ゲームグラフィックス特論

構築: Doxygen 1.8.17

1 ゲームグラフィックス特論の宿題用補助プログラム **GLFW3** 版 **1**

2 名前空間索引 **3**

2.1 名前空間一覧 3

3 階層索引 **5**

3.1 クラス階層 5

4 クラス索引 **7**

4.1 クラス一覧 7

5 ファイル索引 **9**

5.1 ファイル一覧 9

6 名前空間詳解 **11**

6.1 gg 名前空間 11

6.1.1 詳解 15

6.1.2 型定義詳解 16

6.1.2.1 GgVector 16

6.1.3 列挙型詳解 16

6.1.3.1 BindingPoints 16

6.1.4 関数詳解 16

6.1.4.1 _ggError() 16

6.1.4.2 _ggFBOError() 17

6.1.4.3 ggArraysObj() 17

6.1.4.4 ggConjugate() 18

6.1.4.5 ggCreateComputeShader() 18

6.1.4.6 ggCreateNormalMap() 19

6.1.4.7 ggCreateShader() 20

6.1.4.8 ggCross() 20

6.1.4.9 ggDot3() 21

6.1.4.10 ggDot4() [1/2] 21

6.1.4.11 ggDot4() [2/2] 21

6.1.4.12 ggElementsMesh() 22

6.1.4.13 ggElementsObj() 23

6.1.4.14 ggElementsSphere() 23

6.1.4.15 ggEllipse() 24

6.1.4.16 ggEulerQuaternion() [1/2] 24

6.1.4.17 ggEulerQuaternion() [2/2] 25

6.1.4.18 ggFrustum() 26

6.1.4.19 ggIdentity() 26

6.1.4.20 ggIdentityQuaternion() 27

6.1.4.21 ggInit() 27

6.1.4.22 ggInvert() [1/2] 27

6.1.4.23 ggInvert() [2/2] . . . 28
6.1.4.24 ggLength3() . . . 28
6.1.4.25 ggLength4() [1/2] . . . 29
6.1.4.26 ggLength4() [2/2] . . . 29
6.1.4.27 ggLoadComputeShader() . . . 30
6.1.4.28 ggLoadHeight() . . . 30
6.1.4.29 ggLoadImage() . . . 31
6.1.4.30 ggLoadShader() . . . 32
6.1.4.31 ggLoadSimpleObj() [1/2] . . . 32
6.1.4.32 ggLoadSimpleObj() [2/2] . . . 33
6.1.4.33 ggLoadTexture() . . . 34
6.1.4.34 ggLookat() [1/3] . . . 35
6.1.4.35 ggLookat() [2/3] . . . 35
6.1.4.36 ggLookat() [3/3] . . . 36
6.1.4.37 ggMatrixQuaternion() [1/2] . . . 37
6.1.4.38 ggMatrixQuaternion() [2/2] . . . 37
6.1.4.39 ggNorm() . . . 38
6.1.4.40 ggNormal() . . . 38
6.1.4.41 ggNormalize() . . . 39
6.1.4.42 ggNormalize3() . . . 39
6.1.4.43 ggNormalize4() [1/2] . . . 40
6.1.4.44 ggNormalize4() [2/2] . . . 40
6.1.4.45 ggOrthogonal() . . . 41
6.1.4.46 ggPerspective() . . . 42
6.1.4.47 ggPointsCube() . . . 42
6.1.4.48 ggPointsSphere() . . . 43
6.1.4.49 ggQuaternion() [1/2] . . . 43
6.1.4.50 ggQuaternion() [2/2] . . . 44
6.1.4.51 ggQuaternionMatrix() . . . 45
6.1.4.52 ggQuaternionTransposeMatrix() . . . 45
6.1.4.53 ggReadImage() . . . 46
6.1.4.54 ggRectangle() . . . 47
6.1.4.55 ggRotate() [1/5] . . . 47
6.1.4.56 ggRotate() [2/5] . . . 48
6.1.4.57 ggRotate() [3/5] . . . 48
6.1.4.58 ggRotate() [4/5] . . . 49
6.1.4.59 ggRotate() [5/5] . . . 50
6.1.4.60 ggRotateQuaternion() [1/3] . . . 50
6.1.4.61 ggRotateQuaternion() [2/3] . . . 51
6.1.4.62 ggRotateQuaternion() [3/3] . . . 51
6.1.4.63 ggRotateX() . . . 52
6.1.4.64 ggRotateY() . . . 53

6.1.4.65 ggRotateZ() . . . 53
6.1.4.66 ggSaveColor() . . . 54
6.1.4.67 ggSaveDepth() . . . 54
6.1.4.68 ggSaveTga() . . . 55
6.1.4.69 ggScale() [1/3] . . . 55
6.1.4.70 ggScale() [2/3] . . . 56
6.1.4.71 ggScale() [3/3] . . . 56
6.1.4.72 ggSlerp() [1/4] . . . 57
6.1.4.73 ggSlerp() [2/4] . . . 58
6.1.4.74 ggSlerp() [3/4] . . . 58
6.1.4.75 ggSlerp() [4/4] . . . 59
6.1.4.76 ggTranslate() [1/3] . . . 60
6.1.4.77 ggTranslate() [2/3] . . . 60
6.1.4.78 ggTranslate() [3/3] . . . 61
6.1.4.79 ggTranspose() . . . 61
6.1.5 変数詳解 . . . 62
6.1.5.1 ggBufferAlignment . . . 62

7 クラス詳解 **63**
7.1 gg::GgBuffer< T > クラステンプレート . . . 63
7.1.1 詳解 . . . 64
7.1.2 構築子と解体子 . . . 64
7.1.2.1 GgBuffer() [1/2] . . . 64
7.1.2.2 ~GgBuffer() . . . 65
7.1.2.3 GgBuffer() [2/2] . . . 65
7.1.3 関数詳解 . . . 65
7.1.3.1 bind() . . . 65
7.1.3.2 copy() . . . 65
7.1.3.3 getBuffer() . . . 66
7.1.3.4 getCount() . . . 66
7.1.3.5 getStride() . . . 67
7.1.3.6 getTarget() . . . 67
7.1.3.7 map() [1/2] . . . 68
7.1.3.8 map() [2/2] . . . 68
7.1.3.9 operator=() . . . 69
7.1.3.10 read() . . . 69
7.1.3.11 send() . . . 69
7.1.3.12 unbind() . . . 70
7.1.3.13 unmap() . . . 70
7.2 gg::GgColorTexture クラス . . . 70
7.2.1 詳解 . . . 71
7.2.2 構築子と解体子 . . . 71

7.2.2.1 GgColorTexture() [1/3] . . . 71
7.2.2.2 GgColorTexture() [2/3] . . . 71
7.2.2.3 GgColorTexture() [3/3] . . . 72
7.2.2.4 ~GgColorTexture() . . . 73
7.2.3 関数詳解 . . . 73
7.2.3.1 load() [1/2] . . . 73
7.2.3.2 load() [2/2] . . . 73
7.3 gg::GgElements クラス . . . 75
7.3.1 詳解 . . . 76
7.3.2 構築子と解体子 . . . 76
7.3.2.1 GgElements() [1/2] . . . 77
7.3.2.2 GgElements() [2/2] . . . 77
7.3.2.3 ~GgElements() . . . 77
7.3.3 関数詳解 . . . 78
7.3.3.1 draw() . . . 78
7.3.3.2 getIndexBuffer() . . . 78
7.3.3.3 getIndexCount() . . . 79
7.3.3.4 load() . . . 79
7.3.3.5 send() . . . 80
7.4 gg::GgMatrix クラス . . . 80
7.4.1 詳解 . . . 84
7.4.2 構築子と解体子 . . . 84
7.4.2.1 GgMatrix() [1/3] . . . 84
7.4.2.2 GgMatrix() [2/3] . . . 85
7.4.2.3 GgMatrix() [3/3] . . . 86
7.4.2.4 ~GgMatrix() . . . 86
7.4.3 関数詳解 . . . 86
7.4.3.1 add() [1/2] . . . 87
7.4.3.2 add() [2/2] . . . 87
7.4.3.3 divide() [1/2] . . . 88
7.4.3.4 divide() [2/2] . . . 88
7.4.3.5 frustum() . . . 89
7.4.3.6 get() [1/3] . . . 90
7.4.3.7 get() [2/3] . . . 91
7.4.3.8 get() [3/3] . . . 92
7.4.3.9 invert() . . . 92
7.4.3.10 load() [1/2] . . . 92
7.4.3.11 load() [2/2] . . . 93
7.4.3.12 loadAdd() [1/2] . . . 94
7.4.3.13 loadAdd() [2/2] . . . 94
7.4.3.14 loadDivide() [1/2] . . . 95
7.4.3.15 loadDivide() [2/2] . . . 95

7.4.3.16 loadFrustum() . . . 96
7.4.3.17 loadIdentity() . . . 97
7.4.3.18 loadInvert() [1/2] . . . 97
7.4.3.19 loadInvert() [2/2] . . . 98
7.4.3.20 loadLookat() [1/3] . . . 98
7.4.3.21 loadLookat() [2/3] . . . 99
7.4.3.22 loadLookat() [3/3] . . . 99
7.4.3.23 loadMultiply() [1/2] . . . 100
7.4.3.24 loadMultiply() [2/2] . . . 101
7.4.3.25 loadNormal() [1/2] . . . 101
7.4.3.26 loadNormal() [2/2] . . . 102
7.4.3.27 loadOrthogonal() . . . 103
7.4.3.28 loadPerspective() . . . 103
7.4.3.29 loadRotate() [1/5] . . . 104
7.4.3.30 loadRotate() [2/5] . . . 105
7.4.3.31 loadRotate() [3/5] . . . 105
7.4.3.32 loadRotate() [4/5] . . . 106
7.4.3.33 loadRotate() [5/5] . . . 106
7.4.3.34 loadRotateX() . . . 107
7.4.3.35 loadRotateY() . . . 107
7.4.3.36 loadRotateZ() . . . 108
7.4.3.37 loadScale() [1/3] . . . 109
7.4.3.38 loadScale() [2/3] . . . 109
7.4.3.39 loadScale() [3/3] . . . 110
7.4.3.40 loadSubtract() [1/2] . . . 110
7.4.3.41 loadSubtract() [2/2] . . . 111
7.4.3.42 loadTranslate() [1/3] . . . 111
7.4.3.43 loadTranslate() [2/3] . . . 112
7.4.3.44 loadTranslate() [3/3] . . . 113
7.4.3.45 loadTranspose() [1/2] . . . 114
7.4.3.46 loadTranspose() [2/2] . . . 115
7.4.3.47 lookat() [1/3] . . . 115
7.4.3.48 lookat() [2/3] . . . 116
7.4.3.49 lookat() [3/3] . . . 116
7.4.3.50 multiply() [1/2] . . . 117
7.4.3.51 multiply() [2/2] . . . 118
7.4.3.52 normal() . . . 118
7.4.3.53 operator∗() [1/3] . . . 119
7.4.3.54 operator∗() [2/3] . . . 119
7.4.3.55 operator∗() [3/3] . . . 119
7.4.3.56 operator∗=() [1/2] . . . 120
7.4.3.57 operator∗=() [2/2] . . . 120

7.4.3.58 operator+() [1/2] . . . 120
7.4.3.59 operator+() [2/2] . . . 121
7.4.3.60 operator+=() [1/2] . . . 121
7.4.3.61 operator+=() [2/2] . . . 121
7.4.3.62 operator-() [1/2] . . . 122
7.4.3.63 operator-() [2/2] . . . 122
7.4.3.64 operator-=() [1/2] . . . 122
7.4.3.65 operator-=() [2/2] . . . 123
7.4.3.66 operator/() [1/2] . . . 123
7.4.3.67 operator/() [2/2] . . . 123
7.4.3.68 operator/=() [1/2] . . . 124
7.4.3.69 operator/=() [2/2] . . . 124
7.4.3.70 operator=() [1/2] . . . 124
7.4.3.71 operator=() [2/2] . . . 125
7.4.3.72 operator[]() [1/2] . . . 125
7.4.3.73 operator[]() [2/2] . . . 125
7.4.3.74 orthogonal() . . . 126
7.4.3.75 perspective() . . . 127
7.4.3.76 projection() [1/4] . . . 128
7.4.3.77 projection() [2/4] . . . 128
7.4.3.78 projection() [3/4] . . . 128
7.4.3.79 projection() [4/4] . . . 129
7.4.3.80 rotate() [1/5] . . . 129
7.4.3.81 rotate() [2/5] . . . 130
7.4.3.82 rotate() [3/5] . . . 130
7.4.3.83 rotate() [4/5] . . . 131
7.4.3.84 rotate() [5/5] . . . 131
7.4.3.85 rotateX() . . . 132
7.4.3.86 rotateY() . . . 132
7.4.3.87 rotateZ() . . . 133
7.4.3.88 scale() [1/3] . . . 133
7.4.3.89 scale() [2/3] . . . 134
7.4.3.90 scale() [3/3] . . . 134
7.4.3.91 subtract() [1/2] . . . 135
7.4.3.92 subtract() [2/2] . . . 136
7.4.3.93 translate() [1/3] . . . 136
7.4.3.94 translate() [2/3] . . . 137
7.4.3.95 translate() [3/3] . . . 137
7.4.3.96 transpose() . . . 138
7.4.4 フレンドと関連関数の詳解 . . . 139
7.4.4.1 GgQuaternion . . . 139
7.5 gg::GgNormalTexture クラス . . . 139

7.5.1 詳解 . . . 139
7.5.2 構築子と解体子 . . . 140
7.5.2.1 GgNormalTexture() [1/3] . . . 140
7.5.2.2 GgNormalTexture() [2/3] . . . 140
7.5.2.3 GgNormalTexture() [3/3] . . . 140
7.5.2.4 ~GgNormalTexture() . . . 141
7.5.3 関数詳解 . . . 141
7.5.3.1 load() [1/2] . . . 141
7.5.3.2 load() [2/2] . . . 142
7.6 gg::GgPoints クラス . . . 143
7.6.1 詳解 . . . 144
7.6.2 構築子と解体子 . . . 144
7.6.2.1 GgPoints() [1/2] . . . 144
7.6.2.2 GgPoints() [2/2] . . . 144
7.6.2.3 ~GgPoints() . . . 145
7.6.3 関数詳解 . . . 145
7.6.3.1 draw() . . . 145
7.6.3.2 getBuffer() . . . 145
7.6.3.3 getCount() . . . 146
7.6.3.4 load() . . . 146
7.6.3.5 send() . . . 146
7.7 gg::GgPointShader クラス . . . 147
7.7.1 詳解 . . . 148
7.7.2 構築子と解体子 . . . 148
7.7.2.1 GgPointShader() [1/2] . . . 148
7.7.2.2 GgPointShader() [2/2] . . . 148
7.7.2.3 ~GgPointShader() . . . 149
7.7.3 関数詳解 . . . 149
7.7.3.1 get() . . . 149
7.7.3.2 loadMatrix() [1/2] . . . 150
7.7.3.3 loadMatrix() [2/2] . . . 150
7.7.3.4 loadModelviewMatrix() [1/2] . . . 151
7.7.3.5 loadModelviewMatrix() [2/2] . . . 152
7.7.3.6 loadProjectionMatrix() [1/2] . . . 152
7.7.3.7 loadProjectionMatrix() [2/2] . . . 153
7.7.3.8 unuse() . . . 153
7.7.3.9 use() [1/5] . . . 154
7.7.3.10 use() [2/5] . . . 154
7.7.3.11 use() [3/5] . . . 154
7.7.3.12 use() [4/5] . . . 155
7.7.3.13 use() [5/5] . . . 155
7.8 gg::GgQuaternion クラス . . . 156

7.8.1 詳解 . . . 160
7.8.2 構築子と解体子 . . . 160
7.8.2.1 GgQuaternion() [1/5] . . . 160
7.8.2.2 GgQuaternion() [2/5] . . . 160
7.8.2.3 GgQuaternion() [3/5] . . . 161
7.8.2.4 GgQuaternion() [4/5] . . . 161
7.8.2.5 GgQuaternion() [5/5] . . . 162
7.8.2.6 ~GgQuaternion() . . . 162
7.8.3 関数詳解 . . . 162
7.8.3.1 add() [1/4] . . . 163
7.8.3.2 add() [2/4] . . . 163
7.8.3.3 add() [3/4] . . . 164
7.8.3.4 add() [4/4] . . . 164
7.8.3.5 conjugate() . . . 165
7.8.3.6 divide() [1/4] . . . 165
7.8.3.7 divide() [2/4] . . . 166
7.8.3.8 divide() [3/4] . . . 166
7.8.3.9 divide() [4/4] . . . 167
7.8.3.10 euler() [1/2] . . . 168
7.8.3.11 euler() [2/2] . . . 168
7.8.3.12 get() [1/2] . . . 169
7.8.3.13 get() [2/2] . . . 169
7.8.3.14 getConjugateMatrix() [1/3] . . . 170
7.8.3.15 getConjugateMatrix() [2/3] . . . 170
7.8.3.16 getConjugateMatrix() [3/3] . . . 171
7.8.3.17 getMatrix() [1/3] . . . 171
7.8.3.18 getMatrix() [2/3] . . . 171
7.8.3.19 getMatrix() [3/3] . . . 172
7.8.3.20 invert() . . . 172
7.8.3.21 load() [1/4] . . . 173
7.8.3.22 load() [2/4] . . . 174
7.8.3.23 load() [3/4] . . . 175
7.8.3.24 load() [4/4] . . . 175
7.8.3.25 loadAdd() [1/4] . . . 176
7.8.3.26 loadAdd() [2/4] . . . 177
7.8.3.27 loadAdd() [3/4] . . . 177
7.8.3.28 loadAdd() [4/4] . . . 178
7.8.3.29 loadConjugate() [1/2] . . . 178
7.8.3.30 loadConjugate() [2/2] . . . 179
7.8.3.31 loadDivide() [1/4] . . . 180
7.8.3.32 loadDivide() [2/4] . . . 180
7.8.3.33 loadDivide() [3/4] . . . 181

7.8.3.34 loadDivide() [4/4] . . . 181
7.8.3.35 loadEuler() [1/2] . . . 182
7.8.3.36 loadEuler() [2/2] . . . 182
7.8.3.37 loadIdentity() . . . 183
7.8.3.38 loadInvert() [1/2] . . . 184
7.8.3.39 loadInvert() [2/2] . . . 185
7.8.3.40 loadMatrix() [1/2] . . . 186
7.8.3.41 loadMatrix() [2/2] . . . 187
7.8.3.42 loadMultiply() [1/4] . . . 187
7.8.3.43 loadMultiply() [2/4] . . . 188
7.8.3.44 loadMultiply() [3/4] . . . 188
7.8.3.45 loadMultiply() [4/4] . . . 189
7.8.3.46 loadNormalize() [1/2] . . . 189
7.8.3.47 loadNormalize() [2/2] . . . 190
7.8.3.48 loadRotate() [1/3] . . . 191
7.8.3.49 loadRotate() [2/3] . . . 191
7.8.3.50 loadRotate() [3/3] . . . 192
7.8.3.51 loadRotateX() . . . 192
7.8.3.52 loadRotateY() . . . 193
7.8.3.53 loadRotateZ() . . . 193
7.8.3.54 loadSlerp() [1/4] . . . 194
7.8.3.55 loadSlerp() [2/4] . . . 195
7.8.3.56 loadSlerp() [3/4] . . . 195
7.8.3.57 loadSlerp() [4/4] . . . 196
7.8.3.58 loadSubtract() [1/4] . . . 196
7.8.3.59 loadSubtract() [2/4] . . . 197
7.8.3.60 loadSubtract() [3/4] . . . 197
7.8.3.61 loadSubtract() [4/4] . . . 198
7.8.3.62 multiply() [1/4] . . . 199
7.8.3.63 multiply() [2/4] . . . 199
7.8.3.64 multiply() [3/4] . . . 199
7.8.3.65 multiply() [4/4] . . . 200
7.8.3.66 norm() . . . 200
7.8.3.67 normalize() . . . 201
7.8.3.68 operator∗() [1/2] . . . 201
7.8.3.69 operator∗() [2/2] . . . 202
7.8.3.70 operator∗=() [1/2] . . . 202
7.8.3.71 operator∗=() [2/2] . . . 202
7.8.3.72 operator+() [1/2] . . . 203
7.8.3.73 operator+() [2/2] . . . 203
7.8.3.74 operator+=() [1/2] . . . 203
7.8.3.75 operator+=() [2/2] . . . 204

7.8.3.76 operator-() [1/2] . . . 204
7.8.3.77 operator-() [2/2] . . . 204
7.8.3.78 operator-=() [1/2] . . . 205
7.8.3.79 operator-=() [2/2] . . . 205
7.8.3.80 operator/() [1/2] . . . 205
7.8.3.81 operator/() [2/2] . . . 206
7.8.3.82 operator/=() [1/2] . . . 206
7.8.3.83 operator/=() [2/2] . . . 206
7.8.3.84 operator=() [1/2] . . . 207
7.8.3.85 operator=() [2/2] . . . 207
7.8.3.86 rotate() [1/3] . . . 207
7.8.3.87 rotate() [2/3] . . . 208
7.8.3.88 rotate() [3/3] . . . 208
7.8.3.89 rotateX() . . . 209
7.8.3.90 rotateY() . . . 210
7.8.3.91 rotateZ() . . . 210
7.8.3.92 slerp() [1/2] . . . 211
7.8.3.93 slerp() [2/2] . . . 211
7.8.3.94 subtract() [1/4] . . . 212
7.8.3.95 subtract() [2/4] . . . 212
7.8.3.96 subtract() [3/4] . . . 213
7.8.3.97 subtract() [4/4] . . . 213
7.9 gg::GgShader クラス . . . 214
7.9.1 詳解 . . . 214
7.9.2 構築子と解体子 . . . 215
7.9.2.1 GgShader() [1/2] . . . 215
7.9.2.2 ~GgShader() . . . 215
7.9.2.3 GgShader() [2/2] . . . 215
7.9.3 関数詳解 . . . 215
7.9.3.1 get() . . . 216
7.9.3.2 operator=() . . . 216
7.9.3.3 unuse() . . . 216
7.9.3.4 use() . . . 216
7.10 gg::GgShape クラス . . . 217
7.10.1 詳解 . . . 217
7.10.2 構築子と解体子 . . . 218
7.10.2.1 GgShape() [1/2] . . . 218
7.10.2.2 ~GgShape() . . . 218
7.10.2.3 GgShape() [2/2] . . . 218
7.10.3 関数詳解 . . . 218
7.10.3.1 draw() . . . 218
7.10.3.2 get() . . . 219

7.10.3.3 getMode() . . . 220
7.10.3.4 operator=() . . . 220
7.10.3.5 setMode() . . . 220
7.11 gg::GgSimpleObj クラス . . . 220
7.11.1 詳解 . . . 221
7.11.2 構築子と解体子 . . . 221
7.11.2.1 GgSimpleObj() . . . 221
7.11.2.2 ~GgSimpleObj() . . . 221
7.11.3 関数詳解 . . . 222
7.11.3.1 draw() . . . 222
7.11.3.2 get() . . . 222
7.12 gg::GgSimpleShader クラス . . . 223
7.12.1 詳解 . . . 225
7.12.2 構築子と解体子 . . . 225
7.12.2.1 GgSimpleShader() [1/3] . . . 225
7.12.2.2 GgSimpleShader() [2/3] . . . 225
7.12.2.3 GgSimpleShader() [3/3] . . . 226
7.12.2.4 ~GgSimpleShader() . . . 226
7.12.3 関数詳解 . . . 226
7.12.3.1 loadMatrix() [1/4] . . . 226
7.12.3.2 loadMatrix() [2/4] . . . 227
7.12.3.3 loadMatrix() [3/4] . . . 227
7.12.3.4 loadMatrix() [4/4] . . . 228
7.12.3.5 loadModelviewMatrix() [1/4] . . . 229
7.12.3.6 loadModelviewMatrix() [2/4] . . . 229
7.12.3.7 loadModelviewMatrix() [3/4] . . . 230
7.12.3.8 loadModelviewMatrix() [4/4] . . . 230
7.12.3.9 operator=() . . . 231
7.12.3.10 use() [1/13] . . . 231
7.12.3.11 use() [2/13] . . . 232
7.12.3.12 use() [3/13] . . . 233
7.12.3.13 use() [4/13] . . . 233
7.12.3.14 use() [5/13] . . . 234
7.12.3.15 use() [6/13] . . . 235
7.12.3.16 use() [7/13] . . . 235
7.12.3.17 use() [8/13] . . . 236
7.12.3.18 use() [9/13] . . . 236
7.12.3.19 use() [10/13] . . . 237
7.12.3.20 use() [11/13] . . . 238
7.12.3.21 use() [12/13] . . . 238
7.12.3.22 use() [13/13] . . . 239
7.13 gg::GgTexture クラス . . . 239

7.13.1 詳解 . . . 240
7.13.2 構築子と解体子 . . . 240
7.13.2.1 GgTexture() [1/2] . . . 240
7.13.2.2 ~GgTexture() . . . 241
7.13.2.3 GgTexture() [2/2] . . . 241
7.13.3 関数詳解 . . . 241
7.13.3.1 bind() . . . 241
7.13.3.2 getHeight() . . . 241
7.13.3.3 getSize() [1/2] . . . 242
7.13.3.4 getSize() [2/2] . . . 242
7.13.3.5 getTexture() . . . 242
7.13.3.6 getWidth() . . . 243
7.13.3.7 operator=() . . . 243
7.13.3.8 unbind() . . . 243
7.14 gg::GgTrackball クラス . . . 243
7.14.1 詳解 . . . 244
7.14.2 構築子と解体子 . . . 244
7.14.2.1 GgTrackball() . . . 245
7.14.2.2 ~GgTrackball() . . . 245
7.14.3 関数詳解 . . . 245
7.14.3.1 begin() . . . 245
7.14.3.2 end() . . . 246
7.14.3.3 get() . . . 246
7.14.3.4 getMatrix() . . . 246
7.14.3.5 getQuaternion() . . . 247
7.14.3.6 getScale() [1/3] . . . 247
7.14.3.7 getScale() [2/3] . . . 247
7.14.3.8 getScale() [3/3] . . . 247
7.14.3.9 getStart() [1/3] . . . 248
7.14.3.10 getStart() [2/3] . . . 248
7.14.3.11 getStart() [3/3] . . . 248
7.14.3.12 motion() . . . 249
7.14.3.13 region() [1/2] . . . 249
7.14.3.14 region() [2/2] . . . 250
7.14.3.15 reset() . . . 250
7.14.3.16 rotate() . . . 251
7.15 gg::GgTriangles クラス . . . 251
7.15.1 詳解 . . . 252
7.15.2 構築子と解体子 . . . 252
7.15.2.1 GgTriangles() [1/2] . . . 252
7.15.2.2 GgTriangles() [2/2] . . . 253
7.15.2.3 ~GgTriangles() . . . 253

7.15.3 関数詳解 . . . 253
7.15.3.1 draw() . . . 254
7.15.3.2 getBuffer() . . . 254
7.15.3.3 getCount() . . . 254
7.15.3.4 load() . . . 255
7.15.3.5 send() . . . 255
7.16 gg::GgUniformBuffer< T > クラステンプレート . . . 256
7.16.1 詳解 . . . 257
7.16.2 構築子と解体子 . . . 257
7.16.2.1 GgUniformBuffer() [1/3] . . . 257
7.16.2.2 GgUniformBuffer() [2/3] . . . 257
7.16.2.3 GgUniformBuffer() [3/3] . . . 257
7.16.2.4 ~GgUniformBuffer() . . . 258
7.16.3 関数詳解 . . . 258
7.16.3.1 bind() . . . 258
7.16.3.2 copy() . . . 259
7.16.3.3 fill() . . . 259
7.16.3.4 getBuffer() . . . 260
7.16.3.5 getCount() . . . 260
7.16.3.6 getStride() . . . 261
7.16.3.7 getTarget() . . . 261
7.16.3.8 load() [1/2] . . . 262
7.16.3.9 load() [2/2] . . . 262
7.16.3.10 map() [1/2] . . . 264
7.16.3.11 map() [2/2] . . . 264
7.16.3.12 read() . . . 265
7.16.3.13 send() . . . 265
7.16.3.14 unbind() . . . 266
7.16.3.15 unmap() . . . 266
7.17 gg::GgVertex 構造体 . . . 266
7.17.1 詳解 . . . 267
7.17.2 構築子と解体子 . . . 267
7.17.2.1 GgVertex() [1/4] . . . 267
7.17.2.2 GgVertex() [2/4] . . . 267
7.17.2.3 GgVertex() [3/4] . . . 267
7.17.2.4 GgVertex() [4/4] . . . 268
7.17.3 メンバ詳解 . . . 268
7.17.3.1 normal . . . 268
7.17.3.2 position . . . 269
7.18 gg::GgSimpleShader::Light 構造体 . . . 269
7.18.1 詳解 . . . 269
7.18.2 メンバ詳解 . . . 269

