQuaternion & loadAdd (const GgQuaternion &q)
  四元数に別の四元数を加算した結果を格納する.
- GgQuaternion & loadSubtract (GLfloat x, GLfloat y, GLfloat z, GLfloat w)
  四元数から別の四元数を減算した結果を格納する.
- GgQuaternion & loadSubtract (const GgVector &v)
  四元数から別の四元数を減算した結果を格納する.
- GgQuaternion & loadSubtract (const GLfloat *a)
  四元数から別の四元数を減算した結果を格納する.
- GgQuaternion & loadSubtract (const GgQuaternion &q)
  四元数から別の四元数を減算した結果を格納する.
- GgQuaternion & loadMultiply (GLfloat x, GLfloat y, GLfloat z, GLfloat w)
  四元数に別の四元数を乗算した結果を格納する.
- GgQuaternion & loadMultiply (const GgVector &v)
  四元数に別の四元数を乗算した結果を格納する.
- GgQuaternion & loadMultiply (const GLfloat *a)
  四元数に別の四元数を乗算した結果を格納する.
- GgQuaternion & loadMultiply (const GgQuaternion &q)
  四元数に別の四元数を乗算した結果を格納する.
- GgQuaternion & loadDivide (GLfloat x, GLfloat y, GLfloat z, GLfloat w)
  四元を別の四元数で除算した結果を格納する.
- GgQuaternion & loadDivide (const GgVector &v)
  四元を別の四元数で除算した結果を格納する.
- GgQuaternion & loadDivide (const GLfloat *a)
  四元を別の四元数で除算した結果を格納する.
- GgQuaternion & loadDivide (const GgQuaternion &q)
  四元を別の四元数で除算した結果を格納する.
- GgQuaternion add (GLfloat x, GLfloat y, GLfloat z, GLfloat w) const
  四元数に別の四元数を加算した結果を返す.
- GgQuaternion add (const GgVector &v) const
  四元数に別の四元数を加算した結果を返す.
- GgQuaternion add (const GLfloat *a) const
  四元数に別の四元数を加算した結果を返す.
- GgQuaternion add (const GgQuaternion &q) const
  四元数に別の四元数を加算した結果を返す.
- GgQuaternion subtract (GLfloat x, GLfloat y, GLfloat z, GLfloat w) const
  四元数から別の四元数を減算した結果を返す.
- GgQuaternion subtract (const GgVector &v) const
  四元数から別の四元数を減算した結果を返す.
- GgQuaternion subtract (const GLfloat *a) const
  四元数から別の四元数を減算した結果を返す.
- GgQuaternion subtract (const GgQuaternion &q) const
  四元数から別の四元数を減算した結果を返す.
- GgQuaternion multiply (GLfloat x, GLfloat y, GLfloat z, GLfloat w) const
  四元数に別の四元数を乗算した結果を返す.
- GgQuaternion multiply (const GgVector &v) const
  四元数に別の四元数を乗算した結果を返す.

- GgQuaternion multiply (const GLfloat *a) const
  四元数に別の四元数を乗算した結果を返す.
- GgQuaternion multiply (const GgQuaternion &q) const
  四元数に別の四元数を乗算した結果を返す.
- GgQuaternion divide (GLfloat x, GLfloat y, GLfloat z, GLfloat w) const
  四元数を別の四元数で除算した結果を返す.
- GgQuaternion divide (const GgVector &v) const
  四元数を別の四元数で除算した結果を返す.
- GgQuaternion divide (const GLfloat *a) const
  四元数を別の四元数で除算した結果を返す.
- GgQuaternion divide (const GgQuaternion &q) const
  四元数を別の四元数で除算した結果を返す.
- GgQuaternion & operator= (const GLfloat *a)
- GgQuaternion & operator= (const GgQuaternion &q)
- GgQuaternion & operator+= (const GLfloat *a)
- GgQuaternion & operator+= (const GgQuaternion &q)
- GgQuaternion & operator-= (const GLfloat *a)
- GgQuaternion & operator-= (const GgQuaternion &q)
- GgQuaternion & operator*= (const GLfloat *a)
- GgQuaternion & operator*= (const GgQuaternion &q)
- GgQuaternion & operator/= (const GLfloat *a)
- GgQuaternion & operator/= (const GgQuaternion &q)
- GgQuaternion operator+ (const GLfloat *a) const
- GgQuaternion operator+ (const GgQuaternion &q) const
- GgQuaternion operator- (const GLfloat *a) const
- GgQuaternion operator- (const GgQuaternion &q) const
- GgQuaternion operator* (const GLfloat *a) const
- GgQuaternion operator* (const GgQuaternion &q) const
- GgQuaternion operator/ (const GLfloat *a) const
- GgQuaternion operator/ (const GgQuaternion &q) const
- GgQuaternion & loadMatrix (const GLfloat *a)
  回転の変換行列を表す四元数を格納する.
- GgQuaternion & loadMatrix (const GgMatrix &m)
  回転の変換行列 *m* を表す四元数を格納する.
- GgQuaternion & loadIdentity ()
  単位元を格納する.
- GgQuaternion & loadRotate (GLfloat x, GLfloat y, GLfloat z, GLfloat a)
  (x, y, z) を軸として角度 *a* 回転する四元数を格納する.
- GgQuaternion & loadRotate (const GLfloat *v, GLfloat a)
  (v[0], v[1], v[2]) を軸として角度 *a* 回転する四元数を格納する.
- GgQuaternion & loadRotate (const GLfloat *v)
  (v[0], v[1], v[2]) を軸として角度 *v[3]* 回転する四元数を格納する.
- GgQuaternion & loadRotateX (GLfloat a)
  x 軸中心に角度 *a* 回転する四元数を格納する.
- GgQuaternion & loadRotateY (GLfloat a)
  y 軸中心に角度 *a* 回転する四元数を格納する.
- GgQuaternion & loadRotateZ (GLfloat a)
  z 軸中心に角度 *a* 回転する四元数を格納する.
- GgQuaternion rotate (GLfloat x, GLfloat y, GLfloat z, GLfloat a) const
  四元数を *(x, y, z)* を軸として角度 *a* 回転した四元数を返す.
- GgQuaternion rotate (const GLfloat *v, GLfloat a) const
  四元数を *(v[0], v[1], v[2])* を軸として角度 *a* 回転した四元数を返す.

- GgQuaternion rotate (const GLfloat *v) const
  四元数を *(v[0], v[1], v[2])* **を軸として角度** *v[3]* **回転した四元数を返す**.
- GgQuaternion rotateX (GLfloat a) const
  四元数を *x* **軸中心に角度** *a* **回転した四元数を返す**.
- GgQuaternion rotateY (GLfloat a) const
  四元数を *y* **軸中心に角度** *a* **回転した四元数を返す**.
- GgQuaternion rotateZ (GLfloat a) const
  四元数を *z* **軸中心に角度** *a* **回転した四元数を返す**.
- GgQuaternion & loadEuler (GLfloat heading, GLfloat pitch, GLfloat roll)
  オイラー角 *(heading, pitch, roll)* **で与えられた回転を表す四元数を格納する**.
- GgQuaternion & loadEuler (const GLfloat *e)
  オイラー角 *(e[0], e[1], e[2])* **で与えられた回転を表す四元数を格納する**.
- GgQuaternion euler (GLfloat heading, GLfloat pitch, GLfloat roll) const
  四元数をオイラー角 *(heading, pitch, roll)* **で回転した四元数を返す**.
- GgQuaternion euler (const GLfloat *e) const
  四元数をオイラー角 *(e[0], e[1], e[2])* **で回転した四元数を返す**.
- GgQuaternion & loadSlerp (const GLfloat *a, const GLfloat *b, GLfloat t)
  球面線形補間の結果を格納する.
- GgQuaternion & loadSlerp (const GgQuaternion &q, const GgQuaternion &r, GLfloat t)
  球面線形補間の結果を格納する.
- GgQuaternion & loadSlerp (const GgQuaternion &q, const GLfloat *a, GLfloat t)
  球面線形補間の結果を格納する.
- GgQuaternion & loadSlerp (const GLfloat *a, const GgQuaternion &q, GLfloat t)
  球面線形補間の結果を格納する.
- GgQuaternion & loadNormalize (const GLfloat *a)
  引数に指定した四元数を正規化して格納する.
- GgQuaternion & loadNormalize (const GgQuaternion &q)
  引数に指定した四元数を正規化して格納する.
- GgQuaternion & loadConjugate (const GLfloat *a)
  引数に指定した四元数の共役四元数を格納する.
- GgQuaternion & loadConjugate (const GgQuaternion &q)
  引数に指定した四元数の共役四元数を格納する.
- GgQuaternion & loadInvert (const GLfloat *a)
  引数に指定した四元数の逆元を格納する.
- GgQuaternion & loadInvert (const GgQuaternion &q)
  引数に指定した四元数の逆元を格納する.
- GgQuaternion slerp (GLfloat *a, GLfloat t) const
  球面線形補間の結果を返す.
- GgQuaternion slerp (const GgQuaternion &q, GLfloat t) const
  球面線形補間の結果を返す.
- GgQuaternion normalize () const
  正規化する.
- GgQuaternion conjugate () const
  共役四元数に変換する.
- GgQuaternion invert () const
  逆元に変換する.
- const GLfloat * get () const
  四元数を取り出す.
- void get (GLfloat *a) const
  四元数を取り出す.
- void getMatrix (GLfloat *a) const

四元数が表す回転の変換行列を *a* に求める.

- void getMatrix (GgMatrix &m) const

  四元数が表す回転の変換行列を *m* に求める.

- GgMatrix getMatrix () const

  四元数が表す回転の変換行列を取り出す.

- void getConjugateMatrix (GLfloat *a) const

  四元数の共役が表す回転の変換行列を *a* に求める.

- void getConjugateMatrix (GgMatrix &m) const

  四元数の共役が表す回転の変換行列を *m* に求める.

- GgMatrix getConjugateMatrix () const

  四元数の共役が表す回転の変換行列を取り出す.

6.8.1 詳解

四元数.

gg.h の 2659 行目に定義があります。

6.8.2 構築子と解体子

6.8.2.1 gg::GgQuaternion::~GgQuaternion ( ) [inline]

デストラクタ.

gg.h の 2679 行目に定義があります。

6.8.2.2 gg::GgQuaternion::GgQuaternion ( ) [inline]

コンストラクタ.

gg.h の 2682 行目に定義があります。

6.8.2.3 gg::GgQuaternion::GgQuaternion (GLfloat *x*, GLfloat *y*, GLfloat *z*, GLfloat *w*) [inline]

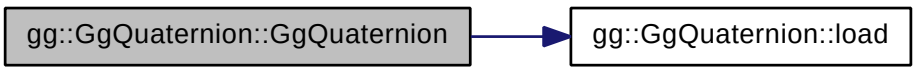
コンストラクタ.

引数

x	四元数の x 要素.
y	四元数の y 要素.
z	四元数の z 要素.
w	四元数の w 要素.

gg.h の 2689 行目に定義があります。

呼び出し関係図:



6.8.2.4 gg::GgQuaternion::GgQuaternion (const GgVector & *v*) [inline]

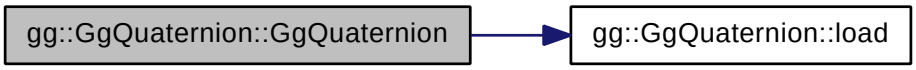
コンストラクタ.

引数

v	四元数を格納した GgVector 型の変数.

gg.h の 2696 行目に定義があります。

呼び出し関係図:



6.8.2.5 gg::GgQuaternion::GgQuaternion (const GLfloat * *a*) [inline]

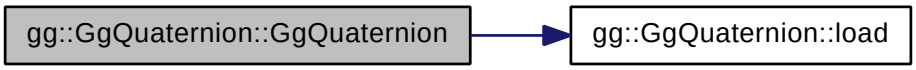
コンストラクタ.

引数

a	四元数を格納した GLfloat 型の 4 要素の配列変数.

gg.h の 2703 行目に定義があります。

呼び出し関係図:



6.8.2.6 gg::GgQuaternion::GgQuaternion (const GgQuaternion & *q*) [inline]

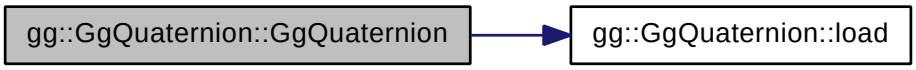
コピーコンストラクタ.

引数

q	GgQuaternion 型の四元数.

gg.h の 2710 行目に定義があります。

呼び出し関係図:



6.8.3 関数詳解

6.8.3.1 GgQuaternion gg::GgQuaternion::add (GLfloat *x*, GLfloat *y*, GLfloat *z*, GLfloat *w*) const [inline]

四元数に別の四元数を加算した結果を返す.

引数

x	加える四元数の x 要素.

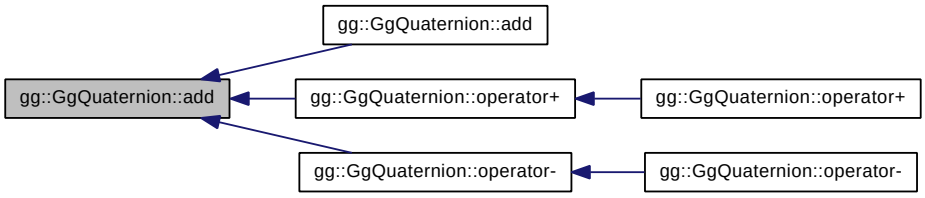
y	加える四元数の y 要素.
z	加える四元数の z 要素.
w	加える四元数の w 要素.

戻り値

(x, y, z, w) を加えた四元数.

gg.h の 2918 行目に定義があります。

被呼び出し関係図:



6.8.3.2 GgQuaternion gg::GgQuaternion::add (const GgVector & *v*) const [inline]

四元数に別の四元数を加算した結果を返す.

引数

v	四元数を格納した GgVector 型の変数.

戻り値

v を加えた四元数.

gg.h の 2931 行目に定義があります。

呼び出し関係図:

6.8.3.3 GgQuaternion gg::GgQuaternion::add (const GLfloat * *a*) const [inline]

四元数に別の四元数を加算した結果を返す.

引数

a	四元数を格納した GLfloat 型の 4 要素の配列変数.

戻り値

a を加えた四元数.

gg.h の 2939 行目に定義があります。

呼び出し関係図:

6.8.3.4 GgQuaternion gg::GgQuaternion::add (const GgQuaternion & *q*) const [inline]

四元数に別の四元数を加算した結果を返す.

引数

q	GgQuaternion 型の四元数.

戻り値

q を加えた四元数.

gg.h の 2947 行目に定義があります。

呼び出し関係図:

6.8.3.5 GgQuaternion gg::GgQuaternion::conjugate ( ) const [inline]

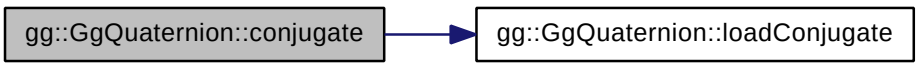
共役四元数に変換する.

戻り値

共役四元数.

gg.h の 3407 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:

6.8.3.6 GgQuaternion gg::GgQuaternion::divide (GLfloat *x*, GLfloat *y*, GLfloat *z*, GLfloat *w*) const [inline]

四元数を別の四元数で除算した結果を返す.

引数

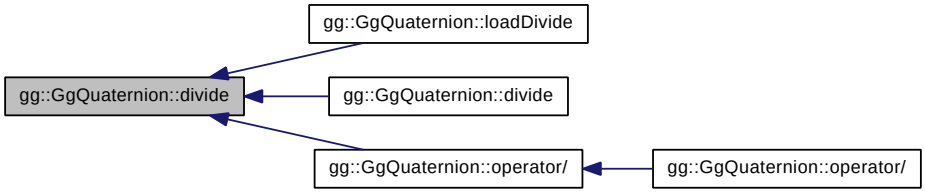
x	割る四元数の x 要素.
y	割る四元数の y 要素.
z	割る四元数の z 要素.
w	割る四元数の w 要素.

戻り値

(x, y, z, w) を割った四元数.

gg.h の 3036 行目に定義があります。

被呼び出し関係図:



6.8.3.7 GgQuaternion gg::GgQuaternion::divide (const GgVector & *v*) const [inline]

四元数を別の四元数で除算した結果を返す.

引数

v	四元数を格納した GgVector 型の変数.

戻り値

v で割った四元数.

gg.h の 3045 行目に定義があります。

呼び出し関係図:

6.8.3.8 GgQuaternion gg::GgQuaternion::divide (const GLfloat * *a*) const [inline]

四元数を別の四元数で除算した結果を返す.

引数

a	四元数を格納した GLfloat 型の 4 要素の配列変数.

戻り値

a で割った四元数.

gg.h の 3053 行目に定義があります。

呼び出し関係図:

6.8.3.9 GgQuaternion gg::GgQuaternion::divide (const GgQuaternion & *q*) const [inline]

四元数を別の四元数で除算した結果を返す.

引数

q	GgQuaternion 型の四元数.

戻り値

q で割った四元数.

gg.h の 3064 行目に定義があります。

呼び出し関係図:

6.8.3.10 GgQuaternion gg::GgQuaternion::euler (GLfloat *heading*, GLfloat *pitch*, GLfloat *roll*) const [inline]

四元数をオイラー角 (heading, pitch, roll) で回転した四元数を返す.

引数

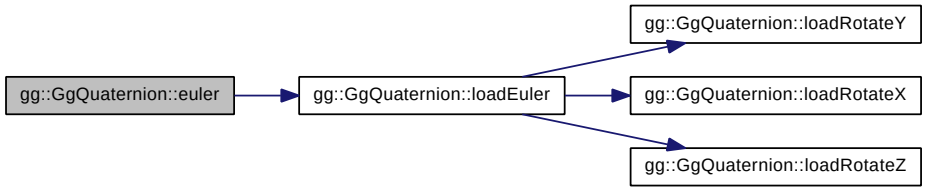
heading	y 軸中心の回転角.
pitch	x 軸中心の回転角.
roll	z 軸中心の回転角.

戻り値

回転した四元数.

gg.h の 3280 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:

6.8.3.11 GgQuaternion gg::GgQuaternion::euler (const GLfloat * e) const [inline]

四元数をオイラー角 (e[0], e[1], e[2]) で回転した四元数を返す.

引数

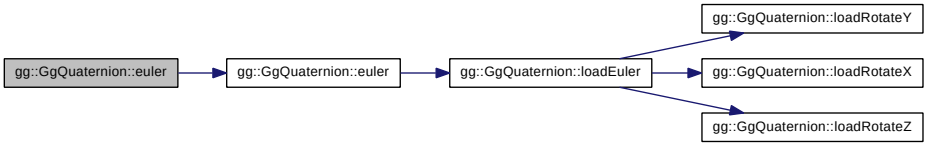
e	オイラー角を表す GLfloat 型の 3 要素の配列変数 (heading, pitch, roll).

戻り値

回転した四元数.

gg.h の 3289 行目に定義があります。

呼び出し関係図:



6.8.3.12 const GLfloat* gg::GgQuaternion::get ( ) const [inline]

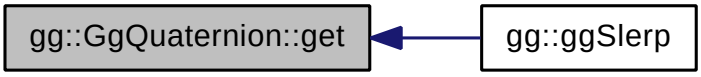
四元数を取り出す.

戻り値

四元数を表す GLfloat 型の 4 要素の配列変数.

gg.h の 3425 行目に定義があります。

被呼び出し関係図:



6.8.3.13 void gg::GgQuaternion::get (GLfloat * a) const [inline]

四元数を取り出す.

引数

a	四元数を格納する GLfloat 型の 4 要素の配列変数.

gg.h の 3432 行目に定義があります。

6.8.3.14 void gg::GgQuaternion::getConjugateMatrix (GLfloat * *a*) const [inline]

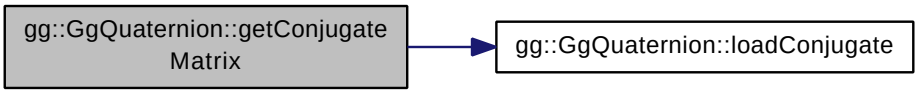
四元数の共役が表す回転の変換行列を a に求める.

引数

a	回転の変換行列を格納する GLfloat 型の 16 要素の配列変数.

gg.h の 3465 行目に定義があります。

呼び出し関係図:



6.8.3.15 void gg::GgQuaternion::getConjugateMatrix (GgMatrix & *m*) const [inline]

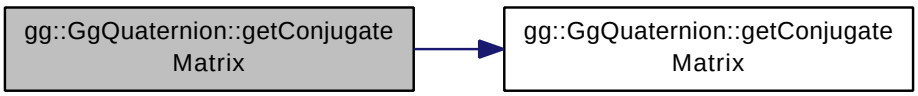
四元数の共役が表す回転の変換行列を m に求める.

引数

m	回転の変換行列を格納する GgMatrix 型の変数.

gg.h の 3474 行目に定義があります。

呼び出し関係図:



6.8.3.16 GgMatrix gg::GgQuaternion::getConjugateMatrix ( ) const [inline]

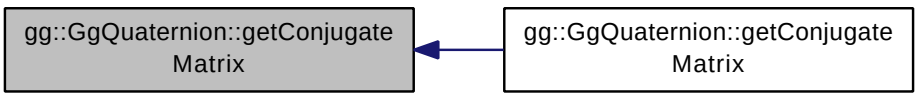
四元数の共役が表す回転の変換行列を取り出す.

戻り値

回転の変換を表す GgMatrix 型の変換行列.

gg.h の 3481 行目に定義があります。

被呼び出し関係図:



6.8.3.17 void gg::GgQuaternion::getMatrix (GLfloat * *a*) const [inline]

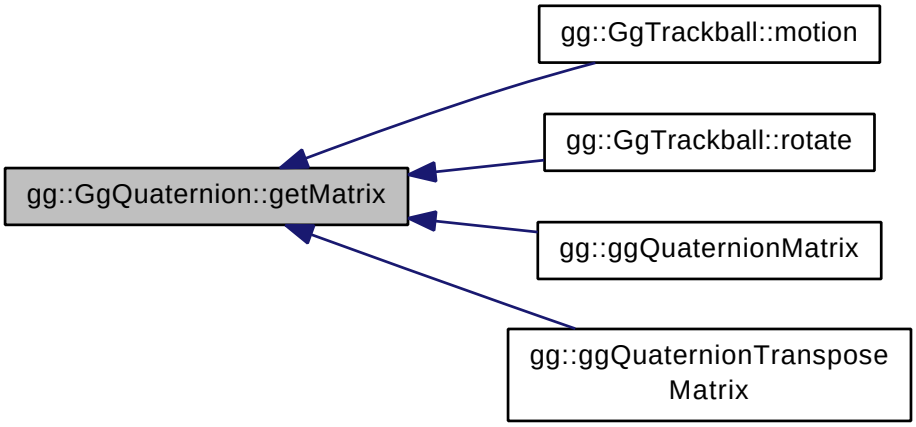
四元数が表す回転の変換行列を a に求める.

引数

a	回転の変換行列を格納する GLfloat 型の 16 要素の配列変数.

gg.h の 3442 行目に定義があります。

被呼び出し関係図:



6.8.3.18 void gg::GgQuaternion::getMatrix (GgMatrix & *m*) const [inline]

四元数が表す回転の変換行列を m に求める.

引数

m	回転の変換行列を格納する GgMatrix 型の変数.

gg.h の 3449 行目に定義があります。

呼び出し関係図:

6.8.3.19 GgMatrix gg::GgQuaternion::getMatrix ( ) const [inline]

四元数が表す回転の変換行列を取り出す.