7.18.2.1 ambient . . . 269
7.18.2.2 diffuse . . . 270
7.18.2.3 position . . . 270
7.18.2.4 specular . . . 270
7.19 gg::GgSimpleShader::LightBuffer クラス . . . 270
7.19.1 詳解 . . . 272
7.19.2 構築子と解体子 . . . 272
7.19.2.1 LightBuffer() [1/2] . . . 272
7.19.2.2 LightBuffer() [2/2] . . . 272
7.19.2.3 ~LightBuffer() . . . 273
7.19.3 関数詳解 . . . 273
7.19.3.1 load() [1/2] . . . 273
7.19.3.2 load() [2/2] . . . 273
7.19.3.3 loadAmbient() [1/2] . . . 274
7.19.3.4 loadAmbient() [2/2] . . . 275
7.19.3.5 loadColor() . . . 275
7.19.3.6 loadDiffuse() [1/2] . . . 275
7.19.3.7 loadDiffuse() [2/2] . . . 277
7.19.3.8 loadPosition() [1/4] . . . 277
7.19.3.9 loadPosition() [2/4] . . . 278
7.19.3.10 loadPosition() [3/4] . . . 278
7.19.3.11 loadPosition() [4/4] . . . 279
7.19.3.12 loadSpecular() [1/2] . . . 279
7.19.3.13 loadSpecular() [2/2] . . . 280
7.19.3.14 select() . . . 280
7.20 gg::GgSimpleShader::Material 構造体 . . . 281
7.20.1 詳解 . . . 282
7.20.2 メンバ詳解 . . . 282
7.20.2.1 ambient . . . 282
7.20.2.2 diffuse . . . 282
7.20.2.3 shininess . . . 282
7.20.2.4 specular . . . 282
7.21 gg::GgSimpleShader::MaterialBuffer クラス . . . 283
7.21.1 詳解 . . . 284
7.21.2 構築子と解体子 . . . 284
7.21.2.1 MaterialBuffer() [1/2] . . . 284
7.21.2.2 MaterialBuffer() [2/2] . . . 285
7.21.2.3 ~MaterialBuffer() . . . 285
7.21.3 関数詳解 . . . 285
7.21.3.1 load() [1/2] . . . 285
7.21.3.2 load() [2/2] . . . 286
7.21.3.3 loadAmbient() [1/2] . . . 286

7.21.3.4 loadAmbient() [2/2] . . . 287
7.21.3.5 loadAmbientAndDiffuse() [1/2] . . . 287
7.21.3.6 loadAmbientAndDiffuse() [2/2] . . . 288
7.21.3.7 loadDiffuse() [1/2] . . . 288
7.21.3.8 loadDiffuse() [2/2] . . . 289
7.21.3.9 loadShininess() [1/2] . . . 289
7.21.3.10 loadShininess() [2/2] . . . 290
7.21.3.11 loadSpecular() [1/2] . . . 290
7.21.3.12 loadSpecular() [2/2] . . . 291
7.21.3.13 select() . . . 291
7.22 Window クラス . . . 292
7.22.1 詳解 . . . 294
7.22.2 構築子と解体子 . . . 295
7.22.2.1 Window() [1/2] . . . 295
7.22.2.2 Window() [2/2] . . . 295
7.22.2.3 ~Window() . . . 295
7.22.3 関数詳解 . . . 295
7.22.3.1 begin() . . . 296
7.22.3.2 commit() . . . 296
7.22.3.3 get() . . . 296
7.22.3.4 getAltArrowX() . . . 296
7.22.3.5 getAltArrowY() . . . 297
7.22.3.6 getAltlArrow() . . . 297
7.22.3.7 getArrow() [1/2] . . . 297
7.22.3.8 getArrow() [2/2] . . . 297
7.22.3.9 getArrowX() . . . 298
7.22.3.10 getArrowY() . . . 298
7.22.3.11 getAspect() . . . 299
7.22.3.12 getControlArrow() . . . 299
7.22.3.13 getControlArrowX() . . . 299
7.22.3.14 getControlArrowY() . . . 300
7.22.3.15 getHeight() . . . 300
7.22.3.16 getKey() . . . 300
7.22.3.17 getMouse() [1/3] . . . 300
7.22.3.18 getMouse() [2/3] . . . 300
7.22.3.19 getMouse() [3/3] . . . 301
7.22.3.20 getMouseX() . . . 301
7.22.3.21 getMouseY() . . . 301
7.22.3.22 getShiftArrow() . . . 302
7.22.3.23 getShiftArrowX() . . . 302
7.22.3.24 getShiftArrowY() . . . 302
7.22.3.25 getSize() [1/2] . . . 302

7.22.3.26 getSize() [2/2] . . . 303
7.22.3.27 getTrackball() . . . 304
7.22.3.28 getTranslation() . . . 304
7.22.3.29 getUserPointer() . . . 305
7.22.3.30 getWheel() [1/3] . . . 305
7.22.3.31 getWheel() [2/3] . . . 305
7.22.3.32 getWheel() [3/3] . . . 306
7.22.3.33 getWheelX() . . . 306
7.22.3.34 getWheelY() . . . 306
7.22.3.35 getWidth() . . . 307
7.22.3.36 init() . . . 307
7.22.3.37 operator bool() . . . 307
7.22.3.38 operator=() . . . 308
7.22.3.39 resetViewport() . . . 308
7.22.3.40 select() . . . 308
7.22.3.41 setClose() . . . 308
7.22.3.42 setKeyboardFunc() . . . 309
7.22.3.43 setMouseFunc() . . . 309
7.22.3.44 setResizeFunc() [1/2] . . . 309
7.22.3.45 setResizeFunc() [2/2] . . . 310
7.22.3.46 setUserPointer() . . . 310
7.22.3.47 shouldClose() . . . 310
7.22.3.48 submit() . . . 311
7.22.3.49 swapBuffers() . . . 311
7.22.3.50 timewarp() . . . 311
7.22.4 メンバ詳解 . . . 311
7.22.4.1 eyeCount . . . 312

8 ファイル詳解 **313**
8.1 gg.cpp ファイル . . . 313
8.2 gg.h ファイル . . . 313
8.2.1 マクロ定義詳解 . . . 319
8.2.1.1 ggError . . . 319
8.2.1.2 ggFBOError . . . 319
8.3 Window.h ファイル . . . 320
8.3.1 マクロ定義詳解 . . . 320
8.3.1.1 USE_IMGUI . . . 320
8.3.1.2 USE_OCULUS_RIFT . . . 320

Chapter 1

ゲームグラフィックス特論の宿題用補助プログラム **GLFW3** 版

Chapter 2

名前空間索引

2.1 名前空間一覧

全名前空間の一覧です。

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 11

Chapter 3

階層索引

3.1 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。

gg::GgBuffer< T > . . . 63
gg::GgColorTexture . . . 70
gg::GgMatrix . . . 80
gg::GgNormalTexture . . . 139
gg::GgPointShader . . . 147
  gg::GgSimpleShader . . . 223
gg::GgQuaternion . . . 156
gg::GgShader . . . 214
gg::GgShape . . . 217
  gg::GgPoints . . . 143
  gg::GgTriangles . . . 251
    gg::GgElements . . . 75
gg::GgSimpleObj . . . 220
gg::GgTexture . . . 239
gg::GgTrackball . . . 243
gg::GgUniformBuffer< T > . . . 256
gg::GgUniformBuffer< Light > . . . 256
  gg::GgSimpleShader::LightBuffer . . . 270
gg::GgUniformBuffer< Material > . . . 256
  gg::GgSimpleShader::MaterialBuffer . . . 283
gg::GgVertex . . . 266
gg::GgSimpleShader::Light . . . 269
gg::GgSimpleShader::Material . . . 281
Window . . . 292

Chapter 4

クラス索引

4.1 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。

gg::GgBuffer< T >
　バッファオブジェクト 63
gg::GgColorTexture
　カラーマップ 70
gg::GgElements
　三角形で表した形状データ (Elements 形式) 75
gg::GgMatrix
　変換行列 80
gg::GgNormalTexture
　法線マップ 139
gg::GgPoints
　点 143
gg::GgPointShader
　点のシェーダ 147
gg::GgQuaternion
　四元数 156
gg::GgShader
　シェーダの基底クラス 214
gg::GgShape
　形状データの基底クラス 217
gg::GgSimpleObj
　Wavefront OBJ 形式のファイル (Arrays 形式) 220
gg::GgSimpleShader
　三角形に単純な陰影付けを行うシェーダ 223
gg::GgTexture
　テクスチャ 239
gg::GgTrackball
　簡易トラックボール処理 243
gg::GgTriangles
　三角形で表した形状データ (Arrays 形式) 251
gg::GgUniformBuffer< T >
　ユニフォームバッファオブジェクト 256
gg::GgVertex
　三角形の頂点データ 266
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 269

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 270
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 281
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 283
Window
ウィンドウ関連の処理 292

Chapter 5

ファイル索引

5.1 ファイル一覧

ファイル一覧です。

gg.cpp 313
gg.h 313
Window.h 320

Chapter 6

名前空間詳解

6.1 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 GgBuffer
  バッファオブジェクト.
- class GgColorTexture
  カラーマップ.
- class GgElements
  三角形で表した形状データ *(Elements* 形式*)*.
- class GgMatrix
  変換行列.
- class GgNormalTexture
  法線マップ.
- class GgPoints
  点.
- class GgPointShader
  点のシェーダ.
- class GgQuaternion
  四元数.
- class GgShader
  シェーダの基底クラス.
- class GgShape
  形状データの基底クラス.
- class GgSimpleObj
  Wavefront OBJ 形式のファイル *(Arrays* 形式*)*.
- class GgSimpleShader
  三角形に単純な陰影付けを行うシェーダ.
- class GgTexture
  テクスチャ.
- class GgTrackball
  簡易トラックボール処理.

- class GgTriangles
  三角形で表した形状データ *(Arrays* 形式*)*.
- class GgUniformBuffer
  ユニフォームバッファオブジェクト.
- struct GgVertex
  三角形の頂点データ.

型定義

- using GgVector = std::array< GLfloat, 4 >
  4 要素の単精度実数の配列.

列挙型

- enum BindingPoints { LightBindingPoint = 0, MaterialBindingPoint }
  光源と材質の *uniform buffer object* の結合ポイント.

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, 初期化に取得される.

6.1.1 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

6.1.2 型定義詳解

6.1.2.1 GgVector

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

4 要素の単精度実数の配列.

gg.h の 1314 行目に定義があります。

6.1.3 列挙型詳解

6.1.3.1 BindingPoints

```
enum gg::BindingPoints
```

光源と材質の uniform buffer object の結合ポイント.

列挙値

LightBindingPoint	光源の uniform buffer object の結合ポイント
MaterialBindingPoint	材質の uniform buffer object の結合ポイント

gg.h の 1300 行目に定義があります。

6.1.4 関数詳解

6.1.4.1 _ggError()

```
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

name	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.
line	エラー発生時に標準エラー出力に出力する数値, 0 なら何も出力しない.

gg.cpp の 2569 行目に定義があります。

6.1.4.2 _ggFBOError()

```
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

name	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.
line	エラー発生時に標準エラー出力に出力する数値, 0 なら何も出力しない.

gg.cpp の 2613 行目に定義があります。

6.1.4.3 ggArraysObj()

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

Wavefront OBJ ファイルを読み込む (Arrays 形式)

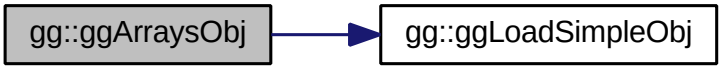
三角形分割された Wavefront OBJ ファイルを読み込んで GgArrays 形式の三角形データを生成する.

引数

name	ファイル名.
normalize	true なら大きさを正規化.

gg.cpp の 5119 行目に定義があります。

呼び出し関係図:



6.1.4.4 ggConjugate()

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

共役四元数を返す.

引数

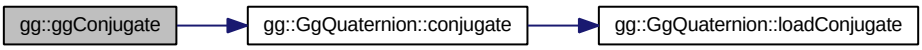
q	GgQuaternion 型の四元数.

戻り値

四元数 q の共役四元数.

gg.h の 3710 行目に定義があります。

呼び出し関係図:



6.1.4.5 ggCreateComputeShader()

```
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 4145 行目に定義があります。

6.1.4.6 ggCreateNormalMap()

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

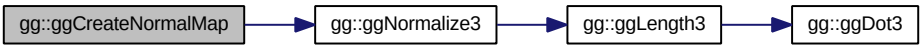
グレースケール画像 (8bit) から法線マップのデータを作成する.

引数

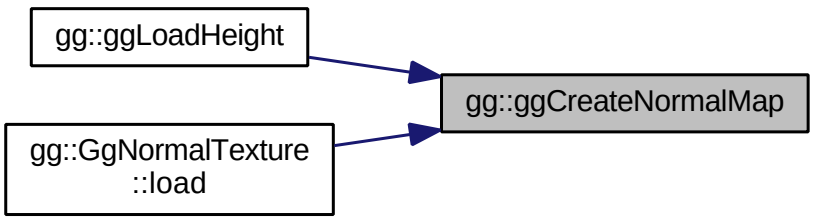
hmap	グレースケール画像のデータ.
width	高さマップのグレースケール画像 hmap の横の画素数.
height	高さマップのグレースケール画像 hmap の縦の画素数.
format	データの書式 (GL_RED, GL_RG, GL_RGB, GL_BGR, GL_RGBA, GL_BGRA).
nz	法線の z 成分の割合.
internal	法線マップを格納するテクスチャの内部フォーマット.
nmap	法線マップを格納する vector.

gg.cpp の 3000 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:



6.1.4.7 ggCreateShader()

```
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 3988 行目に定義があります。

6.1.4.8 ggCross()

```
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 1560 行目に定義があります。

6.1.4.9 ggDot3()

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

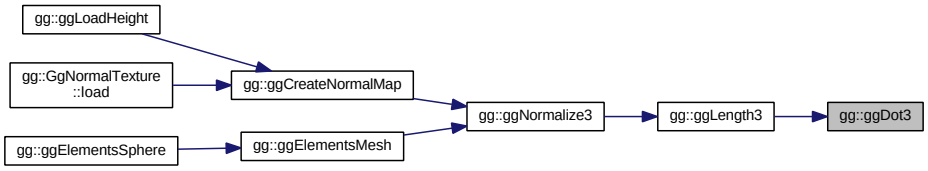
3 要素の内積.

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 1548 行目に定義があります。

被呼び出し関係図:



6.1.4.10 ggDot4() [1/2]

```
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 1635 行目に定義があります。

6.1.4.11 ggDot4() [2/2]

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

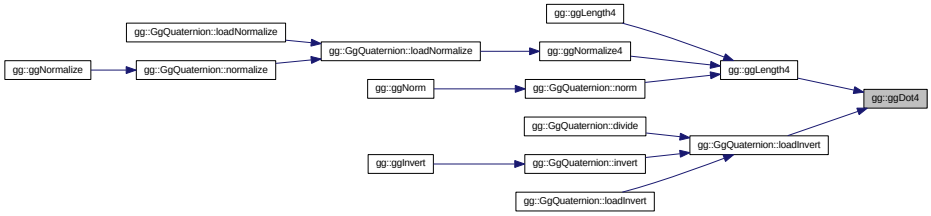
4 要素の内積

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 1624 行目に定義があります。

被呼び出し関係図:



6.1.4.12 ggElementsMesh()

```
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).

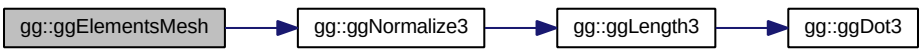
メッシュ状に GgElements 形式の三角形データを生成する.

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 5153 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:

6.1.4.13 ggElementsObj()

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

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

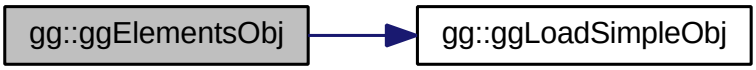
三角形分割された Wavefront OBJ ファイル を読み込んで GgElements 形式の三角形データを生成する.

引数

name	ファイル名.
normalize	true なら大きさを正規化.

gg.cpp の 5135 行目に定義があります。

呼び出し関係図:



6.1.4.14 ggElementsSphere()

```
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 5249 行目に定義があります。

呼び出し関係図:

6.1.4.15 ggEllipse()

```
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 5093 行目に定義があります。

6.1.4.16 ggEulerQuaternion() [1/2]

```
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

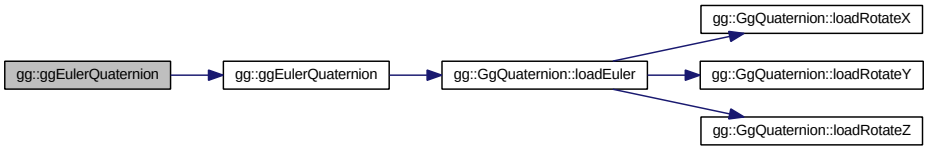
e	オイラー角を表す GLfloat 型の 3 要素の配列変数 (heading, pitch, roll).

戻り値

回転を表す四元数.

gg.h の 3645 行目に定義があります。

呼び出し関係図:



6.1.4.17 ggEulerQuaternion() [2/2]

```
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

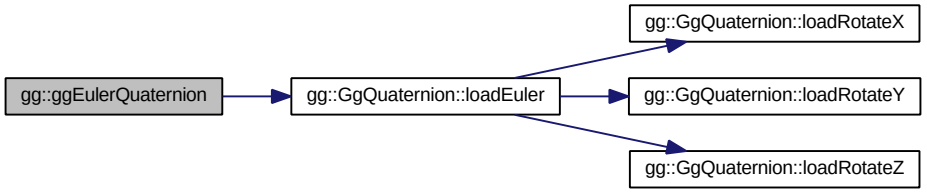
heading	y 軸中心の回転角.
pitch	x 軸中心の回転角.
roll	z 軸中心の回転角.

戻り値

回転を表す四元数.

gg.h の 3636 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:

6.1.4.18 ggFrustum()

```
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

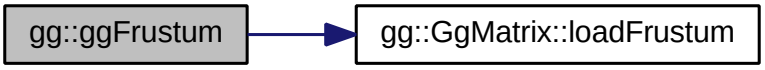
left	ウィンドウの左端の位置.
right	ウィンドウの右端の位置.
bottom	ウィンドウの下端の位置.
top	ウィンドウの上端の位置.
zNear	視点から前方面までの位置.
zFar	視点から後方面までの位置.

戻り値

求めた透視投影変換行列.

gg.h の 2657 行目に定義があります。

呼び出し関係図:



6.1.4.19 ggIdentity()

```
GgMatrix gg::ggIdentity ( )  [inline]
```

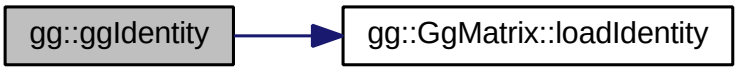
単位行列を返す.

戻り値

単位行列

gg.h の 2439 行目に定義があります。

呼び出し関係図:



6.1.4.20 ggIdentityQuaternion()

```
GgQuaternion gg::ggIdentityQuaternion ( )  [inline]
```

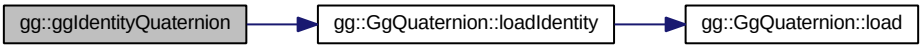
単位四元数を返す

戻り値

単位四元数.

gg.h の 3557 行目に定義があります。

呼び出し関係図:



6.1.4.21 ggInit()

```
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 において 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。

6.1.4.22 ggInvert() [1/2]

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

逆行列を返す.

引数

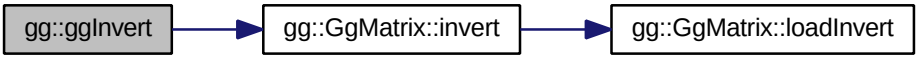
m	元の変換行列.

戻り値

m の逆行列.

gg.h の 2689 行目に定義があります。

呼び出し関係図:



6.1.4.23 ggInvert() [2/2]

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

四元数の逆元を求める.

引数

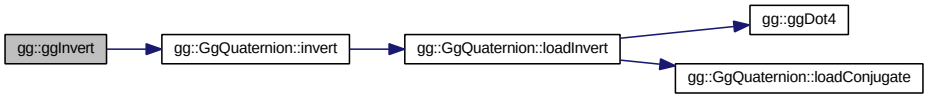
q	GgQuaternion 型の四元数.

戻り値

四元数 q の逆元.

gg.h の 3718 行目に定義があります。

呼び出し関係図:



6.1.4.24 ggLength3()

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

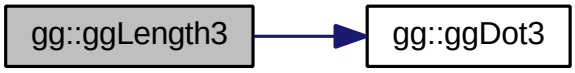
3 要素の長さ.

引数

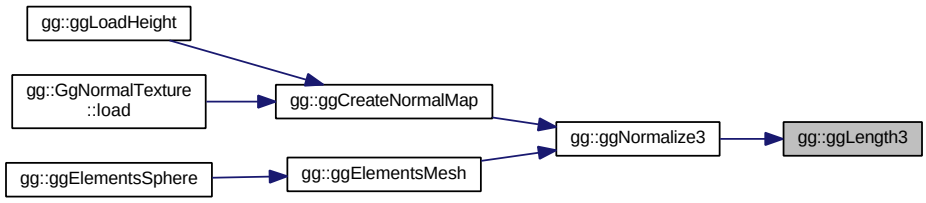
a	GLfloat 型の 3 要素の配列変数.

gg.cpp の 4206 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:



6.1.4.25 ggLength4() [1/2]

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

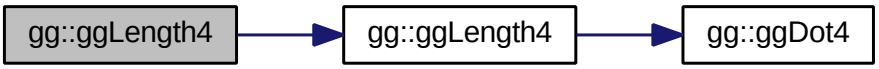
GgVector 型の長さ.

引数

a	GgVector 型の変数.

gg.h の 1579 行目に定義があります。

呼び出し関係図:



6.1.4.26 ggLength4() [2/2]

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

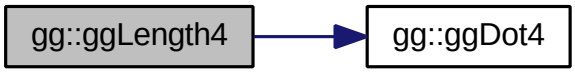
4 要素の長さ.

引数

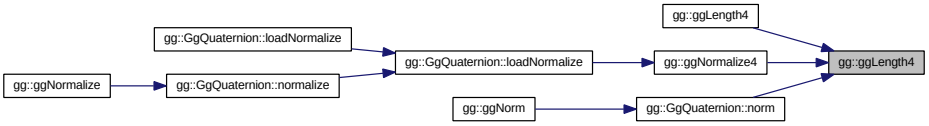
a	GLfloat 型の 4 要素の配列変数.

gg.cpp の 4216 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:



6.1.4.27 ggLoadComputeShader()

```
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

comp	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 4186 行目に定義があります。

6.1.4.28 ggLoadHeight()

```
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

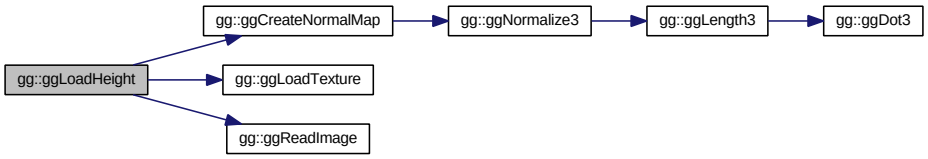
name	読み込むファイル名.
nz	法線の z 成分の割合.
pWidth	読みだした画像ファイルの横の画素数の格納先のポインタ (nullptr なら格納しない). ++
pHeight	読みだした画像ファイルの縦の画素数の格納先のポインタ (nullptr なら格納しない).
internal	glTexImage2D() に指定するテクスチャの内部フォーマット.

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 3080 行目に定義があります。

呼び出し関係図:



6.1.4.29 ggLoadImage()

```
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

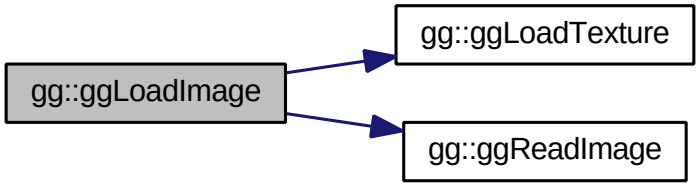
name	読み込むファイル名.
pWidth	読みだした画像ファイルの横の画素数の格納先のポインタ (nullptr なら格納しない). ++
pHeight	読みだした画像ファイルの縦の画素数の格納先のポインタ (nullptr なら格納しない).
internal	glTexImage2D() に指定するテクスチャの内部フォーマット, 0 なら外部フォーマットに合わせる.
wrap	テクスチャのラッピングモード, デフォルトは GL_CLAMP_TO_EDGE.

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 2944 行目に定義があります。

呼び出し関係図:



6.1.4.30 ggLoadShader()

```
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 4123 行目に定義があります。

6.1.4.31 ggLoadSimpleObj() [1/2]

```
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

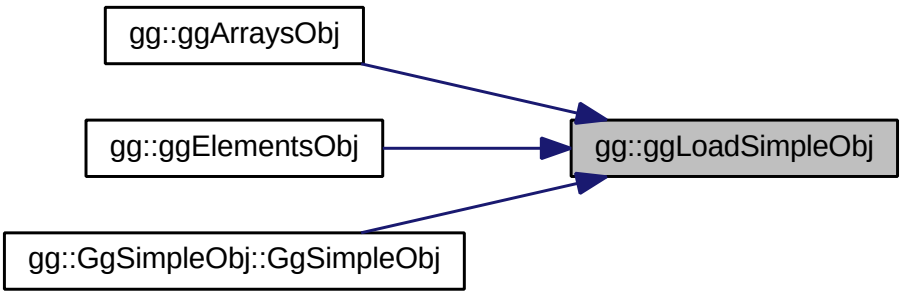
name	読み込む Wavefront OBJ ファイル名.
group	読み込んだデータのポリゴングループごとの最初の三角形の番号と三角形数・材質番号.
material	読み込んだデータのポリゴングループごとの GgSimpleShader::Material 型の材質.
vert	読み込んだデータの頂点属性.
normalize	true なら読み込んだデータの大きさを正規化する.

戻り値

ファイルの読み込みに成功したら true.

gg.cpp の 3733 行目に定義があります。

被呼び出し関係図:



6.1.4.32 ggLoadSimpleObj() [2/2]

```
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 3822 行目に定義があります。

6.1.4.33 ggLoadTexture()

```
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

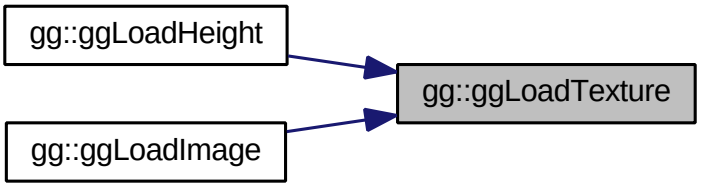
image	テクスチャとして読み込むデータ, nullptr ならテクスチャメモリの確保のみを行う.
width	テクスチャとして読み込むデータ image の横の画素数.
height	テクスチャとして読み込むデータ image の縦の画素数.
format	image のフォーマット.
type	image のデータ型.
internal	テクスチャの内部フォーマット.
wrap	テクスチャのラッピングモード, デフォルトは GL_CLAMP_TO_EDGE.

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:



6.1.4.34 ggLookat() [1/3]

```
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

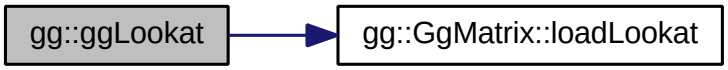
e	視点の位置を格納した GgVector 型の変数.
t	目標点の位置を格納した GgVector 型の変数.
u	上方向のベクトルを格納した GgVector 型の変数.

戻り値

求めたビュー変換行列.

gg.h の 2623 行目に定義があります。

呼び出し関係図:



6.1.4.35 ggLookat() [2/3]

```
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

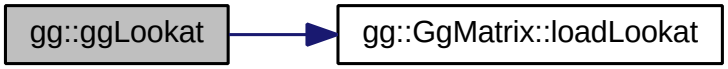
e	視点の位置を格納した GLfloat 型の 3 要素の配列変数.
t	目標点の位置を格納した GLfloat 型の 3 要素の配列変数.
u	上方向のベクトルを格納した GLfloat 型の 3 要素の配列変数.

戻り値

求めたビュー変換行列.

gg.h の 2608 行目に定義があります。

呼び出し関係図:



6.1.4.36 ggLookat() [3/3]

```
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

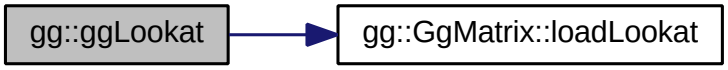
ex	視点の位置の x 座標値.
ey	視点の位置の y 座標値.
ez	視点の位置の z 座標値.
tx	目標点の位置の x 座標値.
ty	目標点の位置の y 座標値.
tz	目標点の位置の z 座標値.
ux	上方向のベクトルの x 成分.
uy	上方向のベクトルの y 成分.
uz	上方向のベクトルの z 成分.

戻り値

求めたビュー変換行列.

gg.h の 2593 行目に定義があります。

呼び出し関係図:



6.1.4.37 ggMatrixQuaternion() [1/2]

```
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

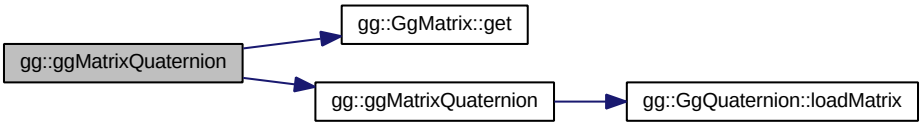
m	GgMatrix 型の変換行列.

戻り値

m による回転の変換に相当する四元数.

gg.h の 3575 行目に定義があります。

呼び出し関係図:



6.1.4.38 ggMatrixQuaternion() [2/2]

```
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

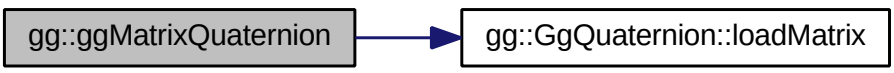
a	GLfloat 型の 16 要素の配列変数.

戻り値

a による回転の変換に相当する四元数.

gg.h の 3566 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:

6.1.4.39 ggNorm()

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

四元数のノルムを返す.

引数

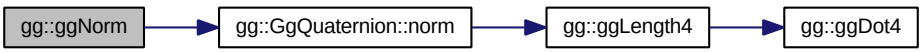
q	GgQuaternion 型の四元数.

戻り値

四元数 q のノルム.

gg.h の 3694 行目に定義があります。

呼び出し関係図:



6.1.4.40 ggNormal()

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

法線変換行列を返す.

引数

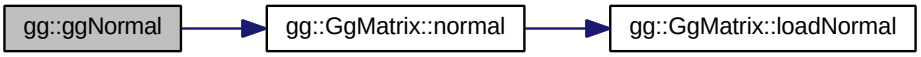
m	元の変換行列.

戻り値

m の法線変換行列.

gg.h の 2697 行目に定義があります。

呼び出し関係図:



6.1.4.41 ggNormalize()

```
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 3702 行目に定義があります。

呼び出し関係図:

6.1.4.42 ggNormalize3()

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

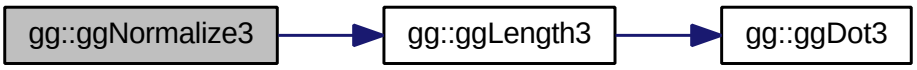
3 要素の正規化.

引数

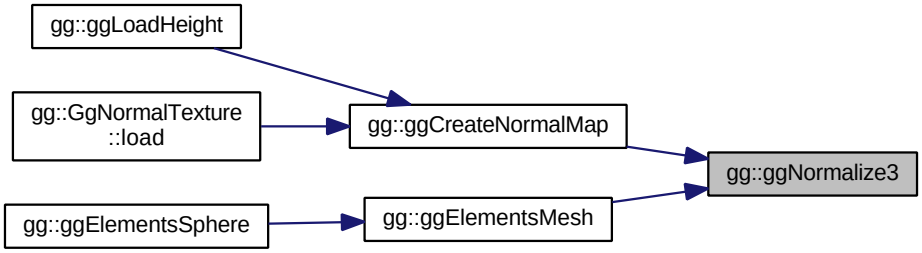
a	GLfloat 型の 3 要素の配列変数.

gg.h の 1531 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:



6.1.4.43 ggNormalize4() [1/2]

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

GgVector 型の正規化.

引数

a	GgVector 型の変数

gg.h の 1606 行目に定義があります。

呼び出し関係図:

6.1.4.44 ggNormalize4() [2/2]

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

4 要素の正規化.

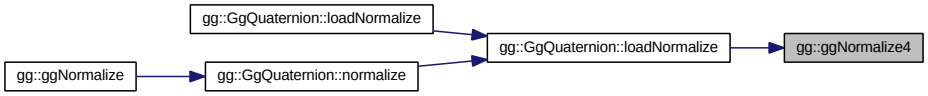
引数

a	GLfloat 型の 4 要素の配列変数.

gg.h の 1589 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:



6.1.4.45 ggOrthogonal()

```
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

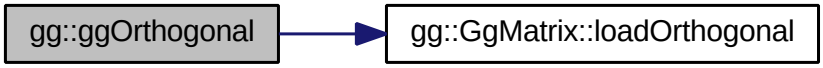
left	ウィンドウの左端の位置.
right	ウィンドウの右端の位置.
bottom	ウィンドウの下端の位置.
top	ウィンドウの上端の位置.
zNear	視点から前方面までの位置.
zFar	視点から後方面までの位置.

戻り値

求めた直交投影変換行列.

gg.h の 2641 行目に定義があります。

呼び出し関係図:



6.1.4.46 ggPerspective()

```
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

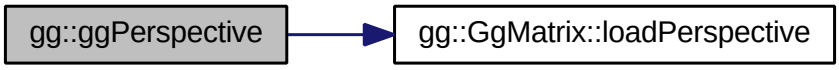
fovy	y 方向の画角.
aspect	縦横比.
zNear	視点から前方面までの位置.
zFar	視点から後方面までの位置.

戻り値

求めた透視投影変換行列.

gg.h の 2671 行目に定義があります。

呼び出し関係図:



6.1.4.47 ggPointsCube()

```
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 5010 行目に定義があります。

6.1.4.48 ggPointsSphere()

```
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 5040 行目に定義があります。

6.1.4.49 ggQuaternion() [1/2]

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

四元数を返す

引数

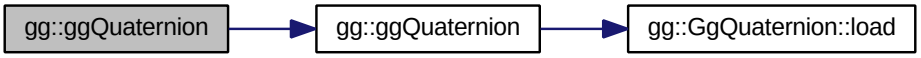
a	GLfloat 型の GLfloat 型の 4 要素の配列変数に格納した四元数.

戻り値

四元数.

gg.h の 3550 行目に定義があります。

呼び出し関係図:



6.1.4.50 ggQuaternion() [2/2]

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

四元数を返す

引数

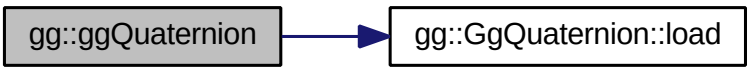
x	四元数の x 要素.
y	四元数の y 要素.
z	四元数の z 要素.
w	四元数の w 要素.

戻り値

四元数.

gg.h の 3541 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:

6.1.4.51 ggQuaternionMatrix()

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

四元数 q の回転の変換行列を返す.

引数

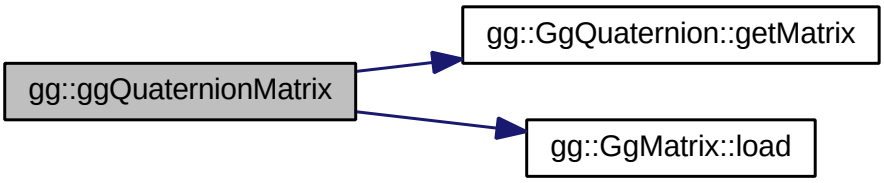
q	元の四元数.

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 3583 行目に定義があります。

呼び出し関係図:



6.1.4.52 ggQuaternionTransposeMatrix()

```
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

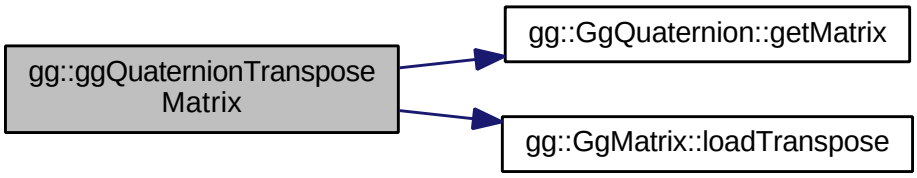
q	元の四元数.

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 3594 行目に定義があります。

呼び出し関係図:



6.1.4.53 ggReadImage()

```
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

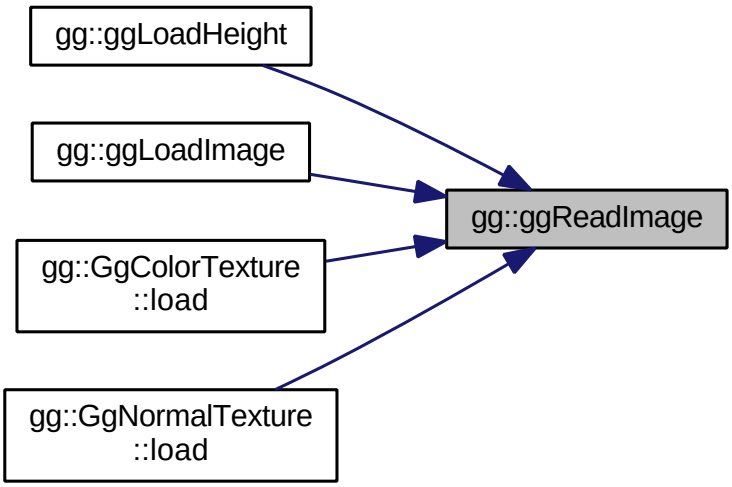
name	読み込むファイル名.
image	読み込んだデータを格納する vector.
pWidth	読み込んだ画像の横の画素数の格納先のポインタ, nullptr なら格納しない.
pHeight	読み込んだ画像の縦の画素数の格納先のポインタ, nullptr なら格納しない.
pFormat	読み込んだファイルの書式 (GL_RED, G_RG, GL_BGR, G_BGRA) の格納先のポインタ, nullptr なら格納しない.

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:



6.1.4.54 ggRectangle()

```
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 5072 行目に定義があります。

6.1.4.55 ggRotate() [1/5]

```
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

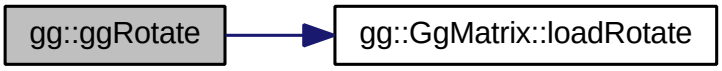
r	回転軸のベクトルと回転角を表す GgVector 型の変数.

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 2576 行目に定義があります。

呼び出し関係図:



6.1.4.56 ggRotate() [2/5]

```
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

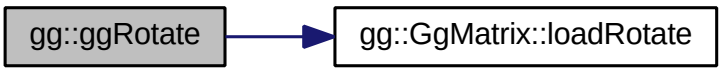
r	回転軸のベクトルを表す GgVector 型の変数.
a	回転角.

戻り値

r を軸に a だけ回転する変換行列.

gg.h の 2558 行目に定義があります。

呼び出し関係図:



6.1.4.57 ggRotate() [3/5]

```
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

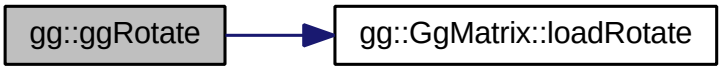
r	回転軸のベクトルと回転角を表す GLfloat 型の 4 要素の配列変数.

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 2567 行目に定義があります。

呼び出し関係図:



6.1.4.58 ggRotate() [4/5]

```
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

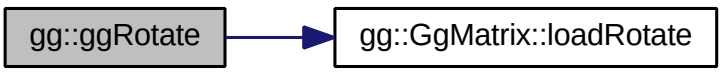
r	回転軸のベクトルを表す GLfloat 型の 3 要素の配列変数.
a	回転角.

戻り値

r を軸に a だけ回転する変換行列.

gg.h の 2548 行目に定義があります。

呼び出し関係図:



6.1.4.59 ggRotate() [5/5]

```
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

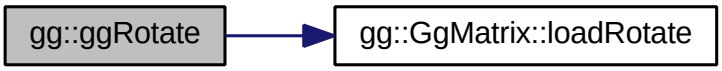
x	回転軸の x 成分.
y	回転軸の y 成分.
z	回転軸の z 成分.
a	回転角.

戻り値

(x, y, z) を軸にさらに a 回転する変換行列.

gg.h の 2538 行目に定義があります。

呼び出し関係図:



6.1.4.60 ggRotateQuaternion() [1/3]

```
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

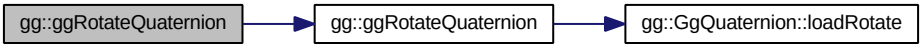
v	軸ベクトルを表す GLfloat 型の 4 要素の配列変数.

戻り値

回転を表す四元数.

gg.h の 3626 行目に定義があります。

呼び出し関係図:



6.1.4.61 ggRotateQuaternion() [2/3]

```
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

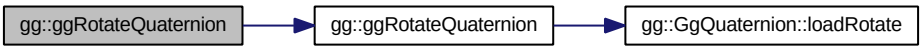
v	軸ベクトルを表す GLfloat 型の 3 要素の配列変数.
a	回転角.

戻り値

回転を表す四元数.

gg.h の 3618 行目に定義があります。

呼び出し関係図:



6.1.4.62 ggRotateQuaternion() [3/3]

```
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

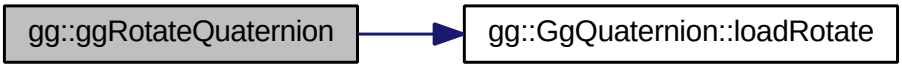
x	軸ベクトルの x 成分.
y	軸ベクトルの y 成分.
z	軸ベクトルの z 成分.
a	回転角.

戻り値

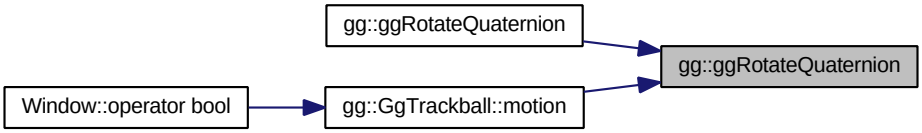
回転を表す四元数.

gg.h の 3608 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:



6.1.4.63 ggRotateX()

```
GgMatrix gg::ggRotateX (
            GLfloat a )  [inline]
```

x 軸中心の回転の変換行列を返す.

引数

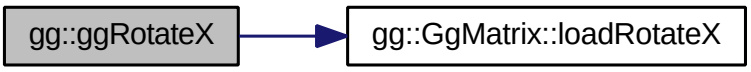
a	回転角.

戻り値

x 軸中心に a だけ回転する変換行列.

gg.h の 2508 行目に定義があります。

呼び出し関係図:



6.1.4.64 ggRotateY()

```
GgMatrix gg::ggRotateY (
            GLfloat a )  [inline]
```

y 軸中心の回転の変換行列を返す.

引数

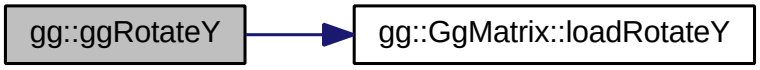
a	回転角.

戻り値

y 軸中心に a だけ回転する変換行列.

gg.h の 2517 行目に定義があります。

呼び出し関係図:



6.1.4.65 ggRotateZ()

```
GgMatrix gg::ggRotateZ (
            GLfloat a )  [inline]
```

z 軸中心の回転の変換行列を返す.

引数

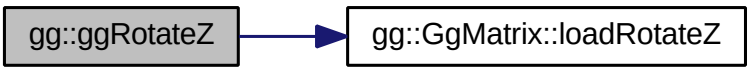
a	回転角.

戻り値

z 軸中心に a だけ回転する変換行列.

gg.h の 2526 行目に定義があります。

呼び出し関係図:



6.1.4.66 ggSaveColor()

```
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:

6.1.4.67 ggSaveDepth()

```
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:

6.1.4.68 ggSaveTga()

```
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

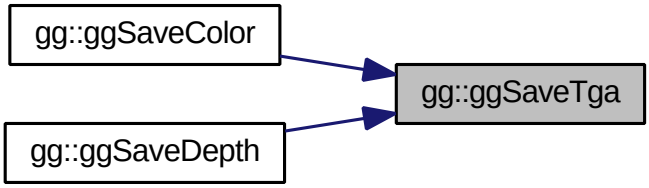
name	保存するファイル名.
buffer	画像データを格納した配列.
width	画像の横の画素数.
height	画像の縦の画素数.
depth	1 画素のバイト数.

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:



6.1.4.69 ggScale() [1/3]

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

拡大縮小の変換行列を返す.

引数

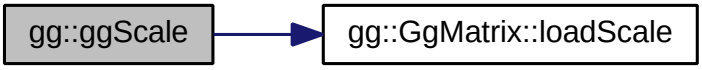
s	拡大率の GgVector 型の変数.

戻り値

拡大縮小の変換行列.

gg.h の 2499 行目に定義があります。

呼び出し関係図:



6.1.4.70 ggScale() [2/3]

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

拡大縮小の変換行列を返す.

引数

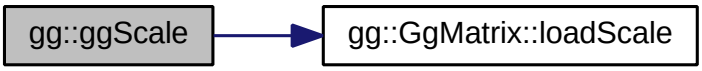
s	拡大率の GLfloat 型の 3 要素の配列変数 (x, y, z).

戻り値

拡大縮小の変換行列.

gg.h の 2490 行目に定義があります。

呼び出し関係図:



6.1.4.71 ggScale() [3/3]

```
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

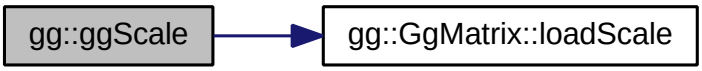
x	x 方向の拡大率.
y	y 方向の拡大率.
z	z 方向の拡大率.
w	拡大率のスケールファクタ (= 1.0f).

戻り値

拡大縮小の変換行列.

gg.h の 2481 行目に定義があります。

呼び出し関係図:



6.1.4.72 ggSlerp() [1/4]

```
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

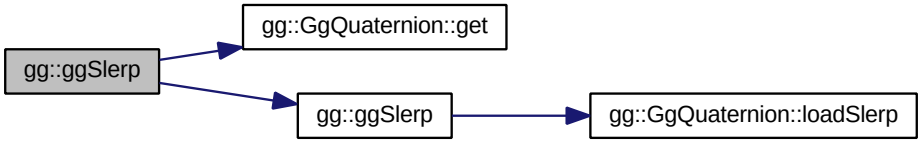
q	GgQuaternion 型の四元数.
r	GgQuaternion 型の四元数.
t	補間パラメータ.

戻り値

q, r を t で内分した四元数.

gg.h の 3666 行目に定義があります。

呼び出し関係図:



6.1.4.73 ggSlerp() [2/4]

```
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

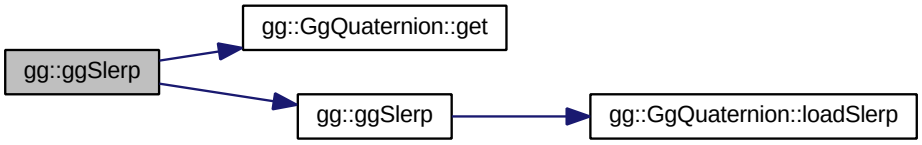
q	GgQuaternion 型の四元数.
a	四元数を格納した GLfloat 型の 4 要素の配列変数.
t	補間パラメータ.

戻り値

q, a を t で内分した四元数.

gg.h の 3676 行目に定義があります。

呼び出し関係図:



6.1.4.74 ggSlerp() [3/4]

```
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

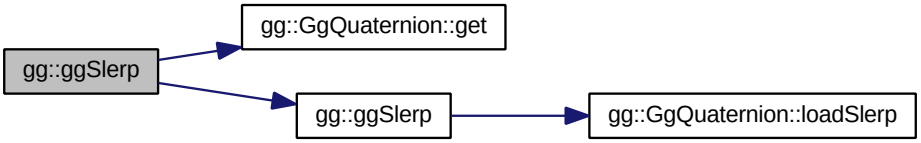
a	四元数を格納した GLfloat 型の 4 要素の配列変数.
q	GgQuaternion 型の四元数.
t	補間パラメータ.

戻り値

a, q を t で内分した四元数.

gg.h の 3686 行目に定義があります。

呼び出し関係図:



6.1.4.75 ggSlerp() [4/4]

```
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

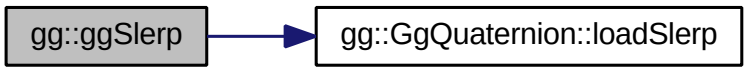
a	四元数を格納した GLfloat 型の 4 要素の配列変数.
b	四元数を格納した GLfloat 型の 4 要素の配列変数.
t	補間パラメータ.

戻り値

a, b を t で内分した四元数.

gg.h の 3655 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:

6.1.4.76 ggTranslate() [1/3]

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

平行移動の変換行列を返す.

引数

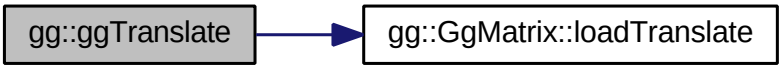
t	移動量の GgVector 型の変数.

戻り値

平行移動の変換行列

gg.h の 2469 行目に定義があります。

呼び出し関係図:



6.1.4.77 ggTranslate() [2/3]

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

平行移動の変換行列を返す.

引数

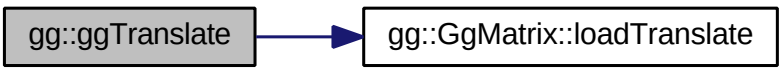
t	移動量の GLfloat 型の 3 要素の配列変数 (x, y, z).

戻り値

平行移動の変換行列

gg.h の 2460 行目に定義があります。

呼び出し関係図:



6.1.4.78 ggTranslate() [3/3]

```
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

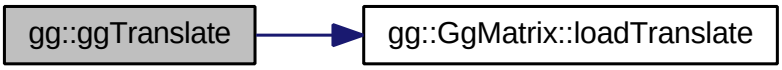
x	x 方向の移動量.
y	y 方向の移動量.
z	z 方向の移動量.
w	移動量のスケールファクタ (= 1.0f).

戻り値

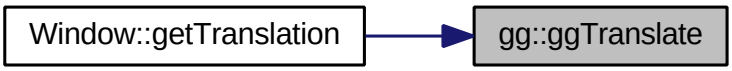
平行移動の変換行列.

gg.h の 2451 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:



6.1.4.79 ggTranspose()

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

転置行列を返す.

引数

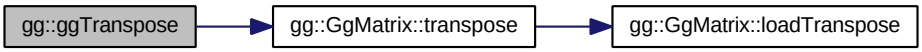
m	元の変換行列.

戻り値

m の転置行列.

gg.h の 2681 行目に定義があります。

呼び出し関係図:



6.1.5 変数詳解

6.1.5.1 ggBufferAlignment

```
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, 初期化に取得される.

使用している GPU のバッファアライメント

Chapter 7

クラス詳解

7.1 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

- GgBuffer (GLenum target, const T *data, GLsizei stride, GLsizei count, GLenum usage)
  コンストラクタ.
- virtual ~GgBuffer ()
  デストラクタ.
- GgBuffer (const GgBuffer< T > &o)=delete
  コピーコンストラクタは使用禁止.
- GgBuffer< T > & operator= (const GgBuffer< T > &o)=delete
  代入演算子は使用禁止.
- 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.

7.1.1 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 4084 行目に定義があります。

7.1.2 構築子と解体子

7.1.2.1 GgBuffer() [1/2]

```
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 4106 行目に定義があります。

7.1.2.2 ~GgBuffer()

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

デストラクタ.

gg.h の 4118 行目に定義があります。

7.1.2.3 GgBuffer() [2/2]

```
template<typename T >
gg::GgBuffer< T >::GgBuffer (
            const GgBuffer< T > & o )  [delete]
```

コピーコンストラクタは使用禁止.

7.1.3 関数詳解

7.1.3.1 bind()

```
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 4160 行目に定義があります。

7.1.3.2 copy()

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

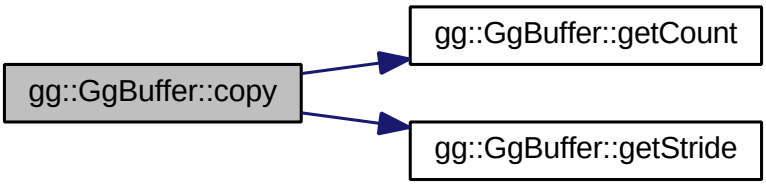
別のバッファオブジェクトからデータを複写する.

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 4234 行目に定義があります。

呼び出し関係図:



7.1.3.3 getBuffer()

```
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 4154 行目に定義があります。

7.1.3.4 getCount()

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

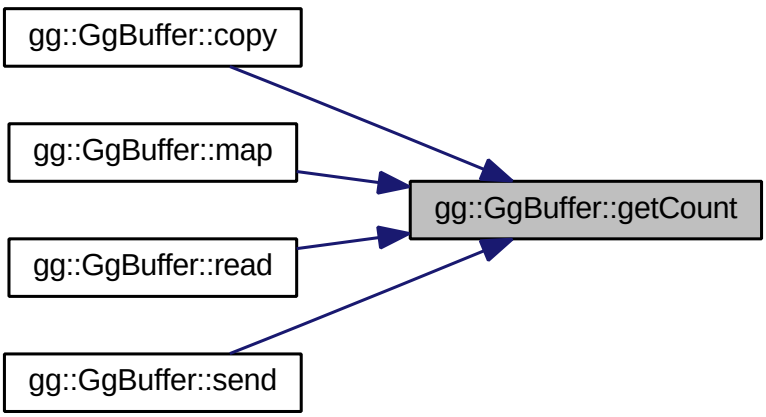
バッファオブジェクトが保持するデータの数を取り出す.

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 4147 行目に定義があります。

被呼び出し関係図:



7.1.3.5 getStride()

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

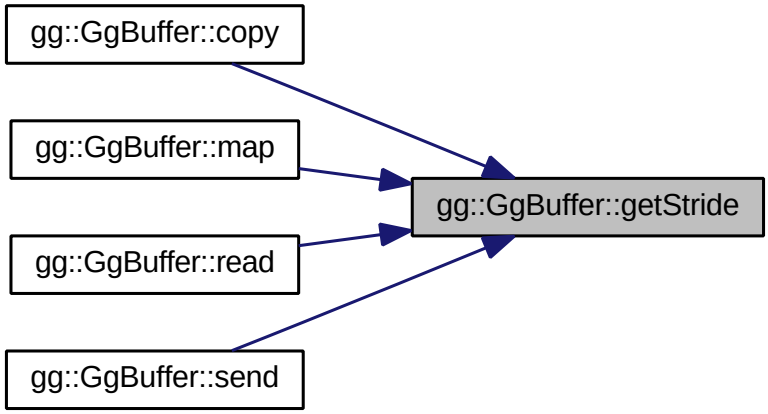
バッファオブジェクトのアライメントを考慮したデータの間隔を取り出す.

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 4140 行目に定義があります。

被呼び出し関係図:



7.1.3.6 getTarget()

```
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 4133 行目に定義があります。

7.1.3.7 map() [1/2]

```
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 4173 行目に定義があります。

7.1.3.8 map() [2/2]

```
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

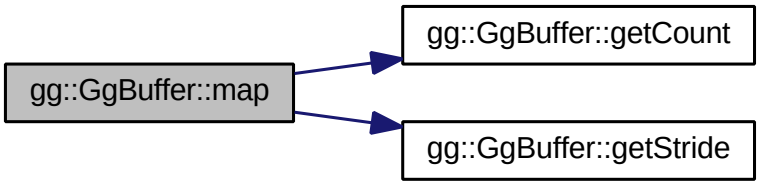
first	マップする範囲のバッファオブジェクトの先頭からの位置.
count	マップするデータの数 (0 ならバッファオブジェクト全体).

戻り値

マップしたメモリの先頭のポインタ.

gg.h の 4183 行目に定義があります。

呼び出し関係図:



7.1.3.9 operator=()

```
template<typename T >
GgBuffer<T>& gg::GgBuffer< T >::operator= (
            const GgBuffer< T > & o )  [delete]
```

代入演算子は使用禁止.

7.1.3.10 read()

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

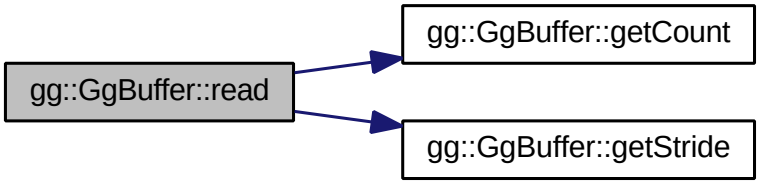
バッファオブジェクトのデータから抽出する.

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 4218 行目に定義があります。

呼び出し関係図:



7.1.3.11 send()

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

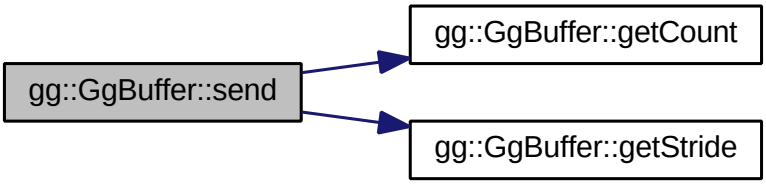
すでに確保したバッファオブジェクトにデータを転送する.

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 4203 行目に定義があります。

呼び出し関係図:



7.1.3.12 unbind()

```
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 4166 行目に定義があります。

7.1.3.13 unmap()

```
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 4194 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

7.2 gg::GgColorTexture クラス

カラーマップ.

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

公開メンバ関数

- 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.
- virtual ~GgColorTexture ()
  デストラクタ.
- 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.

7.2.1 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 3949 行目に定義があります。

7.2.2 構築子と解体子

7.2.2.1 GgColorTexture() [1/3]

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

コンストラクタ.

gg.h の 3957 行目に定義があります。

7.2.2.2 GgColorTexture() [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 3967 行目に定義があります。

呼び出し関係図:

7.2.2.3 GgColorTexture() [3/3]

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

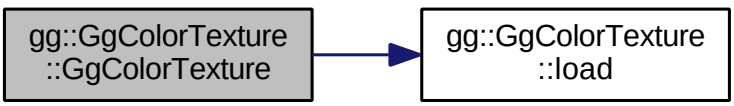
ファイルからデータを読み込んでテクスチャを作成するコンストラクタ.

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.
wrap	テクスチャのラッピングモード, GL_TEXTURE_WRAP_S および GL_TEXTURE_WRAP_T に設定する値.

gg.h の 3978 行目に定義があります。

呼び出し関係図:



7.2.2.4 ～GgColorTexture()

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

デストラクタ.

gg.h の 3984 行目に定義があります。

7.2.3 関数詳解

7.2.3.1 load() [1/2]

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

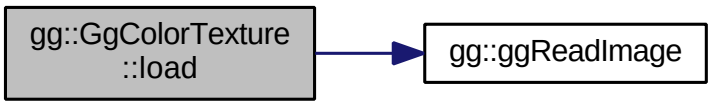
テクスチャを作成してファイルからデータを読み込む.

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 ならファイルの画像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 (GL_CLAMP_TO_EDGE, GL_CLAMP_TO_BORDER, GL_REPEAT, GL_MIRRORED_REPEAT).

gg.cpp の 3118 行目に定義があります。

呼び出し関係図:



7.2.3.2 load() [2/2]

```
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]
```

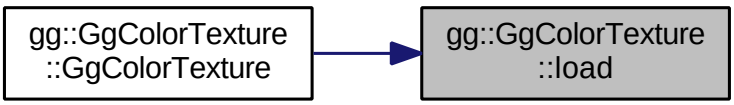
テクスチャを作成してメモリ上のデータを読み込む.

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	glTexImage2D() に指定する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 (GL_CLAMP_TO_EDGE, GL_CLAMP_TO_BORDER, GL_REPEAT, GL_MIRRORED_REPEAT).

gg.h の 3994 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:

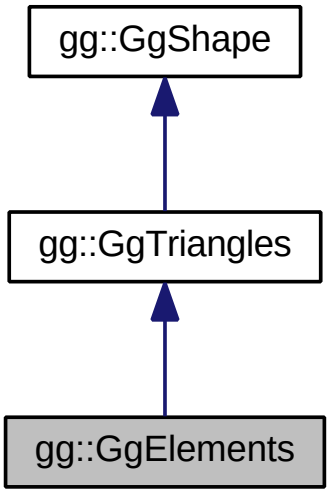
- gg.h
- gg.cpp

7.3 gg::GgElements クラス

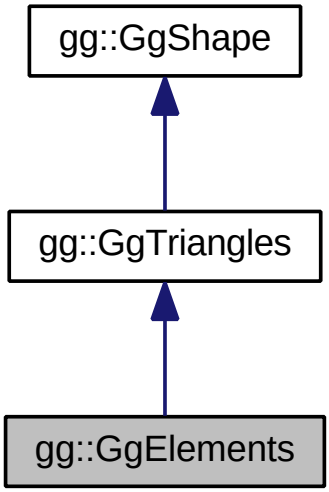
三角形で表した形状データ (Elements 形式).

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

gg::GgElements の継承関係図



gg::GgElements 連携図



公開メンバ関数

- 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.
- virtual ~GgElements ()
  デストラクタ.
- 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.

7.3.1 詳解

三角形で表した形状データ (Elements 形式).

gg.h の 4759 行目に定義があります。

7.3.2 構築子と解体子

7.3.2.1 GgElements() [1/2]

```
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 4769 行目に定義があります。

7.3.2.2 GgElements() [2/2]

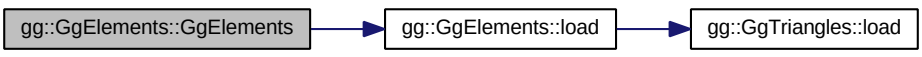
```
gg::GgElements::GgElements (
            const GgVertex * vert,
            GLsizei countv,
            const GLuint * face,
            GLsizei countf,
            GLenum mode = GL_TRIANGLES,
            GLenum usage = GL_STATIC_DRAW )  [inline]
```

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 4780 行目に定義があります。
呼び出し関係図:



7.3.2.3 ~GgElements()

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

デストラクタ.

gg.h の 4788 行目に定義があります。

7.3.3 関数詳解

7.3.3.1 draw()

```
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 4997 行目に定義があります。

呼び出し関係図:

7.3.3.2 getIndexBuffer()

```
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 4799 行目に定義があります。

7.3.3.3 getIndexCount()

```
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 4792 行目に定義があります。

7.3.3.4 load()

```
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.