戻り値

回転の変換を表す GgMatrix 型の変換行列.

gg.h の 3456 行目に定義があります。

被呼び出し関係図:

6.8.3.20 GgQuaternion gg::GgQuaternion::invert ( ) const [inline]

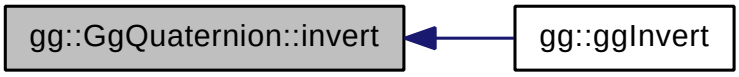
逆元に変換する.

戻り値

四元数の逆元.

gg.h の 3416 行目に定義があります。
呼び出し関係図:

被呼び出し関係図:



6.8.3.21 GgQuaternion& gg::GgQuaternion::load (GLfloat *x*, GLfloat *y*, GLfloat *z*, GLfloat *w*) [inline]

四元数を格納する.

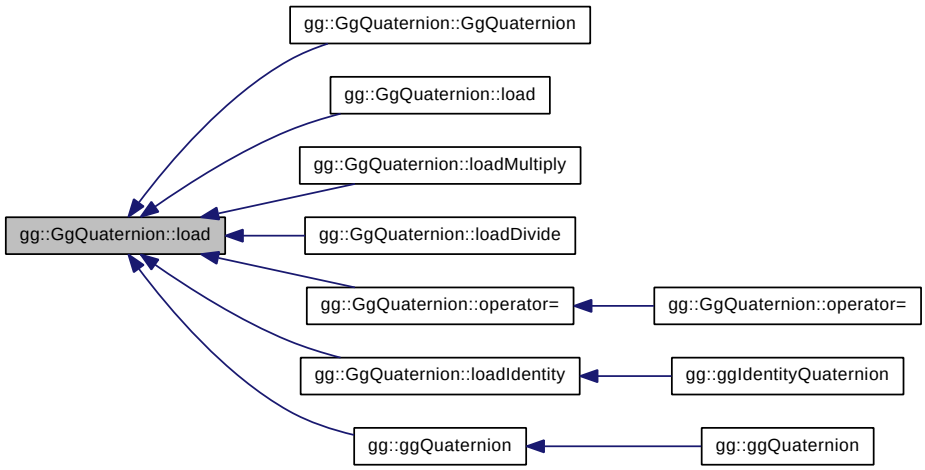
引数

x	四元数の x 要素.
y	四元数の y 要素.
z	四元数の z 要素.
w	四元数の w 要素.

戻り値

設定した四元数.

gg.h の 2728 行目に定義があります。
被呼び出し関係図:



6.8.3.22 GgQuaternion& gg::GgQuaternion::load (const GgVector & *v*) [inline]

四元数を格納する.

引数

v	四元数を格納した GgVector 型の変数.

戻り値

設定した四元数.

gg.h の 2740 行目に定義があります。

6.8.3.23 GgQuaternion& gg::GgQuaternion::load (const GLfloat * *a*) [inline]

四元数を格納する.

引数

a	四元数を格納した GLfloat 型の 4 要素の配列変数.

戻り値

設定した四元数.

gg.h の 2749 行目に定義があります。

呼び出し関係図:

6.8.3.24 GgQuaternion& gg::GgQuaternion::load (const GgQuaternion & *q*) [inline]

四元数を格納する.

引数

q	GgQuaternion 型の四元数.

戻り値

設定した四元数.

gg.h の 2757 行目に定義があります。

呼び出し関係図:

6.8.3.25 GgQuaternion& gg::GgQuaternion::loadAdd (GLfloat *x*, GLfloat *y*, GLfloat *z*, GLfloat *w*) [inline]

四元数に別の四元数を加算した結果を格納する.

引数

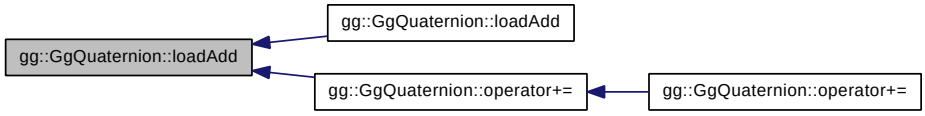
x	加える四元数の x 要素.
y	加える四元数の y 要素.
z	加える四元数の z 要素.
w	加える四元数の w 要素.

戻り値

(x, y, z, w) を加えた四元数.

gg.h の 2768 行目に定義があります。

被呼び出し関係図:



6.8.3.26 GgQuaternion& gg::GgQuaternion::loadAdd (const GgVector & *v*) [inline]

四元数に別の四元数を加算した結果を格納する.

引数

v	四元数を格納した GgVector 型の変数.

戻り値

v を加えた四元数.

gg.h の 2780 行目に定義があります。

呼び出し関係図:

6.8.3.27 GgQuaternion& gg::GgQuaternion::loadAdd (const GLfloat * *a*) [inline]

四元数に別の四元数を加算した結果を格納する.

引数

a	四元数を格納した GLfloat 型の 4 要素の配列変数.

戻り値

a を加えた四元数.

gg.h の 2788 行目に定義があります。

呼び出し関係図:

6.8.3.28 GgQuaternion& gg::GgQuaternion::loadAdd (const GgQuaternion & *q*) [inline]

四元数に別の四元数を加算した結果を格納する.

引数

q	GgQuaternion 型の四元数.

戻り値

q を加えた四元数.

gg.h の 2796 行目に定義があります。

呼び出し関係図:

6.8.3.29 gg::GgQuaternion & gg::GgQuaternion::loadConjugate (const GLfloat * *a*)

引数に指定した四元数の共役四元数を格納する.

引数

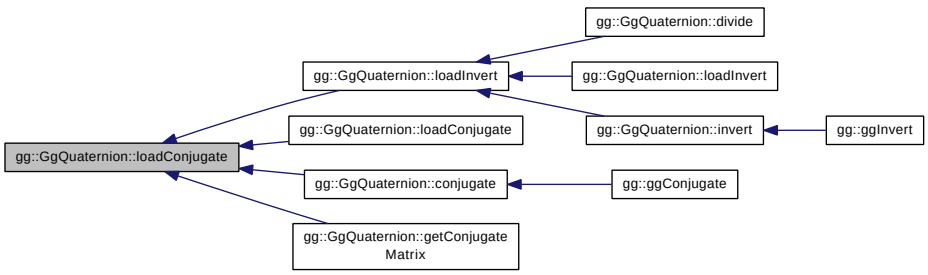
a	四元数を格納した GLfloat 型の 4 要素の配列変数.

戻り値

共役四元数.

gg.cpp の 4796 行目に定義があります。

被呼び出し関係図:



6.8.3.30 GgQuaternion& gg::GgQuaternion::loadConjugate (const GgQuaternion & *q*) [inline]

引数に指定した四元数の共役四元数を格納する.

引数

q	GgQuaternion 型の四元数.

戻り値

共役四元数.

gg.h の 3356 行目に定義があります。

呼び出し関係図:

6.8.3.31 GgQuaternion& gg::GgQuaternion::loadDivide (GLfloat *x*, GLfloat *y*, GLfloat *z*, GLfloat *w*) [inline]

四元を別の四元数で除算した結果を格納する.

引数

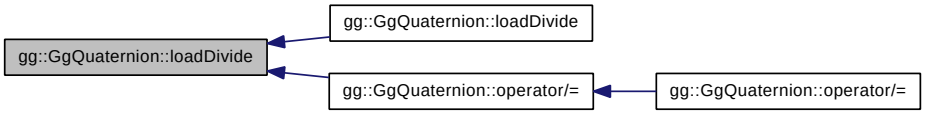
x	割る四元数の x 要素.
y	割る四元数の y 要素.
z	割る四元数の z 要素.
w	割る四元数の w 要素.

戻り値

(x, y, z, w) を割った四元数.

gg.h の 2882 行目に定義があります。

被呼び出し関係図:



6.8.3.32 GgQuaternion& gg::GgQuaternion::loadDivide (const GgVector & *v*) [inline]

四元を別の四元数で除算した結果を格納する.

引数

v	四元数を格納した GgVector 型の変数.

戻り値

v で割った四元数.

gg.h の 2891 行目に定義があります。

呼び出し関係図:

6.8.3.33 GgQuaternion& gg::GgQuaternion::loadDivide (const GLfloat * *a*) [inline]

四元を別の四元数で除算した結果を格納する.

引数

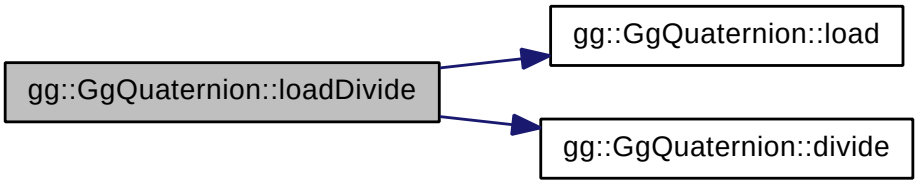
a	四元数を格納した GLfloat 型の 4 要素の配列変数.

戻り値

a で割った四元数.

gg.h の 2899 行目に定義があります。

呼び出し関係図:



6.8.3.34 GgQuaternion& gg::GgQuaternion::loadDivide (const GgQuaternion & *q*) [inline]

四元を別の四元数で除算した結果を格納する.

引数

q	GgQuaternion 型の四元数.

戻り値

q で割った四元数.

gg.h の 2907 行目に定義があります。

呼び出し関係図:

6.8.3.35 gg::GgQuaternion & gg::GgQuaternion::loadEuler (GLfloat *heading*, GLfloat *pitch*, GLfloat *roll*)

オイラー角 (heading, pitch, roll) で与えられた回転を表す四元数を格納する.

引数

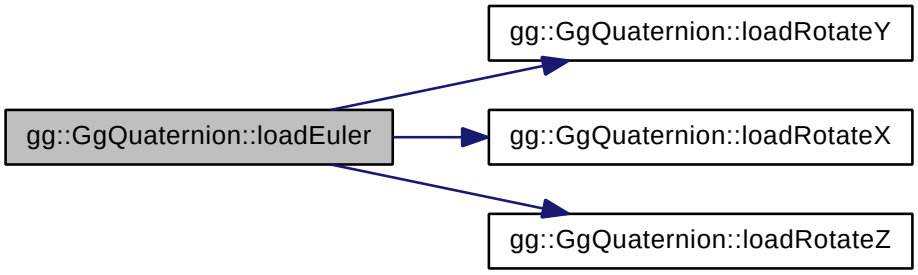
heading	y 軸中心の回転角.
pitch	x 軸中心の回転角.
roll	z 軸中心の回転角.

戻り値

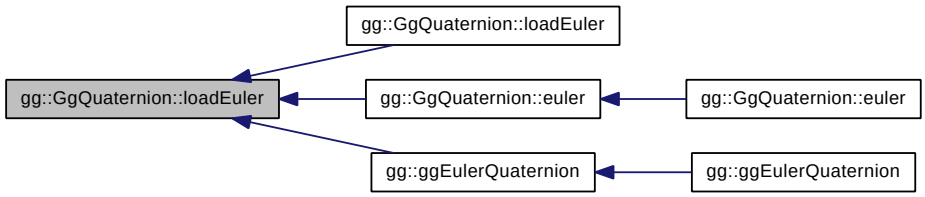
格納した回転を表す四元数.

gg.cpp の 4765 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:



6.8.3.36 GgQuaternion& gg::GgQuaternion::loadEuler (const GLfloat * *e*) [inline]

オイラー角 (e[0], e[1], e[2]) で与えられた回転を表す四元数を格納する.

引数

e	オイラー角を表す GLfloat 型の 3 要素の配列変数 (heading, pitch, roll).

戻り値

格納した回転を表す四元数.

gg.h の 3270 行目に定義があります。

呼び出し関係図:

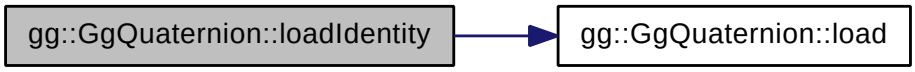
6.8.3.37 GgQuaternion& gg::GgQuaternion::loadIdentity ( ) [inline]

単位元を格納する.

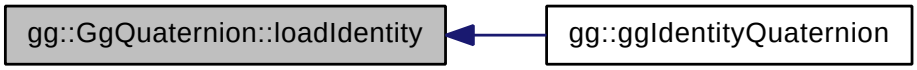
戻り値

格納された単位元.

gg.h の 3162 行目に定義があります。
呼び出し関係図:



被呼び出し関係図:



6.8.3.38 gg::GgQuaternion & gg::GgQuaternion::loadInvert (const GLfloat * a)

引数に指定した四元数の逆元を格納する.

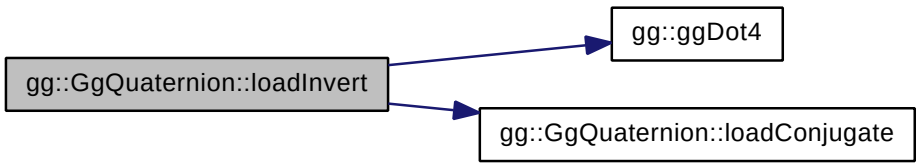
引数

a	四元数を格納した GLfloat 型の 4 要素の配列変数.

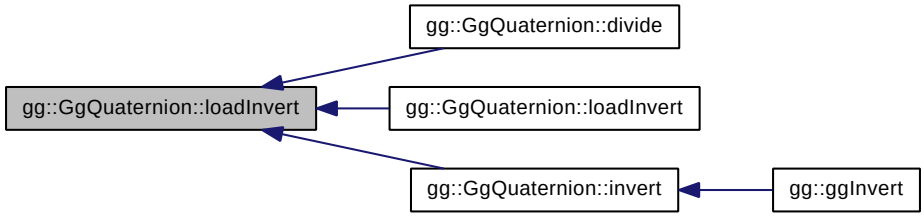
戻り値

四元数の逆元.

gg.cpp の 4810 行目に定義があります。
呼び出し関係図:



被呼び出し関係図:



6.8.3.39 GgQuaternion& gg::GgQuaternion::loadInvert (const GgQuaternion & *q*) [inline]

引数に指定した四元数の逆元を格納する.

引数

q	GgQuaternion 型の四元数.

戻り値

四元数の逆元.

gg.h の 3369 行目に定義があります。
呼び出し関係図:

6.8.3.40 GgQuaternion& gg::GgQuaternion::loadMatrix (const GLfloat * *a*) [inline]

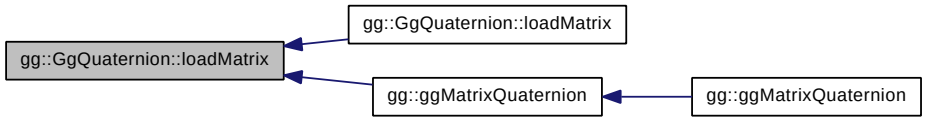
回転の変換行列を表す四元数を格納する.
引数

a	GLfloat 型の 16 要素の変換行列.

戻り値

a による回転の変換に相当する四元数.

gg.h の 3146 行目に定義があります。
被呼び出し関係図:



6.8.3.41 GgQuaternion& gg::GgQuaternion::loadMatrix (const GgMatrix & *m*) [inline]

回転の変換行列 m を表す四元数を格納する.
引数

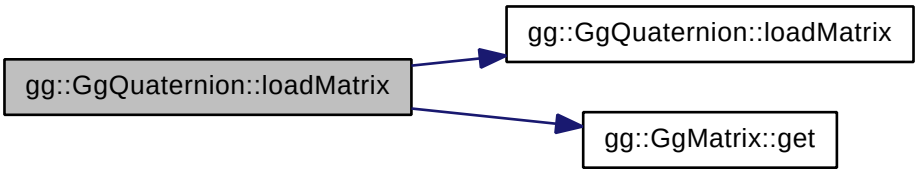
m	Ggmatrix 型の変換行列.

戻り値

m による回転の変換に相当する四元数.

gg.h の 3155 行目に定義があります。

呼び出し関係図:



6.8.3.42 GgQuaternion& gg::GgQuaternion::loadMultiply (GLfloat *x*, GLfloat *y*, GLfloat *z*, GLfloat *w*) [inline]

四元数に別の四元数を乗算した結果を格納する.

引数

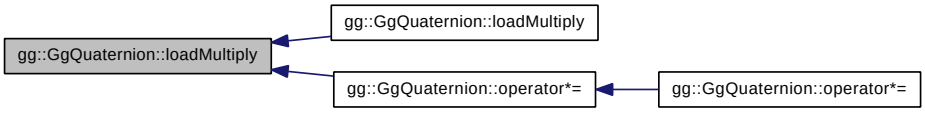
x	掛ける四元数の x 要素.
y	掛ける四元数の y 要素.
z	掛ける四元数の z 要素.
w	掛ける四元数の w 要素.

戻り値

(x, y, z, w) を掛けた四元数.

gg.h の 2846 行目に定義があります。

被呼び出し関係図:



6.8.3.43 GgQuaternion& gg::GgQuaternion::loadMultiply (const GgVector & *v*) [inline]

四元数に別の四元数を乗算した結果を格納する.

引数

v	四元数を格納した GgVector 型の変数.

戻り値

v を乗じた四元数.

gg.h の 2855 行目に定義があります。

呼び出し関係図:

6.8.3.44 GgQuaternion& gg::GgQuaternion::loadMultiply (const GLfloat * *a*) [inline]

四元数に別の四元数を乗算した結果を格納する.

引数

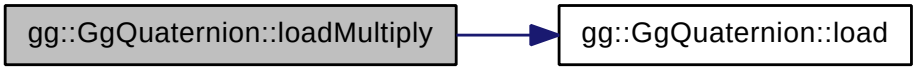
a	四元数を格納した GLfloat 型の 4 要素の配列変数.

戻り値

a を乗じた四元数.

gg.h の 2863 行目に定義があります。

呼び出し関係図:



6.8.3.45 GgQuaternion& gg::GgQuaternion::loadMultiply (const GgQuaternion & *q*) [inline]

四元数に別の四元数を乗算した結果を格納する.

引数

q	GgQuaternion 型の四元数.

戻り値

q を乗じた四元数.

gg.h の 2871 行目に定義があります。

呼び出し関係図:

6.8.3.46 gg::GgQuaternion & gg::GgQuaternion::loadNormalize (const GLfloat * *a*)

引数に指定した四元数を正規化して格納する.

引数

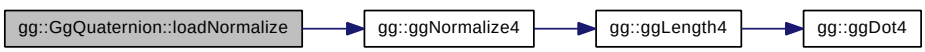
a	四元数を格納した GLfloat 型の 4 要素の配列変数.

戻り値

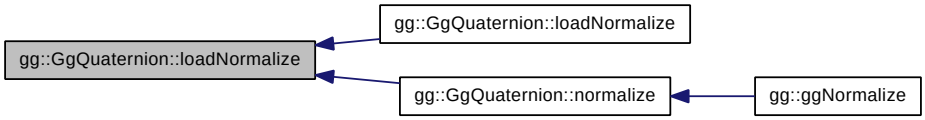
正規化された四元数.

gg.cpp の 4781 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:



6.8.3.47 GgQuaternion& gg::GgQuaternion::loadNormalize (const GgQuaternion & *q*) [inline]

引数に指定した四元数を正規化して格納する.

引数

q	GgQuaternion 型の四元数.

戻り値

正規化された四元数.

gg.h の 3343 行目に定義があります。

呼び出し関係図:

6.8.3.48 gg::GgQuaternion & gg::GgQuaternion::loadRotate (GLfloat *x*, GLfloat *y*, GLfloat *z*, GLfloat *a*)

(x, y, z) を軸として角度 a 回転する四元数を格納する.

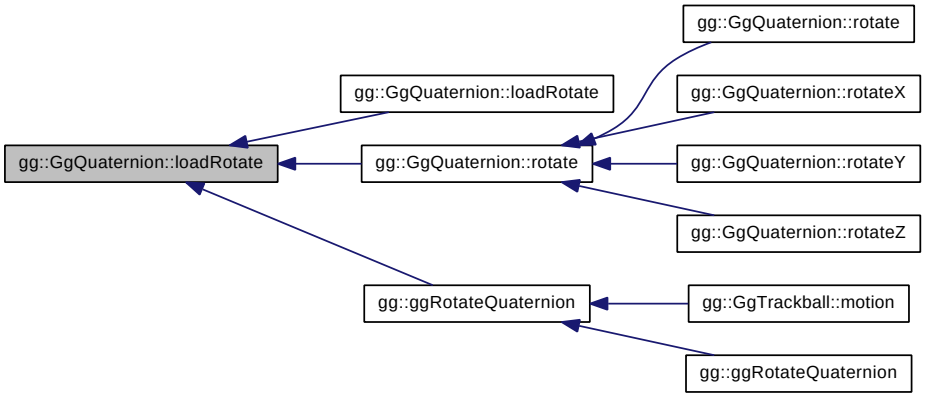
引数

x	軸ベクトルの x 成分.
y	軸ベクトルの y 成分.
z	軸ベクトルの z 成分.
a	回転角.

戻り値

格納された回転を表す四元数.

gg.cpp の 4699 行目に定義があります。
被呼び出し関係図:



6.8.3.49 GgQuaternion& gg::GgQuaternion::loadRotate (const GLfloat * v, GLfloat a) [inline]

(v[0], v[1], v[2]) を軸として角度 a 回転する四元数を格納する.
引数

v	軸ベクトルを表す GLfloat 型の 3 要素の配列変数.
a	回転角.

戻り値

格納された回転を表す四元数.

gg.h の 3179 行目に定義があります。
呼び出し関係図:

6.8.3.50 GgQuaternion& gg::GgQuaternion::loadRotate (const GLfloat * v) [inline]

(v[0], v[1], v[2]) を軸として角度 v[3] 回転する四元数を格納する.
引数

v	軸ベクトルと回転角を格納した GLfloat 型の 4 要素の配列変数.

戻り値

格納された回転を表す四元数.

gg.h の 3187 行目に定義があります。
呼び出し関係図:

6.8.3.51 gg::GgQuaternion & gg::GgQuaternion::loadRotateX (GLfloat *a*)

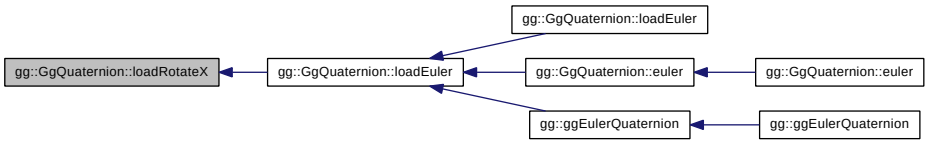
x 軸中心に角度 a 回転する四元数を格納する.
引数

a	回転角.

戻り値

格納された回転を表す四元数.

gg.cpp の 4723 行目に定義があります。
被呼び出し関係図:



6.8.3.52 gg::GgQuaternion & gg::GgQuaternion::loadRotateY (GLfloat *a*)

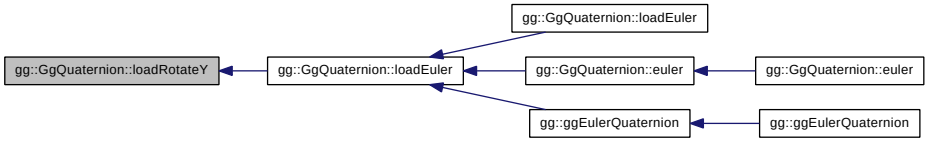
y 軸中心に角度 a 回転する四元数を格納する.
引数

a	回転角.

戻り値

格納された回転を表す四元数.

gg.cpp の 4737 行目に定義があります。
被呼び出し関係図:



6.8.3.53 gg::GgQuaternion & gg::GgQuaternion::loadRotateZ (GLfloat *a*)

z 軸中心に角度 a 回転する四元数を格納する.

引数

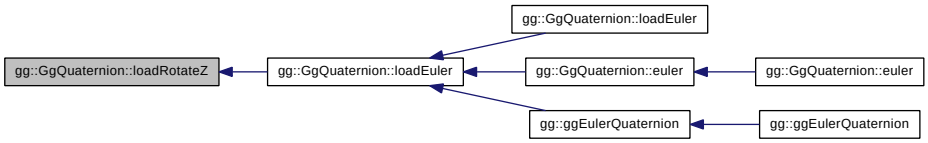
v	軸ベクトルを表す GLfloat 型の 4 要素の配列変数.

戻り値

格納された回転を表す四元数.

gg.cpp の 4751 行目に定義があります。

被呼び出し関係図:



6.8.3.54 GgQuaternion& gg::GgQuaternion::loadSlerp (const GLfloat * *a*, const GLfloat * *b*, GLfloat *t*) [inline]

球面線形補間の結果を格納する.

引数

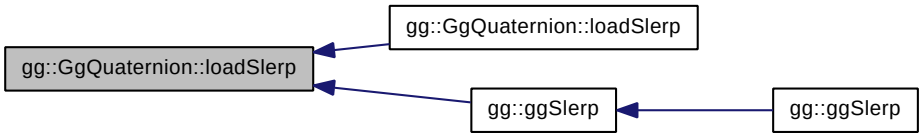
a	四元数を格納した GLfloat 型の 4 要素の配列変数.
b	四元数を格納した GLfloat 型の 4 要素の配列変数.
t	補間パラメータ.

戻り値

格納した a, b を t で内分した四元数.

gg.h の 3299 行目に定義があります。

被呼び出し関係図:



6.8.3.55 GgQuaternion& gg::GgQuaternion::loadSlerp (const GgQuaternion & *q*, const GgQuaternion & *r*, GLfloat *t*) [inline]

球面線形補間の結果を格納する.

引数

q	GgQuaternion 型の四元数.
r	GgQuaternion 型の四元数.

t	補間パラメータ.

戻り値

格納した q, r を t で内分した四元数.

gg.h の 3310 行目に定義があります。

呼び出し関係図:

6.8.3.56 GgQuaternion& gg::GgQuaternion::loadSlerp (const GgQuaternion & *q*, const GLfloat * *a*, GLfloat *t*) [inline]

球面線形補間の結果を格納する.

引数

q	GgQuaternion 型の四元数.
a	四元数を格納した GLfloat 型の 4 要素の配列変数.
t	補間パラメータ.

戻り値

格納した q, a を t で内分した四元数.

gg.h の 3320 行目に定義があります。

呼び出し関係図:

6.8.3.57 GgQuaternion& gg::GgQuaternion::loadSlerp (const GLfloat * *a*, const GgQuaternion & *q*, GLfloat *t*) [inline]

球面線形補間の結果を格納する.

引数

a	四元数を格納した GLfloat 型の 4 要素の配列変数.
q	GgQuaternion 型の四元数.
t	補間パラメータ.

戻り値

格納した a, q を t で内分した四元数.

gg.h の 3330 行目に定義があります。

呼び出し関係図:

6.8.3.58 GgQuaternion& gg::GgQuaternion::loadSubtract (GLfloat *x*, GLfloat *y*, GLfloat *z*, GLfloat *w*) [inline]

四元数から別の四元数を減算した結果を格納する.

引数

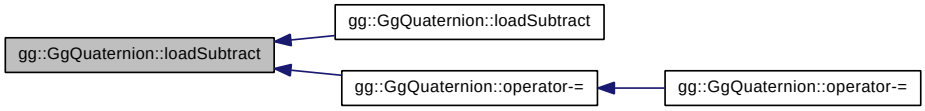
x	引く四元数の x 要素.
y	引く四元数の y 要素.
z	引く四元数の z 要素.
w	引く四元数の w 要素.

戻り値

(x, y, z, w) を引いた四元数.

gg.h の 2807 行目に定義があります。

被呼び出し関係図:



6.8.3.59 GgQuaternion& gg::GgQuaternion::loadSubtract (const GgVector & *v*) [inline]

四元数から別の四元数を減算した結果を格納する.

引数

v	四元数を格納した GgVector 型の変数.

戻り値

v を引いた四元数.

gg.h の 2819 行目に定義があります。

呼び出し関係図:

6.8.3.60 GgQuaternion& gg::GgQuaternion::loadSubtract (const GLfloat * *a*) [inline]

四元数から別の四元数を減算した結果を格納する.

引数

a	四元数を格納した GLfloat 型の 4 要素の配列変数.

戻り値

a を引いた四元数.

gg.h の 2827 行目に定義があります。

呼び出し関係図:

6.8.3.61 GgQuaternion& gg::GgQuaternion::loadSubtract (const GgQuaternion & *q*) [inline]

四元数から別の四元数を減算した結果を格納する.

引数

q	GgQuaternion 型の四元数.

戻り値

q を引いた四元数.

gg.h の 2835 行目に定義があります。

呼び出し関係図:

6.8.3.62 GgQuaternion gg::GgQuaternion::multiply (GLfloat *x*, GLfloat *y*, GLfloat *z*, GLfloat *w*) const [inline]

四元数に別の四元数を乗算した結果を返す.

引数

x	掛ける四元数の x 要素.
y	掛ける四元数の y 要素.
z	掛ける四元数の z 要素.
w	掛ける四元数の w 要素.

戻り値

(x, y, z, w) を掛けた四元数.

gg.h の 2998 行目に定義があります。

6.8.3.63 GgQuaternion gg::GgQuaternion::multiply (const GgVector & *v*) const [inline]

四元数に別の四元数を乗算した結果を返す.

引数

v	四元数を格納した GgVector 型の変数.

戻り値

v を掛けた四元数.

gg.h の 3007 行目に定義があります。

6.8.3.64 GgQuaternion gg::GgQuaternion::multiply (const GLfloat * *a*) const [inline]

四元数に別の四元数を乗算した結果を返す.

引数

a	四元数を格納した GLfloat 型の 4 要素の配列変数.

戻り値

a を掛けた四元数.

gg.h の 3015 行目に定義があります。

6.8.3.65 GgQuaternion gg::GgQuaternion::multiply (const GgQuaternion & *q*) const [inline]

四元数に別の四元数を乗算した結果を返す.

引数

q	GgQuaternion 型の四元数.

戻り値

q を掛けた四元数.

gg.h の 3025 行目に定義があります。

6.8.3.66 GLfloat gg::GgQuaternion::norm ( ) const [inline]

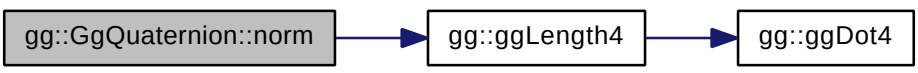
四元数のノルムを求める.

戻り値

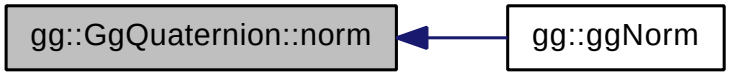
四元数のノルム.

gg.h の 2717 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:



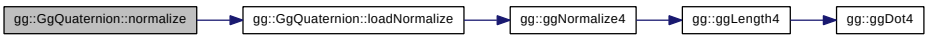
6.8.3.67 GgQuaternion gg::GgQuaternion::normalize ( ) const [inline]

正規化する.

戻り値

正規化された四元数.

gg.h の 3398 行目に定義があります。
呼び出し関係図:



被呼び出し関係図:

6.8.3.68 GgQuaternion gg::GgQuaternion::operator∗ (const GLfloat ∗ *a*) const [inline]

gg.h の 3126 行目に定義があります。
被呼び出し関係図:

6.8.3.69 GgQuaternion gg::GgQuaternion::operator∗ (const GgQuaternion & *q*) const [inline]

gg.h の 3130 行目に定義があります。
呼び出し関係図:

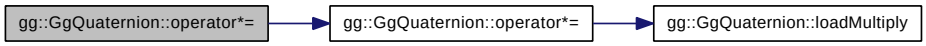
6.8.3.70 GgQuaternion& gg::GgQuaternion::operator*= (const GLfloat * a) [inline]

gg.h の 3094 行目に定義があります。
呼び出し関係図:

被呼び出し関係図:

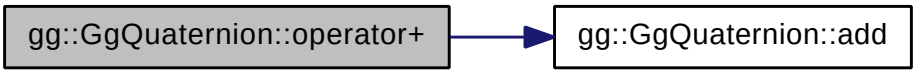
6.8.3.71 GgQuaternion& gg::GgQuaternion::operator*= (const GgQuaternion & q) [inline]

gg.h の 3098 行目に定義があります。
呼び出し関係図:



6.8.3.72 GgQuaternion gg::GgQuaternion::operator+ (const GLfloat * a) const [inline]

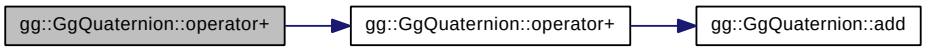
gg.h の 3110 行目に定義があります。
呼び出し関係図:



被呼び出し関係図:

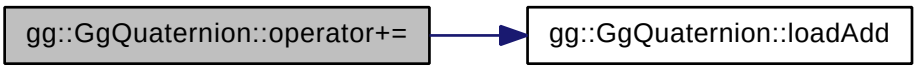
6.8.3.73 GgQuaternion gg::GgQuaternion::operator+ (const GgQuaternion & q) const [inline]

gg.h の 3114 行目に定義があります。
呼び出し関係図:



6.8.3.74 GgQuaternion& gg::GgQuaternion::operator+= (const GLfloat * a) [inline]

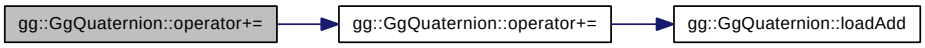
gg.h の 3078 行目に定義があります。
呼び出し関係図:



被呼び出し関係図:

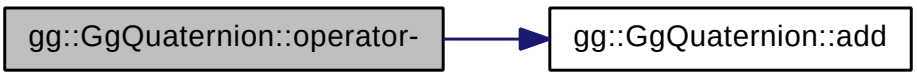
6.8.3.75 GgQuaternion& gg::GgQuaternion::operator+= (const GgQuaternion & q) [inline]

gg.h の 3082 行目に定義があります。
呼び出し関係図:



6.8.3.76 GgQuaternion gg::GgQuaternion::operator- (const GLfloat * a) const [inline]

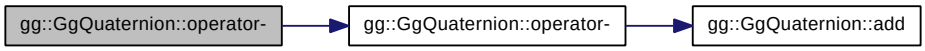
gg.h の 3118 行目に定義があります。
呼び出し関係図:



被呼び出し関係図:

6.8.3.77 GgQuaternion gg::GgQuaternion::operator- (const GgQuaternion & q) const [inline]

gg.h の 3122 行目に定義があります。
呼び出し関係図:



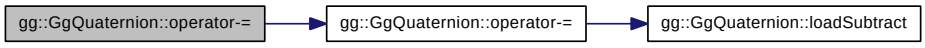
6.8.3.78 GgQuaternion& gg::GgQuaternion::operator-= (const GLfloat * a) [inline]

gg.h の 3086 行目に定義があります。
呼び出し関係図:

被呼び出し関係図:

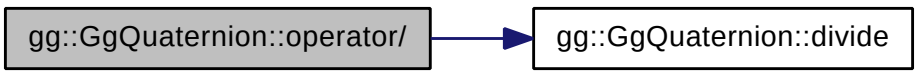
6.8.3.79 GgQuaternion& gg::GgQuaternion::operator-= (const GgQuaternion & q) [inline]

gg.h の 3090 行目に定義があります。
呼び出し関係図:



6.8.3.80 GgQuaternion gg::GgQuaternion::operator/ (const GLfloat * a) const [inline]

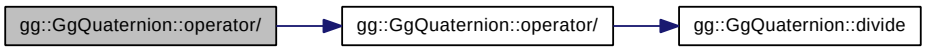
gg.h の 3134 行目に定義があります。
呼び出し関係図:



被呼び出し関係図:

6.8.3.81 GgQuaternion gg::GgQuaternion::operator/ (const GgQuaternion & q) const [inline]

gg.h の 3138 行目に定義があります。
呼び出し関係図:



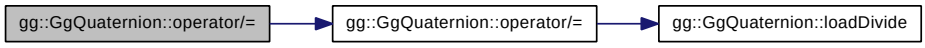
6.8.3.82 GgQuaternion& gg::GgQuaternion::operator/= (const GLfloat * a) [inline]

gg.h の 3102 行目に定義があります。
呼び出し関係図:

被呼び出し関係図:

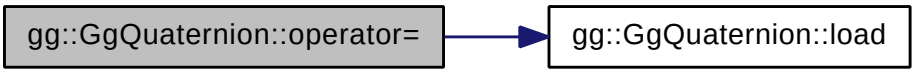
6.8.3.83 GgQuaternion& gg::GgQuaternion::operator/= (const GgQuaternion & q) [inline]

gg.h の 3106 行目に定義があります。
呼び出し関係図:



6.8.3.84 GgQuaternion& gg::GgQuaternion::operator= (const GLfloat * a) [inline]

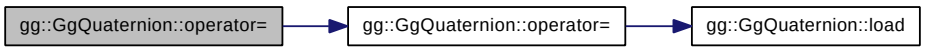
gg.h の 3070 行目に定義があります。
呼び出し関係図:



被呼び出し関係図:

6.8.3.85 GgQuaternion& gg::GgQuaternion::operator= (const GgQuaternion & q) [inline]

gg.h の 3074 行目に定義があります。
呼び出し関係図:



6.8.3.86 GgQuaternion gg::GgQuaternion::rotate (GLfloat *x*, GLfloat *y*, GLfloat *z*, GLfloat *a*) const [inline]

四元数を (x, y, z) を軸として角度 a 回転した四元数を返す.

引数

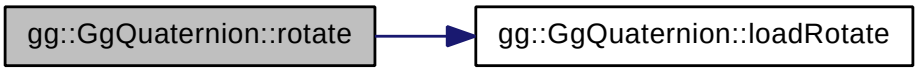
x	軸ベクトルの x 成分.
y	軸ベクトルの y 成分.
z	軸ベクトルの z 成分.
a	回転角.

戻り値

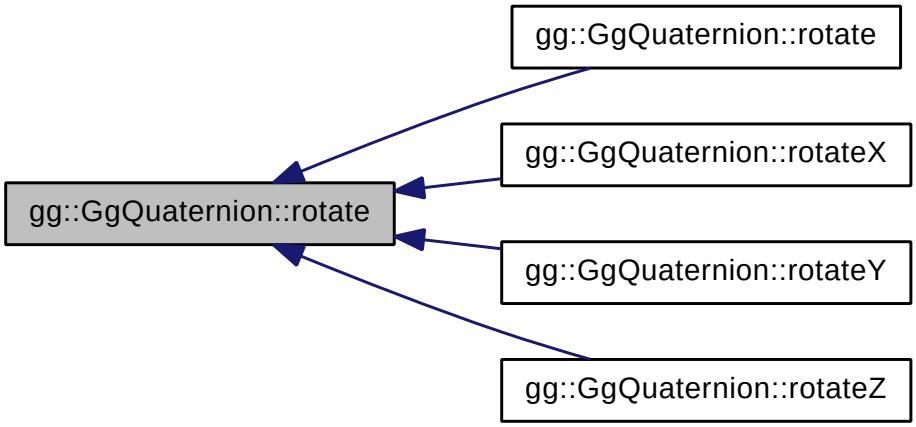
回転した四元数.

gg.h の 3213 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:



6.8.3.87 GgQuaternion gg::GgQuaternion::rotate (const GLfloat * *v*, GLfloat *a*) const [inline]

四元数を (v[0], v[1], v[2]) を軸として角度 a 回転した四元数を返す.

引数

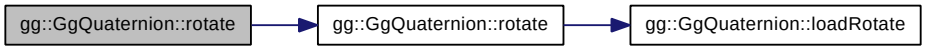
v	軸ベクトルを表す GLfloat 型の 3 要素の配列変数.
a	回転角.

戻り値

回転した四元数.

gg.h の 3223 行目に定義があります。

呼び出し関係図:



6.8.3.88 GgQuaternion gg::GgQuaternion::rotate (const GLfloat * v) const [inline]

四元数を (v[0], v[1], v[2]) を軸として角度 v[3] 回転した四元数を返す.

引数

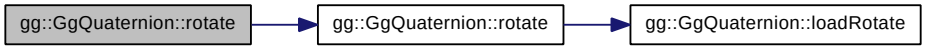
v	軸ベクトルを表す GLfloat 型の 4 要素の配列変数.

戻り値

回転した四元数.

gg.h の 3231 行目に定義があります。

呼び出し関係図:



6.8.3.89 GgQuaternion gg::GgQuaternion::rotateX (GLfloat *a*) const [inline]

四元数を x 軸中心に角度 a 回転した四元数を返す.

引数

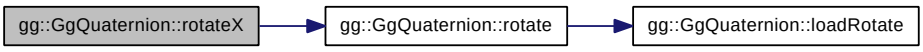
a	回転角.

戻り値

回転した四元数.

gg.h の 3239 行目に定義があります。

呼び出し関係図:



6.8.3.90 GgQuaternion gg::GgQuaternion::rotateY (GLfloat *a*) const [inline]

四元数を y 軸中心に角度 a 回転した四元数を返す.

引数

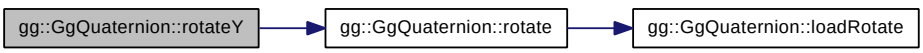
a	回転角.

戻り値

回転した四元数.

gg.h の 3247 行目に定義があります。

呼び出し関係図:



6.8.3.91 GgQuaternion gg::GgQuaternion::rotateZ (GLfloat *a*) const [inline]

四元数を z 軸中心に角度 a 回転した四元数を返す.

引数

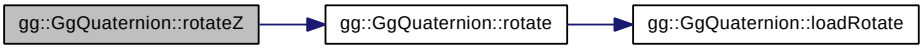
a	回転角.

戻り値

回転した四元数.

gg.h の 3255 行目に定義があります。

呼び出し関係図:



6.8.3.92 GgQuaternion gg::GgQuaternion::slerp (GLfloat * *a*, GLfloat *t*) const [inline]

球面線形補間の結果を返す.

引数

a	四元数を格納した GLfloat 型の 4 要素の配列変数.
t	補間パラメータ.

戻り値

四元数を a に対して t で内分した結果.

gg.h の 3378 行目に定義があります。

6.8.3.93 GgQuaternion gg::GgQuaternion::slerp (const GgQuaternion & *q*, GLfloat *t*) const [inline]

球面線形補間の結果を返す.

引数

q	GgQuaternion 型の四元数.
t	補間パラメータ.

戻り値

四元数を q に対して t で内分した結果.

gg.h の 3389 行目に定義があります。

6.8.3.94 GgQuaternion gg::GgQuaternion::subtract (GLfloat *x*, GLfloat *y*, GLfloat *z*, GLfloat *w*) const [inline]

四元数から別の四元数を減算した結果を返す.

引数

x	引く四元数の x 要素.
y	引く四元数の y 要素.
z	引く四元数の z 要素.
w	引く四元数の w 要素.

戻り値

(x, y, z, w) を引いた四元数.

gg.h の 2958 行目に定義があります。

被呼び出し関係図:

6.8.3.95 GgQuaternion gg::GgQuaternion::subtract (const GgVector & *v*) const [inline]

四元数から別の四元数を減算した結果を返す.

引数

v	四元数を格納した GgVector 型の変数.

戻り値

v を引いた四元数.

gg.h の 2971 行目に定義があります。

呼び出し関係図:

6.8.3.96 GgQuaternion gg::GgQuaternion::subtract (const GLfloat * *a*) const [inline]

四元数から別の四元数を減算した結果を返す.

引数

a	四元数を格納した GLfloat 型の 4 要素の配列変数.

戻り値

a を引いた四元数.

gg.h の 2979 行目に定義があります。

呼び出し関係図:

6.8.3.97 GgQuaternion gg::GgQuaternion::subtract (const GgQuaternion & *q*) const [inline]

四元数から別の四元数を減算した結果を返す.

引数

q	GgQuaternion 型の四元数.

戻り値

q を引いた四元数.

gg.h の 2987 行目に定義があります。

呼び出し関係図:

このクラス詳解は次のファイルから抽出されました:

- gg.h
- gg.cpp

6.9 gg::GgShader クラス

シェーダの基底クラス.

`#include <gg.h>`

公開メンバ関数

- virtual ~GgShader ()
  デストラクタ.
- GgShader (const char *vert, const char *frag=0, const char *geom=0, int nvarying=0, const char **varyings=0)
  コンストラクタ.
- void use () const
  シェーダプログラムの使用を開始する.
- void unuse () const
  シェーダプログラムの使用を終了する.
- GLuint get () const
  シェーダのプログラム名を得る.

6.9.1 詳解

シェーダの基底クラス.

シェーダのクラスはこのクラスを派生して作る.

gg.h の 4899 行目に定義があります。

6.9.2 構築子と解体子

6.9.2.1 virtual gg::GgShader::~GgShader ( ) [inline], [virtual]

デストラクタ.

gg.h の 4913 行目に定義があります。

6.9.2.2 gg::GgShader::GgShader (const char * *vert*, const char * *frag* = 0, const char * *geom* = 0, int *nvarying* = 0, const char ** *varyings* = 0) [inline]

コンストラクタ.

引数

vert	バーテックスシェーダのソースファイル名.
frag	フラグメントシェーダのソースファイル名 (0 なら不使用).
geom	ジオメトリシェーダのソースファイル名 (0 なら不使用).
nvarying	フィードバックする varying 変数の数 (0 なら不使用).
varyings	フィードバックする varying 変数のリスト.

gg.h の 4926 行目に定義があります。

6.9.3 関数詳解

6.9.3.1 GLuint gg::GgShader::get ( ) const [inline]

シェーダのプログラム名を得る.

戻り値

シェーダのプログラム名.

gg.h の 4945 行目に定義があります。

6.9.3.2 void gg::GgShader::unuse ( ) const [inline]

シェーダプログラムの使用を終了する.

gg.h の 4938 行目に定義があります。

6.9.3.3 void gg::GgShader::use ( ) const [inline]

シェーダプログラムの使用を開始する.

gg.h の 4932 行目に定義があります。

このクラス詳解は次のファイルから抽出されました:

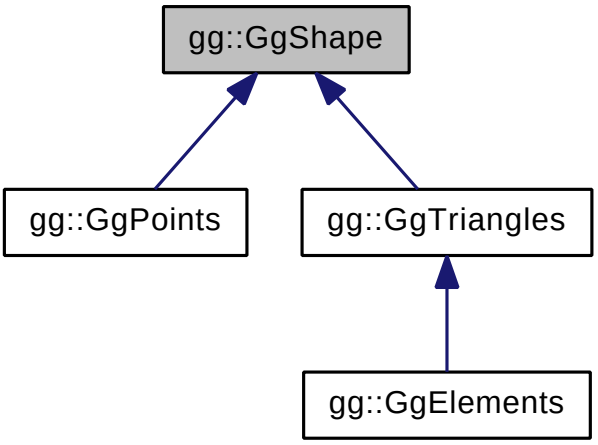
- gg.h

6.10 gg::GgShape クラス

形状データの基底クラス.