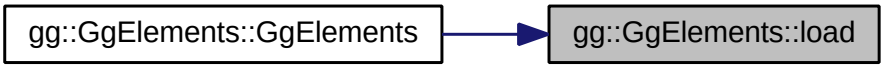
引数

vert	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 4824 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:



7.3.3.5 send()

```
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

vert	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 4811 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

7.4 gg::GgMatrix クラス

変換行列.

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

公開メンバ関数

- 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 ()
  デストラクタ.
- 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.
- GLfloat get (int i) const
  変換行列の要素を取り出す.
- const GLfloat & operator[ ] (std::size_t i) const
  変換行列の要素にアクセスする.
- GLfloat & operator[ ] (std::size_t i)
  変換行列の要素にアクセスする.

フレンド

- class GgQuaternion

7.4.1 詳解

変換行列.

gg.h の 1643 行目に定義があります。

7.4.2 構築子と解体子

7.4.2.1 GgMatrix() [1/3]

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

コンストラクタ.

gg.h の 1660 行目に定義があります。

7.4.2.2 GgMatrix() [2/3]

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

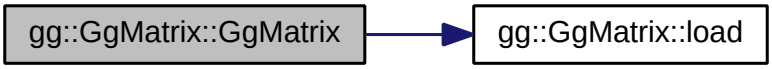
コンストラクタ.

引数

a	GLfloat 型の 16 要素の配列変数.

gg.h の 1664 行目に定義があります。

呼び出し関係図:



7.4.2.3 GgMatrix() [3/3]

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

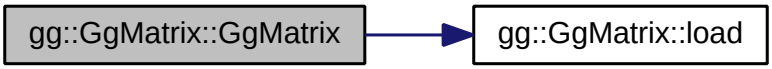
コピーコンストラクタ.

引数

m	GgMatrix 型の変数.

gg.h の 1671 行目に定義があります。

呼び出し関係図:



7.4.2.4 ~GgMatrix()

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

デストラクタ.

gg.h の 1677 行目に定義があります。

7.4.3 関数詳解

7.4.3.1 add() [1/2]

```
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

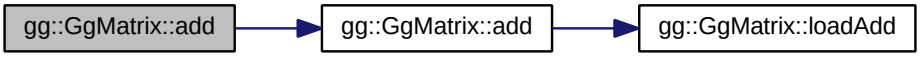
m	GgMatrix 型の変数.

戻り値

変換行列に m を加えた GgMatrix 型の値.

gg.h の 1774 行目に定義があります。

呼び出し関係図:



7.4.3.2 add() [2/2]

```
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

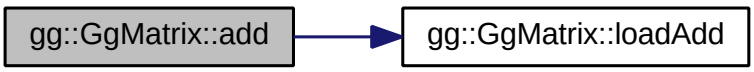
a	GLfloat 型の 16 要素の配列変数.

戻り値

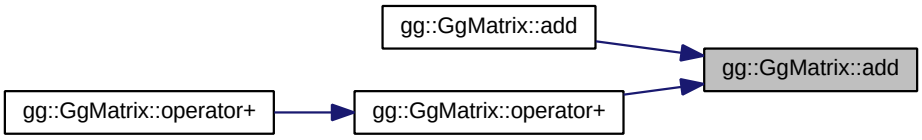
変換行列に a を加えた GgMatrix 型の値.

gg.h の 1765 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:



7.4.3.3 divide() [1/2]

```
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 1828 行目に定義があります。

呼び出し関係図:

7.4.3.4 divide() [2/2]

```
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

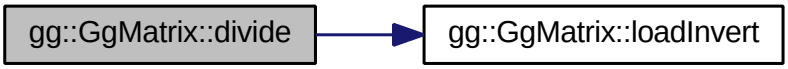
a	GLfloat 型の 16 要素の配列変数.

戻り値

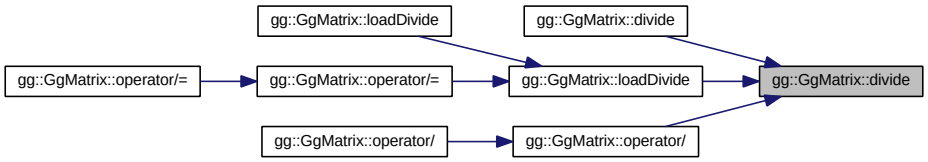
変換行列を a で割った GgMatrix 型の値.

gg.h の 1817 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:



7.4.3.5 frustum()

```
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

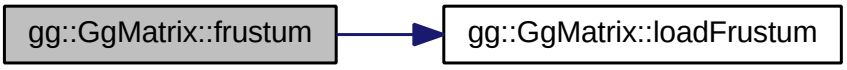
left	ウィンドウの左端の位置.
right	ウィンドウの右端の位置.
bottom	ウィンドウの下端の位置.
top	ウィンドウの上端の位置.
zNear	視点から前方面までの位置.
zFar	視点から後方面までの位置.

戻り値

透視投影変換行列を乗じた変換行列.

gg.h の 2314 行目に定義があります。

呼び出し関係図:



7.4.3.6 get() [1/3]

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

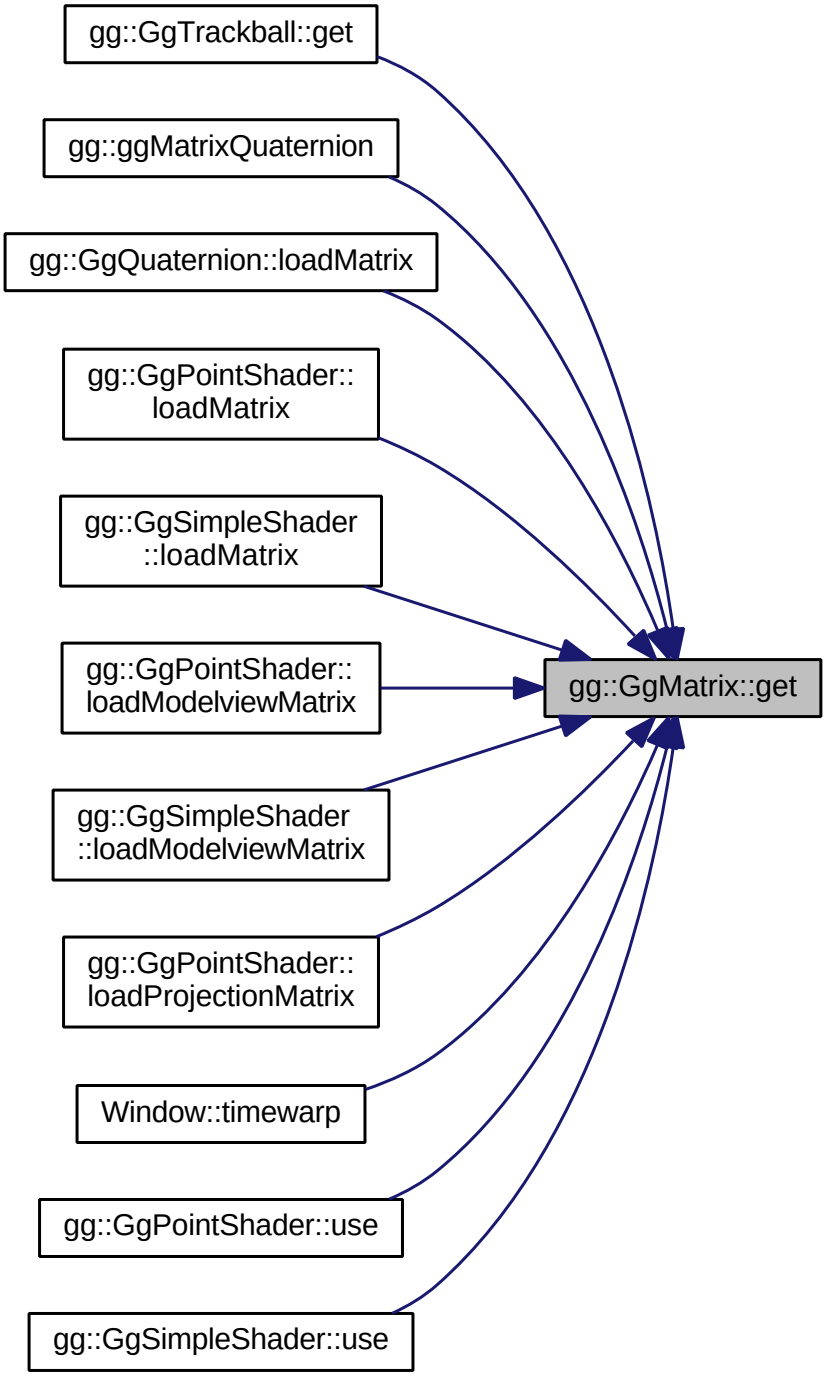
変換行列を取り出す.

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 2403 行目に定義があります。

被呼び出し関係図:



7.4.3.7 get() [2/3]

```
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 2410 行目に定義があります。

7.4.3.8 get() [3/3]

```
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 の i 番目の要素.

gg.h の 2417 行目に定義があります。

7.4.3.9 invert()

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

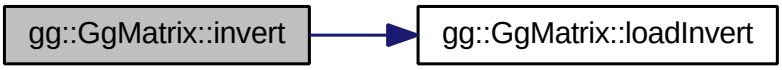
逆行列を返す.

戻り値

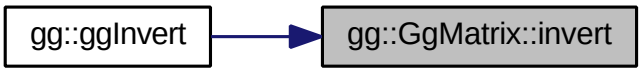
逆行列.

gg.h の 2345 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:



7.4.3.10 load() [1/2]

```
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 1691 行目に定義があります。

呼び出し関係図:

7.4.3.11 load() [2/2]

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

配列変数の値を格納する.

引数

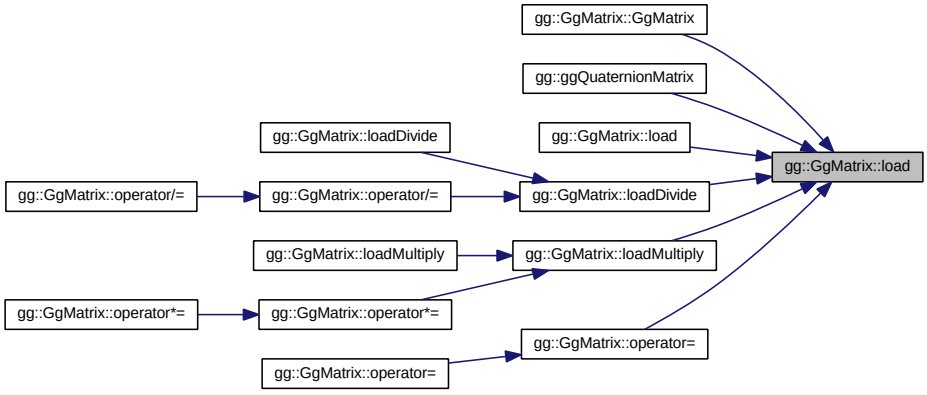
a	GLfloat 型の 16 要素の配列変数.

戻り値

a を代入した GgMatrix 型の値.

gg.h の 1682 行目に定義があります。

被呼び出し関係図:



7.4.3.12 loadAdd() [1/2]

```
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 1708 行目に定義があります。

呼び出し関係図:

7.4.3.13 loadAdd() [2/2]

```
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

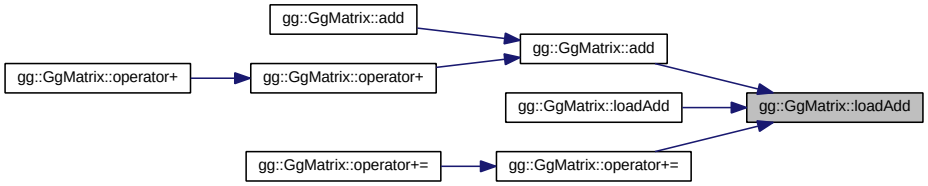
a	GLfloat 型の 16 要素の配列変数.

戻り値

変換行列に a を加えた GgMatrix 型の値.

gg.h の 1699 行目に定義があります。

被呼び出し関係図:



7.4.3.14 loadDivide() [1/2]

```
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

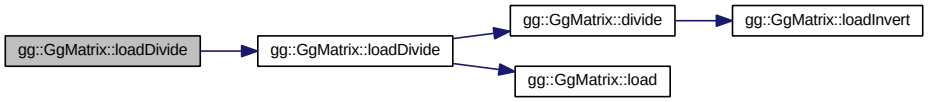
m	GgMatrix 型の変数.

戻り値

変換行列に m を乗じた GgMatrix 型の値.

gg.h の 1757 行目に定義があります。

呼び出し関係図:



7.4.3.15 loadDivide() [2/2]

```
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

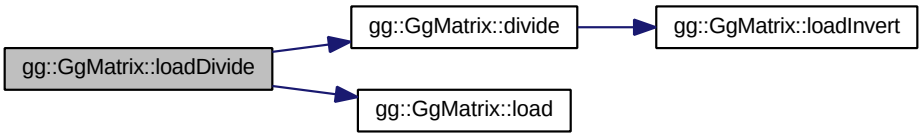
a	GLfloat 型の 16 要素の配列変数.

戻り値

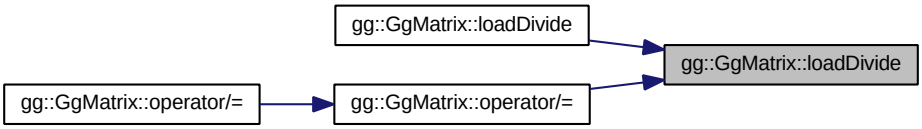
変換行列に a を乗じた GgMatrix 型の値.

gg.h の 1749 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:



7.4.3.16 loadFrustum()

```
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

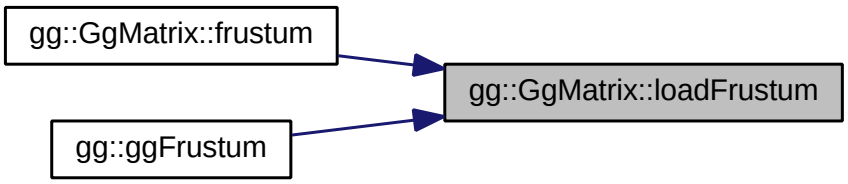
left	ウィンドウの左端の位置.
right	ウィンドウの右端の位置.
bottom	ウィンドウの下端の位置.
top	ウィンドウの上端の位置.
zNear	視点から前方面までの位置.
zFar	視点から後方面までの位置.

戻り値

設定した透視投影変換行列.

gg.cpp の 4590 行目に定義があります。

被呼び出し関係図:



7.4.3.17 loadIdentity()

```
gg::GgMatrix & gg::GgMatrix::loadIdentity ( )
```

単位行列を格納する.

gg.cpp の 4248 行目に定義があります。

被呼び出し関係図:

7.4.3.18 loadInvert() [1/2]

```
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 2104 行目に定義があります。

呼び出し関係図:

7.4.3.19 loadInvert() [2/2]

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

逆行列を格納する.

引数

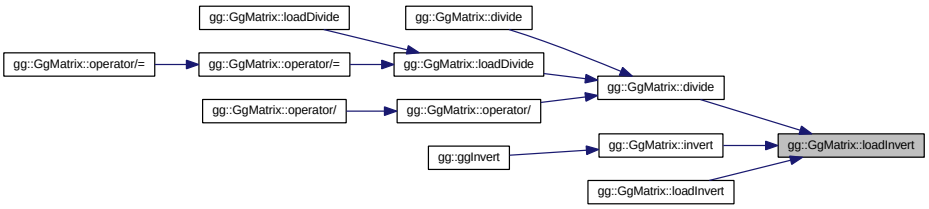
a	GLfloat 型の 16 要素の変換行列.

戻り値

設定した a の逆行列.

gg.cpp の 4405 行目に定義があります。

被呼び出し関係図:



7.4.3.20 loadLookat() [1/3]

```
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 2045 行目に定義があります。

呼び出し関係図:

7.4.3.21 loadLookat() [2/3]

```
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 2035 行目に定義があります。

呼び出し関係図:

7.4.3.22 loadLookat() [3/3]

```
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

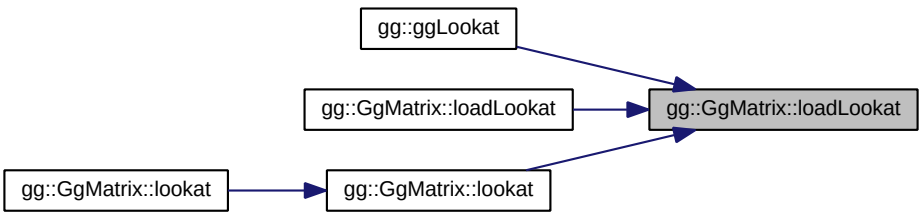
ex	視点の位置の x 座標値.
ey	視点の位置の y 座標値.
ez	視点の位置の z 座標値.
tx	目標点の位置の x 座標値.
ty	目標点の位置の y 座標値.
tz	目標点の位置の z 座標値.
ux	上方向のベクトルの x 成分.
uy	上方向のベクトルの y 成分.
uz	上方向のベクトルの z 成分.

戻り値

設定したビュー変換行列.

gg.cpp の 4507 行目に定義があります。

被呼び出し関係図:



7.4.3.23 loadMultiply() [1/2]

```
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

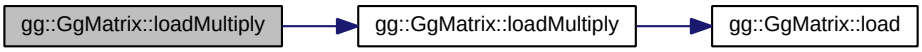
m	GgMatrix 型の変数.

戻り値

変換行列に m を掛けた GgMatrix 型の値.

gg.h の 1741 行目に定義があります。

呼び出し関係図:



7.4.3.24 loadMultiply() [2/2]

```
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

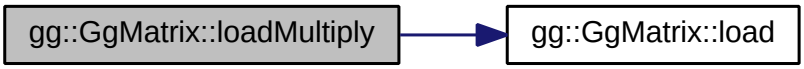
a	GLfloat 型の 16 要素の配列変数.

戻り値

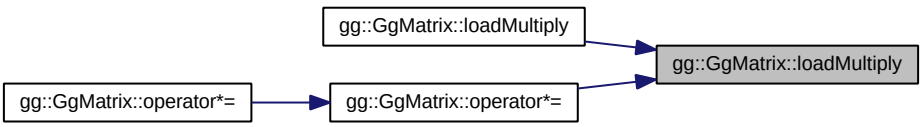
変換行列に a を掛けた GgMatrix 型の値.

gg.h の 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:



7.4.3.25 loadNormal() [1/2]

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

法線変換行列を格納する.

引数

m	GgMatrix 型の変換行列.

戻り値

設定した m の法線変換行列.

gg.h の 2117 行目に定義があります。

呼び出し関係図:

7.4.3.26 loadNormal() [2/2]

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

法線変換行列を格納する.

引数

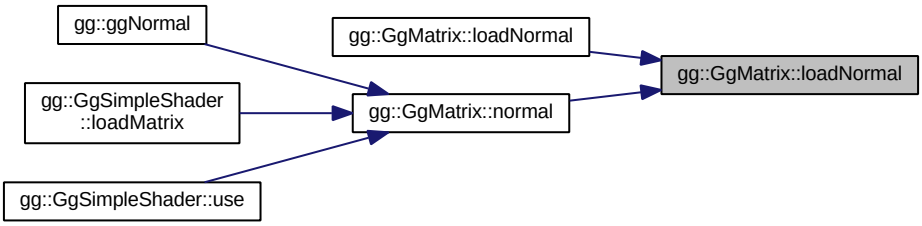
a	GLfloat 型の 16 要素の変換行列.

戻り値

設定した m の法線変換行列.

gg.cpp の 4487 行目に定義があります。

被呼び出し関係図:



7.4.3.27 loadOrthogonal()

```
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

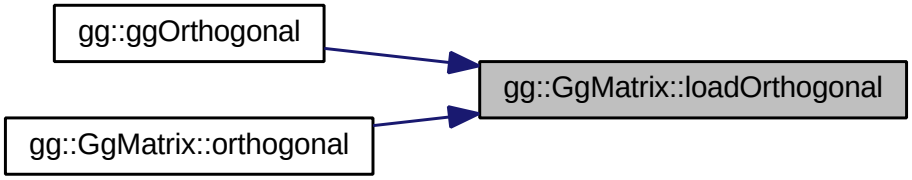
left	ウィンドウの左端の位置.
right	ウィンドウの右端の位置.
bottom	ウィンドウの下端の位置.
top	ウィンドウの上端の位置.
zNear	視点から前方面までの位置.
zFar	視点から後方面までの位置.

戻り値

設定した直交投影変換行列.

gg.cpp の 4563 行目に定義があります。

被呼び出し関係図:



7.4.3.28 loadPerspective()

```
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

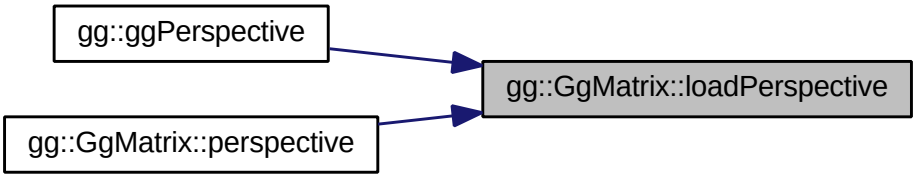
fovy	y 方向の画角.
aspect	縦横比.
zNear	視点から前方面までの位置.
zFar	視点から後方面までの位置.

戻り値

設定した透視投影変換行列.

gg.cpp の 4617 行目に定義があります。

被呼び出し関係図:



7.4.3.29 loadRotate() [1/5]

```
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 2010 行目に定義があります。

呼び出し関係図:

7.4.3.30 loadRotate() [2/5]

```
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 1994 行目に定義があります。

呼び出し関係図:

7.4.3.31 loadRotate() [3/5]

```
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 2002 行目に定義があります。

呼び出し関係図:

7.4.3.32 loadRotate() [4/5]

```
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 1985 行目に定義があります。

呼び出し関係図:

7.4.3.33 loadRotate() [5/5]

```
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

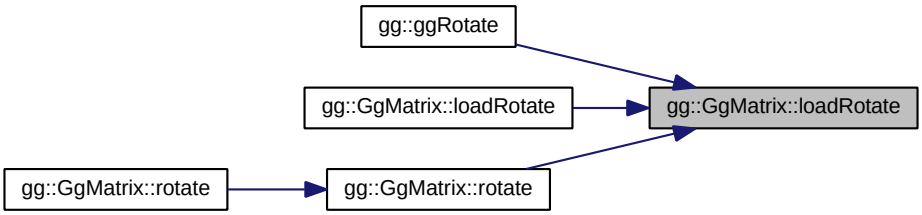
x	回転軸の x 成分.
y	回転軸の y 成分.
z	回転軸の z 成分.
a	回転角.

戻り値

設定した変換行列.

gg.cpp の 4341 行目に定義があります。

被呼び出し関係図:



7.4.3.34 loadRotateX()

```
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

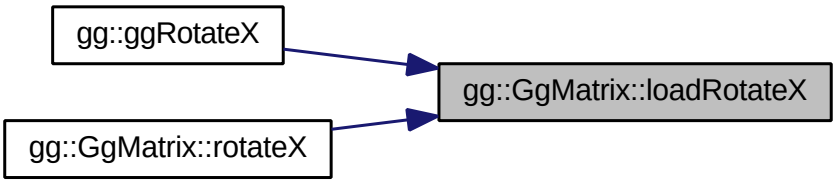
a	回転角.

戻り値

設定した変換行列.

gg.cpp の 4293 行目に定義があります。

被呼び出し関係図:



7.4.3.35 loadRotateY()

```
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

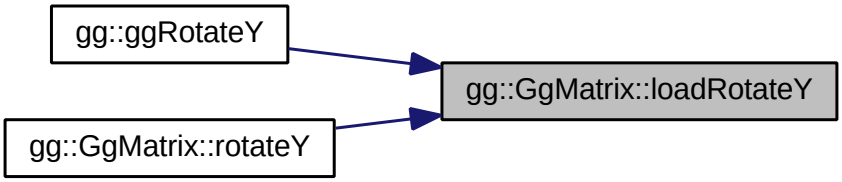
a	回転角.

戻り値

設定した変換行列.

gg.cpp の 4309 行目に定義があります。

被呼び出し関係図:



7.4.3.36 loadRotateZ()

```
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

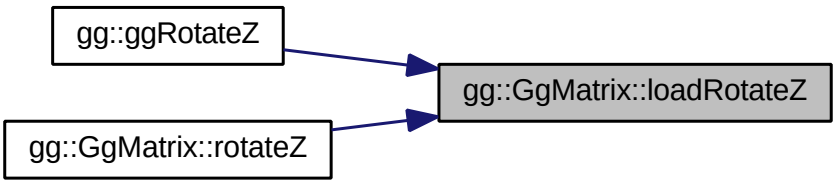
a	回転角.

戻り値

設定した変換行列.

gg.cpp の 4325 行目に定義があります。

被呼び出し関係図:



7.4.3.37 loadScale() [1/3]

```
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 1953 行目に定義があります。

呼び出し関係図:

7.4.3.38 loadScale() [2/3]

```
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 1945 行目に定義があります。

呼び出し関係図:

7.4.3.39 loadScale() [3/3]

```
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

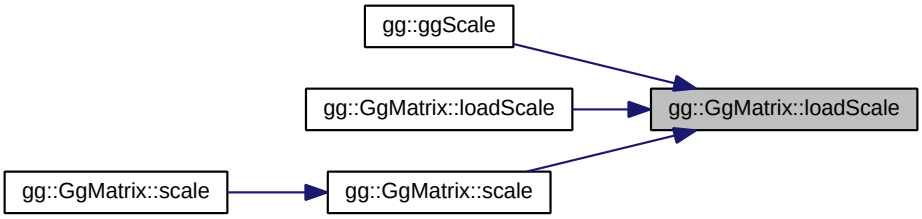
x	x 方向の拡大率.
y	y 方向の拡大率.
z	z 方向の拡大率.
w	w 拡大率のスケールファクタ (= 1.0f).

戻り値

設定した変換行列.

gg.cpp の 4277 行目に定義があります。

被呼び出し関係図:



7.4.3.40 loadSubtract() [1/2]

```
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 1725 行目に定義があります。

呼び出し関係図:

7.4.3.41 loadSubtract() [2/2]

```
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

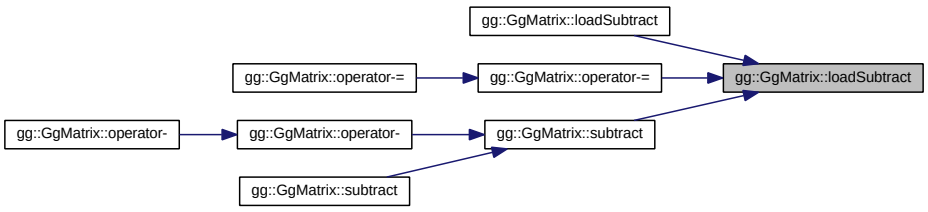
a	GLfloat 型の 16 要素の配列変数.

戻り値

変換行列に a を引いた GgMatrix 型の値.

gg.h の 1716 行目に定義があります。

被呼び出し関係図:



7.4.3.42 loadTranslate() [1/3]

```
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 1929 行目に定義があります。

呼び出し関係図:

7.4.3.43 loadTranslate() [2/3]

```
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 1921 行目に定義があります。

呼び出し関係図:

7.4.3.44 loadTranslate() [3/3]

```
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

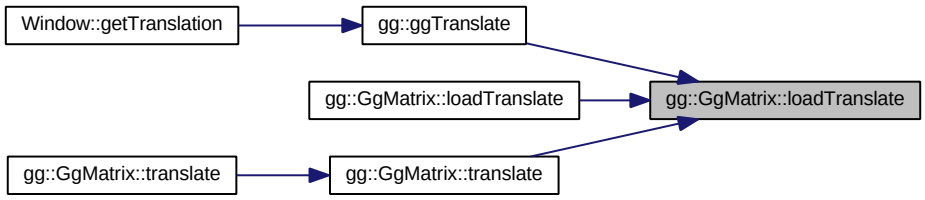
x	x 方向の移動量.
y	y 方向の移動量.
z	z 方向の移動量.
w	w 移動量のスケールファクタ (= 1.0f).