`#include <gg.h>`

gg::GgShape の継承関係図



公開メンバ関数

- virtual ~GgShape ()
  デストラクタ.
- GgShape (GLenum mode=0)
  コンストラクタ.
- GgShape (const GgShape &o)
  コピーコンストラクタ.
- GgShape & operator= (const GgShape &o)
- GLuint get () const
  頂点配列オブジェクト名を取り出す.
- void setMode (GLenum mode)
  基本図形の設定.
- GLenum getMode () const
  基本図形の検査.
- virtual void draw (GLint first=0, GLsizei count=0) const
  図形の描画, **派生クラスでこの手続きをオーバーライドする**.

6.10.1 詳解

形状データの基底クラス.

形状データのクラスはこのクラスを派生して作る. 基本図形の種類と頂点配列オブジェクトを保持する.

gg.h の 4455 行目に定義があります。

6.10.2 構築子と解体子

6.10.2.1 virtual gg::GgShape::~GgShape ( ) [inline], [virtual]

デストラクタ.

gg.h の 4466 行目に定義があります。

6.10.2.2 gg::GgShape::GgShape (GLenum *mode* = 0) [inline]

コンストラクタ.

引数

mode	基本図形の種類.

gg.h の 4474 行目に定義があります。

6.10.2.3 gg::GgShape::GgShape (const GgShape & *o*) [inline]

コピーコンストラクタ.

gg.h の 4483 行目に定義があります。

6.10.3 関数詳解

6.10.3.1 virtual void gg::GgShape::draw (GLint *first* = 0, GLsizei *count* = 0) const [inline], [virtual]

図形の描画, 派生クラスでこの手続きをオーバーライドする.

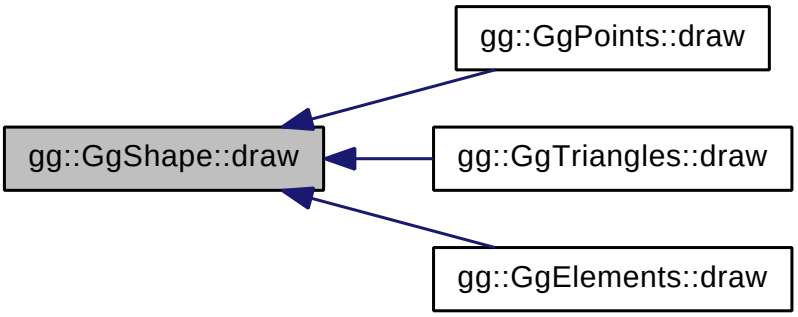
引数

first	描画する最初のアイテム.
count	描画するアイテムの数, 0 なら全部のアイテムを描画する.

gg::GgElements, gg::GgTriangles, gg::GgPointsで再実装されています。

gg.h の 4524 行目に定義があります。

被呼び出し関係図:



6.10.3.2 GLuint gg::GgShape::get ( ) const [inline]

頂点配列オブジェクト名を取り出す.

戻り値

頂点配列オブジェクト名.

gg.h の 4502 行目に定義があります。

6.10.3.3 GLenum gg::GgShape::getMode ( ) const [inline]

基本図形の検査.

戻り値

この頂点配列オブジェクトの基本図形の種類.

gg.h の 4516 行目に定義があります。

6.10.3.4 GgShape& gg::GgShape::operator= (const GgShape & *o*) [inline]

gg.h の 4490 行目に定義があります。

6.10.3.5 void gg::GgShape::setMode (GLenum *mode*) [inline]

基本図形の設定.

引数

mode	基本図形の種類.

gg.h の 4509 行目に定義があります。

このクラス詳解は次のファイルから抽出されました:

- gg.h

6.11 gg::GgSimpleObj クラス

Wavefront OBJ 形式のファイル (Arrays 形式).

`#include <gg.h>`

公開メンバ関数

- virtual ~GgSimpleObj ()
  デストラクタ.
- GgSimpleObj (const char *name, const GgSimpleShader *shader=nullptr, bool normalize=false)
  コンストラクタ.
- GgSimpleObj (const char *name, const GgSimpleShader &shader, bool normalize=false)
  コンストラクタ.
- const GgTriangles * get () const
  形状データの取り出し.
- void selectShader (const GgSimpleShader *shader)
  Wavefront OBJ **形式のデータを描画する際に用いるシェーダを指定する**.
- void selectShader (const GgSimpleShader &shader)
  Wavefront OBJ **形式のデータを描画する際に用いるシェーダを指定する**.
- const GgSimpleShader * getShader () const
  Wavefront **この** *OBJ* **形式のデータを描画する際に用いるシェーダを取り出す**.
- virtual void draw (GLint first=0, GLsizei count=0) const
  Wavefront OBJ **形式のデータを描画する手続き**.

6.11.1 詳解

Wavefront OBJ 形式のファイル (Arrays 形式).

gg.h の 5536 行目に定義があります。

6.11.2 構築子と解体子

6.11.2.1 virtual gg::GgSimpleObj::~GgSimpleObj ( ) [inline], [virtual]

デストラクタ.

gg.h の 5553 行目に定義があります。

6.11.2.2 gg::GgSimpleObj::GgSimpleObj (const char * *name*, const GgSimpleShader * *shader* = nullptr, bool *normalize* = false)

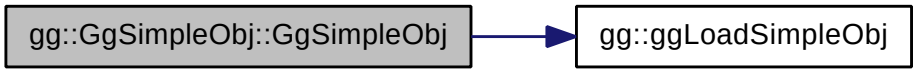
コンストラクタ.

引数

name	三角形分割された Alias OBJ 形式のファイルのファイル名.
shader	この図形の描画に用いる GgSimpleShader 型のシェーダオブジェクトのポインタ.
normalize	true なら図形のサイズを [-1, 1] に正規化する.

gg.cpp の 5636 行目に定義があります。

呼び出し関係図:



6.11.2.3 gg::GgSimpleObj::GgSimpleObj (const char * *name*, const GgSimpleShader & *shader*, bool *normalize* = false) [inline]

コンストラクタ.

引数

name	三角形分割された Alias OBJ 形式のファイルのファイル名.
shader	この図形の描画に用いる GgSimpleShader 型のシェーダオブジェクト.
normalize	true なら図形のサイズを [-1, 1] に正規化する.

gg.h の 5569 行目に定義があります。

6.11.3 関数詳解

6.11.3.1 void gg::GgSimpleObj::draw (GLint *first* = 0, GLsizei *count* = 0) const [virtual]

Wavefront OBJ 形式のデータを描画する手続き.

引数

first	描画する最初のパーツ番号.
count	描画するパーツの数, 0 なら全部のパーツを描く.

gg.cpp の 5663 行目に定義があります。

6.11.3.2 const GgTriangles* gg::GgSimpleObj::get ( ) const [inline]

形状データの取り出し.

戻り値

GgTriangles 型の形状データのポインタ.

gg.h の 5574 行目に定義があります。

6.11.3.3 const GgSimpleShader* gg::GgSimpleObj::getShader ( ) const [inline]

Wavefront この OBJ 形式のデータを描画する際に用いるシェーダを取り出す.

戻り値

この図形の描画に用いる GgSimpleShader 型のシェーダオブジェクトのポインタ.

gg.h の 5595 行目に定義があります。

6.11.3.4 void gg::GgSimpleObj::selectShader (const GgSimpleShader * shader) [inline]

Wavefront OBJ 形式のデータを描画する際に用いるシェーダを指定する.

引数

shader	この図形の描画に用いる GgSimpleShader 型のシェーダオブジェクトのポインタ.

gg.h の 5581 行目に定義があります。

被呼び出し関係図:

6.11.3.5 void gg::GgSimpleObj::selectShader (const GgSimpleShader & shader) [inline]

Wavefront OBJ 形式のデータを描画する際に用いるシェーダを指定する.

引数

shader	この図形の描画に用いる GgSimpleShader 型のシェーダオブジェクト.

gg.h の 5588 行目に定義があります。

呼び出し関係図:

このクラス詳解は次のファイルから抽出されました:

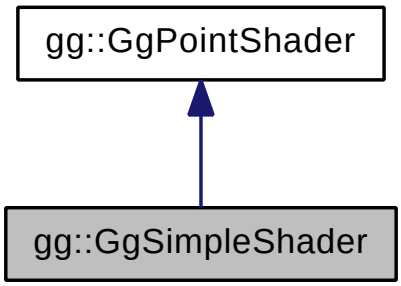
- gg.h
- gg.cpp

6.12 gg::GgSimpleShader クラス

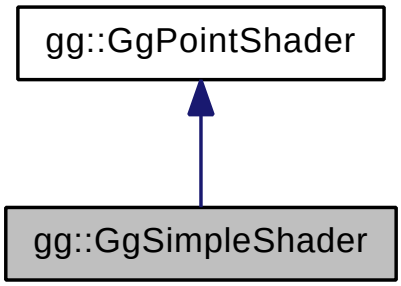
三角形に単純な陰影付けを行うシェーダ.

#include <gg.h>

gg::GgSimpleShader の継承関係図



gg::GgSimpleShader 連携図



クラス

- struct Light
  三角形に単純な陰影付けを行うシェーダが参照する光源データ.
- class LightBuffer
  三角形に単純な陰影付けを行うシェーダが参照する光源データのユニフォームバッファオブジェクト.
- struct Material
  三角形に単純な陰影付けを行うシェーダが参照する材質データ.
- class MaterialBuffer
  三角形に単純な陰影付けを行うシェーダが参照する材質データのユニフォームバッファオブジェクト.

公開メンバ関数

- virtual ~GgSimpleShader ()
  デストラクタ.
- GgSimpleShader ()
  コンストラクタ.
- GgSimpleShader (const char *vert, const char *frag=0, const char *geom=0, GLint nvarying=0, const char **varyings=0)
  コンストラクタ
- GgSimpleShader (const GgSimpleShader &o)
  コピーコンストラクタ.
- GgSimpleShader & operator= (const GgSimpleShader &o)
- virtual void loadMatrix (const GLfloat *mp, const GLfloat *mv, const GLfloat *mn) const
  変換行列を設定する.

- virtual void loadMatrix (const GgMatrix &mp, const GgMatrix &mv, const GgMatrix &mn) const
  変換行列を設定する.
- virtual void loadMatrix (const GLfloat *mp, const GLfloat *mv) const
  変換行列を設定する.
- virtual void loadMatrix (const GgMatrix &mp, const GgMatrix &mv) const
  変換行列を設定する.
- void selectMaterial (const MaterialBuffer *material, GLint i=0) const
  材質を選択する
- void selectMaterial (const MaterialBuffer &material, GLint i=0) const
  材質を選択する
- void selectLight (const LightBuffer *light, GLint i=0) const
  光源を選択する
- void selectLight (const LightBuffer &light, GLint i=0) const
  光源を選択する
- void use () const
  シェーダプログラムの使用を開始する.
- void use (const LightBuffer *light) const
  光源を指定してシェーダプログラムの使用を開始する.
- void use (const LightBuffer &light) const
  光源を指定してシェーダプログラムの使用を開始する.
- void use (const LightBuffer &light, const GLfloat *mp, const GLfloat *mv, const GLfloat *mn) const
  光源と変換行列を指定してシェーダプログラムの使用を開始する.
- void use (const LightBuffer &light, const GgMatrix &mp, const GgMatrix &mv, const GgMatrix &mn) const
  光源と変換行列を指定してシェーダプログラムの使用を開始する.
- void use (const LightBuffer &light, const GgMatrix &mp, const GgMatrix &mv) const
  光源と変換行列を指定してシェーダプログラムの使用を開始する.
- void use (const LightBuffer &light, const GLfloat *mp, const GLfloat *mv) const
  光源と変換行列を指定してシェーダプログラムの使用を開始する.

6.12.1 詳解

三角形に単純な陰影付けを行うシェーダ.

gg.h の 5034 行目に定義があります。

6.12.2 構築子と解体子

6.12.2.1 virtual gg::GgSimpleShader::~GgSimpleShader ( ) [inline], [virtual]

デストラクタ.

gg.h の 5049 行目に定義があります。

6.12.2.2 gg::GgSimpleShader::GgSimpleShader ( ) [inline]

コンストラクタ.

gg.h の 5052 行目に定義があります。

6.12.2.3 gg::GgSimpleShader::GgSimpleShader (const char * *vert*, const char * *frag* = 0, const char * *geom* = 0, GLint *nvarying* = 0, const char ** *varyings* = 0)

コンストラクタ

引数

vert	バーテックスシェーダのソースファイル名.
frag	フラグメントシェーダのソースファイル名 (0 なら不使用).
geom	ジオメトリシェーダのソースファイル名 (0 なら不使用).
nvarying	フィードバックする varying 変数の数 (0 なら不使用).
varyings	フィードバックする varying 変数のリスト.

gg.cpp の 5622 行目に定義があります。

6.12.2.4 gg::GgSimpleShader::GgSimpleShader (const GgSimpleShader & *o*) [inline]

コピーコンストラクタ.

gg.h の 5064 行目に定義があります。

6.12.3 関数詳解

6.12.3.1 virtual void gg::GgSimpleShader::loadMatrix (const GLfloat * *mp*, const GLfloat * *mv*, const GLfloat * *mn*) const [inline], [virtual]

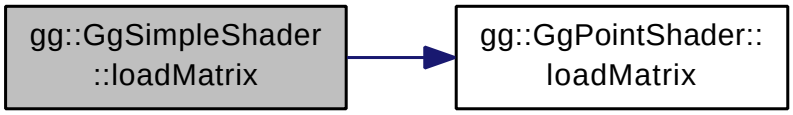
変換行列を設定する.

引数

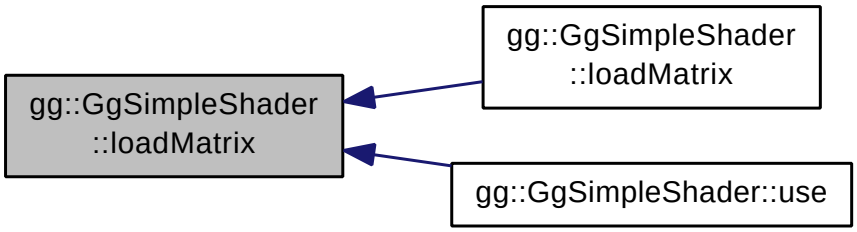
mp	GLfloat 型の 16 要素の配列変数に格納された投影変換行列.
mv	GLfloat 型の 16 要素の配列変数に格納されたモデルビュー変換行列.
mv	GLfloat 型の 16 要素の配列変数に格納されたモデルビュー変換行列の法線変換行列.

gg.h の 5088 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:



6.12.3.2 virtual void gg::GgSimpleShader::loadMatrix (const GgMatrix & *mp*, const GgMatrix & *mv*, const GgMatrix & *mn*) const [inline], [virtual]

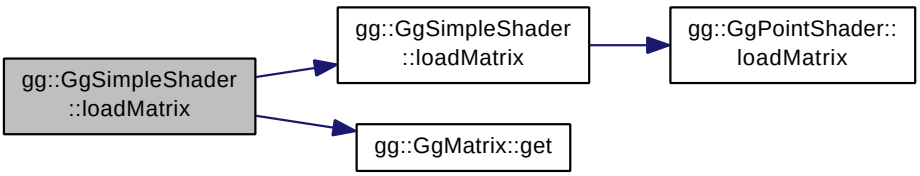
変換行列を設定する.

引数

mp	GgMatrix 型の投影変換行列.
mv	GgMatrix 型のモデルビュー変換行列.
mn	GgMatrix 型のモデルビュー変換行列の法線変換行列.

gg.h の 5098 行目に定義があります。

呼び出し関係図:



6.12.3.3 virtual void gg::GgSimpleShader::loadMatrix (const GLfloat * *mp*, const GLfloat * *mv*) const [inline], [virtual]

変換行列を設定する.

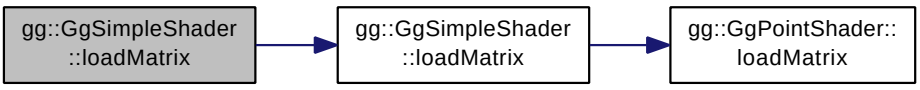
引数

mp	GLfloat 型の 16 要素の配列変数に格納された投影変換行列.
mv	GLfloat 型の 16 要素の配列変数に格納されたモデルビュー変換行列.

gg::GgPointShaderを再実装しています。

gg.h の 5106 行目に定義があります。

呼び出し関係図:



6.12.3.4 virtual void gg::GgSimpleShader::loadMatrix (const GgMatrix & *mp*, const GgMatrix & *mv*) const [inline], [virtual]

変換行列を設定する.

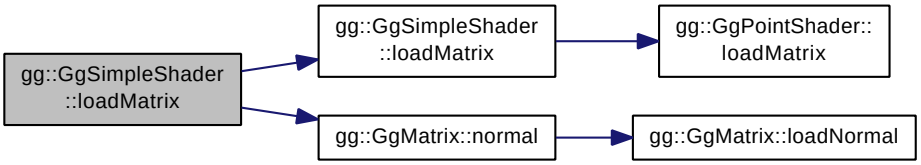
引数

mp	GgMatrix 型の投影変換行列.
mv	GgMatrix 型のモデルビュー変換行列.

gg::GgPointShaderを再実装しています。

gg.h の 5114 行目に定義があります。

呼び出し関係図:



6.12.3.5 GgSimpleShader& gg::GgSimpleShader::operator= (const GgSimpleShader & *o*) [inline]

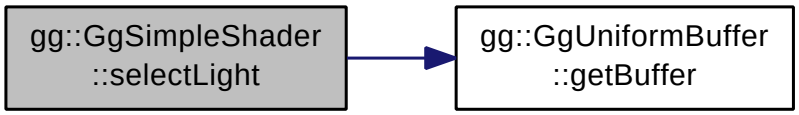
gg.h の 5071 行目に定義があります。

6.12.3.6 void gg::GgSimpleShader::selectLight (const LightBuffer * *light*, GLint *i* = 0) const [inline]

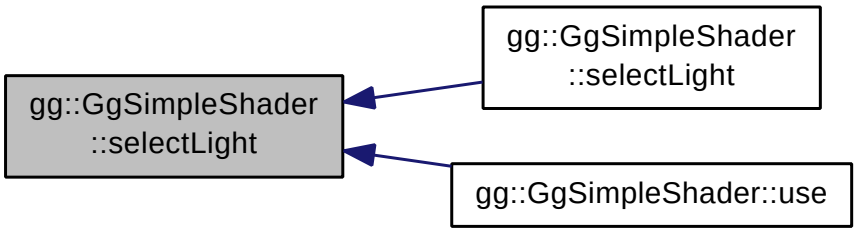
光源を選択する

gg.h の 5418 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:

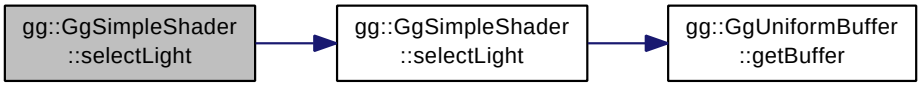


6.12.3.7 void gg::GgSimpleShader::selectLight (const LightBuffer & *light*, GLint *i* = 0) const [inline]

光源を選択する

gg.h の 5426 行目に定義があります。

呼び出し関係図:

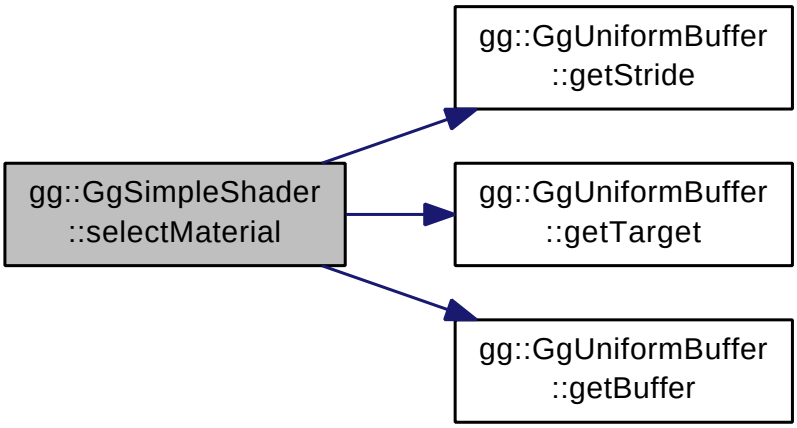


6.12.3.8 void gg::GgSimpleShader::selectMaterial (const MaterialBuffer * material, GLint i = 0) const [inline]

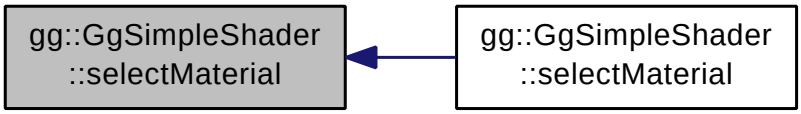
材質を選択する

gg.h の 5404 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:

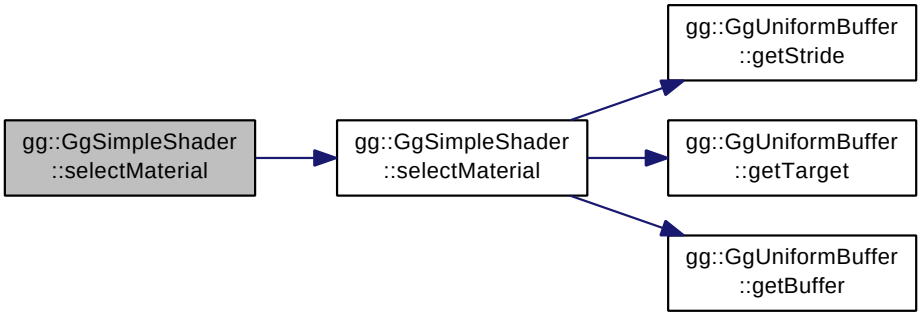


6.12.3.9 void gg::GgSimpleShader::selectMaterial (const MaterialBuffer & material, GLint i = 0) const [inline]

材質を選択する

gg.h の 5412 行目に定義があります。

呼び出し関係図:



6.12.3.10 void gg::GgSimpleShader::use ( ) const [inline]

シェーダプログラムの使用を開始する.

gg.h の 5432 行目に定義があります。

呼び出し関係図:

被呼び出し関係図:

6.12.3.11 void gg::GgSimpleShader::use (const LightBuffer * *light*) const [inline]

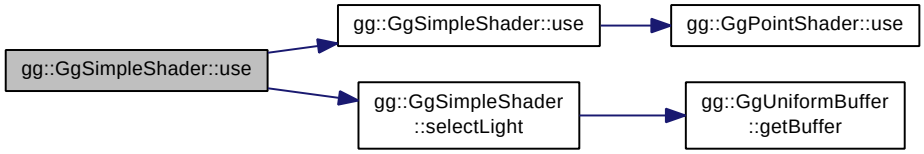
光源を指定してシェーダプログラムの使用を開始する.

引数

light	光源の特性の gg::LightBuffer 構造体のポインタ.

gg.h の 5440 行目に定義があります。

呼び出し関係図:



6.12.3.12 void gg::GgSimpleShader::use (const LightBuffer & *light*) const [inline]

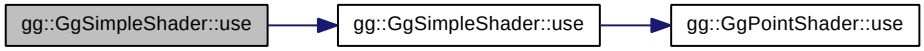
光源を指定してシェーダプログラムの使用を開始する.

引数

light	光源の特性の gg::LightBuffer 構造体.

gg.h の 5451 行目に定義があります。

呼び出し関係図:



6.12.3.13 void gg::GgSimpleShader::use (const LightBuffer & *light*, const GLfloat * *mp*, const GLfloat * *mv*, const GLfloat * *mn*) const [inline]

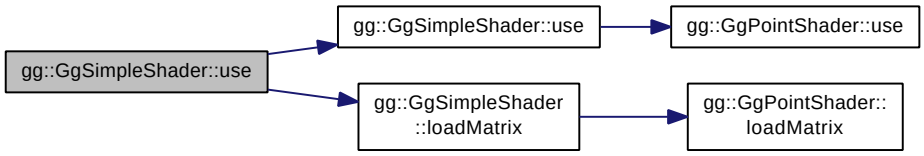
光源と変換行列を指定してシェーダプログラムの使用を開始する.

引数

light	光源の特性の gg::LightBuffer 構造体
mp	GLfloat 型の 16 要素の配列変数に格納された投影変換行列.
mv	GLfloat 型の 16 要素の配列変数に格納されたモデルビュー変換行列.
mv	GLfloat 型の 16 要素の配列変数に格納されたモデルビュー変換行列の法線変換行列.

gg.h の 5461 行目に定義があります。

呼び出し関係図:



6.12.3.14 void gg::GgSimpleShader::use (const LightBuffer & *light*, const GgMatrix & *mp*, const GgMatrix & *mv*, const GgMatrix & *mn*) const [inline]

光源と変換行列を指定してシェーダプログラムの使用を開始する.

引数

light	光源の特性の gg::LightBuffer 構造体
mp	GgMatrix 型の投影変換行列.
mv	GgMatrix 型のモデルビュー変換行列.
mn	GgMatrix 型のモデルビュー変換行列の法線変換行列.

gg.h の 5475 行目に定義があります。

呼び出し関係図:

6.12.3.15 void gg::GgSimpleShader::use (const LightBuffer & *light*, const GgMatrix & *mp*, const GgMatrix & *mv*) const [inline]

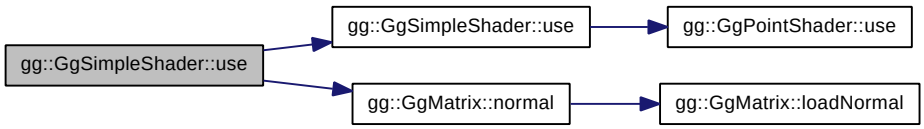
光源と変換行列を指定してシェーダプログラムの使用を開始する.

引数

light	光源の特性の gg::LightBuffer 構造体
mp	GgMatrix 型の投影変換行列.
mv	GgMatrix 型のモデルビュー変換行列.

gg.h の 5484 行目に定義があります。

呼び出し関係図:



6.12.3.16 void gg::GgSimpleShader::use (const LightBuffer & *light*, const GLfloat * *mp*, const GLfloat * *mv*) const [inline]

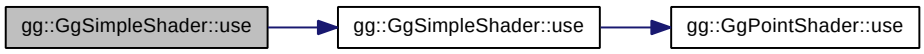
光源と変換行列を指定してシェーダプログラムの使用を開始する.

引数

light	光源の特性の gg::LightBuffer 構造体
mp	GLfloat 型の 16 要素の配列変数に格納された投影変換行列.
mv	GLfloat 型の 16 要素の配列変数に格納されたモデルビュー変換行列.

gg.h の 5493 行目に定義があります。

呼び出し関係図:



このクラス詳解は次のファイルから抽出されました:

- gg.h
- gg.cpp

6.13 gg::GgTexture クラス

テクスチャ.

`#include <gg.h>`

公開メンバ関数

- virtual ~GgTexture ()
  デストラクタ.
- GgTexture (const GLvoid *image, GLsizei width, GLsizei height, GLenum format=GL_BGR, GLenum type=GL_UNSIGNED_BYTE, GLenum internal=GL_RGBA, GLenum wrap=GL_CLAMP_TO_EDGE)
  メモリ上のデータからテクスチャを作成するコンストラクタ.
- void bind () const
  テクスチャの使用開始 *(**このテクスチャを使用する際に呼び出す**).*
- void unbind () const
  テクスチャの使用終了 *(**このテクスチャを使用しなくなったら呼び出す**).*
- GLsizei getWidth () const
  使用しているテクスチャの横の画素数を取り出す.
- GLsizei getHeight () const
  使用しているテクスチャの縦の画素数を取り出す.
- void getSize (GLsizei *size) const

使用しているテクスチャのサイズを取り出す.

- const GLsizei * getSize () const

  使用しているテクスチャのサイズを取り出す.

- GLuint getTexture () const

  使用しているテクスチャのテクスチャ名を得る.

6.13.1 詳解

テクスチャ.

画像データを読み込んでテクスチャマップを作成する.

gg.h の 3811 行目に定義があります。

6.13.2 構築子と解体子

6.13.2.1 virtual gg::GgTexture::~GgTexture ( ) [inline], [virtual]

デストラクタ.

gg.h の 3828 行目に定義があります。

6.13.2.2 gg::GgTexture::GgTexture (const GLvoid * *image*, GLsizei *width*, GLsizei *height*, GLenum *format* = GL_BGR, GLenum *type* = GL_UNSIGNED_BYTE, GLenum *internal* = GL_RGBA, GLenum *wrap* = GL_CLAMP_TO_EDGE) [inline]

メモリ上のデータからテクスチャを作成するコンストラクタ.

引数

image	テクスチャとして用いる画像データ, nullptr ならデータを読み込まない.
width	テクスチャの横の画素数.
height	テクスチャの縦の画素数.
format	読み込む画像のフォーマット.
type	画像のデータ型.
internal	テクスチャの内部フォーマット.
wrap	テクスチャのラッピングモード, デフォルトは GL_CLAMP_TO_EDGE.

gg.h の 3842 行目に定義があります。

6.13.3 関数詳解

6.13.3.1 void gg::GgTexture::bind ( ) const [inline]

テクスチャの使用開始 (このテクスチャを使用する際に呼び出す).

gg.h の 3850 行目に定義があります。

6.13.3.2 GLsizei gg::GgTexture::getHeight ( ) const [inline]

使用しているテクスチャの縦の画素数を取り出す.

戻り値

テクスチャの縦の画素数.

gg.h の 3870 行目に定義があります。

被呼び出し関係図:

6.13.3.3 void gg::GgTexture::getSize (GLsizei * *size*) const [inline]

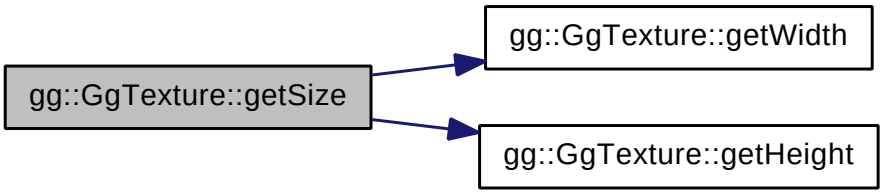
使用しているテクスチャのサイズを取り出す.

引数

テクスチャのサイズを格納する	GLsizei 型の 2 要素の配列変数.

gg.h の 3877 行目に定義があります。

呼び出し関係図:



6.13.3.4 const GLsizei* gg::GgTexture::getSize ( ) const [inline]

使用しているテクスチャのサイズを取り出す.

戻り値

テクスチャのサイズを格納した配列へのポインタ.

gg.h の 3885 行目に定義があります。

6.13.3.5 GLuint gg::GgTexture::getTexture ( ) const [inline]

使用しているテクスチャのテクスチャ名を得る.

戻り値

テクスチャ名.

gg.h の 3892 行目に定義があります。

6.13.3.6 GLsizei gg::GgTexture::getWidth ( ) const [inline]

使用しているテクスチャの横の画素数を取り出す.

戻り値

テクスチャの横の画素数.

gg.h の 3863 行目に定義があります。
被呼び出し関係図:

6.13.3.7 void gg::GgTexture::unbind ( ) const [inline]

テクスチャの使用終了 (このテクスチャを使用しなくなったら呼び出す).
gg.h の 3856 行目に定義があります。
このクラス詳解は次のファイルから抽出されました:

- gg.h

6.14 gg::GgTrackball クラス

簡易トラックボール処理.

```
#include <gg.h>
```

公開メンバ関数

- virtual ~GgTrackball ()
  デストラクタ.
- GgTrackball ()
  コンストラクタ.
- void region (float w, float h)
  トラックボール処理するマウスの移動範囲を指定する.
- void region (int w, int h)
  トラックボール処理するマウスの移動範囲を指定する.
- void begin (float x, float y)
  トラックボール処理を開始する.
- void motion (float x, float y)
  回転の変換行列を計算する.
- void rotate (const GgQuaternion &q)
  トラックボールの回転角を修正する.
- void end (float x, float y)
  トラックボール処理を停止する.
- void reset ()
  トラックボールをリセットする

- const GLfloat * getStart () const
  トラックボール処理の開始位置を取り出す.
- GLfloat getStart (int direction) const
  トラックボール処理の開始位置を取り出す.
- void getStart (GLfloat *position) const
  トラックボール処理の開始位置を取り出す.
- const GLfloat * getScale () const
  トラックボール処理の換算係数を取り出す.
- GLfloat getScale (int direction) const
  トラックボール処理の換算係数を取り出す.
- void getScale (GLfloat *factor) const
  トラックボール処理の換算係数を取り出す.
- const GgQuaternion & getQuaternion () const
  現在の回転の四元数を取り出す.
- const GgMatrix & getMatrix () const
  現在の回転の変換行列を取り出す.
- const GLfloat * get () const
  現在の回転の変換行列を取り出す.

6.14.1 詳解

簡易トラックボール処理.

gg.h の 3680 行目に定義があります。

6.14.2 構築子と解体子

6.14.2.1 virtual gg::GgTrackball::~GgTrackball ( ) [inline], [virtual]

デストラクタ.

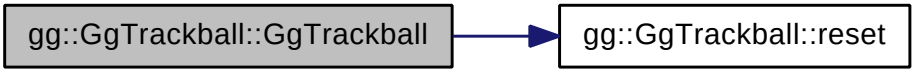
gg.h の 3692 行目に定義があります。

6.14.2.2 gg::GgTrackball::GgTrackball ( ) [inline]

コンストラクタ.

gg.h の 3695 行目に定義があります。

呼び出し関係図:



6.14.3 関数詳解

6.14.3.1 void gg::GgTrackball::begin (float *x*, float *y*)

トラックボール処理を開始する.

マウスのドラッグ開始時 (マウスボタンを押したとき) に呼び出す.

引数

x	現在のマウスの x 座標.
y	現在のマウスの y 座標.

gg.cpp の 4865 行目に定義があります。

6.14.3.2 void gg::GgTrackball::end (float *x*, float *y*)

トラックボール処理を停止する.

マウスのドラッグ終了時 (マウスボタンを離したとき) に呼び出す.

引数

x	現在のマウスの x 座標.
y	現在のマウスの y 座標.

gg.cpp の 4929 行目に定義があります。

6.14.3.3 const GLfloat* gg::GgTrackball::get ( ) const [inline]

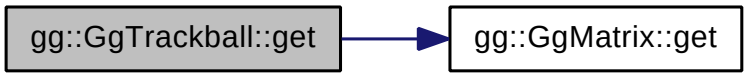
現在の回転の変換行列を取り出す.

戻り値

回転の変換を表す GLfloat 型の 16 要素の配列.

gg.h の 3800 行目に定義があります。

呼び出し関係図:



6.14.3.4 const GgMatrix& gg::GgTrackball::getMatrix ( ) const [inline]

現在の回転の変換行列を取り出す.

戻り値

回転の変換を表す GgMatrix 型の変換行列.

gg.h の 3793 行目に定義があります。

6.14.3.5 const GgQuaternion& gg::GgTrackball::getQuaternion ( ) const [inline]

現在の回転の四元数を取り出す.

戻り値

回転の変換を表す Quaternion 型の四元数.

gg.h の 3786 行目に定義があります。

6.14.3.6 const GLfloat* gg::GgTrackball::getScale ( ) const [inline]

トラックボール処理の換算係数を取り出す.

戻り値

トラックボールの換算係数のポインタ.

gg.h の 3764 行目に定義があります。

6.14.3.7 GLfloat gg::GgTrackball::getScale (int *direction*) const [inline]

トラックボール処理の換算係数を取り出す.

引数

direction	0 なら x 方向, 1 なら y 方向.

gg.h の 3771 行目に定義があります。

6.14.3.8 void gg::GgTrackball::getScale (GLfloat * *factor*) const [inline]

トラックボール処理の換算係数を取り出す.

引数

factor	トラックボールの換算係数を格納する 2 要素の配列.

gg.h の 3778 行目に定義があります。

6.14.3.9 const GLfloat* gg::GgTrackball::getStart ( ) const [inline]

トラックボール処理の開始位置を取り出す.

戻り値

トラックボールの開始位置のポインタ.

gg.h の 3742 行目に定義があります。

6.14.3.10 GLfloat gg::GgTrackball::getStart (int *direction*) const [inline]

トラックボール処理の開始位置を取り出す.

引数

direction	0 なら x 方向, 1 なら y 方向.

gg.h の 3749 行目に定義があります。

6.14.3.11 void gg::GgTrackball::getStart (GLfloat * *position*) const [inline]

トラックボール処理の開始位置を取り出す.

引数

\position	トラックボールの開始位置を格納する 2 要素の配列.

gg.h の 3756 行目に定義があります。

6.14.3.12 void gg::GgTrackball::motion (float *x*, float *y*)

回転の変換行列を計算する.

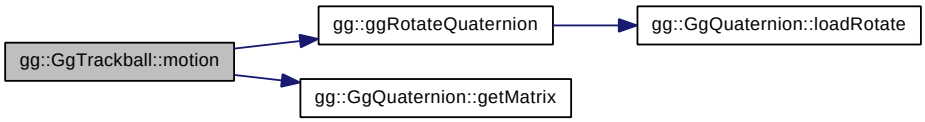
マウスのドラッグ中に呼び出す.

引数

x	現在のマウスの x 座標.
y	現在のマウスの y 座標.

gg.cpp の 4881 行目に定義があります。

呼び出し関係図:



6.14.3.13 void gg::GgTrackball::region (float *w*, float *h*)

トラックボール処理するマウスの移動範囲を指定する.

ウィンドウのリサイズ時に呼び出す.

引数

w	領域の横幅.
h	領域の高さ.

gg.cpp の 4852 行目に定義があります。

被呼び出し関係図:

6.14.3.14 void gg::GgTrackball::region (int *w*, int *h*) [inline]

トラックボール処理するマウスの移動範囲を指定する.

ウィンドウのリサイズ時に呼び出す.

引数

w	領域の横幅.

h	領域の高さ.

gg.h の 3710 行目に定義があります。

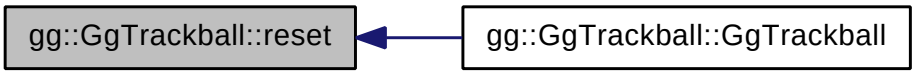
呼び出し関係図:

6.14.3.15 void gg::GgTrackball::reset ( )

トラックボールをリセットする

gg.cpp の 4834 行目に定義があります。

被呼び出し関係図:



6.14.3.16 void gg::GgTrackball::rotate (const GgQuaternion & *q*)

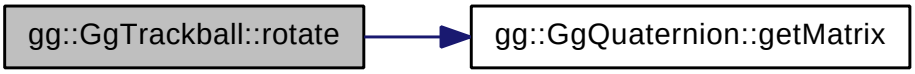
トラックボールの回転角を修正する.

引数

q	修正分の回転角の四元数.

gg.cpp の 4908 行目に定義があります。

呼び出し関係図:



このクラス詳解は次のファイルから抽出されました:

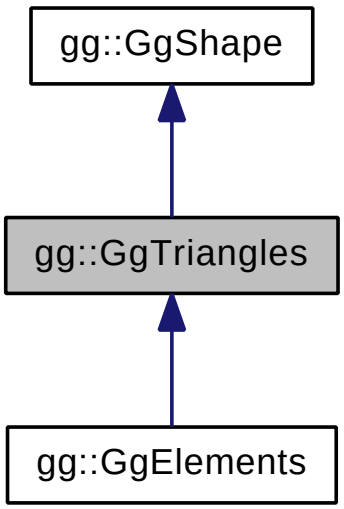
- gg.h
- gg.cpp

6.15 gg::GgTriangles クラス

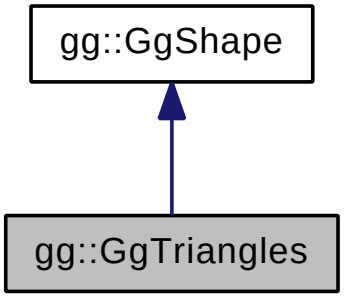
三角形で表した形状データ (Arrays 形式).

`#include <gg.h>`

gg::GgTriangles の継承関係図



gg::GgTriangles 連携図



公開メンバ関数

- virtual ~GgTriangles ()
  デストラクタ.
- GgTriangles (GLenum mode=GL_TRIANGLES)
  コンストラクタ
- GgTriangles (const GgVertex *vert, GLsizei count, GLenum mode=GL_TRIANGLES, GLenum usage=GL_STATIC_DRAW)
  コンストラクタ.
- GLsizei getCount () const
  データの数を取り出す.
- GLuint getBuffer () const
  頂点属性を格納した頂点バッファオブジェクト名を取り出す.
- void send (const GgVertex *vert, GLint first=0, GLsizei count=0) const
  既存のバッファオブジェクトに頂点属性を転送する.
- void load (const GgVertex *vert, GLsizei count, GLenum usage=GL_STATIC_DRAW)
  バッファオブジェクトを確保して頂点属性を格納する.
- virtual void draw (GLint first=0, GLsizei count=0) const
  三角形の描画.

6.15.1 詳解

三角形で表した形状データ (Arrays 形式).

gg.h の 4644 行目に定義があります。

6.15.2 構築子と解体子

6.15.2.1 virtual gg::GgTriangles::~GgTriangles ( ) [inline], [virtual]

デストラクタ.

gg.h の 4653 行目に定義があります。

6.15.2.2 gg::GgTriangles::GgTriangles (GLenum *mode* = GL_TRIANGLES) [inline]

コンストラクタ

引数

mode	描画する基本図形の種類.

gg.h の 4657 行目に定義があります。

6.15.2.3 gg::GgTriangles::GgTriangles (const GgVertex * *vert*, GLsizei *count*, GLenum *mode* = GL_TRIANGLES, GLenum *usage* = GL_STATIC_DRAW) [inline]

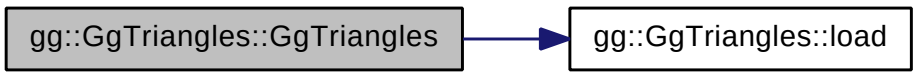
コンストラクタ.

引数

vert	この図形の頂点属性の配列 (nullptr ならデータを転送しない).
count	頂点数.
mode	描画する基本図形の種類.
usage	バッファオブジェクトの使い方.

gg.h の 4666 行目に定義があります。

呼び出し関係図:



6.15.3 関数詳解

6.15.3.1 void gg::GgTriangles::draw (GLint *first* = 0, GLsizei *count* = 0) const [virtual]

三角形の描画.

引数

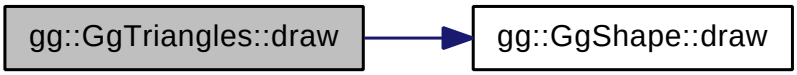
first	描画を開始する最初の三角形番号.
count	描画する三角形数, 0 なら全部の三角形を描く.

gg::GgShapeを再実装しています。

gg::GgElementsで再実装されています。

gg.cpp の 4956 行目に定義があります。

呼び出し関係図:



6.15.3.2 GLuint gg::GgTriangles::getBuffer ( ) const [inline]

頂点属性を格納した頂点バッファオブジェクト名を取り出す.

戻り値

この図形の頂点属性を格納した頂点バッファオブジェクト名.

gg.h の 4682 行目に定義があります。

6.15.3.3 GLsizei gg::GgTriangles::getCount ( ) const [inline]

データの数を取り出す.

戻り値

この図形の頂点属性の数 (頂点数).

gg.h の 4675 行目に定義があります。

6.15.3.4 void gg::GgTriangles::load (const GgVertex * *vert*, GLsizei *count*, GLenum *usage* = GL_STATIC_DRAW) [inline]

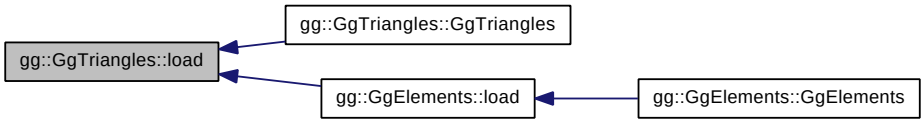
バッファオブジェクトを確保して頂点属性を格納する.

引数

vert	頂点属性が格納されてている領域の先頭のポインタ.
count	頂点のデータの数 (頂点数).
usage	バッファオブジェクトの使い方.

gg.h の 4700 行目に定義があります。

被呼び出し関係図:



6.15.3.5 void gg::GgTriangles::send (const GgVertex * *vert*, GLint *first* = 0, GLsizei *count* = 0) const [inline]

既存のバッファオブジェクトに頂点属性を転送する.

引数

vert	転送元の頂点属性が格納されてている領域の先頭のポインタ.
first	転送先のバッファオブジェクトの先頭の要素番号.
count	転送する頂点の位置データの数 (0 ならバッファオブジェクト全体).

gg.h の 4691 行目に定義があります。

被呼び出し関係図:

このクラス詳解は次のファイルから抽出されました:

- gg.h
- gg.cpp

6.16 gg::GgUniformBuffer< T > クラステンプレート

ユニフォームバッファオブジェクト.

```
#include <gg.h>
```

公開メンバ関数

- virtual ~GgUniformBuffer ()
  デストラクタ.
- GgUniformBuffer ()
  コンストラクタ.
- GgUniformBuffer (const T *data, GLsizei count, GLenum usage=GL_STATIC_DRAW)
  ユニフォームバッファオブジェクトのブロックごとにデータを転送するコンストラクタ.
- GgUniformBuffer (const T &data, GLsizei count, GLenum usage=GL_STATIC_DRAW)
  ユニフォームバッファオブジェクトの全ブロックに同じデータを格納するコンストラクタ.
- GLuint getTarget () const
  ユニフォームバッファオブジェクトのターゲットを取り出す.
- GLsizeiptr getStride () const
  ユニフォームバッファオブジェクトのアライメントを考慮したデータの間隔を取り出す.
- GLsizei getCount () const
  データの数を取り出す.
- GLuint getBuffer () const
  ユニフォームバッファオブジェクト名を取り出す.
- void bind () const
  ユニフォームバッファオブジェクトを結合する.
- void unbind () const
  ユニフォームバッファオブジェクトを解放する.
- void * map () const
  ユニフォームバッファオブジェクトをマップする.
- void * map (GLint first, GLsizei count) const
  ユニフォームバッファオブジェクトの指定した範囲をマップする.
- void unmap () const

バッファオブジェクトをアンマップする.