戻り値

設定した変換行列.

gg.cpp の 4261 行目に定義があります。

被呼び出し関係図:



7.4.3.45 loadTranspose() [1/2]

```
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 2091 行目に定義があります。

呼び出し関係図:

7.4.3.46 loadTranspose() [2/2]

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

転置行列を格納する.

引数

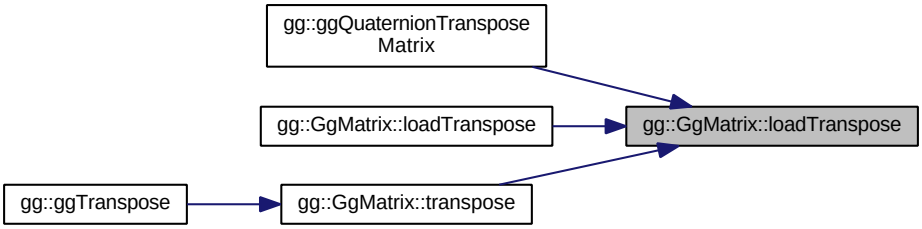
a	GLfloat 型の 16 要素の変換行列.

戻り値

設定した a の転置行列.

gg.cpp の 4380 行目に定義があります。

被呼び出し関係図:



7.4.3.47 lookat() [1/3]

```
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

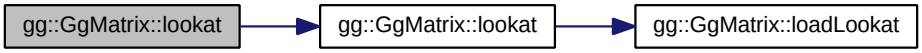
e	視点の位置を格納した GgVector 型の変数.
t	目標点の位置を格納した GgVector 型の変数.
u	上方向のベクトルを格納した GgVector 型の変数.

戻り値

ビュー変換行列を乗じた変換行列.

gg.h の 2285 行目に定義があります。

呼び出し関係図:



7.4.3.48 lookat() [2/3]

```
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 2275 行目に定義があります。

呼び出し関係図:

7.4.3.49 lookat() [3/3]

```
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

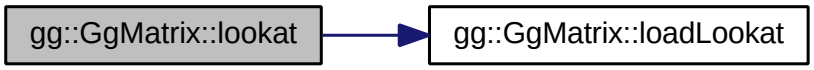
ex	視点の位置の x 座標値.
ey	視点の位置の y 座標値.
ez	視点の位置の z 座標値.
tx	目標点の位置の x 座標値.
ty	目標点の位置の y 座標値.
tz	目標点の位置の z 座標値.
ux	上方向のベクトルの x 成分.
uy	上方向のベクトルの y 成分.
uz	上方向のベクトルの z 成分.

戻り値

ビュー変換行列を乗じた変換行列.

gg.h の 2262 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:

7.4.3.50 multiply() [1/2]

```
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 1809 行目に定義があります。

7.4.3.51 multiply() [2/2]

```
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 1799 行目に定義があります。

7.4.3.52 normal()

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

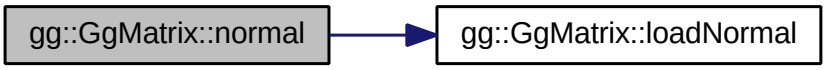
法線変換行列を返す.

戻り値

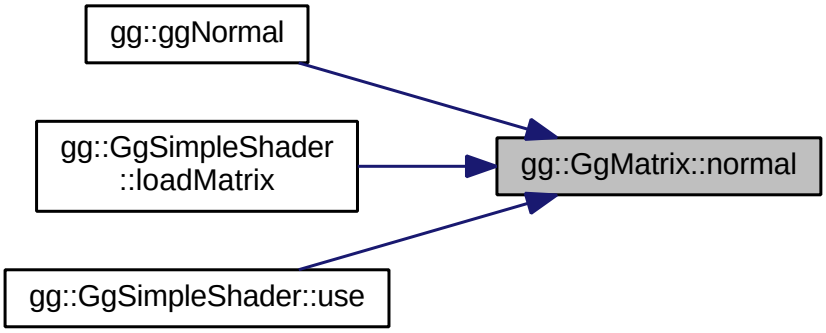
法線変換行列.

gg.h の 2353 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:



7.4.3.53 operator∗() [1/3]

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

gg.h の 1894 行目に定義があります。

呼び出し関係図:

7.4.3.54 operator∗() [2/3]

```
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 2394 行目に定義があります。

7.4.3.55 operator∗() [3/3]

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

gg.h の 1890 行目に定義があります。

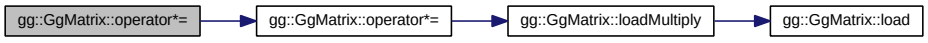
被呼び出し関係図:

7.4.3.56 operator*=() [1/2]

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

gg.h の 1862 行目に定義があります。

呼び出し関係図:

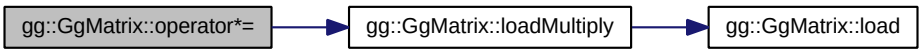


7.4.3.57 operator*=() [2/2]

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

gg.h の 1858 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:

7.4.3.58 operator+() [1/2]

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

gg.h の 1878 行目に定義があります。

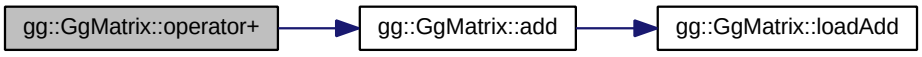
呼び出し関係図:

7.4.3.59 operator+() [2/2]

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

gg.h の 1874 行目に定義があります。

呼び出し関係図:



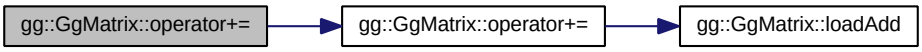
被呼び出し関係図:

7.4.3.60 operator+=() [1/2]

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

gg.h の 1846 行目に定義があります。

呼び出し関係図:



7.4.3.61 operator+=() [2/2]

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

gg.h の 1842 行目に定義があります。

呼び出し関係図:

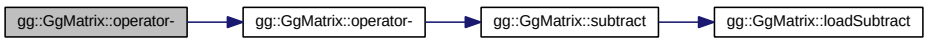
被呼び出し関係図:

7.4.3.62 operator-() [1/2]

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

gg.h の 1886 行目に定義があります。

呼び出し関係図:

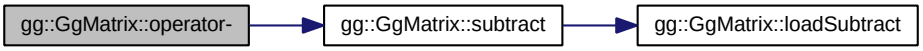


7.4.3.63 operator-() [2/2]

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

gg.h の 1882 行目に定義があります。

呼び出し関係図:



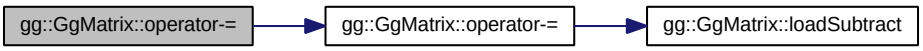
被呼び出し関係図:

7.4.3.64 operator-=() [1/2]

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

gg.h の 1854 行目に定義があります。

呼び出し関係図:



7.4.3.65 operator-=() [2/2]

```
GgMatrix& gg::GgMatrix::operator-= (
            const GLfloat * a )  [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:

7.4.3.66 operator/() [1/2]

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

gg.h の 1902 行目に定義があります。

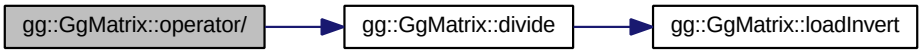
呼び出し関係図:

7.4.3.67 operator/() [2/2]

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

gg.h の 1898 行目に定義があります。

呼び出し関係図:



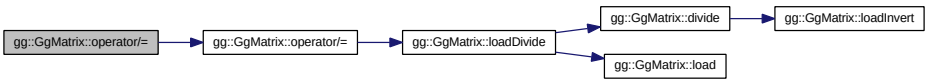
被呼び出し関係図:

7.4.3.68 operator/=() [1/2]

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

gg.h の 1870 行目に定義があります。

呼び出し関係図:

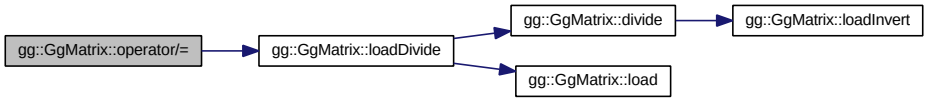


7.4.3.69 operator/=() [2/2]

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

gg.h の 1866 行目に定義があります。

呼び出し関係図:



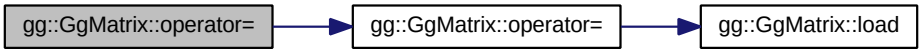
被呼び出し関係図:

7.4.3.70 operator=() [1/2]

```
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:

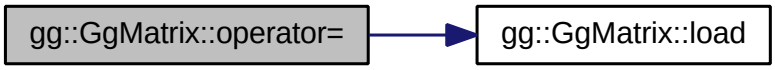


7.4.3.71 operator=() [2/2]

```
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:

7.4.3.72 operator[]() [1/2]

```
GLfloat& gg::GgMatrix::operator[] (
            std::size_t i )  [inline]
```

変換行列の要素にアクセスする.

戻り値

変換行列を格納した GLfloat 型の 16 要素の配列変数 の i 番目の要素の参照.

gg.h の 2431 行目に定義があります。

7.4.3.73 operator[]() [2/2]

```
const GLfloat& gg::GgMatrix::operator[] (
            std::size_t i ) const  [inline]
```

変換行列の要素にアクセスする.

戻り値

変換行列を格納した GLfloat 型の 16 要素の配列変数 の i 番目の要素の参照.

gg.h の 2424 行目に定義があります。

7.4.3.74 orthogonal()

```
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

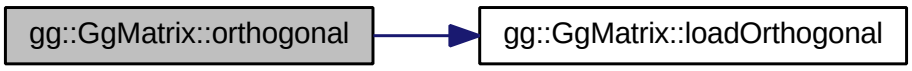
left	ウィンドウの左端の位置.
right	ウィンドウの右端の位置.
bottom	ウィンドウの下端の位置.
top	ウィンドウの上端の位置.
zNear	視点から前方面までの位置.
zFar	視点から後方面までの位置.

戻り値

直交投影変換行列を乗じた変換行列.

gg.h の 2298 行目に定義があります。

呼び出し関係図:



7.4.3.75 perspective()

```
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

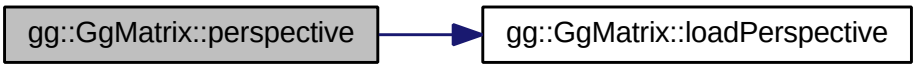
fovy	y 方向の画角.
aspect	縦横比.
zNear	視点から前方面までの位置.
zFar	視点から後方面までの位置.

戻り値

透視投影変換行列を乗じた変換行列.

gg.h の 2328 行目に定義があります。

呼び出し関係図:



7.4.3.76 projection() [1/4]

```
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 2386 行目に定義があります。

7.4.3.77 projection() [2/4]

```
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 2378 行目に定義があります。

7.4.3.78 projection() [3/4]

```
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 2370 行目に定義があります。

7.4.3.79 projection() [4/4]

```
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 2362 行目に定義があります。

7.4.3.80 rotate() [1/5]

```
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 2246 行目に定義があります。

呼び出し関係図:

7.4.3.81 rotate() [2/5]

```
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 2230 行目に定義があります。

呼び出し関係図:

7.4.3.82 rotate() [3/5]

```
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 2238 行目に定義があります。

呼び出し関係図:

7.4.3.83 rotate() [4/5]

```
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 2221 行目に定義があります。

呼び出し関係図:

7.4.3.84 rotate() [5/5]

```
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

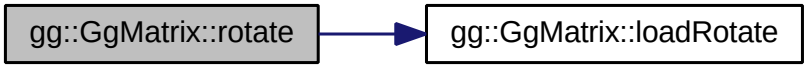
x	回転軸の x 成分.
y	回転軸の y 成分.
z	回転軸の z 成分.
a	回転角.

戻り値

(x, y, z) を軸にさらに a 回転した変換行列.

gg.h の 2211 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:

7.4.3.85 rotateX()

```
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

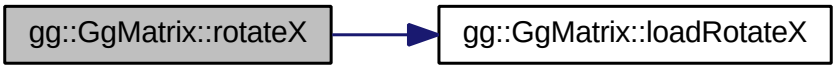
a	回転角.

戻り値

x 軸中心にさらに a 回転した変換行列.

gg.h の 2181 行目に定義があります。

呼び出し関係図:



7.4.3.86 rotateY()

```
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

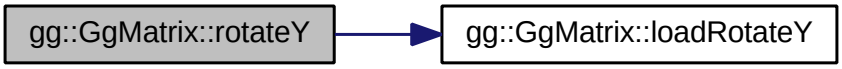
a	回転角.

戻り値

y 軸中心にさらに a 回転した変換行列.

gg.h の 2190 行目に定義があります。

呼び出し関係図:



7.4.3.87 rotateZ()

```
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

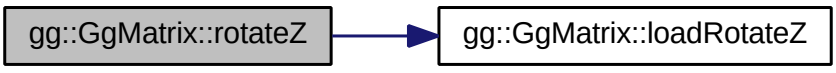
a	回転角.

戻り値

z 軸中心にさらに a 回転した変換行列.

gg.h の 2199 行目に定義があります。

呼び出し関係図:



7.4.3.88 scale() [1/3]

```
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

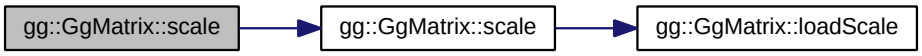
s	拡大率の GgVector 型の変数.

戻り値

拡大縮小した結果の変換行列.

gg.h の 2173 行目に定義があります。

呼び出し関係図:



7.4.3.89 scale() [2/3]

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

拡大縮小変換を乗じた結果を返す.

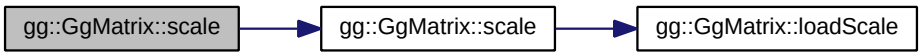
引数

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 2165 行目に定義があります。
呼び出し関係図:



7.4.3.90 scale() [3/3]

```
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

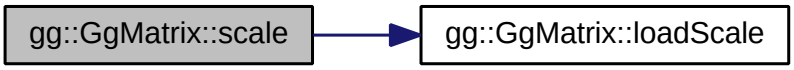
x	x 方向の拡大率.
y	y 方向の拡大率.
z	z 方向の拡大率.
w	w 移動量のスケールファクタ (= 1.0f).

戻り値

拡大縮小した結果の変換行列.

gg.h の 2156 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:

7.4.3.91 subtract() [1/2]

```
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

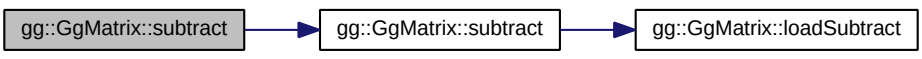
m	GgMatrix 型の変数.

戻り値

変換行列に m を引いた GgMatrix 型の値.

gg.h の 1791 行目に定義があります。

呼び出し関係図:



7.4.3.92 subtract() [2/2]

```
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

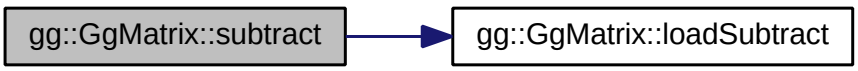
a	GLfloat 型の 16 要素の配列変数.

戻り値

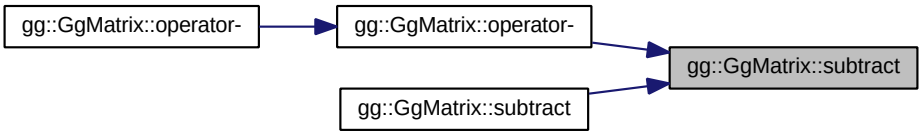
変換行列に a を引いた GgMatrix 型の値.

gg.h の 1782 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:



7.4.3.93 translate() [1/3]

```
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

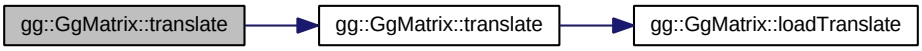
t	移動量の GgVector 型の変数.

戻り値

平行移動した結果の変換行列.

gg.h の 2145 行目に定義があります。

呼び出し関係図:



7.4.3.94 translate() [2/3]

```
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

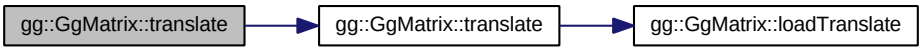
t	移動量の GLfloat 型の 3 要素の配列変数 (x, y, z).

戻り値

平行移動した結果の変換行列.

gg.h の 2137 行目に定義があります。

呼び出し関係図:



7.4.3.95 translate() [3/3]

```
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

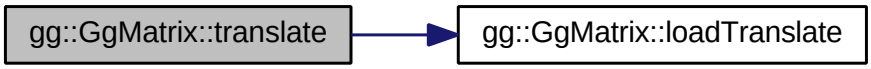
x	x 方向の移動量.
y	y 方向の移動量.
z	z 方向の移動量.
w	w 移動量のスケールファクタ (= 1.0f).

戻り値

平行移動した結果の変換行列.

gg.h の 2128 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:

7.4.3.96 transpose()

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

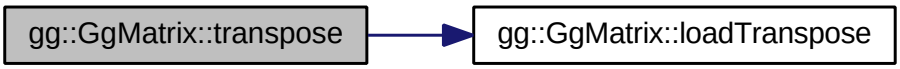
転置行列を返す.

戻り値

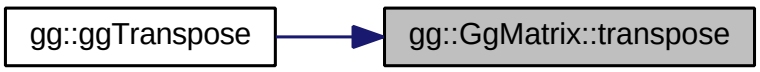
転置行列.

gg.h の 2337 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:



7.4.4 フレンドと関連関数の詳解

7.4.4.1 GgQuaternion

```
friend class GgQuaternion  [friend]
```

gg.h の 1655 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

7.5 gg::GgNormalTexture クラス

法線マップ.

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

公開メンバ関数

- 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.
- virtual ~GgNormalTexture ()
  デストラクタ.
- 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.

7.5.1 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 4014 行目に定義があります。

7.5.2 構築子と解体子

7.5.2.1 GgNormalTexture() [1/3]

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

コンストラクタ.

gg.h の 4022 行目に定義があります。

7.5.2.2 GgNormalTexture() [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]
```

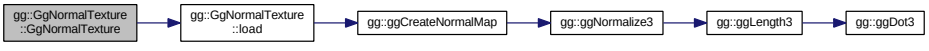
メモリ上のデータから法線マップのテクスチャを作成するコンストラクタ.

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 4031 行目に定義があります。

呼び出し関係図:



7.5.2.3 GgNormalTexture() [3/3]

```
gg::GgNormalTexture::GgNormalTexture (
            const char * name,
```

```
            float nz = 1.0f,
            GLenum internal = GL_RGBA )  [inline]
```

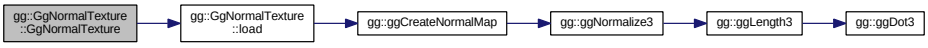
ファイルからデータを読み込んで法線マップのテクスチャを作成するコンストラクタ.

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 4042 行目に定義があります。

呼び出し関係図:



7.5.2.4 ~GgNormalTexture()

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

デストラクタ.

gg.h の 4049 行目に定義があります。

7.5.3 関数詳解

7.5.3.1 load() [1/2]

```
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 3163 行目に定義があります。

呼び出し関係図:

7.5.3.2 load() [2/2]

```
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.

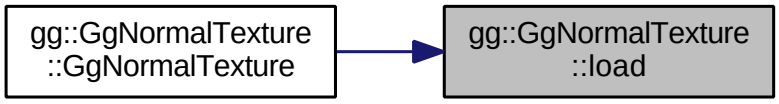
引数

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 4058 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:



このクラス詳解は次のファイルから抽出されました:

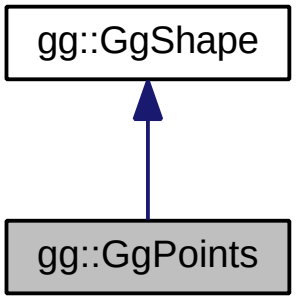
- gg.h
- gg.cpp

7.6 gg::GgPoints クラス

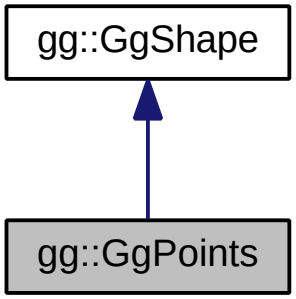
点.

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

gg::GgPoints の継承関係図



gg::GgPoints 連携図



公開メンバ関数

- 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_STATI↩ C_DRAW)
  コンストラクタ.
- virtual ~GgPoints ()
  デストラクタ.
- 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.

7.6.1 詳解

点.

gg.h の 4567 行目に定義があります。

7.6.2 構築子と解体子

7.6.2.1 GgPoints() [1/2]

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

コンストラクタ.

gg.h の 4576 行目に定義があります。

7.6.2.2 GgPoints() [2/2]

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

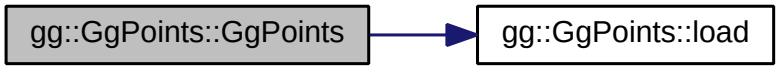
コンストラクタ.

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 4585 行目に定義があります。

呼び出し関係図:



7.6.2.3 ～GgPoints()

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

デストラクタ.

gg.h の 4592 行目に定義があります。

7.6.3 関数詳解

7.6.3.1 draw()

```
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 4973 行目に定義があります。

呼び出し関係図:

7.6.3.2 getBuffer()

```
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 4603 行目に定義があります。

7.6.3.3 getCount()

```
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 4596 行目に定義があります。

7.6.3.4 load()

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

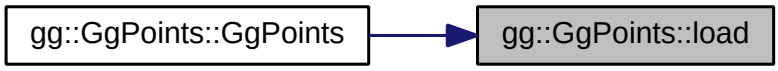
バッファオブジェクトを確保して頂点の位置データを格納する.

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 4621 行目に定義があります。

被呼び出し関係図:



7.6.3.5 send()

```
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 4612 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:

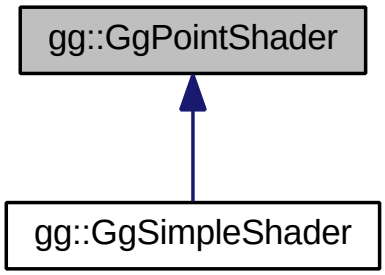
- gg.h
- gg.cpp

7.7 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

- 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 ~GgPointShader ()
  デストラクタ.
- virtual void loadProjectionMatrix (const GLfloat *mp) const
  投影変換行列を設定する.
- virtual void loadProjectionMatrix (const GgMatrix &mp) const
  投影変換行列を設定する.
- virtual void loadModelviewMatrix (const GLfloat *mv) const
  モデルビュー変換行列を設定する.
- virtual void loadModelviewMatrix (const GgMatrix &mv) const
  モデルビュー変換行列を設定する.

- virtual void loadMatrix (const GLfloat *mp, const GLfloat *mv) const
  投影変換行列とモデルビュー変換行列を設定する.
- virtual void loadMatrix (const GgMatrix &mp, const GgMatrix &mv) const
  投影変換行列とモデルビュー変換行列を設定する.
- virtual void use () const
  シェーダプログラムの使用を開始する.
- void use (const GLfloat *mp) const
  投影変換行列を設定してシェーダプログラムの使用を開始する.
- void use (const GgMatrix &mp) const
  投影変換行列を設定してシェーダプログラムの使用を開始する.
- void use (const GLfloat *mp, const GLfloat *mv) const
  投影変換行列とモデルビューを設定してシェーダプログラムの使用を開始する.
- void use (const GgMatrix &mp, const GgMatrix &mv) const
  投影変換行列とモデルビューを設定して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.

7.7.1 詳解

点のシェーダ.

gg.h の 4988 行目に定義があります。

7.7.2 構築子と解体子

7.7.2.1 GgPointShader() [1/2]

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

コンストラクタ.

gg.h の 5002 行目に定義があります。

7.7.2.2 GgPointShader() [2/2]

```
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 5010 行目に定義があります。

7.7.2.3 ~GgPointShader()

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

デストラクタ.

gg.h の 5025 行目に定義があります。

7.7.3 関数詳解

7.7.3.1 get()

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

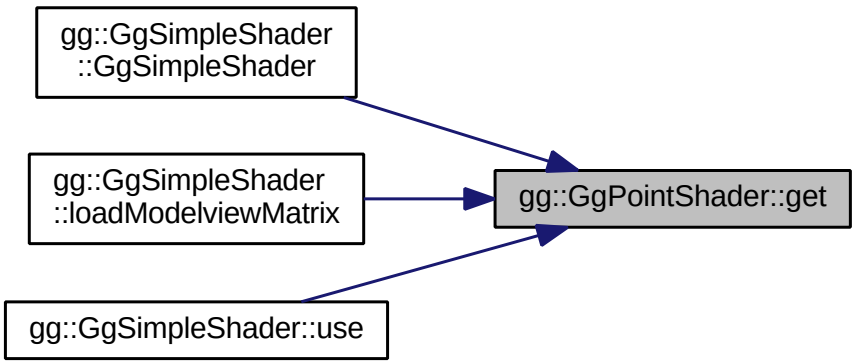
シェーダのプログラム名を得る.

戻り値

シェーダのプログラム名.

gg.h の 5118 行目に定義があります。

被呼び出し関係図:



7.7.3.2 loadMatrix() [1/2]

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

投影変換行列とモデルビュー変換行列を設定する.

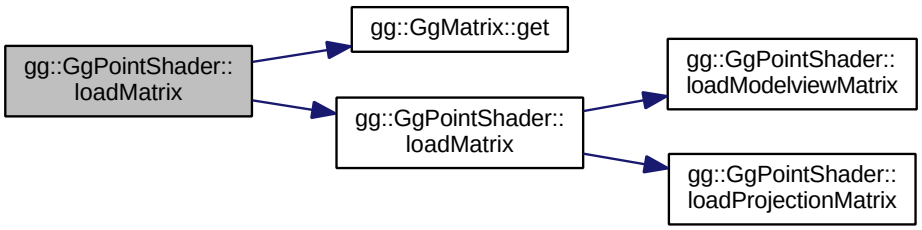
引数

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 5067 行目に定義があります。

呼び出し関係図:



7.7.3.3 loadMatrix() [2/2]

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

投影変換行列とモデルビュー変換行列を設定する.

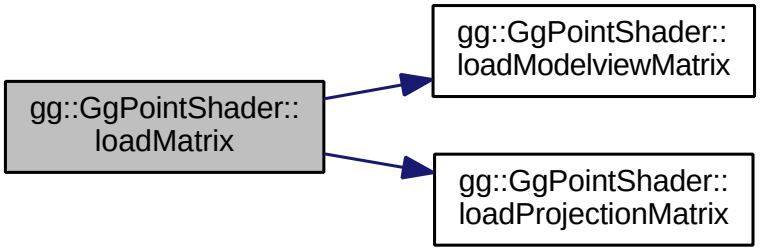
引数

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.

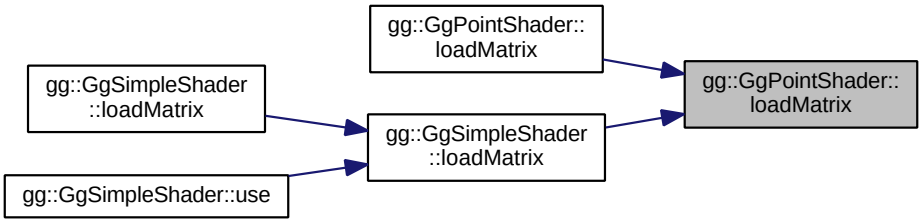
gg::GgSimpleShaderで再実装されています。

gg.h の 5058 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:



7.7.3.4 loadModelviewMatrix() [1/2]

```
virtual void gg::GgPointShader::loadModelviewMatrix (
            const GgMatrix & mv ) const  [inline], [virtual]
```

モデルビュー変換行列を設定する.

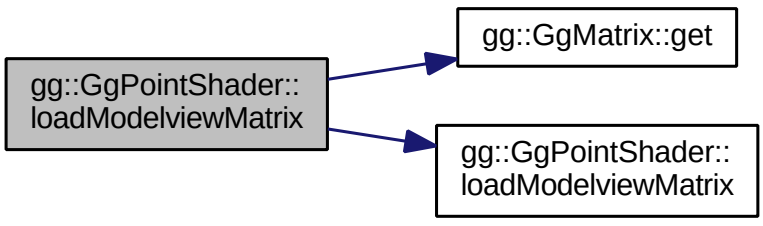
引数

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 5050 行目に定義があります。

呼び出し関係図:



7.7.3.5 loadModelviewMatrix() [2/2]

```
virtual void gg::GgPointShader::loadModelviewMatrix (
            const GLfloat * mv ) const  [inline], [virtual]
```

モデルビュー変換行列を設定する.

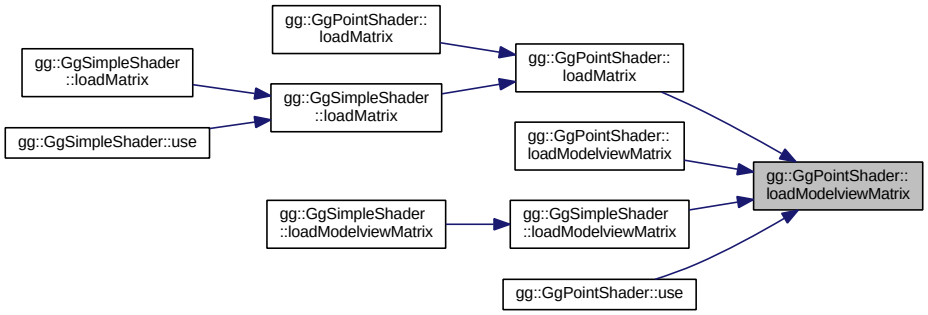
引数

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 5043 行目に定義があります。

被呼び出し関係図:



7.7.3.6 loadProjectionMatrix() [1/2]

```
virtual void gg::GgPointShader::loadProjectionMatrix (
            const GgMatrix & mp ) const  [inline], [virtual]
```

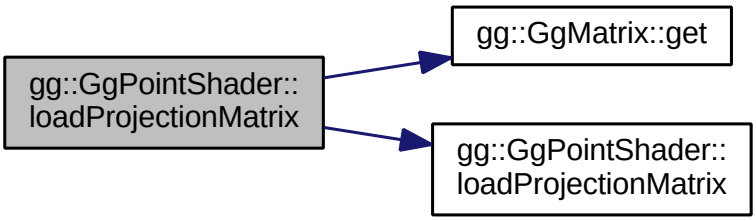
投影変換行列を設定する.

引数

mp	GgMatrix 型の投影変換行列.

gg.h の 5036 行目に定義があります。

呼び出し関係図:



7.7.3.7 loadProjectionMatrix() [2/2]

```
virtual void gg::GgPointShader::loadProjectionMatrix (
            const GLfloat * mp ) const  [inline], [virtual]
```

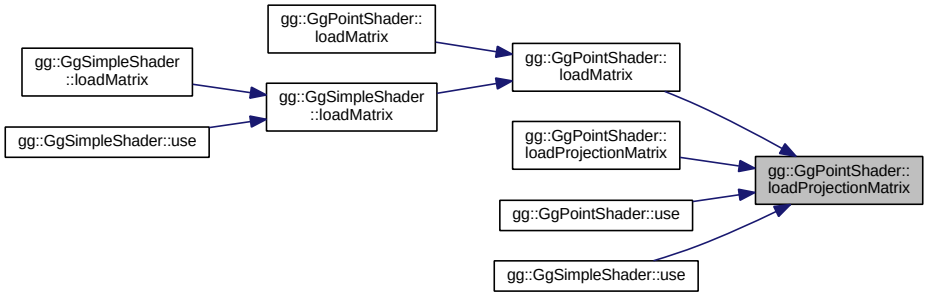
投影変換行列を設定する.

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.

gg.h の 5029 行目に定義があります。

被呼び出し関係図:



7.7.3.8 unuse()

```
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 5111 行目に定義があります。

7.7.3.9 use() [1/5]

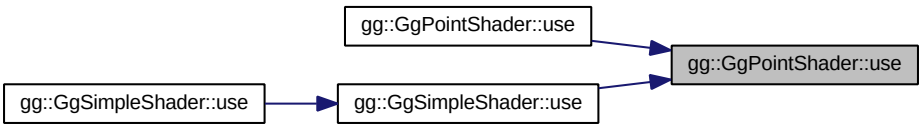
```
virtual void gg::GgPointShader::use ( ) const  [inline], [virtual]
```

シェーダプログラムの使用を開始する.

gg::GgSimpleShaderで再実装されています。

gg.h の 5073 行目に定義があります。

被呼び出し関係図:



7.7.3.10 use() [2/5]

```
void gg::GgPointShader::use (
            const GgMatrix & mp ) const  [inline]
```

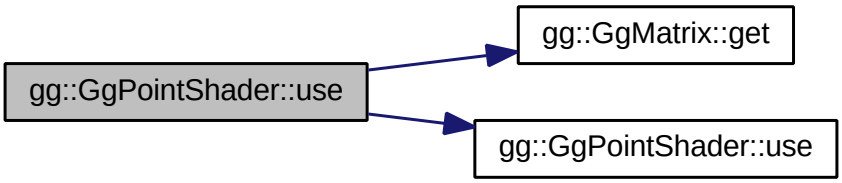
投影変換行列を設定してシェーダプログラムの使用を開始する.