- void load (const T *data, GLsizei count, GLenum usage=GL_STATIC_DRAW)

  ユニフォームバッファオブジェクトを確保してブロックごとにデータを転送する.
- void load (const T &data, GLsizei count, GLenum usage=GL_STATIC_DRAW)

  ユニフォームバッファオブジェクトを確保して全てのブロックに同じデータを格納する.
- void send (const GLvoid *data, GLint offset=0, GLsizei size=sizeof(T), GLint first=0, GLsizei count=0) const

  ユニフォームバッファオブジェクトを確保してユニフォームバッファオブジェクトのブロックごとのメンバを同じデータで埋める.
- void fill (const GLvoid *data, GLint offset=0, GLsizei size=sizeof(T), GLint first=0, GLsizei count=0) const

  ユニフォームバッファオブジェクトの全ブロックのメンバーを同じデータを格納する.
- void read (GLvoid *data, GLint offset=0, GLsizei size=sizeof(T), GLint first=0, GLsizei count=0) const

  ユニフォームバッファオブジェクトからデータを抽出する.
- void copy (GLuint src_buffer, GLint src_first=0, GLint dst_first=0, GLsizei count=0) const

  別のバッファオブジェクトからデータを複写する.

6.16.1 詳解

template<typename T>class gg::GgUniformBuffer< T >

ユニフォームバッファオブジェクト.

ユニフォーム変数を格納するバッファオブジェクトの基底クラス.

gg.h の 4212 行目に定義があります。

6.16.2 構築子と解体子

6.16.2.1 template<typename T> virtual gg::GgUniformBuffer< T >::~GgUniformBuffer ( ) [inline], [virtual]

デストラクタ.

gg.h の 4220 行目に定義があります。

6.16.2.2 template<typename T> gg::GgUniformBuffer< T >::GgUniformBuffer ( ) [inline]

コンストラクタ.

gg.h の 4223 行目に定義があります。

6.16.2.3 template<typename T> gg::GgUniformBuffer< T >::GgUniformBuffer (const T * *data*, GLsizei *count*, GLenum *usage* = GL_STATIC_DRAW) [inline]

ユニフォームバッファオブジェクトのブロックごとにデータを転送するコンストラクタ.

引数

data	データが格納されている領域の先頭のポインタ (nullptr ならデータを転送しない).
count	データの数.
usage	バッファオブジェクトの使い方.

gg.h の 4229 行目に定義があります。

6.16.2.4 template<typename T> gg::GgUniformBuffer< T >::GgUniformBuffer (const T & *data*, GLsizei *count*, GLenum *usage* = GL_STATIC_DRAW) [inline]

ユニフォームバッファオブジェクトの全ブロックに同じデータを格納するコンストラクタ.

引数

data	格納するデータ.
count	格納する数.
usage	バッファオブジェクトの使い方.

gg.h の 4238 行目に定義があります。

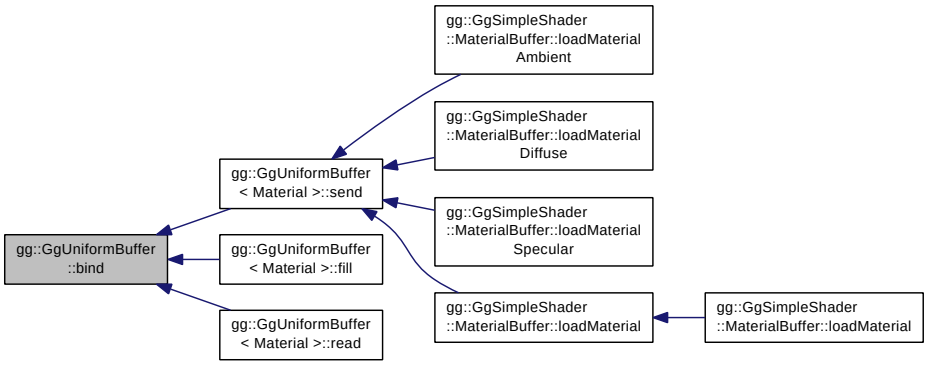
6.16.3 関数詳解

6.16.3.1 template<typename T> void gg::GgUniformBuffer< T >::bind ( ) const [inline]

ユニフォームバッファオブジェクトを結合する.

gg.h の 4272 行目に定義があります。

被呼び出し関係図:



6.16.3.2 template<typename T> void gg::GgUniformBuffer< T >::copy (GLuint *src_buffer*, GLint *src_first* = 0, GLint *dst_first* = 0, GLsizei *count* = 0) const [inline]

別のバッファオブジェクトからデータを複写する.

引数

src_buffer	複写元のバッファオブジェクト名.
src_first	複写元 (buffer) の先頭のデータの位置.
dst_first	複写先 (getBuffer()) の先頭のデータの位置.
count	複写するデータの数 (0 ならバッファオブジェクト全体).

gg.h の 4426 行目に定義があります。

6.16.3.3 template<typename T> void gg::GgUniformBuffer< T >::fill (const GLvoid * *data*, GLint *offset* = 0, GLsizei *size* = sizeof (T), GLint *first* = 0, GLsizei *count* = 0) const [inline]

ユニフォームバッファオブジェクトの全ブロックのメンバーを同じデータを格納する.

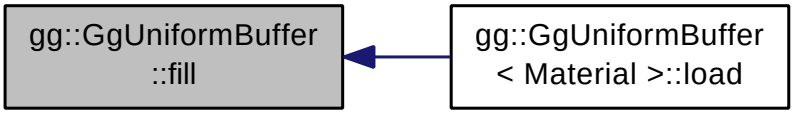
引数

data	格納するデータ.
offset	格納先のメンバのブロックの先頭からのバイトオフセット.

size	格納するデータの一個あたりのバイト数.
first	格納先のバッファオブジェクトのブロックの先頭の番号.
count	格納するデータの数.

gg.h の 4372 行目に定義があります。

被呼び出し関係図:



6.16.3.4 template<typename T> GLuint gg::GgUniformBuffer< T >::getBuffer ( ) const [inline]

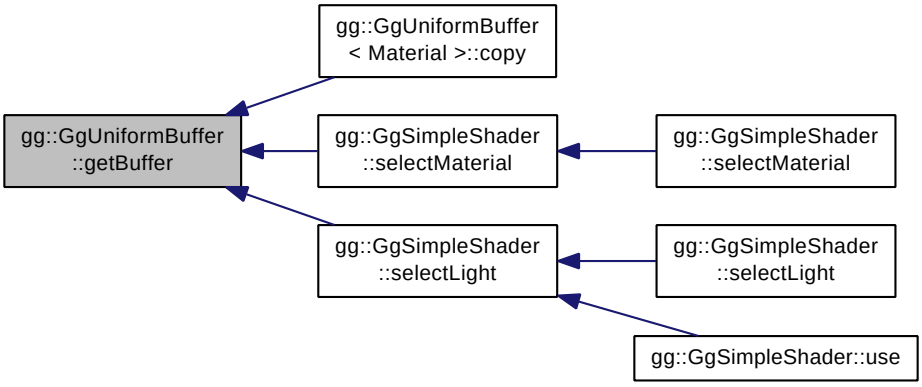
ユニフォームバッファオブジェクト名を取り出す.

戻り値

このユニフォームバッファオブジェクト名.

gg.h の 4266 行目に定義があります。

被呼び出し関係図:



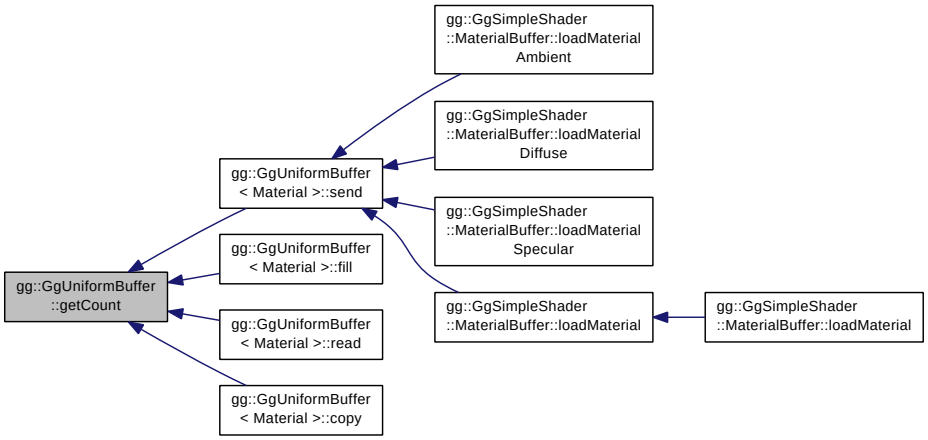
6.16.3.5 template<typename T> GLsizei gg::GgUniformBuffer< T >::getCount ( ) const [inline]

データの数を取り出す.

戻り値

このユニフォームバッファオブジェクトのデータの数.

gg.h の 4259 行目に定義があります。
被呼び出し関係図:



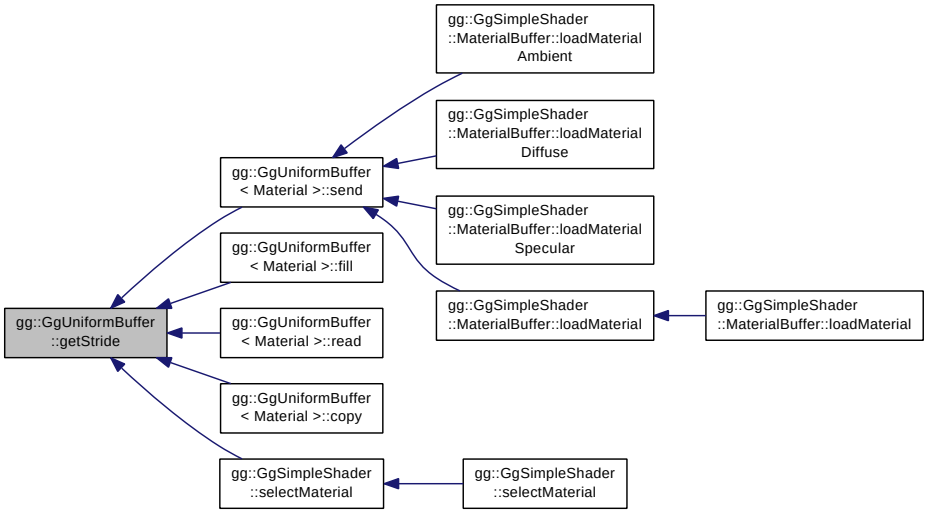
6.16.3.6 template<typename T> GLsizeiptr gg::GgUniformBuffer< T >::getStride ( ) const [inline]

ユニフォームバッファオブジェクトのアライメントを考慮したデータの間隔を取り出す.

戻り値

このユニフォームバッファオブジェクトのデータの間隔.

gg.h の 4252 行目に定義があります。
被呼び出し関係図:



6.16.3.7 template<typename T> GLuint gg::GgUniformBuffer< T >::getTarget ( ) const [inline]

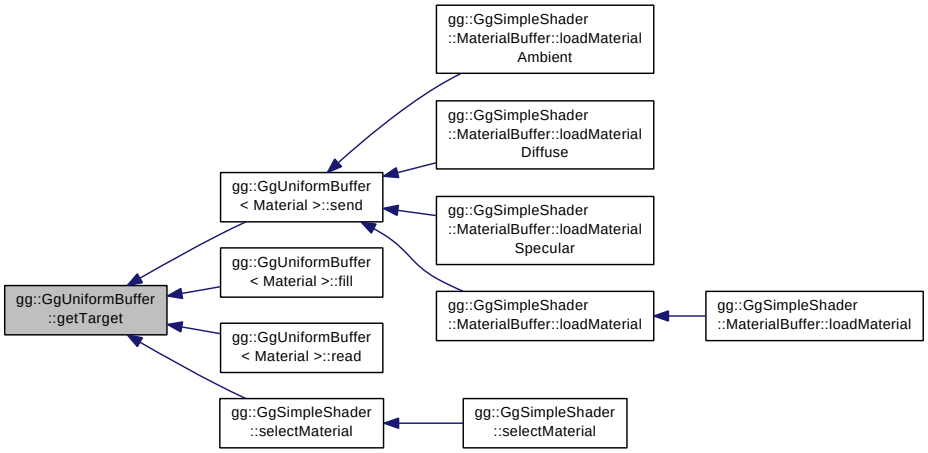
ユニフォームバッファオブジェクトのターゲットを取り出す.

戻り値

このユニフォームバッファオブジェクトのターゲット.

gg.h の 4245 行目に定義があります。

被呼び出し関係図:



6.16.3.8 template<typename T> void gg::GgUniformBuffer< T >::load (const T * *data*, GLsizei *count*, GLenum *usage* = GL_STATIC_DRAW) [inline]

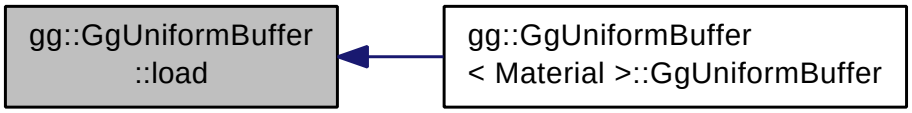
ユニフォームバッファオブジェクトを確保してブロックごとにデータを転送する.

引数

data	データが格納されている領域の先頭のポインタ (nullptr ならデータを転送しない).
count	データの数.
usage	バッファオブジェクトの使い方.

gg.h の 4309 行目に定義があります。

被呼び出し関係図:



6.16.3.9 template<typename T> void gg::GgUniformBuffer< T >::load (const T & *data*, GLsizei *count*, GLenum *usage* = GL_STATIC_DRAW) [inline]

ユニフォームバッファオブジェクトを確保して全てのブロックに同じデータを格納する.

引数

data	格納するデータ.
count	格納する数.

usage	バッファオブジェクトの使い方.

gg.h の 4325 行目に定義があります。

6.16.3.10 template<typename T> void* gg::GgUniformBuffer< T >::map ( ) const [inline]

ユニフォームバッファオブジェクトをマップする.

戻り値

マップしたメモリの先頭のポインタ.

gg.h の 4285 行目に定義があります。

6.16.3.11 template<typename T> void* gg::GgUniformBuffer< T >::map (GLint *first*, GLsizei *count*) const [inline]

ユニフォームバッファオブジェクトの指定した範囲をマップする.

引数

first	マップする範囲のバッファオブジェクトの先頭からの位置.
count	マップするデータの数 (0 ならバッファオブジェクト全体).

戻り値

マップしたメモリの先頭のポインタ.

gg.h の 4294 行目に定義があります。

6.16.3.12 template<typename T> void gg::GgUniformBuffer< T >::read (GLvoid * *data*, GLint *offset* = 0, GLsizei *size* = sizeof (T), GLint *first* = 0, GLsizei *count* = 0) const [inline]

ユニフォームバッファオブジェクトからデータを抽出する.

引数

data	抽出先の領域の先頭のポインタ.
offset	抽出元のユニフォームバッファオブジェクトのメンバのブロックの先頭からのバイトオフセット.
size	抽出するデータの一個あたりのバイト数.
first	抽出元のユニフォームバッファオブジェクトのブロックの先頭の番号.
count	抽出するデータの数 (0 ならユニフォームバッファオブジェクト全体).

gg.h の 4398 行目に定義があります。

6.16.3.13 template<typename T> void gg::GgUniformBuffer< T >::send (const GLvoid * *data*, GLint *offset* = 0, GLsizei *size* = sizeof (T), GLint *first* = 0, GLsizei *count* = 0) const [inline]

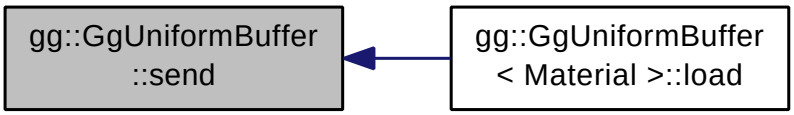
ユニフォームバッファオブジェクトを確保してユニフォームバッファオブジェクトのブロックごとのメンバを同じデータで埋める.

引数

data	データが格納されている領域の先頭のポインタ.
offset	格納先のメンバのブロックの先頭からのバイトオフセット.
size	格納するデータの一個あたりのバイト数.
first	格納先のバッファオブジェクトのブロックの先頭の番号.
count	格納するデータの数.

gg.h の 4343 行目に定義があります。

被呼び出し関係図:



6.16.3.14 template<typename T> void gg::GgUniformBuffer< T >::unbind ( ) const [inline]

ユニフォームバッファオブジェクトを解放する.

gg.h の 4278 行目に定義があります。

6.16.3.15 template<typename T> void gg::GgUniformBuffer< T >::unmap ( ) const [inline]

バッファオブジェクトをアンマップする.

gg.h の 4300 行目に定義があります。

このクラス詳解は次のファイルから抽出されました:

- gg.h

6.17 gg::GgVertex 構造体

三角形の頂点データ.

`#include <gg.h>`

公開メンバ関数

- GgVertex ()
  法線.
- GgVertex (const GgVector &pos, const GgVector &norm)
  コンストラクタ.
- GgVertex (GLfloat px, GLfloat py, GLfloat pz, GLfloat nx, GLfloat ny, GLfloat nz)
  コンストラクタ.
- GgVertex (const GLfloat *pos, const GLfloat *norm)
  コンストラクタ.

公開変数類

- GgVector position
- GgVector normal
  位置.

6.17.1 詳解

三角形の頂点データ.

gg.h の 4606 行目に定義があります。

6.17.2 構築子と解体子

6.17.2.1 gg::GgVertex::GgVertex ( ) [inline]

法線.

コンストラクタ.

gg.h の 4612 行目に定義があります。

6.17.2.2 gg::GgVertex::GgVertex (const GgVector & *pos*, const GgVector & *norm*) [inline]

コンストラクタ.

引数

pos	GgVector 型の位置データ.
norm	GgVector 型の法線データ.

gg.h の 4617 行目に定義があります。

6.17.2.3 gg::GgVertex::GgVertex (GLfloat *px*, GLfloat *py*, GLfloat *pz*, GLfloat *nx*, GLfloat *ny*, GLfloat *nz*) [inline]

コンストラクタ.

引数

px	GgVector 型の位置データの x 成分.
py	GgVector 型の位置データの y 成分.
pz	GgVector 型の位置データの z 成分.
nx	GgVector 型の法線データの x 成分.
ny	GgVector 型の法線データの y 成分.
nz	GgVector 型の法線データの z 成分.

gg.h の 4628 行目に定義があります。

6.17.2.4 gg::GgVertex::GgVertex (const GLfloat * *pos*, const GLfloat * *norm*) [inline]

コンストラクタ.

引数

pos	3 要素の GLfloat 型の位置データのポインタ.
norm	3 要素の GLfloat 型の法線データのポインタ.

gg.h の 4636 行目に定義があります。

6.17.3 メンバ詳解

6.17.3.1 GgVector gg::GgVertex::normal

位置.

gg.h の 4609 行目に定義があります。

6.17.3.2 GgVector gg::GgVertex::position

gg.h の 4608 行目に定義があります。
この構造体詳解は次のファイルから抽出されました:

- gg.h

6.18 gg::GgSimpleShader::Light 構造体

三角形に単純な陰影付けを行うシェーダが参照する光源データ.

```
#include <gg.h>
```

公開変数類

- GgVector ambient
- GgVector diffuse

  光源強度の環境光成分.
- GgVector specular

  光源強度の拡散反射光成分.
- GgVector position

  光源強度の鏡面反射光成分.

6.18.1 詳解

三角形に単純な陰影付けを行うシェーダが参照する光源データ.
gg.h の 5122 行目に定義があります。

6.18.2 メンバ詳解

6.18.2.1 GgVector gg::GgSimpleShader::Light::ambient

gg.h の 5124 行目に定義があります。

6.18.2.2 GgVector gg::GgSimpleShader::Light::diffuse

光源強度の環境光成分.
gg.h の 5125 行目に定義があります。

6.18.2.3 GgVector gg::GgSimpleShader::Light::position

光源強度の鏡面反射光成分.
gg.h の 5127 行目に定義があります。

6.18.2.4 GgVector gg::GgSimpleShader::Light::specular

光源強度の拡散反射光成分.

gg.h の 5126 行目に定義があります。

この構造体詳解は次のファイルから抽出されました:

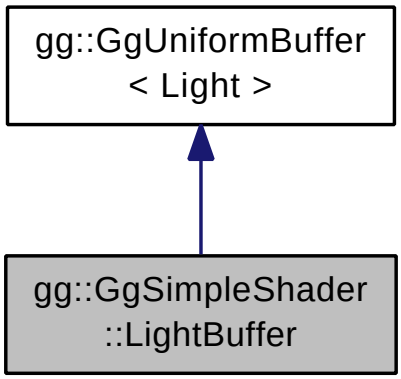
- gg.h

6.19 gg::GgSimpleShader::LightBuffer クラス

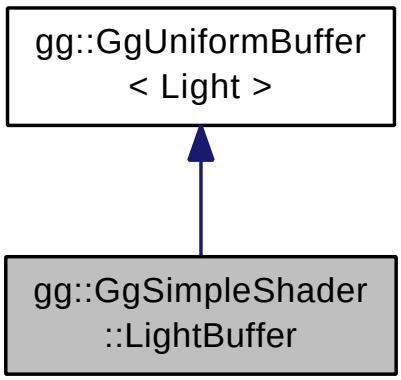
三角形に単純な陰影付けを行うシェーダが参照する光源データのユニフォームバッファオブジェクト.

`#include <gg.h>`

gg::GgSimpleShader::LightBuffer の継承関係図



gg::GgSimpleShader::LightBuffer 連携図



公開メンバ関数

- virtual ~LightBuffer ()
  デストラクタ.
- LightBuffer (const Light *light=nullptr, GLsizei count=1, GLenum usage=GL_STATIC_DRAW)
  デフォルトコンストラクタ.
- LightBuffer (const Light &light, GLsizei count=1, GLenum usage=GL_STATIC_DRAW)

同じデータで埋めるコンストラクタ.

- void loadLightAmbient (GLfloat r, GLfloat g, GLfloat b, GLfloat a=1.0f, GLint first=0, GLsizei count=1) const
  光源の強度の環境光成分を設定する.
- void loadLightAmbient (const GLfloat ∗ambient, GLint first=0, GLsizei count=1) const
  光源の強度の環境光成分を設定する.
- void loadLightDiffuse (GLfloat r, GLfloat g, GLfloat b, GLfloat a=1.0f, GLint first=0, GLsizei count=1) const
  光源の強度の拡散反射光成分を設定する.
- void loadLightDiffuse (const GLfloat ∗diffuse, GLint first=0, GLsizei count=1) const
  光源の強度の拡散反射光成分を設定する.
- void loadLightSpecular (GLfloat r, GLfloat g, GLfloat b, GLfloat a=1.0f, GLint first=0, GLsizei count=1) const
  光源の強度の鏡面反射光成分を設定する.
- void loadLightSpecular (const GLfloat ∗specular, GLint first=0, GLsizei count=1) const
  光源の強度の鏡面反射光成分を設定する.
- void loadLightMaterial (const Light &material, GLint first=0, GLsizei count=1) const
  光源の色を設定するが位置は変更しない.
- void loadLightPosition (GLfloat x, GLfloat y, GLfloat z, GLfloat w=1.0f, GLint first=0, GLsizei count=1) const
  光源の位置を設定する.
- void loadLightPosition (const GgVector &position, GLint first=0, GLsizei count=1) const
  光源の位置を設定する.
- void loadLightPosition (const GLfloat ∗position, GLint first=0, GLsizei count=1) const
  光源の位置を設定する.
- void loadLightPosition (const GgVector ∗position, GLint first=0, GLsizei count=1) const
  光源の位置を設定する.
- void loadLight (const Light ∗light, GLint first=0, GLsizei count=1) const
  光源の色と位置を設定する.
- void loadLight (const Light &light, GLint first=0, GLsizei count=1) const
  光源の色と位置を設定する.

6.19.1 詳解

三角形に単純な陰影付けを行うシェーダが参照する光源データのユニフォームバッファオブジェクト.

gg.h の 5133 行目に定義があります。

6.19.2 構築子と解体子

6.19.2.1 virtual gg::GgSimpleShader::LightBuffer::∼LightBuffer ( ) [inline], [virtual]

デストラクタ.

gg.h の 5139 行目に定義があります。

6.19.2.2 gg::GgSimpleShader::LightBuffer::LightBuffer (const Light ∗ *light* = nullptr, GLsizei *count* = 1, GLenum *usage* = GL_STATIC_DRAW) [inline]

デフォルトコンストラクタ.

引数

light	GgSimpleShader::Light 型の光源データのポインタ.
count	バッファ中の GgSimpleShader::Light 型の光源データの数.

gg.h の 5144 行目に定義があります。

6.19.2.3 gg::GgSimpleShader::LightBuffer::LightBuffer (const Light & *light*, GLsizei *count* = 1, GLenum *usage* = GL_STATIC_DRAW) [inline]

同じデータで埋めるコンストラクタ.

引数

light	GgSimpleShader::Light 型の光源データ.
count	バッファ中の GgSimpleShader::Light 型の光源データの数.

gg.h の 5150 行目に定義があります。

6.19.3 関数詳解

6.19.3.1 void gg::GgSimpleShader::LightBuffer::loadLight (const Light * *light*, GLint *first* = 0, GLsizei *count* = 1) const [inline]

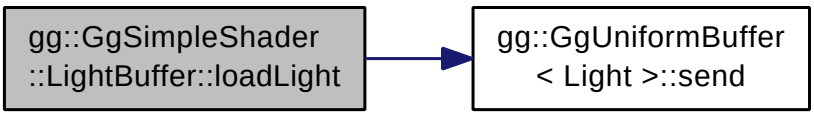
光源の色と位置を設定する.

引数

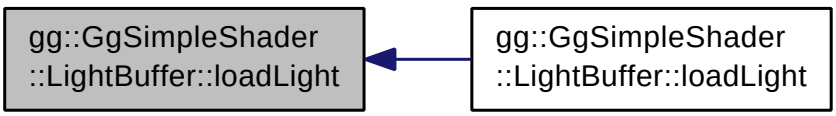
light	光源の特性の GgSimpleShader::Light 構造体のポインタ
first	値を設定する光源データの最初の番号, デフォルトは 0.
count	値を設定する光源データの数, デフォルトは 1.

gg.h の 5253 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:



6.19.3.2 void gg::GgSimpleShader::LightBuffer::loadLight (const Light & *light*, GLint *first* = 0, GLsizei *count* = 1) const [inline]

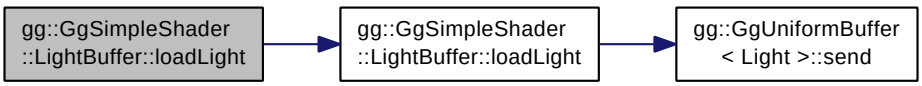
光源の色と位置を設定する.

引数

light	光源の特性の GgSimpleShader::Light 構造体
first	値を設定する光源データの最初の番号, デフォルトは 0.
count	値を設定する光源データの数, デフォルトは 1.

gg.h の 5262 行目に定義があります。

呼び出し関係図:



6.19.3.3 void gg::GgSimpleShader::LightBuffer::loadLightAmbient (GLfloat *r*, GLfloat *g*, GLfloat *b*, GLfloat *a* = 1.0f, GLint *first* = 0, GLsizei *count* = 1) const

光源の強度の環境光成分を設定する.

引数

r	光源の強度の環境光成分の赤成分.
g	光源の強度の環境光成分の緑成分.
b	光源の強度の環境光成分の青成分.
first	値を設定する光源データの最初の番号, デフォルトは 0.
count	値を設定する光源データの数, デフォルトは 1.

gg.cpp の 5267 行目に定義があります。

6.19.3.4 void gg::GgSimpleShader::LightBuffer::loadLightAmbient (const GLfloat * *ambient*, GLint *first* = 0, GLsizei *count* = 1) const [inline]

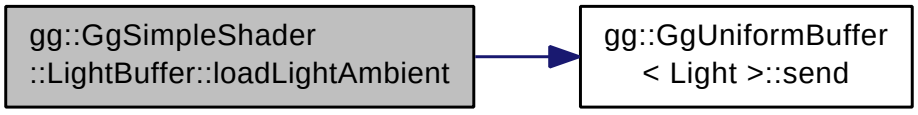
光源の強度の環境光成分を設定する.

引数

ambient	光源の強度の環境光成分を格納した GLfloat 型の 4 要素の配列変数.
first	値を設定する光源データの最初の番号, デフォルトは 0.
count	値を設定する光源データの数, デフォルトは 1.

gg.h の 5165 行目に定義があります。

呼び出し関係図:



6.19.3.5 void gg::GgSimpleShader::LightBuffer::loadLightDiffuse (GLfloat *r*, GLfloat *g*, GLfloat *b*, GLfloat *a* = 1.0f, GLint *first* = 0, GLsizei *count* = 1) const

光源の強度の拡散反射光成分を設定する.

引数

r	光源の強度の拡散反射光成分の赤成分.
g	光源の強度の拡散反射光成分の緑成分.
b	光源の強度の拡散反射光成分の青成分.
a	光源の強度の拡散反射光成分の不透明度, デフォルトは 1.
first	値を設定する光源データの最初の番号, デフォルトは 0.
count	値を設定する光源データの数, デフォルトは 1.

gg.cpp の 5296 行目に定義があります。

6.19.3.6 void gg::GgSimpleShader::LightBuffer::loadLightDiffuse (const GLfloat * *diffuse*, GLint *first* = 0, GLsizei *count* = 1) const [inline]

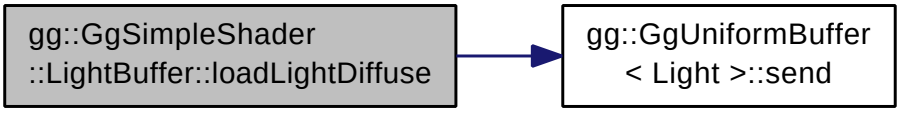
光源の強度の拡散反射光成分を設定する.

引数

diffuse	光源の強度の拡散反射光成分を格納した GLfloat 型の 4 要素の配列変数.
first	値を設定する光源データの最初の番号, デフォルトは 0.
count	値を設定する光源データの数, デフォルトは 1.

gg.h の 5184 行目に定義があります。

呼び出し関係図:



6.19.3.7 void gg::GgSimpleShader::LightBuffer::loadLightMaterial (const Light & *material*, GLint *first* = 0, GLsizei *count* = 1) const

光源の色を設定するが位置は変更しない.

引数

material	光源の特性の GgSimpleShader::Light 構造体.
first	値を設定する光源データの最初の番号, デフォルトは 0.
count	値を設定する光源データの数, デフォルトは 1.

gg.cpp の 5351 行目に定義があります。

6.19.3.8 void gg::GgSimpleShader::LightBuffer::loadLightPosition (GLfloat *x*, GLfloat *y*, GLfloat *z*, GLfloat *w* = 1.0f, GLint *first* = 0, GLsizei *count* = 1) const

光源の位置を設定する.

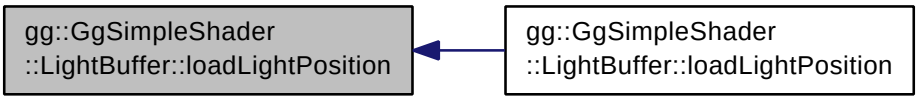
引数

x	光源の位置の x 座標.
y	光源の位置の y 座標.
z	光源の位置の z 座標.

w	光源の位置の w 座標, デフォルトは 1.
first	値を設定する光源データの最初の番号, デフォルトは 0.
count	値を設定する光源データの数, デフォルトは 1.

gg.cpp の 5379 行目に定義があります。

被呼び出し関係図:



6.19.3.9 void gg::GgSimpleShader::LightBuffer::loadLightPosition (const GgVector & *position*, GLint *first* = 0, GLsizei *count* = 1) const

光源の位置を設定する.

引数

position	光源の位置の同次座標を格納した GgVector 型の変数.
first	値を設定する光源データの最初の番号, デフォルトは 0.
count	値を設定する光源データの数, デフォルトは 1.

gg.cpp の 5405 行目に定義があります。

6.19.3.10 void gg::GgSimpleShader::LightBuffer::loadLightPosition (const GLfloat * *position*, GLint *first* = 0, GLsizei *count* = 1) const [inline]

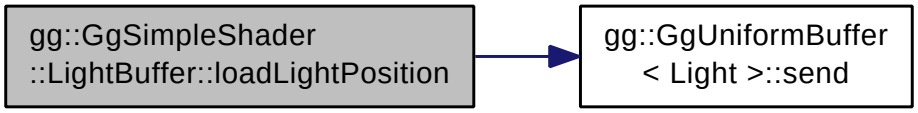
光源の位置を設定する.

引数

position	光源の位置の同次座標を格納した GLfloat 型の 4 要素の配列変数.
first	値を設定する光源データの最初の番号, デフォルトは 0.
count	値を設定する光源データの数, デフォルトは 1.

gg.h の 5234 行目に定義があります。

呼び出し関係図:



6.19.3.11 void gg::GgSimpleShader::LightBuffer::loadLightPosition (const GgVector * *position*, GLint *first* = 0, GLsizei *count* = 1) const [inline]

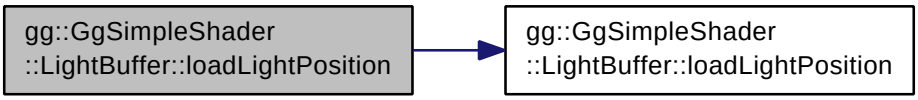
光源の位置を設定する.

引数

position	光源の位置の同次座標を格納した GgVector 型の配列.
first	値を設定する光源データの最初の番号, デフォルトは 0.
count	値を設定する光源データの数, デフォルトは 1.

gg.h の 5244 行目に定義があります。

呼び出し関係図:



6.19.3.12 void gg::GgSimpleShader::LightBuffer::loadLightSpecular (GLfloat *r*, GLfloat *g*, GLfloat *b*, GLfloat *a* = `1.0f`, GLint *first* = `0`, GLsizei *count* = `1`) const

光源の強度の鏡面反射光成分を設定する.

引数

r	光源の強度の鏡面反射光成分の赤成分.
g	光源の強度の鏡面反射光成分の緑成分.
b	光源の強度の鏡面反射光成分の青成分.
a	光源の強度の鏡面反射光成分の不透明度, デフォルトは 1.
first	値を設定する光源データの最初の番号, デフォルトは 0.
count	値を設定する光源データの数, デフォルトは 1.

gg.cpp の 5325 行目に定義があります。

6.19.3.13 void gg::GgSimpleShader::LightBuffer::loadLightSpecular (const GLfloat * *specular*, GLint *first* = `0`, GLsizei *count* = `1`) const `[inline]`

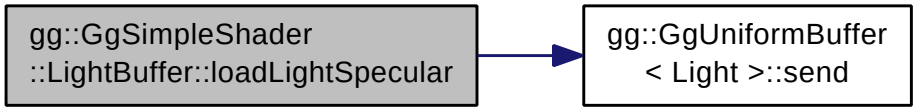
光源の強度の鏡面反射光成分を設定する.

引数

specular	光源の強度の鏡面反射光成分を格納した GLfloat 型の 4 要素の配列変数.
first	値を設定する光源データの最初の番号, デフォルトは 0.
count	値を設定する光源データの数, デフォルトは 1.

gg.h の 5203 行目に定義があります。

呼び出し関係図:



このクラス詳解は次のファイルから抽出されました:

- gg.h
- gg.cpp

6.20 gg::GgSimpleShader::Material 構造体

三角形に単純な陰影付けを行うシェーダが参照する材質データ.

`#include <gg.h>`

公開変数類

- GgVector ambient
- GgVector diffuse

  環境光に対する反射係数.
- GgVector specular

  拡散反射係数.
- GLfloat shininess

  鏡面反射係数.

6.20.1 詳解

三角形に単純な陰影付けを行うシェーダが参照する材質データ.

gg.h の 5271 行目に定義があります。

6.20.2 メンバ詳解

6.20.2.1 GgVector gg::GgSimpleShader::Material::ambient

gg.h の 5273 行目に定義があります。

6.20.2.2 GgVector gg::GgSimpleShader::Material::diffuse

環境光に対する反射係数.

gg.h の 5274 行目に定義があります。

6.20.2.3 GLfloat gg::GgSimpleShader::Material::shininess

鏡面反射係数.

gg.h の 5276 行目に定義があります。

6.20.2.4 GgVector gg::GgSimpleShader::Material::specular

拡散反射係数.

gg.h の 5275 行目に定義があります。

この構造体詳解は次のファイルから抽出されました:

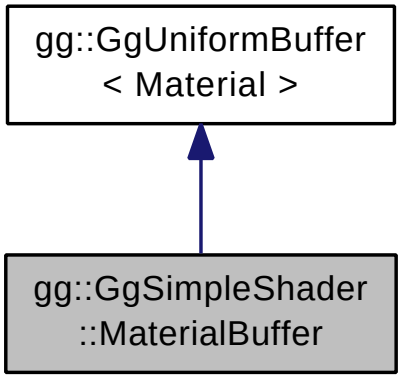
- gg.h

6.21 gg::GgSimpleShader::MaterialBuffer クラス

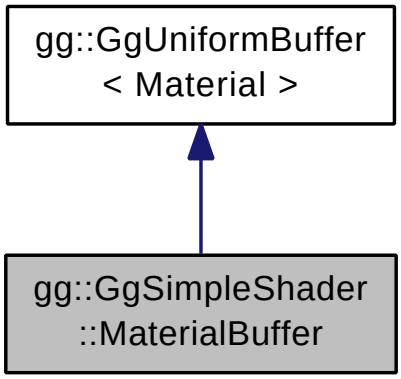
三角形に単純な陰影付けを行うシェーダが参照する材質データのユニフォームバッファオブジェクト.

`#include <gg.h>`

gg::GgSimpleShader::MaterialBuffer の継承関係図



gg::GgSimpleShader::MaterialBuffer 連携図



公開メンバ関数

- virtual ~MaterialBuffer ()
  デストラクタ
- MaterialBuffer (const Material *material=nullptr, GLsizei count=1, GLenum usage=GL_STATIC_DRAW)
  デフォルトコンストラクタ
- MaterialBuffer (const Material &material, GLsizei count=1, GLenum usage=GL_STATIC_DRAW)
  同じデータで埋めるコンストラクタ
- void loadMaterialAmbient (GLfloat r, GLfloat g, GLfloat b, GLfloat a=1.0f, GLint first=0, GLsizei count=1) const
  環境光に対する反射係数を設定する.
- void loadMaterialAmbient (const GLfloat *ambient, GLint first=0, GLsizei count=1) const
  環境光に対する反射係数を設定する.
- void loadMaterialDiffuse (GLfloat r, GLfloat g, GLfloat b, GLfloat a=1.0f, GLint first=0, GLsizei count=1) const
  拡散反射係数を設定する.
- void loadMaterialDiffuse (const GLfloat *diffuse, GLint first=0, GLsizei count=1) const
  拡散反射係数を設定する.

- void loadMaterialAmbientAndDiffuse (GLfloat r, GLfloat g, GLfloat b, GLfloat a=1.0f, GLint first=0, GLsizei count=1) const
  環境光に対する反射係数と拡散反射係数を設定する.
- void loadMaterialAmbientAndDiffuse (const GLfloat *color, GLint first=0, GLsizei count=1) const
  環境光に対する反射係数と拡散反射係数を設定する.
- void loadMaterialSpecular (GLfloat r, GLfloat g, GLfloat b, GLfloat a=1.0f, GLint first=0, GLsizei count=1) const
  鏡面反射係数を設定する.
- void loadMaterialSpecular (const GLfloat *specular, GLint first=0, GLsizei count=1) const
  鏡面反射係数を設定する.
- void loadMaterialShininess (GLfloat shininess, GLint first=0, GLsizei count=1) const
  輝き係数を設定する.
- void loadMaterialShininess (const GLfloat *shininess, GLint first=0, GLsizei count=1) const
  輝き係数を設定する.
- void loadMaterial (const Material *material, GLint first=0, GLsizei count=1) const
  材質を設定する.
- void loadMaterial (const Material &material, GLint first=0, GLsizei count=1) const
  材質を設定する.

6.21.1 詳解

三角形に単純な陰影付けを行うシェーダが参照する材質データのユニフォームバッファオブジェクト.

gg.h の 5282 行目に定義があります。

6.21.2 構築子と解体子

6.21.2.1 virtual gg::GgSimpleShader::MaterialBuffer::~MaterialBuffer ( ) [inline], [virtual]

デストラクタ

gg.h の 5288 行目に定義があります。

6.21.2.2 gg::GgSimpleShader::MaterialBuffer::MaterialBuffer (const Material * *material* = nullptr, GLsizei *count* = 1, GLenum *usage* = GL_STATIC_DRAW) [inline]

デフォルトコンストラクタ

引数

material	GgSimpleShader::Material 型の材質データのポインタ.
count	バッファ中の GgSimpleShader::Material 型の材質データの数.

gg.h の 5293 行目に定義があります。

6.21.2.3 gg::GgSimpleShader::MaterialBuffer::MaterialBuffer (const Material & *material*, GLsizei *count* = 1, GLenum *usage* = GL_STATIC_DRAW) [inline]

同じデータで埋めるコンストラクタ

引数

material	GgSimpleShader::Material 型の材質データ.
count	バッファ中の GgSimpleShader::Material 型の材質データの数.

gg.h の 5299 行目に定義があります。

6.21.3 関数詳解

6.21.3.1 void gg::GgSimpleShader::MaterialBuffer::loadMaterial (const Material * *material*, GLint *first* = 0, GLsizei *count* = 1) const [inline]

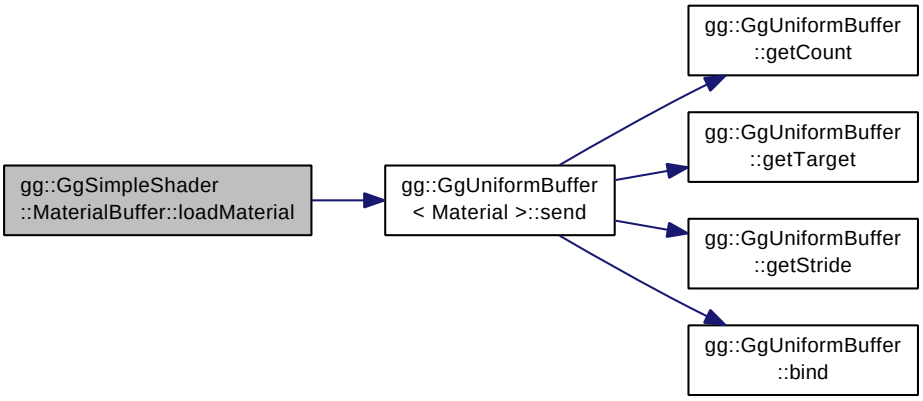
材質を設定する.

引数

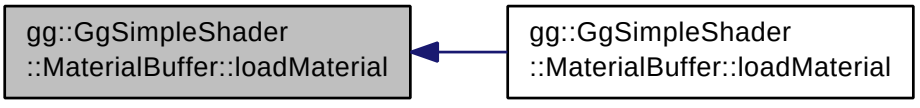
material	光源の特性の GgSimpleShader::Material 構造体のポインタ.
first	値を設定する材質データの最初の番号, デフォルトは 0.
count	値を設定する材質データの数, デフォルトは 1.

gg.h の 5388 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:



6.21.3.2 void gg::GgSimpleShader::MaterialBuffer::loadMaterial (const Material & *material*, GLint *first* = 0, GLsizei *count* = 1) const [inline]

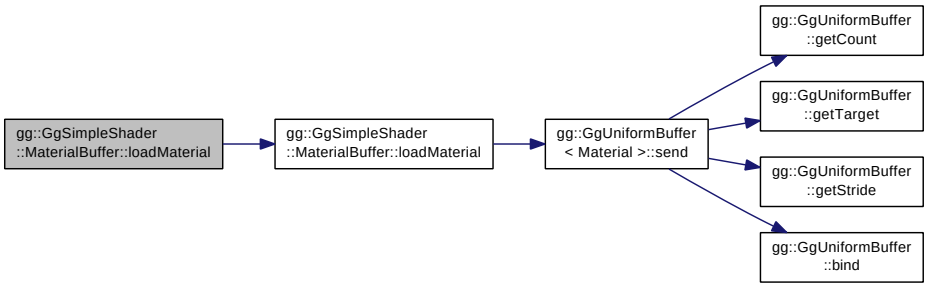
材質を設定する.

引数

material	光源の特性の GgSimpleShader::Material 構造体.
first	値を設定する材質データの最初の番号, デフォルトは 0.
count	値を設定する材質データの数, デフォルトは 1.

gg.h の 5397 行目に定義があります。

呼び出し関係図:



6.21.3.3 void gg::GgSimpleShader::MaterialBuffer::loadMaterialAmbient (GLfloat *r*, GLfloat *g*, GLfloat *b*, GLfloat *a* = `1.0f`, GLint *first* = `0`, GLsizei *count* = `1`) const

環境光に対する反射係数を設定する.

引数

r	環境光に対する反射係数の赤成分.
g	環境光に対する反射係数の緑成分.
b	環境光に対する反射係数の青成分.
a	環境光に対する反射係数の不透明度, デフォルトは 1.
first	値を設定する材質データの最初の番号, デフォルトは 0.
count	値を設定する材質データの数, デフォルトは 1.

gg.cpp の 5431 行目に定義があります。

6.21.3.4 void gg::GgSimpleShader::MaterialBuffer::loadMaterialAmbient (const GLfloat * *ambient*, GLint *first* = `0`, GLsizei *count* = `1`) const [inline]

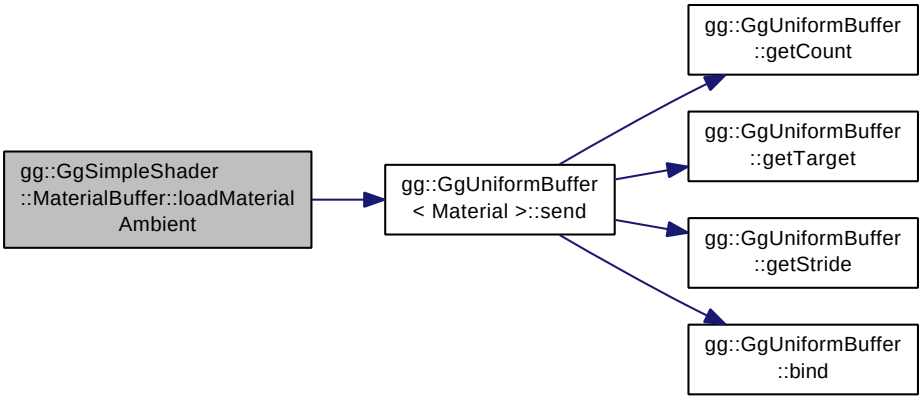
環境光に対する反射係数を設定する.