引数

mp	GgMatrix 型の投影変換行列.

gg.h の 5088 行目に定義があります。

呼び出し関係図:



7.7.3.11 use() [3/5]

```
void gg::GgPointShader::use (
            const GgMatrix & mp,
            const GgMatrix & mv ) const  [inline]
```

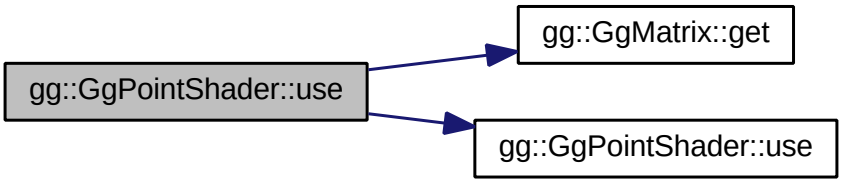
投影変換行列とモデルビューを設定してシェーダプログラムの使用を開始する.

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.h の 5105 行目に定義があります。

呼び出し関係図:



7.7.3.12 use() [4/5]

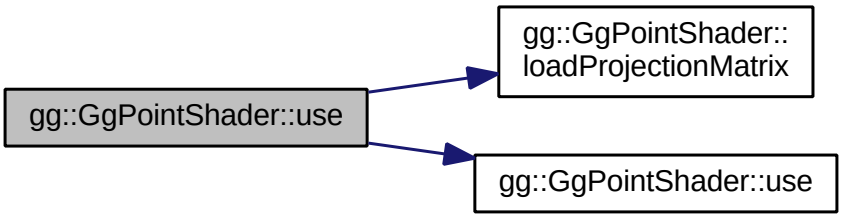
```
void gg::GgPointShader::use (
            const GLfloat * mp ) const  [inline]
```

投影変換行列を設定してシェーダプログラムの使用を開始する.

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.

gg.h の 5080 行目に定義があります。
呼び出し関係図:



7.7.3.13 use() [5/5]

```
void gg::GgPointShader::use (
            const GLfloat * mp,
            const GLfloat * mv ) const  [inline]
```

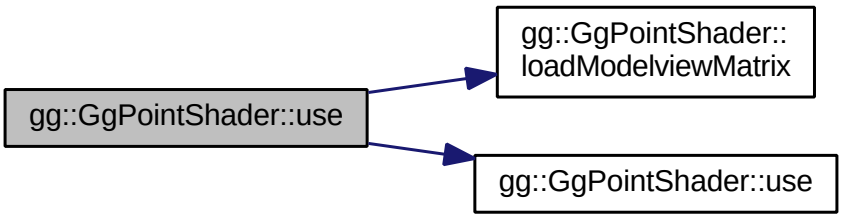
投影変換行列とモデルビューを設定してシェーダプログラムの使用を開始する.

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.h の 5096 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

7.8 gg::GgQuaternion クラス

四元数.

`#include <gg.h>`

公開メンバ関数

- 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.
- ~GgQuaternion ()
  デストラクタ.
- 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.

7.8.1 詳解

四元数.

gg.h の 2705 行目に定義があります。

7.8.2 構築子と解体子

7.8.2.1 GgQuaternion() [1/5]

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

コンストラクタ.

gg.h の 2725 行目に定義があります。

7.8.2.2 GgQuaternion() [2/5]

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

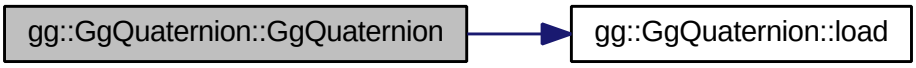
コンストラクタ.

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 2732 行目に定義があります。

呼び出し関係図:



7.8.2.3 GgQuaternion() [3/5]

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

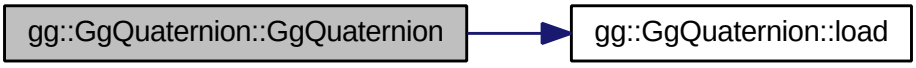
コンストラクタ.

引数

v	四元数を格納した GgVector 型の変数.

gg.h の 2739 行目に定義があります。

呼び出し関係図:



7.8.2.4 GgQuaternion() [4/5]

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

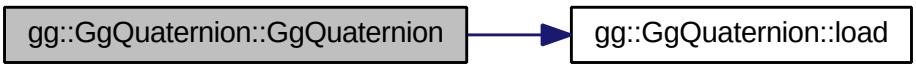
コンストラクタ.

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 2746 行目に定義があります。

呼び出し関係図:



7.8.2.5 GgQuaternion() [5/5]

```
gg::GgQuaternion::GgQuaternion (
            const GgQuaternion & q ) [inline]
```

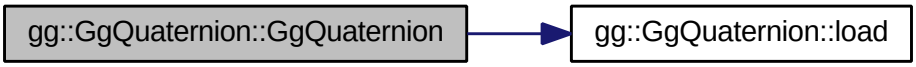
コピーコンストラクタ.

引数

q	GgQuaternion 型の四元数.

gg.h の 2753 行目に定義があります。

呼び出し関係図:



7.8.2.6 ~GgQuaternion()

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

デストラクタ.

gg.h の 2759 行目に定義があります。

7.8.3 関数詳解

7.8.3.1 add() [1/4]

```
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 2993 行目に定義があります。

呼び出し関係図:

7.8.3.2 add() [2/4]

```
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 2977 行目に定義があります。

呼び出し関係図:

7.8.3.3 add() [3/4]

```
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 2985 行目に定義があります。

呼び出し関係図:

7.8.3.4 add() [4/4]

```
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

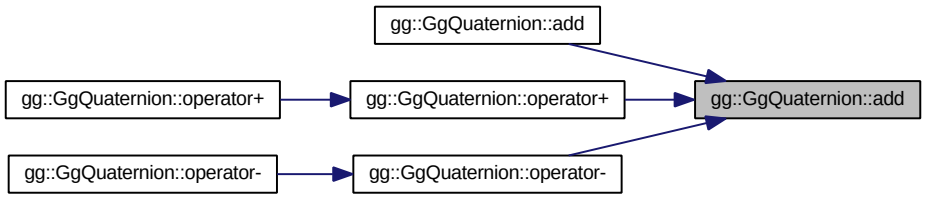
x	加える四元数の x 要素.
y	加える四元数の y 要素.
z	加える四元数の z 要素.
w	加える四元数の w 要素.

戻り値

(x, y, z, w) を加えた四元数.

gg.h の 2964 行目に定義があります。

被呼び出し関係図:



7.8.3.5 conjugate()

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

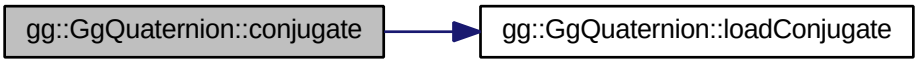
共役四元数に変換する.

戻り値

共役四元数.

gg.h の 3453 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:

7.8.3.6 divide() [1/4]

```
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 3110 行目に定義があります。

呼び出し関係図:

7.8.3.7 divide() [2/4]

```
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 3091 行目に定義があります。

呼び出し関係図:

7.8.3.8 divide() [3/4]

```
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 3099 行目に定義があります。

呼び出し関係図:

7.8.3.9 divide() [4/4]

```
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

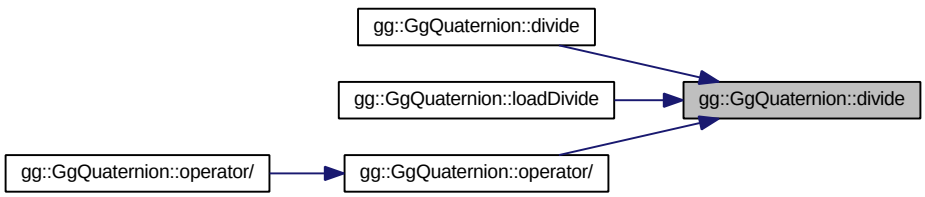
x	割る四元数の x 要素.
y	割る四元数の y 要素.
z	割る四元数の z 要素.
w	割る四元数の w 要素.

戻り値

(x, y, z, w) を割った四元数.

gg.h の 3082 行目に定義があります。

被呼び出し関係図:



7.8.3.10 euler() [1/2]

```
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

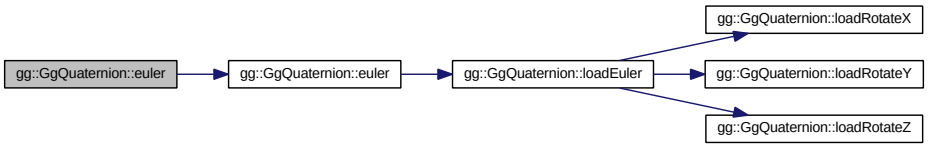
e	オイラー角を表す GLfloat 型の 3 要素の配列変数 (heading, pitch, roll).

戻り値

回転した四元数.

gg.h の 3335 行目に定義があります。

呼び出し関係図:



7.8.3.11 euler() [2/2]

```
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

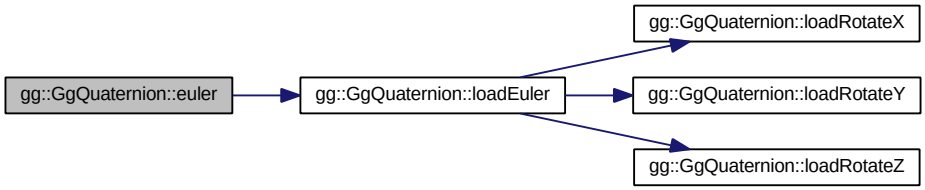
heading	y 軸中心の回転角.
pitch	x 軸中心の回転角.
roll	z 軸中心の回転角.

戻り値

回転した四元数.

gg.h の 3326 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:

7.8.3.12 get() [1/2]

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

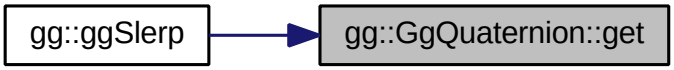
四元数を取り出す.

戻り値

四元数を表す GLfloat 型の 4 要素の配列変数.

gg.h の 3471 行目に定義があります。

被呼び出し関係図:



7.8.3.13 get() [2/2]

```
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 3478 行目に定義があります。

7.8.3.14 getConjugateMatrix() [1/3]

```
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 3527 行目に定義があります。

被呼び出し関係図:

7.8.3.15 getConjugateMatrix() [2/3]

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

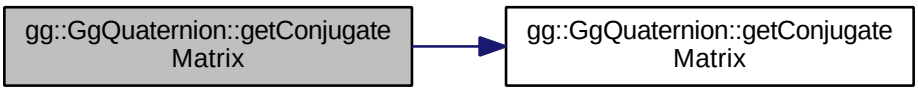
四元数の共役が表す回転の変換行列を m に求める.

引数

m	回転の変換行列を格納する GgMatrix 型の変数.

gg.h の 3520 行目に定義があります。

呼び出し関係図:



7.8.3.16 getConjugateMatrix() [3/3]

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

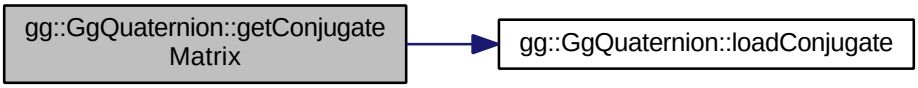
四元数の共役が表す回転の変換行列を a に求める.

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 3511 行目に定義があります。

呼び出し関係図:



7.8.3.17 getMatrix() [1/3]

```
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 3502 行目に定義があります。

被呼び出し関係図:

7.8.3.18 getMatrix() [2/3]

```
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 3495 行目に定義があります。

呼び出し関係図:

7.8.3.19 getMatrix() [3/3]

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

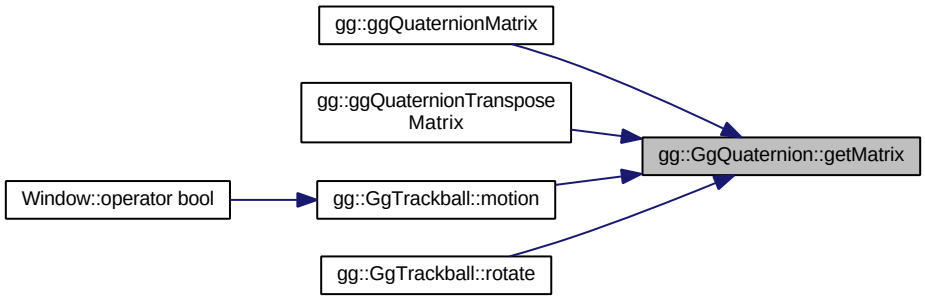
四元数が表す回転の変換行列を a に求める.

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 3488 行目に定義があります。

被呼び出し関係図:



7.8.3.20 invert()

```
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 3462 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:

7.8.3.21 load() [1/4]

```
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 2803 行目に定義があります。

呼び出し関係図:

7.8.3.22 load() [2/4]

```
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 2786 行目に定義があります。

7.8.3.23 load() [3/4]

```
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 2795 行目に定義があります。

呼び出し関係図:

7.8.3.24 load() [4/4]

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

四元数を格納する.

引数

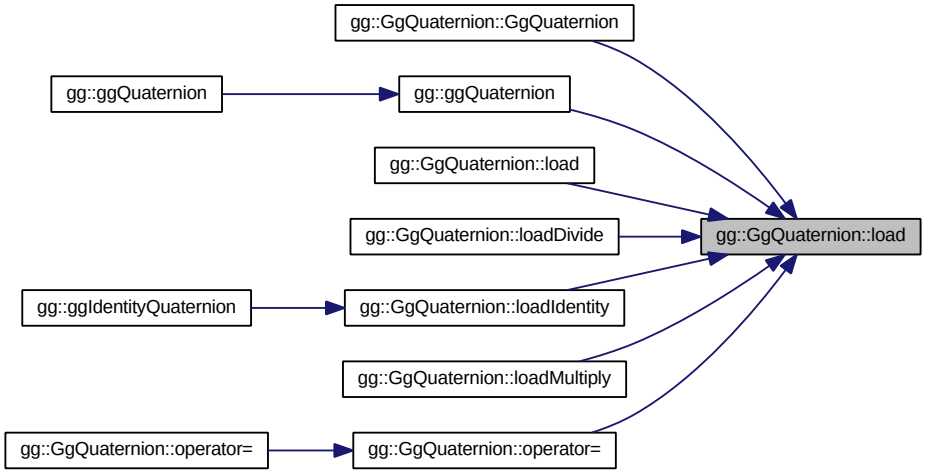
x	四元数の x 要素.
y	四元数の y 要素.
z	四元数の z 要素.
w	四元数の w 要素.

戻り値

設定した四元数.

gg.h の 2774 行目に定義があります。

被呼び出し関係図:



7.8.3.25 loadAdd() [1/4]

```
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 2842 行目に定義があります。

呼び出し関係図:

7.8.3.26 loadAdd() [2/4]

```
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 2826 行目に定義があります。

呼び出し関係図:

7.8.3.27 loadAdd() [3/4]

```
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 2834 行目に定義があります。

呼び出し関係図:

7.8.3.28 loadAdd() [4/4]

```
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

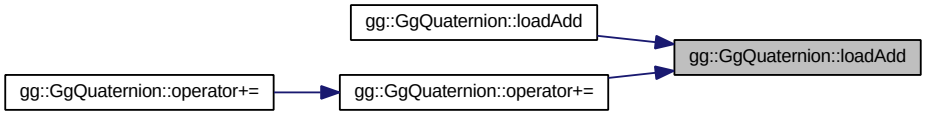
x	加える四元数の x 要素.
y	加える四元数の y 要素.
z	加える四元数の z 要素.
w	加える四元数の w 要素.

戻り値

(x, y, z, w) を加えた四元数.

gg.h の 2814 行目に定義があります。

被呼び出し関係図:



7.8.3.29 loadConjugate() [1/2]

```
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 3402 行目に定義があります。

呼び出し関係図:

7.8.3.30 loadConjugate() [2/2]

```
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

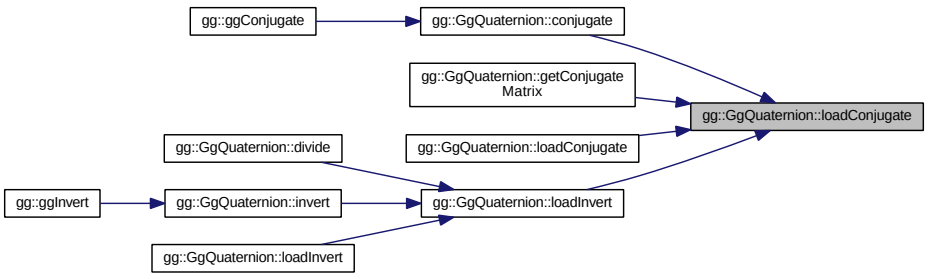
a	四元数を格納した GLfloat 型の 4 要素の配列変数.

戻り値

共役四元数.

gg.cpp の 4825 行目に定義があります。

被呼び出し関係図:



7.8.3.31 loadDivide() [1/4]

```
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 2953 行目に定義があります。

呼び出し関係図:

7.8.3.32 loadDivide() [2/4]

```
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 2937 行目に定義があります。

呼び出し関係図:

7.8.3.33 loadDivide() [3/4]

```
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

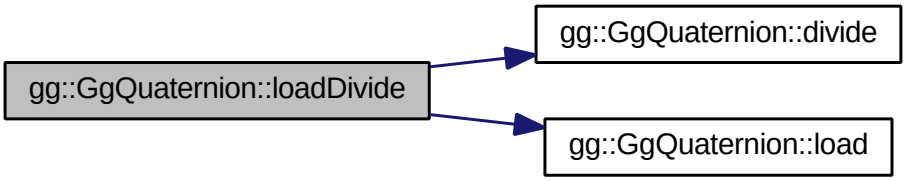
a	四元数を格納した GLfloat 型の 4 要素の配列変数.

戻り値

a で割った四元数.

gg.h の 2945 行目に定義があります。

呼び出し関係図:



7.8.3.34 loadDivide() [4/4]

```
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

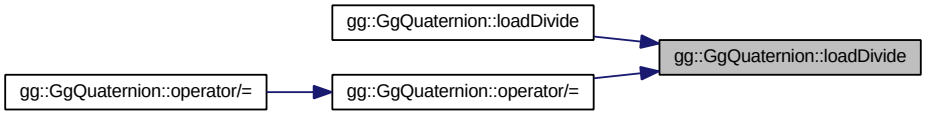
x	割る四元数の x 要素.
y	割る四元数の y 要素.
z	割る四元数の z 要素.
w	割る四元数の w 要素.

戻り値

(x, y, z, w) を割った四元数.

gg.h の 2928 行目に定義があります。

被呼び出し関係図:



7.8.3.35 loadEuler() [1/2]

```
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 3316 行目に定義があります。

呼び出し関係図:

7.8.3.36 loadEuler() [2/2]

```
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

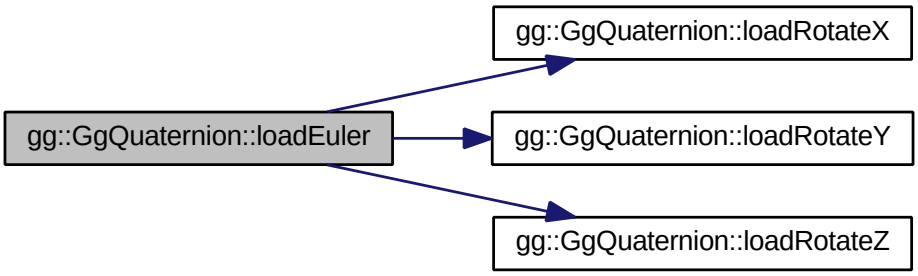
heading	y 軸中心の回転角.
pitch	x 軸中心の回転角.
roll	z 軸中心の回転角.

戻り値

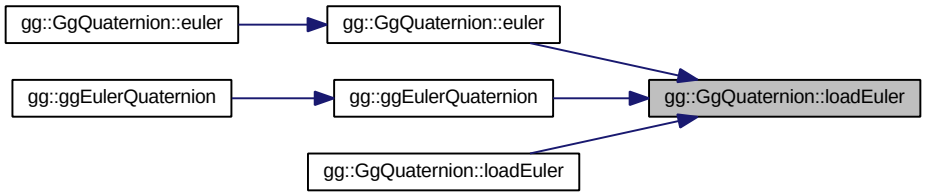
格納した回転を表す四元数.

gg.cpp の 4794 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:



7.8.3.37 loadIdentity()

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

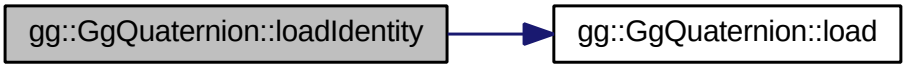
単位元を格納する.

戻り値

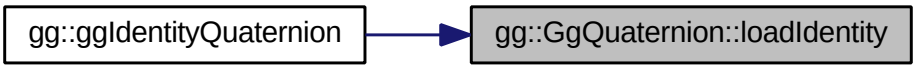
格納された単位元.

gg.h の 3208 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:



7.8.3.38 loadInvert() [1/2]

```
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 3415 行目に定義があります。

呼び出し関係図:

7.8.3.39 loadInvert() [2/2]

```
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

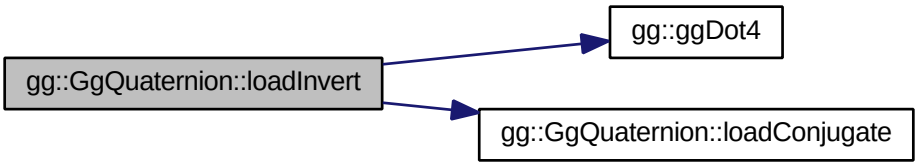
a	四元数を格納した GLfloat 型の 4 要素の配列変数.

戻り値

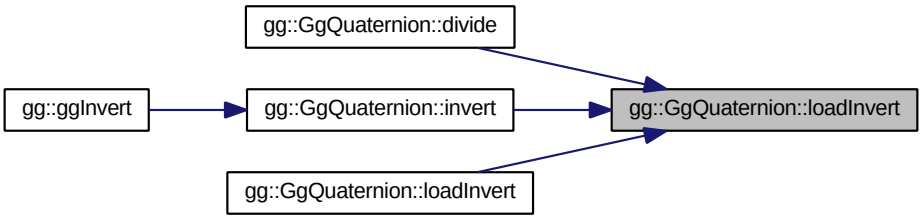
四元数の逆元.

gg.cpp の 4839 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:



7.8.3.40 loadMatrix() [1/2]

```
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

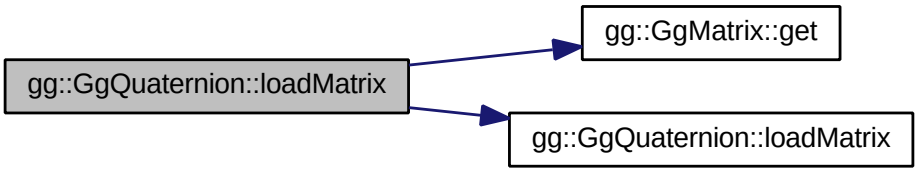
m	Ggmatrix 型の変換行列.

戻り値

m による回転の変換に相当する四元数.

gg.h の 3201 行目に定義があります。

呼び出し関係図:



7.8.3.41 loadMatrix() [2/2]

```
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

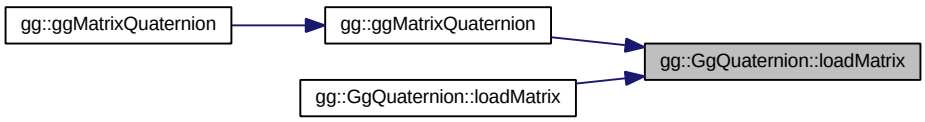
a	GLfloat 型の 16 要素の変換行列.

戻り値

a による回転の変換に相当する四元数.

gg.h の 3192 行目に定義があります。

被呼び出し関係図:



7.8.3.42 loadMultiply() [1/4]

```
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 2917 行目に定義があります。

呼び出し関係図:

7.8.3.43 loadMultiply() [2/4]

```
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 2901 行目に定義があります。

呼び出し関係図:

7.8.3.44 loadMultiply() [3/4]

```
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

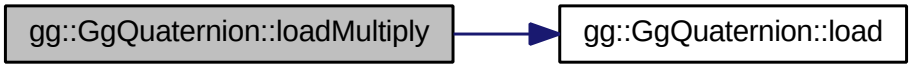
a	四元数を格納した GLfloat 型の 4 要素の配列変数.

戻り値

a を乗じた四元数.

gg.h の 2909 行目に定義があります。

呼び出し関係図:



7.8.3.45 loadMultiply() [4/4]

```
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

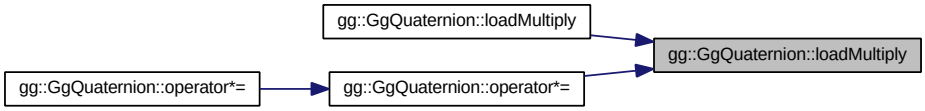
x	掛ける四元数の x 要素.
y	掛ける四元数の y 要素.
z	掛ける四元数の z 要素.
w	掛ける四元数の w 要素.

戻り値

(x, y, z, w) を掛けた四元数.

gg.h の 2892 行目に定義があります。

被呼び出し関係図:



7.8.3.46 loadNormalize() [1/2]

```
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 3389 行目に定義があります。

呼び出し関係図:

7.8.3.47 loadNormalize() [2/2]

```
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

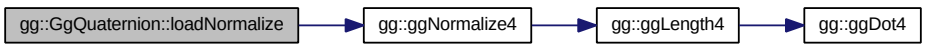
a	四元数を格納した GLfloat 型の 4 要素の配列変数.

戻り値

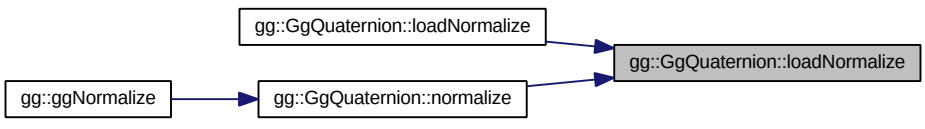
正規化された四元数.

gg.cpp の 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:



7.8.3.48 loadRotate() [1/3]

```
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 3233 行目に定義があります。

呼び出し関係図:

7.8.3.49 loadRotate() [2/3]

```
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 3225 行目に定義があります。

呼び出し関係図:

7.8.3.50 loadRotate() [3/3]

```
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

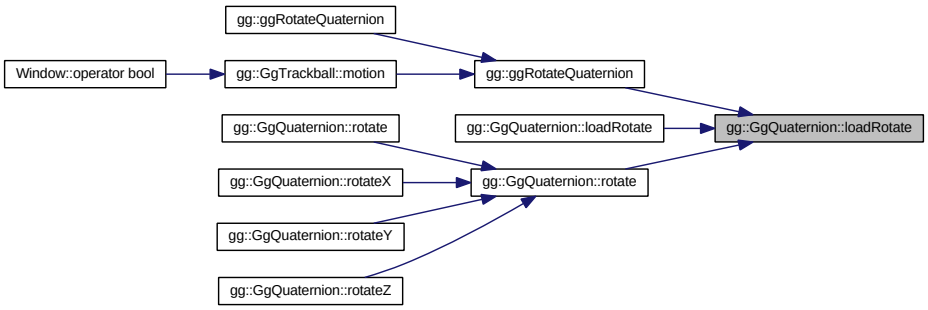
x	軸ベクトルの x 成分.
y	軸ベクトルの y 成分.
z	軸ベクトルの z 成分.
a	回転角.

戻り値

格納された回転を表す四元数.

gg.cpp の 4728 行目に定義があります。

被呼び出し関係図:



7.8.3.51 loadRotateX()

```
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

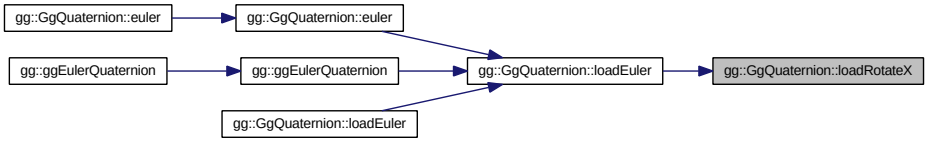
a	回転角.

戻り値

格納された回転を表す四元数.

gg.cpp の 4752 行目に定義があります。

被呼び出し関係図:



7.8.3.52 loadRotateY()

```
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

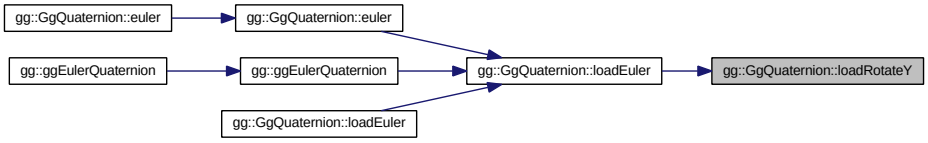
a	回転角.

戻り値

格納された回転を表す四元数.

gg.cpp の 4766 行目に定義があります。

被呼び出し関係図:



7.8.3.53 loadRotateZ()

```
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

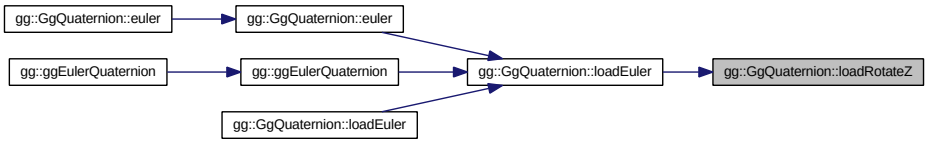
a	回転角.

戻り値

格納された回転を表す四元数.

gg.cpp の 4780 行目に定義があります。

被呼び出し関係図:



7.8.3.54 loadSlerp() [1/4]

```
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 3356 行目に定義があります。

呼び出し関係図:

7.8.3.55 loadSlerp() [2/4]

```
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 3366 行目に定義があります。

呼び出し関係図:

7.8.3.56 loadSlerp() [3/4]

```
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 3376 行目に定義があります。

呼び出し関係図:

7.8.3.57 loadSlerp() [4/4]

```
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

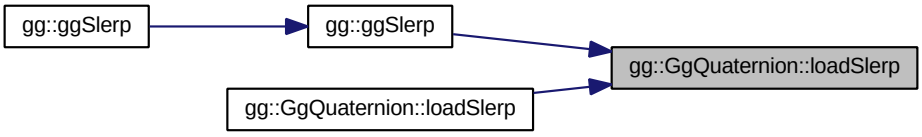
a	四元数を格納した GLfloat 型の 4 要素の配列変数.
b	四元数を格納した GLfloat 型の 4 要素の配列変数.
t	補間パラメータ.

戻り値

格納した a, b を t で内分した四元数.

gg.h の 3345 行目に定義があります。

被呼び出し関係図:



7.8.3.58 loadSubtract() [1/4]

```
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 2881 行目に定義があります。

呼び出し関係図:

7.8.3.59 loadSubtract() [2/4]

```
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 2865 行目に定義があります。

呼び出し関係図:

7.8.3.60 loadSubtract() [3/4]

```
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 2873 行目に定義があります。

呼び出し関係図:

7.8.3.61 loadSubtract() [4/4]

```
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

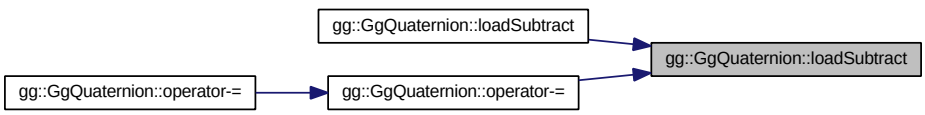
x	引く四元数の x 要素.
y	引く四元数の y 要素.
z	引く四元数の z 要素.
w	引く四元数の w 要素.

戻り値

(x, y, z, w) を引いた四元数.

gg.h の 2853 行目に定義があります。

被呼び出し関係図:



7.8.3.62 multiply() [1/4]

```
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 3071 行目に定義があります。

7.8.3.63 multiply() [2/4]

```
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 3053 行目に定義があります。

7.8.3.64 multiply() [3/4]

```
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 3061 行目に定義があります。

7.8.3.65 multiply() [4/4]

```
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 3044 行目に定義があります。

7.8.3.66 norm()

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

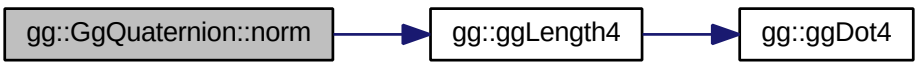
四元数のノルムを求める.

戻り値

四元数のノルム.

gg.h の 2763 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:

7.8.3.67 normalize()

```
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 3444 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:

7.8.3.68 operator*() [1/2]

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

gg.h の 3176 行目に定義があります。

呼び出し関係図:

7.8.3.69 operator*() [2/2]

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

gg.h の 3172 行目に定義があります。

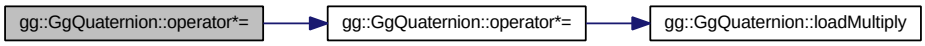
被呼び出し関係図:

7.8.3.70 operator*=() [1/2]

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

gg.h の 3144 行目に定義があります。

呼び出し関係図:



7.8.3.71 operator*=() [2/2]

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

gg.h の 3140 行目に定義があります。

呼び出し関係図:

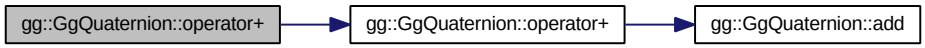
被呼び出し関係図:

7.8.3.72 operator+() [1/2]

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

gg.h の 3160 行目に定義があります。

呼び出し関係図:

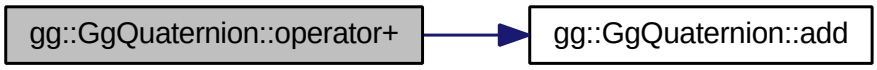


7.8.3.73 operator+() [2/2]

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

gg.h の 3156 行目に定義があります。

呼び出し関係図:



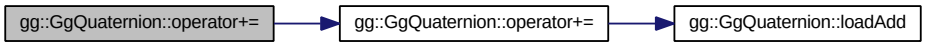
被呼び出し関係図:

7.8.3.74 operator+=() [1/2]

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

gg.h の 3128 行目に定義があります。

呼び出し関係図:



7.8.3.75 operator+=() [2/2]

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

gg.h の 3124 行目に定義があります。

呼び出し関係図:

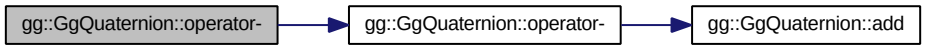
被呼び出し関係図:

7.8.3.76 operator-() [1/2]

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

gg.h の 3168 行目に定義があります。

呼び出し関係図:

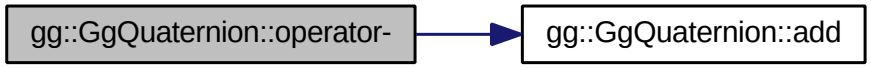


7.8.3.77 operator-() [2/2]

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

gg.h の 3164 行目に定義があります。

呼び出し関係図:



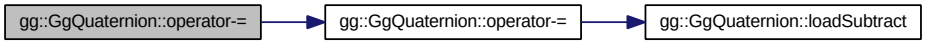
被呼び出し関係図:

7.8.3.78 operator-=() [1/2]

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

gg.h の 3136 行目に定義があります。

呼び出し関係図:



7.8.3.79 operator-=() [2/2]

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

gg.h の 3132 行目に定義があります。

呼び出し関係図:

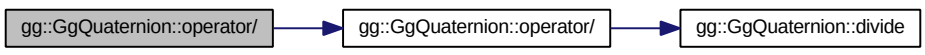
被呼び出し関係図:

7.8.3.80 operator/() [1/2]

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

gg.h の 3184 行目に定義があります。

呼び出し関係図:



7.8.3.81 operator/() [2/2]

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

gg.h の 3180 行目に定義があります。

呼び出し関係図:

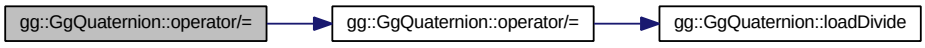
被呼び出し関係図:

7.8.3.82 operator/=() [1/2]

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

gg.h の 3152 行目に定義があります。

呼び出し関係図:



7.8.3.83 operator/=() [2/2]

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

gg.h の 3148 行目に定義があります。

呼び出し関係図:

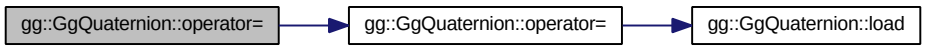
被呼び出し関係図:

7.8.3.84 operator=() [1/2]

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

gg.h の 3120 行目に定義があります。

呼び出し関係図:

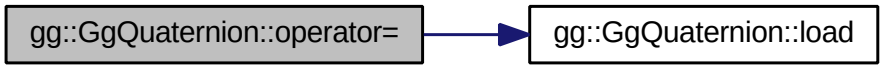


7.8.3.85 operator=() [2/2]

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

gg.h の 3116 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:

7.8.3.86 rotate() [1/3]

```
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

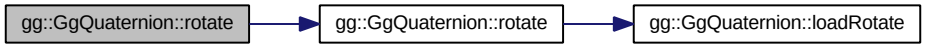
v	軸ベクトルを表す GLfloat 型の 4 要素の配列変数.

戻り値

回転した四元数.

gg.h の 3277 行目に定義があります。

呼び出し関係図:



7.8.3.87 rotate() [2/3]

```
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

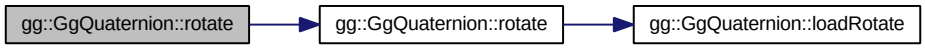
v	軸ベクトルを表す GLfloat 型の 3 要素の配列変数.
a	回転角.

戻り値

回転した四元数.

gg.h の 3269 行目に定義があります。

呼び出し関係図:



7.8.3.88 rotate() [3/3]

```
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

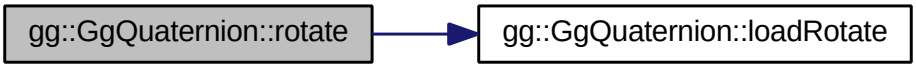
x	軸ベクトルの x 成分.
y	軸ベクトルの y 成分.
z	軸ベクトルの z 成分.
a	回転角.

戻り値

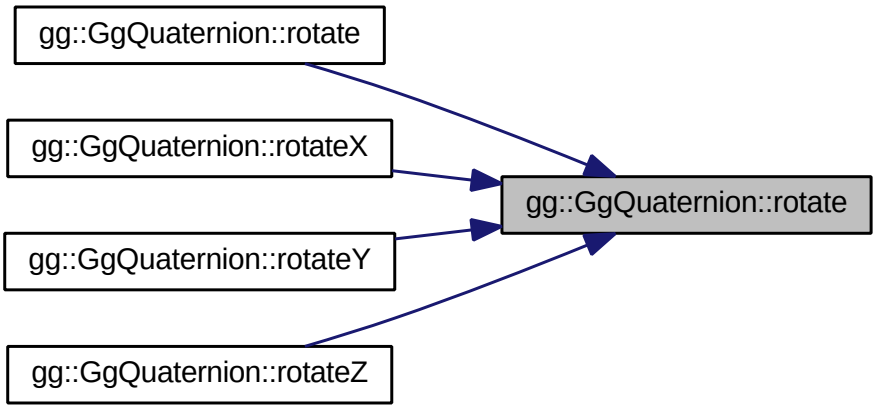
回転した四元数.

gg.h の 3259 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:



7.8.3.89 rotateX()

```
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

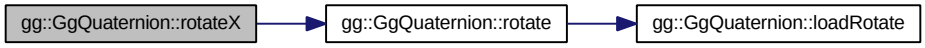
a	回転角.

戻り値

回転した四元数.

gg.h の 3285 行目に定義があります。

呼び出し関係図:



7.8.3.90 rotateY()

```
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

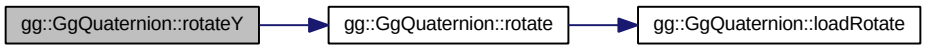
a	回転角.

戻り値

回転した四元数.

gg.h の 3293 行目に定義があります。

呼び出し関係図:



7.8.3.91 rotateZ()

```
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

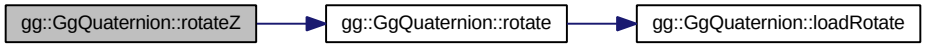
a	回転角.

戻り値

回転した四元数.

gg.h の 3301 行目に定義があります。

呼び出し関係図:



7.8.3.92 slerp() [1/2]

```
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 3435 行目に定義があります。

7.8.3.93 slerp() [2/2]

```
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 3424 行目に定義があります。

7.8.3.94 subtract() [1/4]

```
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 3033 行目に定義があります。

呼び出し関係図:

7.8.3.95 subtract() [2/4]

```
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 3017 行目に定義があります。

呼び出し関係図:

7.8.3.96 subtract() [3/4]

```
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 3025 行目に定義があります。

呼び出し関係図:

7.8.3.97 subtract() [4/4]

```
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 3004 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

7.9 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

- 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.
- virtual ~GgShader ()
  デストラクタ.
- GgShader (const GgShader &o)=delete
  コピーコンストラクタは使用禁止.
- GgShader & operator= (const GgShader &o)=delete
  代入演算子は使用禁止.
- 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.

7.9.1 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 4933 行目に定義があります。

7.9.2 構築子と解体子

7.9.2.1 GgShader() [1/2]

```
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 4946 行目に定義があります。

7.9.2.2 ~GgShader()

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

デストラクタ.

gg.h の 4952 行目に定義があります。

7.9.2.3 GgShader() [2/2]

```
gg::GgShader::GgShader (
            const GgShader & o )  [delete]
```

コピーコンストラクタは使用禁止.

7.9.3 関数詳解

7.9.3.1 get()

```
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 4979 行目に定義があります。

7.9.3.2 operator=()

```
GgShader& gg::GgShader::operator= (
            const GgShader & o )  [delete]
```

代入演算子は使用禁止.

7.9.3.3 unuse()

```
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 4972 行目に定義があります。

7.9.3.4 use()

```
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 4966 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:

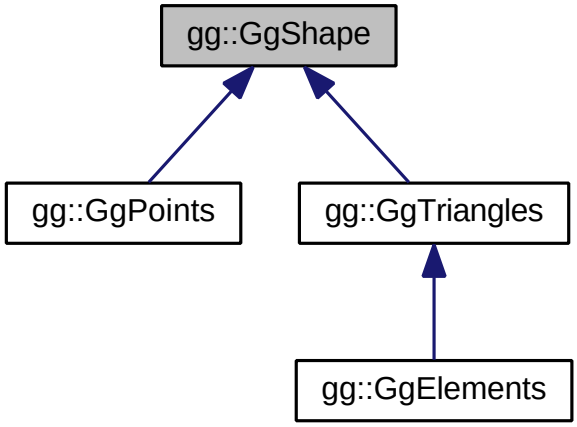
- gg.h

7.10 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

- GgShape (GLenum mode=0)
  コンストラクタ.
- virtual ~GgShape ()
  デストラクタ.
- GgShape (const GgShape &o)=delete
  コピーコンストラクタは使用禁止.
- GgShape & operator= (const GgShape &o)=delete
  代入演算子は使用禁止.
- 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.

7.10.1 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 4502 行目に定義があります。

7.10.2 構築子と解体子

7.10.2.1 GgShape() [1/2]

```
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 4514 行目に定義があります。

7.10.2.2 ~GgShape()

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

デストラクタ.

gg.h の 4522 行目に定義があります。

7.10.2.3 GgShape() [2/2]

```
gg::GgShape::GgShape (
            const GgShape & o )  [delete]
```

コピーコンストラクタは使用禁止.

7.10.3 関数詳解

7.10.3.1 draw()

```
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.

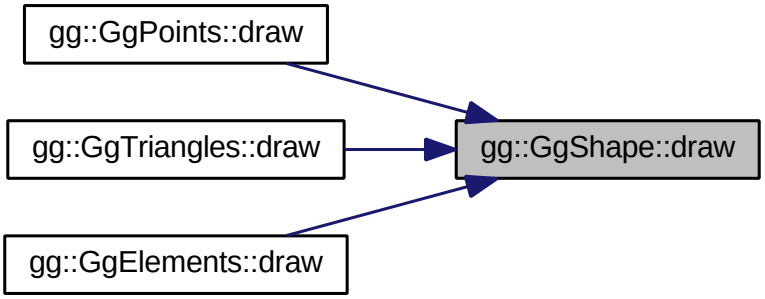
引数

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 4558 行目に定義があります。

被呼び出し関係図:



7.10.3.2 get()

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

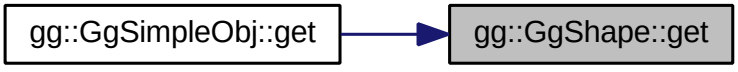
頂点配列オブジェクト名を取り出す.

戻り値

頂点配列オブジェクト名.

gg.h の 4536 行目に定義があります。

被呼び出し関係図:



7.10.3.3 getMode()

```
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 4550 行目に定義があります。

7.10.3.4 operator=()

```
GgShape& gg::GgShape::operator= (
            const GgShape & o )  [delete]
```

代入演算子は使用禁止.

7.10.3.5 setMode()

```
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 4543 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

7.11 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

- GgSimpleObj (const char *name, bool normalize=false)
  コンストラクタ.
- virtual ~GgSimpleObj ()
  デストラクタ.
- const GgTriangles * get () const
  形状データの取り出し.
- 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.

7.11.1 詳解

Wavefront OBJ 形式のファイル (Arrays 形式).

gg.h の 5724 行目に定義があります。

7.11.2 構築子と解体子

7.11.2.1 GgSimpleObj()

```
gg::GgSimpleObj::GgSimpleObj (
            const char * name,
            bool normalize = false )
```

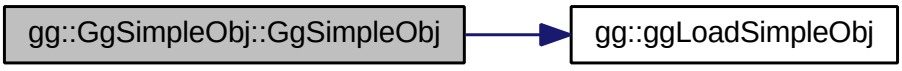
コンストラクタ.

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.
normalize	true なら図形のサイズを [-1, 1] に正規化する.

gg.cpp の 5665 行目に定義があります。

呼び出し関係図:



7.11.2.2 ~GgSimpleObj()

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

デストラクタ.

gg.h の 5743 行目に定義があります。

7.11.3 関数詳解

7.11.3.1 draw()

```
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 5687 行目に定義があります。

7.11.3.2 get()

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

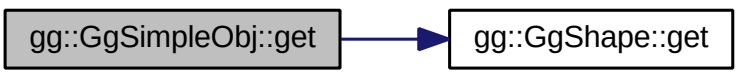
形状データの取り出し.

戻り値

GgTriangles 型の形状データのポインタ.

gg.h の 5747 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:

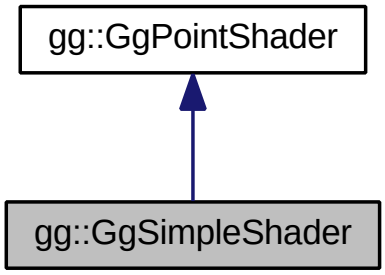
- gg.h
- gg.cpp

7.12 gg::GgSimpleShader クラス

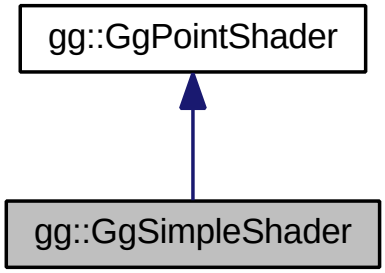
三角形に単純な陰影付けを行うシェーダ.

```
#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

- 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.
- virtual ~GgSimpleShader ()
  デストラクタ.
- GgSimpleShader & operator= (const GgSimpleShader &o)
- virtual void loadModelviewMatrix (const GLfloat *mv, const GLfloat *mn) const
  モデルビュー変換行列と法線変換行列を設定する.
- virtual void loadModelviewMatrix (const GgMatrix &mv, const GgMatrix &mn) const
  モデルビュー変換行列と法線変換行列を設定する.
- virtual void loadModelviewMatrix (const GLfloat *mv) const
  モデルビュー変換行列とそれから求めた法線変換行列を設定する.
- virtual void loadModelviewMatrix (const GgMatrix &mv) const
  モデルビュー変換行列とそれから求めた法線変換行列を設定する.
- virtual void loadMatrix (const GLfloat *mp, const GLfloat *mv, const GLfloat *mn) const
  投影変換行列とモデルビュー変換行列と法線変換行列を設定する.
- virtual void loadMatrix (const GgMatrix &mp, const GgMatrix &mv, const GgMatrix &mn) const
  投影変換行列とモデルビュー変換行列と法線変換行列を設定する.
- virtual void loadMatrix (const GLfloat *mp, const GLfloat *mv) const
  投影変換行列とモデルビュー変換行列を設定しモデルビュー変換行列から求めた法線変換行列を設定する.
- virtual void loadMatrix (const GgMatrix &mp, const GgMatrix &mv) const
  投影変換行列とモデルビュー変換行列を設定しモデルビュー変換行列から求めた法線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 use (const GLfloat *mp, const GLfloat *mv, const GLfloat *mn) const
  投影変換行列とモデルビュー変換行列と法線変換行列を設定してシェーダプログラムの使用を開始する.
- void use (const GgMatrix &mp, const GgMatrix &mv, const GgMatrix &mn) const
  投影変換行列とモデルビュー変換行列と法線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 GLfloat *mp, const GLfloat *mv) const
  投影変換行列とモデルビュー変換行列を設定しモデルビュー変換行列から求めた法線変換行列を設定してシェーダプログラムの使用を開始する.
- void use (const GgMatrix &mp, const GgMatrix &mv) const
  投影変換行列とモデルビュー変換行列を設定しモデルビュー変換行列から求めた法線変換行列を設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, GLint i=0) 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, GLint i=0) 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 GLfloat *mp, const GLfloat *mv, const GLfloat *mn, const LightBuffer *light, GLint i=0) const
  光源を指定し投影変換行列とモデルビュー変換行列と法線変換行列を設定してシェーダプログラムの使用を開始する.
- void use (const GgMatrix &mp, const GgMatrix &mv, const GgMatrix &mn, const LightBuffer &light, GLint i=0) const
  光源を指定し投影変換行列とモデルビュー変換行列と法線変換行列を設定してシェーダプログラムの使用を開始する.
- void use (const GLfloat *mp, const GLfloat *mv, const LightBuffer *light, GLint i=0) const
  光源を指定し投影変換行列とモデルビュー変換行列を設定しモデルビュー変換行列から求めた法線変換行列を設定してシェーダプログラムの使用を開始する.
- void use (const GgMatrix &mp, const GgMatrix &mv, const LightBuffer &light, GLint i=0) const

光源を指定し投影変換行列とモデルビュー変換行列を設定しモデルビュー変換行列から求めた法線変換行列を設定してシェーダプログラムの使用を開始する.

- void use (const GLfloat *mp, const LightBuffer *light, GLint i=0) const
  光源を指定し投影変換行列を設定してシェーダプログラムの使用を開始する.
- void use (const GgMatrix &mp, const LightBuffer &light, GLint i=0) const
  光源を指定し投影変換行列を設定してシェーダプログラムの使用を開始する.

7.12.1 詳解

三角形に単純な陰影付けを行うシェーダ.

gg.h の 5127 行目に定義があります。

7.12.2 構築子と解体子

7.12.2.1 GgSimpleShader() [1/3]

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

コンストラクタ.

gg.h の 5142 行目に定義があります。

7.12.2.2 GgSimpleShader() [2/3]

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

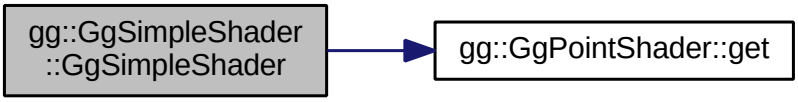
コンストラクタ.

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 5651 行目に定義があります。

呼び出し関係図:



7.12.2.3 GgSimpleShader() [3/3]

```
gg::GgSimpleShader::GgSimpleShader (
            const GgSimpleShader & o )  [inline]
```

コピーコンストラクタ.

gg.h の 5154 行目に定義があります。

7.12.2.4 ~GgSimpleShader()

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

デストラクタ.

gg.h の 5161 行目に定義があります。

7.12.3 関数詳解

7.12.3.1 loadMatrix() [1/4]

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

投影変換行列とモデルビュー変換行列を設定しモデルビュー変換行列から求めた法線変換行列を設定する.

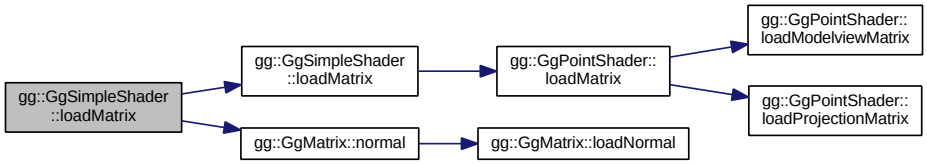
引数

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 5238 行目に定義があります。

呼び出し関係図:



7.12.3.2 loadMatrix() [2/4]

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

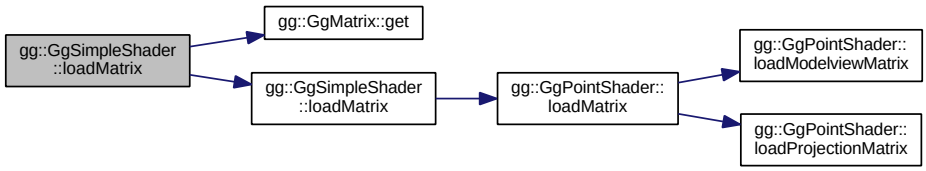
投影変換行列とモデルビュー変換行列と法線変換行列を設定する.

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 5222 行目に定義があります。

呼び出し関係図:



7.12.3.3 loadMatrix() [3/4]

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

投影変換行列とモデルビュー変換行列を設定しモデルビュー変換行列から求めた法線変換行列を設定する.

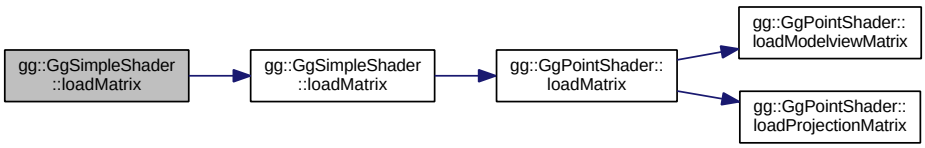
引数

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 5230 行目に定義があります。

呼び出し関係図:



7.12.3.4 loadMatrix() [4/4]

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

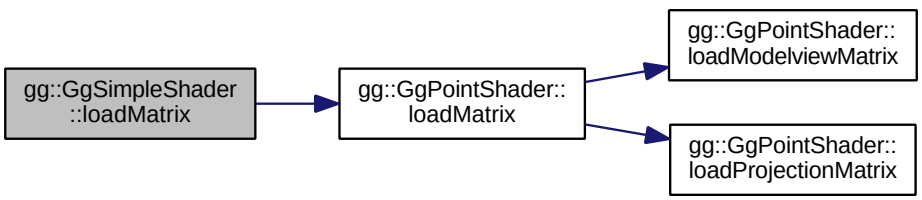
投影変換行列とモデルビュー変換行列と法線変換行列を設定する.

引数

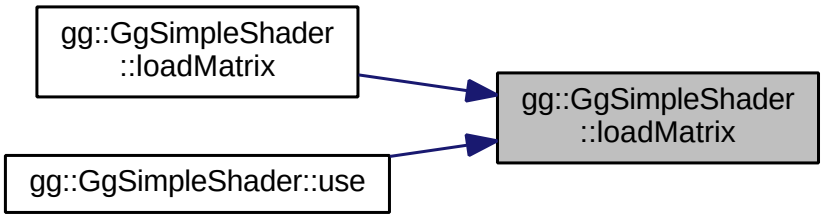
mp	GLfloat 型の 16 要素の配列変数に格納された投影変換行列.
mv	GLfloat 型の 16 要素の配列変数に格納されたモデルビュー変換行列.
mn	GLfloat 型の 16 要素の配列変数に格納されたモデルビュー変換行列の法線変換行列.

gg.h の 5212 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:



7.12.3.5 loadModelviewMatrix() [1/4]

```
virtual void gg::GgSimpleShader::loadModelviewMatrix (
            const GgMatrix & mv ) const  [inline], [virtual]
```

モデルビュー変換行列とそれから求めた法線変換行列を設定する.

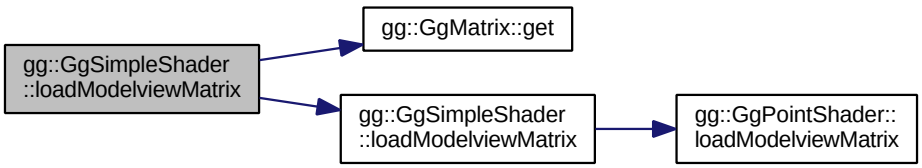
引数

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 5203 行目に定義があります。

呼び出し関係図:



7.12.3.6 loadModelviewMatrix() [2/4]