引数

ambient	環境光に対する反射係数を格納した GLfloat 型の 4 要素の配列変数.

gg.h の 5313 行目に定義があります。

呼び出し関係図:



6.21.3.5 void gg::GgSimpleShader::MaterialBuffer::loadMaterialAmbientAndDiffuse (GLfloat *r*, GLfloat *g*, GLfloat *b*, GLfloat *a* = 1.0f, GLint *first* = 0, GLsizei *count* = 1) const

環境光に対する反射係数と拡散反射係数を設定する.

引数

r	環境光に対する反射係数と拡散反射係数の赤成分.
g	環境光に対する反射係数と拡散反射係数の緑成分.
b	環境光に対する反射係数と拡散反射係数の青成分.
a	環境光に対する反射係数と拡散反射係数の不透明度, デフォルトは 1.
first	値を設定する材質データの最初の番号, デフォルトは 0.
count	値を設定する材質データの数, デフォルトは 1.

gg.cpp の 5489 行目に定義があります。

6.21.3.6 void gg::GgSimpleShader::MaterialBuffer::loadMaterialAmbientAndDiffuse (const GLfloat * *color*, GLint *first* = 0, GLsizei *count* = 1) const

環境光に対する反射係数と拡散反射係数を設定する.

引数

color	環境光に対する反射係数と拡散反射係数を格納した GLfloat 型の 4 要素の配列変数.
first	値を設定する材質データの最初の番号, デフォルトは 0.
count	値を設定する材質データの数, デフォルトは 1.

gg.cpp の 5515 行目に定義があります。

6.21.3.7 void gg::GgSimpleShader::MaterialBuffer::loadMaterialDiffuse (GLfloat *r*, GLfloat *g*, GLfloat *b*, GLfloat *a* = 1.0f, GLint *first* = 0, GLsizei *count* = 1) const

拡散反射係数を設定する.

引数

r	拡散反射係数の赤成分.

g	拡散反射係数の緑成分.
b	拡散反射係数の青成分.
a	拡散反射係数の不透明度, デフォルトは 1.
first	値を設定する材質データの最初の番号, デフォルトは 0.
count	値を設定する材質データの数, デフォルトは 1.

gg.cpp の 5460 行目に定義があります。

6.21.3.8 void gg::GgSimpleShader::MaterialBuffer::loadMaterialDiffuse (const GLfloat * *diffuse*, GLint *first* = 0, GLsizei *count* = 1) const [inline]

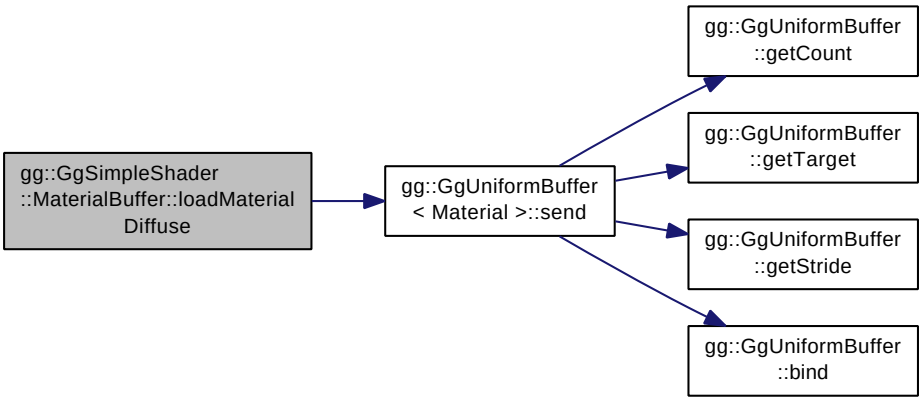
拡散反射係数を設定する.

引数

diffuse	拡散反射係数を格納した GLfloat 型の 4 要素の配列変数.
first	値を設定する材質データの最初の番号, デフォルトは 0.
count	値を設定する材質データの数, デフォルトは 1.

gg.h の 5332 行目に定義があります。

呼び出し関係図:



6.21.3.9 void gg::GgSimpleShader::MaterialBuffer::loadMaterialShininess (GLfloat *shininess*, GLint *first* = 0, GLsizei *count* = 1) const

輝き係数を設定する.

引数

shininess	輝き係数.
first	値を設定する材質データの最初の番号, デフォルトは 0.
count	値を設定する材質データの数, デフォルトは 1.

gg.cpp の 5584 行目に定義があります。

6.21.3.10 void gg::GgSimpleShader::MaterialBuffer::loadMaterialShininess (const GLfloat * *shininess*, GLint *first* = 0, GLsizei *count* = 1) const

輝き係数を設定する.

引数

shininess	輝き係数.
first	値を設定する材質データの最初の番号, デフォルトは 0.
count	値を設定する材質データの数, デフォルトは 1.

gg.cpp の 5605 行目に定義があります。

6.21.3.11 void gg::GgSimpleShader::MaterialBuffer::loadMaterialSpecular (GLfloat *r*, GLfloat *g*, GLfloat *b*, GLfloat *a* = `1.0f`, GLint *first* = `0`, GLsizei *count* = `1`) const

鏡面反射係数を設定する.

引数

r	鏡面反射係数の赤成分.
g	鏡面反射係数の緑成分.
b	鏡面反射係数の青成分.
a	鏡面反射係数の不透明度, デフォルトは 1.
first	値を設定する材質データの最初の番号, デフォルトは 0.
count	値を設定する材質データの数, デフォルトは 1.

gg.cpp の 5558 行目に定義があります。

6.21.3.12 void gg::GgSimpleShader::MaterialBuffer::loadMaterialSpecular (const GLfloat * *specular*, GLint *first* = `0`, GLsizei *count* = `1`) const [inline]

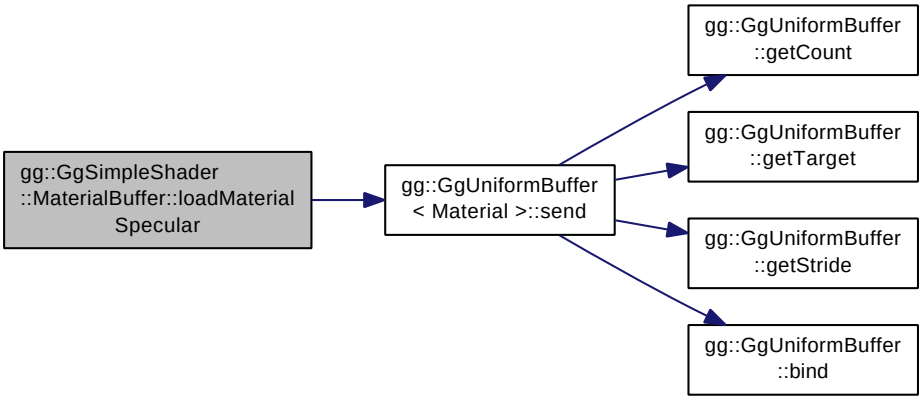
鏡面反射係数を設定する.

引数

specular	鏡面反射係数を格納した GLfloat 型の 4 要素の配列変数.
first	値を設定する材質データの最初の番号, デフォルトは 0.
count	値を設定する材質データの数, デフォルトは 1.

gg.h の 5366 行目に定義があります。

呼び出し関係図:



このクラス詳解は次のファイルから抽出されました:

- gg.h
- gg.cpp

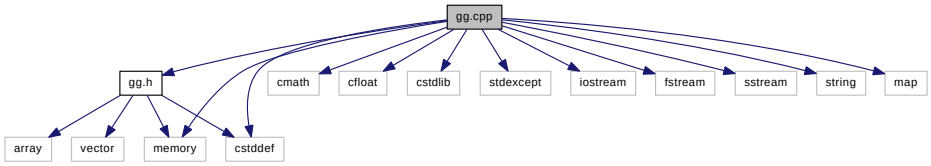
Chapter 7

ファイル詳解

7.1 gg.cpp ファイル

```
#include "gg.h"
#include <cmath>
#include <cfloat>
#include <cstdlib>
#include <cstddef>
#include <stdexcept>
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <memory>
#include <map>
```

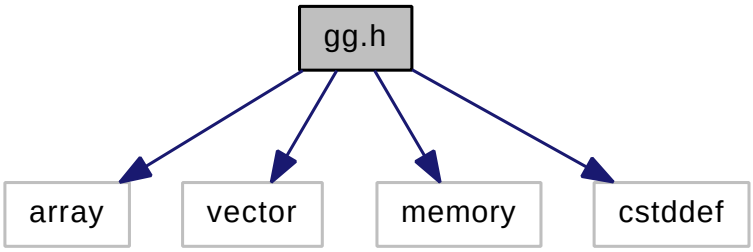
gg.cpp の依存先関係図:



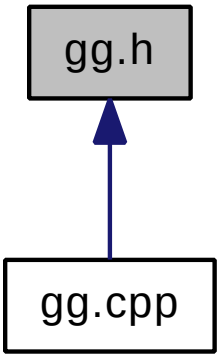
7.2 gg.h ファイル

```
#include <array>
#include <vector>
#include <memory>
#include <cstddef>
```

gg.h の依存先関係図:



被依存関係図:



クラス

- class gg::GgMatrix
  変換行列.
- class gg::GgQuaternion
  四元数.
- class gg::GgTrackball
  簡易トラックボール処理.
- class gg::GgTexture
  テクスチャ.
- class gg::GgColorTexture
  カラーマップ.
- class gg::GgNormalTexture
  法線マップ.
- class gg::GgBuffer< T >
  バッファオブジェクト.
- class gg::GgUniformBuffer< T >
  ユニフォームバッファオブジェクト.
- class gg::GgShape
  形状データの基底クラス.
- class gg::GgPoints
  点.
- struct gg::GgVertex
  三角形の頂点データ.
- class gg::GgTriangles

三角形で表した形状データ *(Arrays* 形式*)*.

- class gg::GgElements
  三角形で表した形状データ *(Elements* 形式*)*.
- class gg::GgShader
  シェーダの基底クラス.
- class gg::GgPointShader
  点のシェーダ.
- class gg::GgSimpleShader
  三角形に単純な陰影付けを行うシェーダ.
- struct gg::GgSimpleShader::Light
  三角形に単純な陰影付けを行うシェーダが参照する光源データ.
- class gg::GgSimpleShader::LightBuffer
  三角形に単純な陰影付けを行うシェーダが参照する光源データのユニフォームバッファオブジェクト.
- struct gg::GgSimpleShader::Material
  三角形に単純な陰影付けを行うシェーダが参照する材質データ.
- class gg::GgSimpleShader::MaterialBuffer
  三角形に単純な陰影付けを行うシェーダが参照する材質データのユニフォームバッファオブジェクト.
- class gg::GgSimpleObj
  Wavefront OBJ 形式のファイル *(Arrays* 形式*)*.

名前空間

- gg
  ゲームグラフィックス特論の宿題用補助プログラムの名前空間

マクロ定義

- #define ggError()
- #define ggFBOError()

型定義

- using gg::GgVector = std::array< GLfloat, 4 >
  4 要素の単精度実数の配列

関数

- void gg::ggInit ()
  ゲームグラフィックス特論の都合にもとづく初期化を行う.
- void gg::_ggError (const char *name=nullptr, unsigned int line=0)
  OpenGL のエラーをチェックする.
- void gg::_ggFBOError (const char *name=nullptr, unsigned int line=0)
  FBO のエラーをチェックする.
- bool gg::ggSaveTga (const char *name, const void *buffer, unsigned int width, unsigned int height, unsigned int depth)
  配列の内容を *TGA* ファイルに保存する.
- bool gg::ggSaveColor (const char *name)
  カラーバッファの内容を *TGA* ファイルに保存する.
- bool gg::ggSaveDepth (const char *name)
  デプスバッファの内容を *TGA* ファイルに保存する.

- bool gg::ggReadImage (const char *name, std::vector< GLubyte > &image, GLsizei *pWidth, GLsizei *pHeight, GLenum *pFormat)

  TGA **ファイル** *(8/16/24/32bit)* **をメモリに読み込む**.
- GLuint gg::ggLoadTexture (const GLvoid *image, GLsizei width, GLsizei height, GLenum format=GL_BGR, GLenum type=GL_UNSIGNED_BYTE, GLenum internal=GL_RGB, GLenum wrap=GL_CLAMP_TO_EDGE)

  テクスチャメモリを確保して画像データをテクスチャとして読み込む.
- GLuint gg::ggLoadImage (const char *name, GLsizei *pWidth=nullptr, GLsizei *pHeight=nullptr, GLenum internal=0, GLenum wrap=GL_CLAMP_TO_EDGE)

  テクスチャメモリを確保して *TGA* **画像ファイルを読み込む**.
- void gg::ggCreateNormalMap (const GLubyte *hmap, GLsizei width, GLsizei height, GLenum format, GLfloat nz, GLenum internal, std::vector< GgVector > &nmap)

  グレースケール画像 *(8bit)* **から法線マップのデータを作成する**.
- GLuint gg::ggLoadHeight (const char *name, float nz, GLsizei *pWidth=nullptr, GLsizei *pHeight=nullptr, GLenum internal=GL_RGBA)

  テクスチャメモリを確保して *TGA* **画像ファイルを読み込み法線マップを作成する**.
- GLuint gg::ggCreateShader (const char *vsrc, const char *fsrc=nullptr, const char *gsrc=nullptr, GLint nvarying=0, const char *const varyings[]=nullptr, const char *vtext="vertex shader", const char *ftext="fragment shader", const char *gtext="geometry shader")

  シェーダのソースプログラムの文字列を読み込んでプログラムオブジェクトを作成する.
- GLuint gg::ggLoadShader (const char *vert, const char *frag=nullptr, const char *geom=nullptr, GLint nvarying=0, const char *const varyings[]=nullptr)

  シェーダのソースファイルを読み込んでプログラムオブジェクトを作成する.
- GLuint gg::ggCreateComputeShader (const char *csrc, const char *ctext="compute shader")

  コンピュートシェーダのソースプログラムの文字列を読み込んでプログラムオブジェクトを作成する.
- GLuint gg::ggLoadComputeShader (const char *comp)

  コンピュートシェーダのソースファイルを読み込んでプログラムオブジェクトを作成する.
- GLfloat gg::ggLength3 (const GLfloat *a)

  3 **要素の長さ**.
- void gg::ggNormalize3 (GLfloat *a)

  3 **要素の正規化**.
- GLfloat gg::ggDot3 (const GLfloat *a, const GLfloat *b)

  3 **要素の内積**.
- void gg::ggCross (GLfloat *c, const GLfloat *a, const GLfloat *b)

  3 **要素の外積**.
- GLfloat gg::ggLength4 (const GLfloat *a)

  4 **要素の長さ**.
- GLfloat gg::ggLength4 (const GgVector &a)

  GgVector **型の長さ**.
- void gg::ggNormalize4 (GLfloat *a)

  4 **要素の正規化**.
- void gg::ggNormalize4 (GgVector &a)

  GgVector **型の正規化**.
- GLfloat gg::ggDot4 (const GLfloat *a, const GLfloat *b)

  4 **要素の内積**
- GLfloat gg::ggDot4 (const GgVector &a, const GgVector &b)

  GgVector **型の内積**
- GgMatrix gg::ggIdentity ()

  単位行列を返す.
- GgMatrix gg::ggTranslate (GLfloat x, GLfloat y, GLfloat z, GLfloat w=1.0f)

  平行移動の変換行列を返す.
- GgMatrix gg::ggTranslate (const GLfloat *t)

  平行移動の変換行列を返す.

- GgMatrix gg::ggTranslate (const GgVector &t)
  平行移動の変換行列を返す.
- GgMatrix gg::ggScale (GLfloat x, GLfloat y, GLfloat z, GLfloat w=1.0f)
  拡大縮小の変換行列を返す.
- GgMatrix gg::ggScale (const GLfloat *s)
  拡大縮小の変換行列を返す.
- GgMatrix gg::ggScale (const GgVector &s)
  拡大縮小の変換行列を返す.
- GgMatrix gg::ggRotateX (GLfloat a)
  x **軸中心の回転の変換行列を返す**.
- GgMatrix gg::ggRotateY (GLfloat a)
  y **軸中心の回転の変換行列を返す**.
- GgMatrix gg::ggRotateZ (GLfloat a)
  z **軸中心の回転の変換行列を返す**.
- GgMatrix gg::ggRotate (GLfloat x, GLfloat y, GLfloat z, GLfloat a)
  (x, y, z) **方向のベクトルを軸とする回転の変換行列を乗じた結果を返す**.
- GgMatrix gg::ggRotate (const GLfloat *r, GLfloat a)
  r **方向のベクトルを軸とする回転の変換行列を乗じた結果を返す**.
- GgMatrix gg::ggRotate (const GgVector &r, GLfloat a)
  r **方向のベクトルを軸とする回転の変換行列を乗じた結果を返す**.
- GgMatrix gg::ggRotate (const GLfloat *r)
  r **方向のベクトルを軸とする回転の変換行列を乗じた結果を返す**.
- GgMatrix gg::ggRotate (const GgVector &r)
  r **方向のベクトルを軸とする回転の変換行列を乗じた結果を返す**.
- GgMatrix gg::ggLookat (GLfloat ex, GLfloat ey, GLfloat ez, GLfloat tx, GLfloat ty, GLfloat tz, GLfloat ux, GLfloat uy, GLfloat uz)
  ビュー変換行列を返す.
- GgMatrix gg::ggLookat (const GLfloat *e, const GLfloat *t, const GLfloat *u)
  ビュー変換行列を返す.
- GgMatrix gg::ggLookat (const GgVector &e, const GgVector &t, const GgVector &u)
  ビュー変換行列を返す.
- GgMatrix gg::ggOrthogonal (GLfloat left, GLfloat right, GLfloat bottom, GLfloat top, GLfloat zNear, GLfloat zFar)
  直交投影変換行列を返す.
- GgMatrix gg::ggFrustum (GLfloat left, GLfloat right, GLfloat bottom, GLfloat top, GLfloat zNear, GLfloat zFar)
  透視透視投影変換行列を返す.
- GgMatrix gg::ggPerspective (GLfloat fovy, GLfloat aspect, GLfloat zNear, GLfloat zFar)
  画角を指定して透視投影変換行列を返す.
- GgMatrix gg::ggTranspose (const GgMatrix &m)
  転置行列を返す.
- GgMatrix gg::ggInvert (const GgMatrix &m)
  逆行列を返す.
- GgMatrix gg::ggNormal (const GgMatrix &m)
  法線変換行列を返す.
- GgQuaternion gg::ggQuaternion (GLfloat x, GLfloat y, GLfloat z, GLfloat w)
  四元数を返す
- GgQuaternion gg::ggQuaternion (const GLfloat *a)
  四元数を返す
- GgQuaternion gg::ggIdentityQuaternion ()
  単位四元数を返す
- GgQuaternion gg::ggMatrixQuaternion (const GLfloat *a)

回転の変換行列 *m* を表す四元数を返す.

- GgQuaternion gg::ggMatrixQuaternion (const GgMatrix &m)

  回転の変換行列 *m* を表す四元数を返す.
- GgMatrix gg::ggQuaternionMatrix (const GgQuaternion &q)

  四元数 *q* の回転の変換行列を返す.
- GgMatrix gg::ggQuaternionTransposeMatrix (const GgQuaternion &q)

  四元数 *q* の回転の転置した変換行列を返す.
- GgQuaternion gg::ggRotateQuaternion (GLfloat x, GLfloat y, GLfloat z, GLfloat a)

  ***(x, y, z)* を軸として角度 *a* 回転する四元数を返す**.
- GgQuaternion gg::ggRotateQuaternion (const GLfloat *v, GLfloat a)

  ***(v[0], v[1], v[2])* を軸として角度 *a* 回転する四元数を返す**.
- GgQuaternion gg::ggRotateQuaternion (const GLfloat *v)

  ***(v[0], v[1], v[2])* を軸として角度 *v[3]* 回転する四元数を返す**.
- GgQuaternion gg::ggEulerQuaternion (GLfloat heading, GLfloat pitch, GLfloat roll)

  オイラー角 *(heading, pitch, roll)* で与えられた回転を表す四元数を返す.
- GgQuaternion gg::ggEulerQuaternion (const GLfloat *e)

  オイラー角 *(e[0], e[1], e[2])* で与えられた回転を表す四元数を返す.
- GgQuaternion gg::ggSlerp (const GLfloat *a, const GLfloat *b, GLfloat t)

  二つの四元数の球面線形補間の結果を返す.
- GgQuaternion gg::ggSlerp (const GgQuaternion &q, const GgQuaternion &r, GLfloat t)

  二つの四元数の球面線形補間の結果を返す.
- GgQuaternion gg::ggSlerp (const GgQuaternion &q, const GLfloat *a, GLfloat t)

  二つの四元数の球面線形補間の結果を返す.
- GgQuaternion gg::ggSlerp (const GLfloat *a, const GgQuaternion &q, GLfloat t)

  二つの四元数の球面線形補間の結果を返す.
- GLfloat gg::ggNorm (const GgQuaternion &q)

  四元数のノルムを返す.
- GgQuaternion gg::ggNormalize (const GgQuaternion &q)

  正規化した四元数を返す.
- GgQuaternion gg::ggConjugate (const GgQuaternion &q)

  共役四元数を返す.
- GgQuaternion gg::ggInvert (const GgQuaternion &q)

  四元数の逆元を求める.
- GgPoints * gg::ggPointsCube (GLsizei countv, GLfloat length=1.0f, GLfloat cx=0.0f, GLfloat cy=0.0f, GLfloat cz=0.0f)

  点群を立方体状に生成する.
- GgPoints * gg::ggPointsSphere (GLsizei countv, GLfloat radius=0.5f, GLfloat cx=0.0f, GLfloat cy=0.0f, GLfloat cz=0.0f)

  点群を球状に生成する.
- GgTriangles * gg::ggRectangle (GLfloat width=1.0f, GLfloat height=1.0f)

  矩形状に *2* 枚の三角形を生成する.
- GgTriangles * gg::ggEllipse (GLfloat width=1.0f, GLfloat height=1.0f, GLuint slices=16)

  楕円状に三角形を生成する.
- GgTriangles * gg::ggArraysObj (const char *name, bool normalize=false)

  Wavefront OBJ **ファイルを読み込む** *(Arrays* **形式***)*
- GgElements * gg::ggElementsObj (const char *name, bool normalize=false)

  Wavefront OBJ **ファイル を読み込む** *(Elements* **形式***)*.
- GgElements * gg::ggElementsMesh (GLuint slices, GLuint stacks, const GLfloat(*pos)[3], const GLfloat(*norm)[3]=nullptr)

  メッシュ形状を作成する *(Elements* **形式***)*.
- GgElements * gg::ggElementsSphere (GLfloat radius=1.0f, int slices=16, int stacks=8)

- bool gg::ggLoadSimpleObj (const char *name, std::vector< std::array< GLuint, 3 > > &group, std::vector< GgSimpleShader::Material > &material, std::vector< GgVertex > &vert, bool normalize=false)

  三角形分割された *OBJ* **ファイルと** *MTL* **ファイルを読み込む** *(Arrays* **形式***)*
- bool gg::ggLoadSimpleObj (const char *name, std::vector< std::array< GLuint, 3 > > &group, std::vector< GgSimpleShader::Material > &material, std::vector< GgVertex > &vert, std::vector< GLuint > &face, bool normalize=false)

  三角形分割された *OBJ* **ファイルを読み込む** *(Elements* **形式***).*

変数

- GLint gg::ggBufferAlignment

  使用している *GPU* **のバッファオブジェクトのアライメント**

7.2.1 マクロ定義詳解

7.2.1.1 #define ggError( )

gg.h の 1324 行目に定義があります。

7.2.1.2 #define ggFBOError( )

gg.h の 1338 行目に定義があります。

Index