```
virtual void gg::GgSimpleShader::loadModelviewMatrix (
            const GgMatrix & mv,
            const GgMatrix & mn ) const  [inline], [virtual]
```

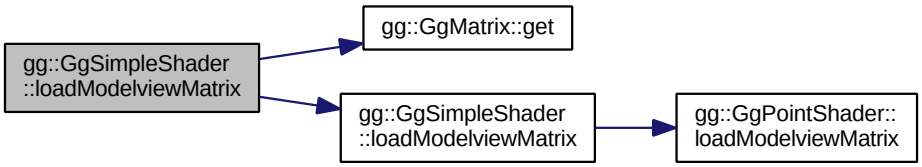
モデルビュー変換行列と法線変換行列を設定する.

引数

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 5189 行目に定義があります。

呼び出し関係図:



7.12.3.7 loadModelviewMatrix() [3/4]

```
virtual void gg::GgSimpleShader::loadModelviewMatrix (
            const GLfloat * mv ) const  [inline], [virtual]
```

モデルビュー変換行列とそれから求めた法線変換行列を設定する.

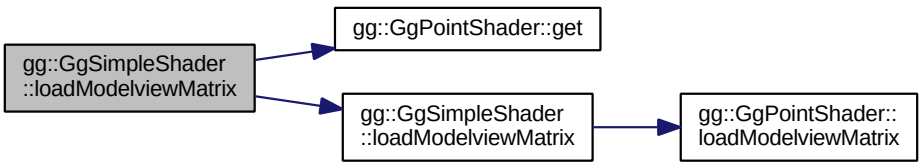
引数

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 5196 行目に定義があります。

呼び出し関係図:



7.12.3.8 loadModelviewMatrix() [4/4]

```
virtual void gg::GgSimpleShader::loadModelviewMatrix (
            const GLfloat * mv,
            const GLfloat * mn ) const  [inline], [virtual]
```

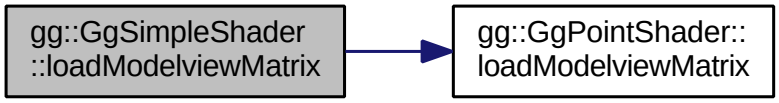
モデルビュー変換行列と法線変換行列を設定する.

引数

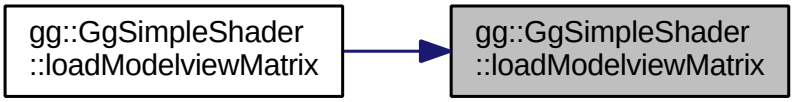
mv	GLfloat 型の 16 要素の配列変数に格納されたモデルビュー変換行列.
mn	GLfloat 型の 16 要素の配列変数に格納されたモデルビュー変換行列の法線変換行列.

gg.h の 5180 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:



7.12.3.9 operator=()

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

gg.h の 5164 行目に定義があります。

7.12.3.10 use() [1/13]

```
void gg::GgSimpleShader::use ( ) const  [inline], [virtual]
```

シェーダプログラムの使用を開始する.

gg::GgPointShaderを再実装しています。

gg.h の 5553 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:

7.12.3.11 use() [2/13]

```
void gg::GgSimpleShader::use (
            const GgMatrix & mp,
            const GgMatrix & mv ) const  [inline]
```

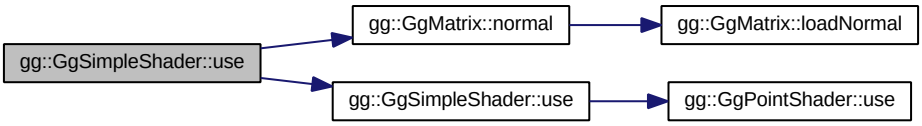
投影変換行列とモデルビュー変換行列を設定しモデルビュー変換行列から求めた法線変換行列を設定してシェーダプログラムの使用を開始する.

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.h の 5592 行目に定義があります。

呼び出し関係図:



7.12.3.12 use() [3/13]

```
void gg::GgSimpleShader::use (
            const GgMatrix & mp,
            const GgMatrix & mv,
            const GgMatrix & mn ) const  [inline]
```

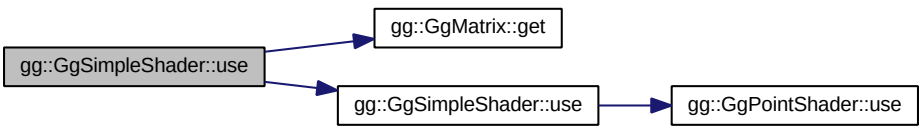
投影変換行列とモデルビュー変換行列と法線変換行列を指定してシェーダプログラムの使用を開始する.

引数

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 5576 行目に定義があります。

呼び出し関係図:



7.12.3.13 use() [4/13]

```
void gg::GgSimpleShader::use (
            const GgMatrix & mp,
```

```
            const GgMatrix & mv,
            const GgMatrix & mn,
            const LightBuffer & light,
            GLint i = 0 ) const  [inline]
```

光源を指定し投影変換行列とモデルビュー変換行列と法線変換行列を設定してシェーダプログラムの使用を開始する.

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.
light	光源の特性の gg::LightBuffer 構造体.
i	光源データの uniform block のインデックス.

gg.h の 5638 行目に定義があります。

呼び出し関係図:

7.12.3.14 use() [5/13]

```
void gg::GgSimpleShader::use (
            const GgMatrix & mp,
            const GgMatrix & mv,
            const LightBuffer & light,
            GLint i = 0 ) const  [inline]
```

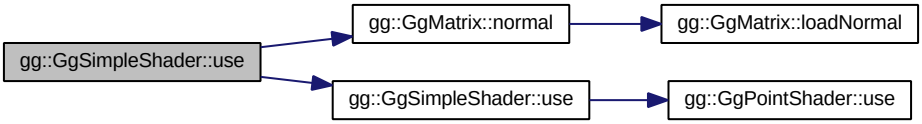
光源を指定し投影変換行列とモデルビュー変換行列を設定しモデルビュー変換行列から求めた法線変換行列を設定してシェーダプログラムの使用を開始する.

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.
light	光源の特性の gg::LightBuffer 構造体.
i	光源データの uniform block のインデックス.

gg.h の 5658 行目に定義があります。

呼び出し関係図:



7.12.3.15 use() [6/13]

```
void gg::GgSimpleShader::use (
            const GgMatrix & mp,
            const LightBuffer & light,
            GLint i = 0 ) const  [inline]
```

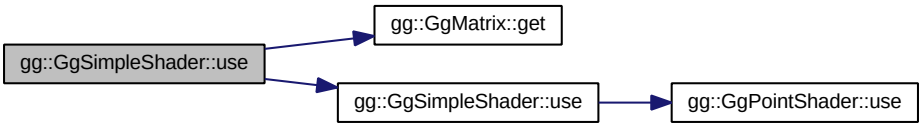
光源を指定し投影変換行列を設定してシェーダプログラムの使用を開始する.

引数

mp	GgMatrix 型の投影変換行列.
light	光源の特性の gg::LightBuffer 構造体.
i	光源データの uniform block のインデックス.

gg.h の 5680 行目に定義があります。

呼び出し関係図:



7.12.3.16 use() [7/13]

```
void gg::GgSimpleShader::use (
            const GLfloat * mp,
            const GLfloat * mv ) const  [inline]
```

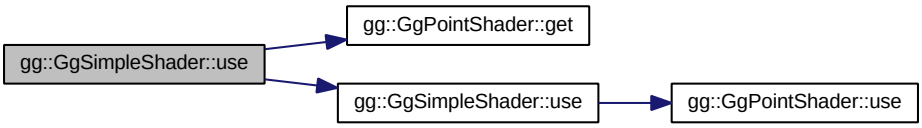
投影変換行列とモデルビュー変換行列を設定しモデルビュー変換行列から求めた法線変換行列を設定してシェーダプログラムの使用を開始する.

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.h の 5584 行目に定義があります。

呼び出し関係図:



7.12.3.17 use() [8/13]

```
void gg::GgSimpleShader::use (
            const GLfloat * mp,
            const GLfloat * mv,
            const GLfloat * mn ) const  [inline]
```

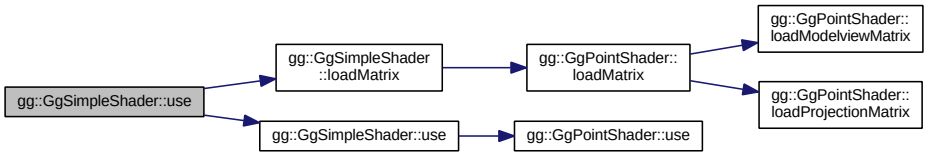
投影変換行列とモデルビュー変換行列と法線変換行列を設定してシェーダプログラムの使用を開始する.

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.
mn	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 5563 行目に定義があります。

呼び出し関係図:



7.12.3.18 use() [9/13]

```
void gg::GgSimpleShader::use (
            const GLfloat * mp,
            const GLfloat * mv,
            const GLfloat * mn,
            const LightBuffer * light,
            GLint i = 0 ) const  [inline]
```

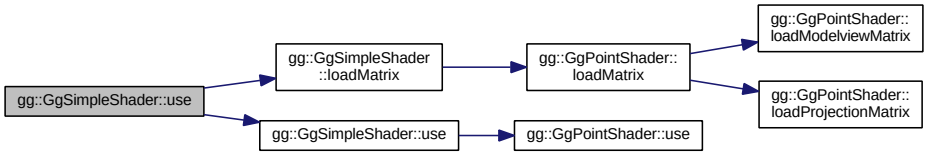
光源を指定し投影変換行列とモデルビュー変換行列と法線変換行列を設定してシェーダプログラムの使用を開始する.

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.
mn	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.
light	光源の特性の gg::LightBuffer 構造体のポインタ.
i	光源データの uniform block のインデックス.

gg.h の 5623 行目に定義があります。

呼び出し関係図:



7.12.3.19 use() [10/13]

```
void gg::GgSimpleShader::use (
            const GLfloat * mp,
            const GLfloat * mv,
            const LightBuffer * light,
            GLint i = 0 ) const  [inline]
```

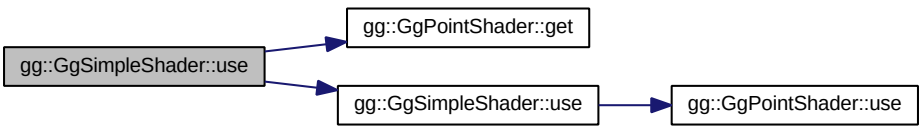
光源を指定し投影変換行列とモデルビュー変換行列を設定しモデルビュー変換行列から求めた法線変換行列を設定してシェーダプログラムの使用を開始する.

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.
light	光源の特性の gg::LightBuffer 構造体のポインタ.
i	光源データの uniform block のインデックス.

gg.h の 5648 行目に定義があります。

呼び出し関係図:



7.12.3.20 use() [11/13]

```
void gg::GgSimpleShader::use (
              const GLfloat * mp,
              const LightBuffer * light,
              GLint i = 0 ) const  [inline]
```

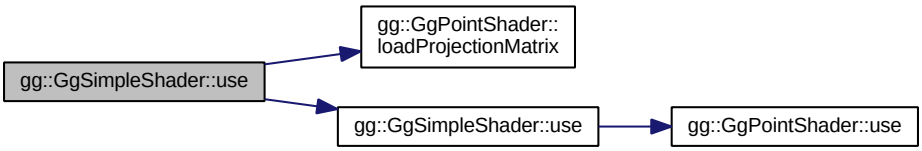
光源を指定し投影変換行列を設定してシェーダプログラムの使用を開始する.

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.
light	光源の特性の gg::LightBuffer 構造体のポインタ.
i	光源データの uniform block のインデックス.

gg.h の 5667 行目に定義があります。

呼び出し関係図:



7.12.3.21 use() [12/13]

```
void gg::GgSimpleShader::use (
              const LightBuffer & light,
              GLint i = 0 ) const  [inline]
```

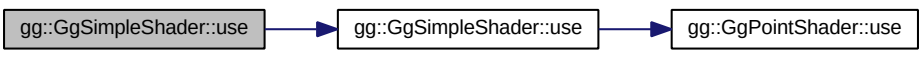
光源を指定してシェーダプログラムの使用を開始する.

引数

light	光源の特性の gg::LightBuffer 構造体.
i	光源データの uniform block のインデックス.

gg.h の 5612 行目に定義があります。

呼び出し関係図:



7.12.3.22 use() [13/13]

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

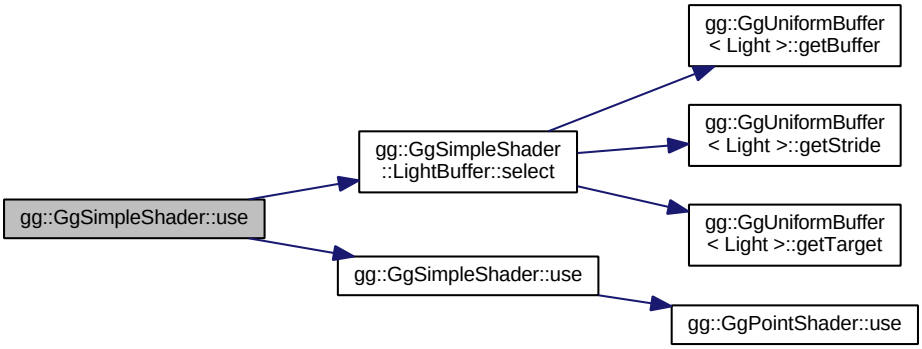
光源を指定してシェーダプログラムの使用を開始する.

引数

light	光源の特性の gg::LightBuffer 構造体のポインタ.
i	光源データの uniform block のインデックス.

gg.h の 5600 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

7.13 gg::GgTexture クラス

テクスチャ.

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

公開メンバ関数

- 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.
- virtual ~GgTexture ()
  デストラクタ.
- GgTexture (const GgTexture &o)=delete

コピーコンストラクタは使用禁止.

- GgTexture & operator= (const GgTexture &o)=delete
  代入演算子は使用禁止.
- 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.

7.13.1 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 3857 行目に定義があります。

7.13.2 構築子と解体子

7.13.2.1 GgTexture() [1/2]

```
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 3875 行目に定義があります。

7.13.2.2 ～GgTexture()

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

デストラクタ.

gg.h の 3883 行目に定義があります。

7.13.2.3 GgTexture() [2/2]

```
gg::GgTexture::GgTexture (
            const GgTexture & o )  [delete]
```

コピーコンストラクタは使用禁止.

7.13.3 関数詳解

7.13.3.1 bind()

```
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 3896 行目に定義があります。

7.13.3.2 getHeight()

```
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 3916 行目に定義があります。

被呼び出し関係図:

7.13.3.3 getSize() [1/2]

```
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 3931 行目に定義があります。

7.13.3.4 getSize() [2/2]

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

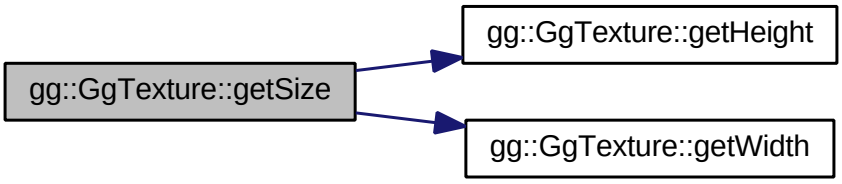
使用しているテクスチャのサイズを取り出す.

引数

size	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 3923 行目に定義があります。

呼び出し関係図:



7.13.3.5 getTexture()

```
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 3938 行目に定義があります。

7.13.3.6 getWidth()

```
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 3909 行目に定義があります。

被呼び出し関係図:

7.13.3.7 operator=()

```
GgTexture& gg::GgTexture::operator= (
            const GgTexture & o )  [delete]
```

代入演算子は使用禁止.

7.13.3.8 unbind()

```
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 3902 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

7.14 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

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

7.14.1 詳解

簡易トラックボール処理.

gg.h の 3726 行目に定義があります。

7.14.2 構築子と解体子

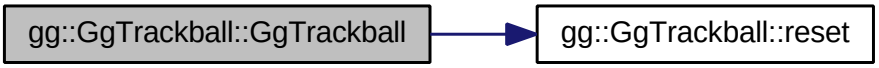
7.14.2.1 GgTrackball()

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

コンストラクタ.

gg.h の 3738 行目に定義があります。

呼び出し関係図:



7.14.2.2 ~GgTrackball()

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

デストラクタ.

gg.h の 3744 行目に定義があります。

7.14.3 関数詳解

7.14.3.1 begin()

```
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 4894 行目に定義があります。

7.14.3.2 end()

```
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 4958 行目に定義があります。

7.14.3.3 get()

```
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 3846 行目に定義があります。

呼び出し関係図:

7.14.3.4 getMatrix()

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

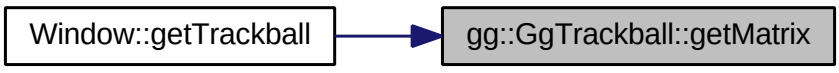
現在の回転の変換行列を取り出す.

戻り値

回転の変換を表す GgMatrix 型の変換行列.

gg.h の 3839 行目に定義があります。

被呼び出し関係図:



7.14.3.5 getQuaternion()

```
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 3832 行目に定義があります。

7.14.3.6 getScale() [1/3]

```
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 3810 行目に定義があります。

7.14.3.7 getScale() [2/3]

```
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 3824 行目に定義があります。

7.14.3.8 getScale() [3/3]

```
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 3817 行目に定義があります。

7.14.3.9 getStart() [1/3]

```
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 3788 行目に定義があります。

7.14.3.10 getStart() [2/3]

```
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 3802 行目に定義があります。

7.14.3.11 getStart() [3/3]

```
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 3795 行目に定義があります。

7.14.3.12 motion()

```
void gg::GgTrackball::motion (
            float x,
            float y )
```

回転の変換行列を計算する.

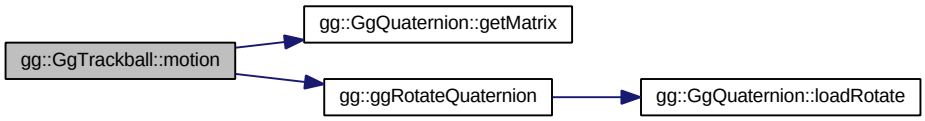
マウスのドラッグ中に呼び出す.

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 4910 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:

7.14.3.13 region() [1/2]

```
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 4881 行目に定義があります。

被呼び出し関係図:

7.14.3.14 region() [2/2]

```
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 3756 行目に定義があります。

呼び出し関係図:

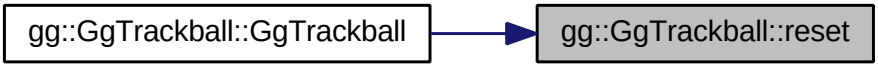
7.14.3.15 reset()

```
void gg::GgTrackball::reset ( )
```

トラックボールをリセットする

gg.cpp の 4863 行目に定義があります。

被呼び出し関係図:



7.14.3.16 rotate()

```
void gg::GgTrackball::rotate (
            const GgQuaternion & q )
```

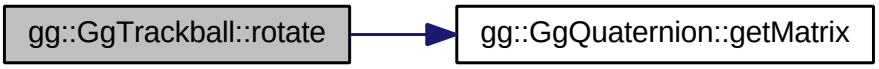
トラックボールの回転角を修正する.

引数

q	修正分の回転角の四元数.

gg.cpp の 4937 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:

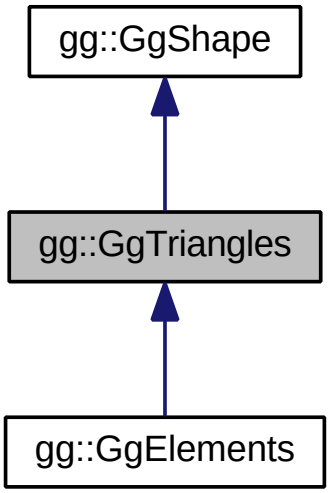
- gg.h
- gg.cpp

7.15 gg::GgTriangles クラス

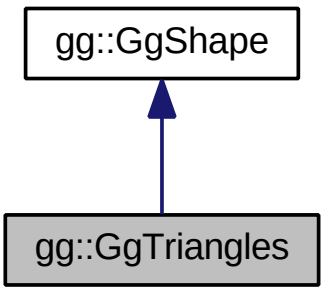
三角形で表した形状データ (Arrays 形式).

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

gg::GgTriangles の継承関係図



gg::GgTriangles 連携図



公開メンバ関数

- 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.
- virtual ~GgTriangles ()
  デストラクタ.
- 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.

7.15.1 詳解

三角形で表した形状データ (Arrays 形式).

gg.h の 4678 行目に定義があります。

7.15.2 構築子と解体子

7.15.2.1 GgTriangles() [1/2]

```
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 4688 行目に定義があります。

7.15.2.2 GgTriangles() [2/2]

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

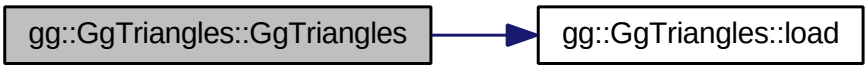
コンストラクタ.

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 4697 行目に定義があります。

呼び出し関係図:



7.15.2.3 ~GgTriangles()

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

デストラクタ.

gg.h の 4705 行目に定義があります。

7.15.3 関数詳解

7.15.3.1 draw()

```
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 4985 行目に定義があります。

呼び出し関係図:

7.15.3.2 getBuffer()

```
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 4716 行目に定義があります。

7.15.3.3 getCount()

```
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 4709 行目に定義があります。

7.15.3.4 load()

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

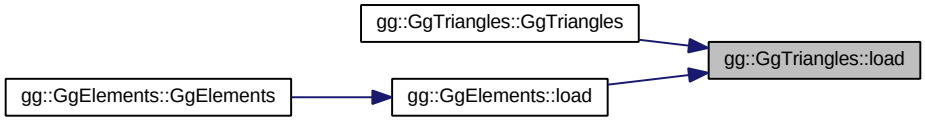
バッファオブジェクトを確保して頂点属性を格納する.

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 4734 行目に定義があります。

被呼び出し関係図:



7.15.3.5 send()

```
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 4725 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

7.16 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

- 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.
- virtual ~GgUniformBuffer ()
  デストラクタ.
- 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.

7.16.1 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 4259 行目に定義があります。

7.16.2 構築子と解体子

7.16.2.1 GgUniformBuffer() [1/3]

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

コンストラクタ.

gg.h の 4267 行目に定義があります。

7.16.2.2 GgUniformBuffer() [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 4273 行目に定義があります。

7.16.2.3 GgUniformBuffer() [3/3]

```
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 4282 行目に定義があります。

7.16.2.4 ~GgUniformBuffer()

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

デストラクタ.

gg.h の 4288 行目に定義があります。

7.16.3 関数詳解

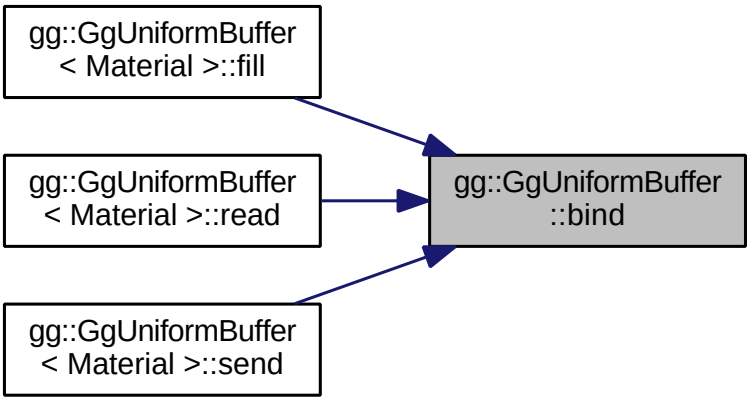
7.16.3.1 bind()

```
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 4319 行目に定義があります。

被呼び出し関係図:



7.16.3.2 copy()

```
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 4473 行目に定義があります。

7.16.3.3 fill()

```
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]
```

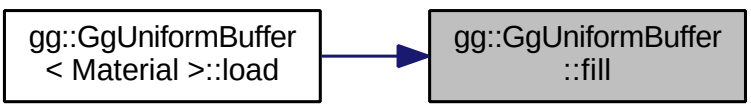
ユニフォームバッファオブジェクトの全ブロックのメンバーを同じデータを格納する.

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 4419 行目に定義があります。

被呼び出し関係図:



7.16.3.4 getBuffer()

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

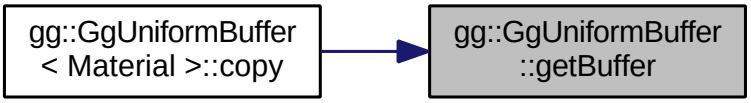
ユニフォームバッファオブジェクト名を取り出す.

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 4313 行目に定義があります。

被呼び出し関係図:



7.16.3.5 getCount()

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

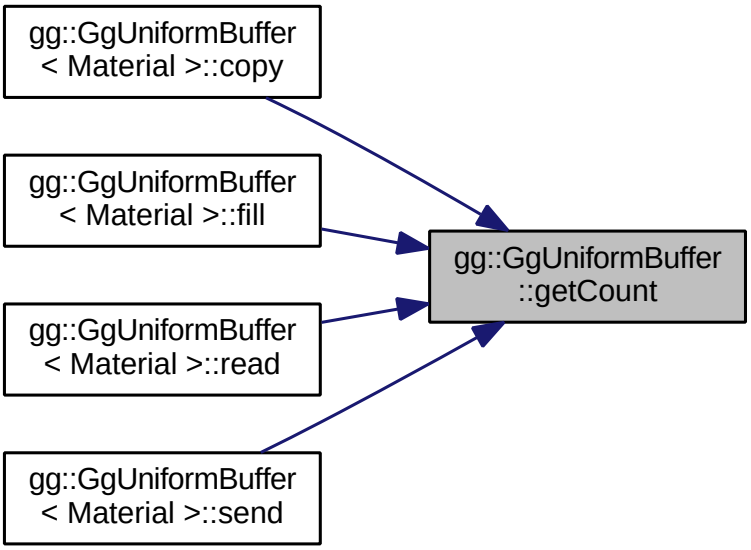
データの数を取り出す.

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 4306 行目に定義があります。

被呼び出し関係図:



7.16.3.6 getStride()

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

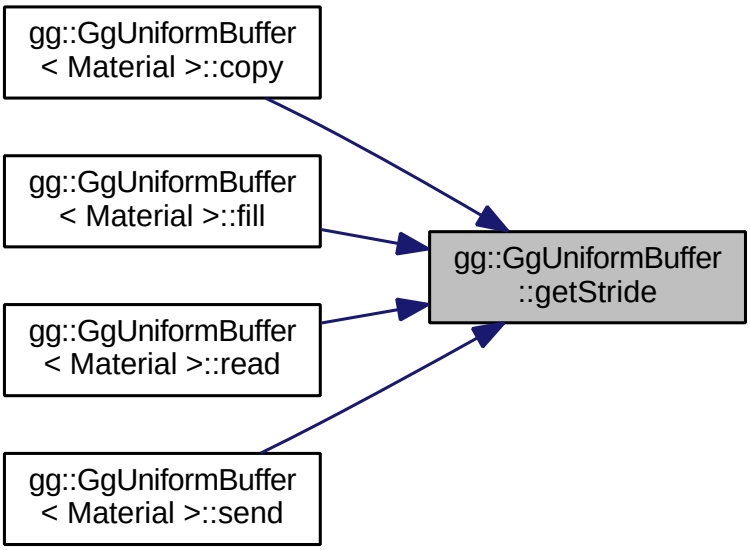
ユニフォームバッファオブジェクトのアライメントを考慮したデータの間隔を取り出す.

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 4299 行目に定義があります。

被呼び出し関係図:



7.16.3.7 getTarget()

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

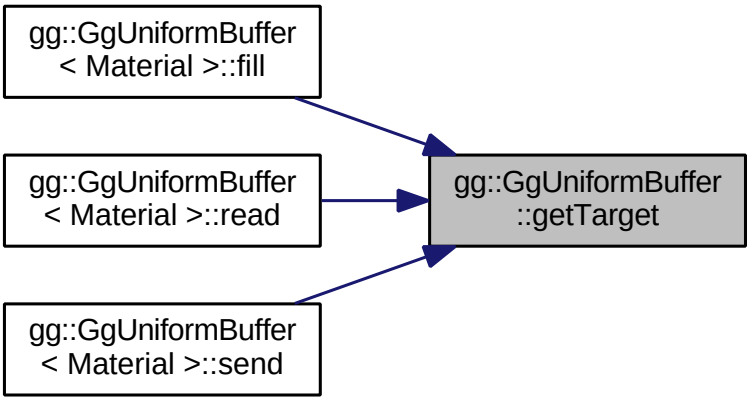
ユニフォームバッファオブジェクトのターゲットを取り出す.

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 4292 行目に定義があります。

被呼び出し関係図:



7.16.3.8 load() [1/2]

```
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 4372 行目に定義があります。

7.16.3.9 load() [2/2]

```
template<typename T >
void gg::GgUniformBuffer< T >::load (
            const T * data,
```

```
        GLsizei count,
        GLenum usage = GL_STATIC_DRAW )  [inline]
```

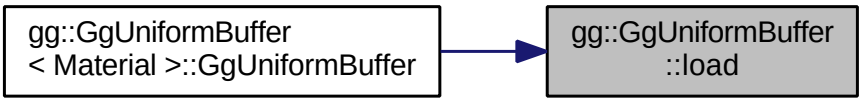
ユニフォームバッファオブジェクトを確保してブロックごとにデータを転送する.

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 4356 行目に定義があります。

被呼び出し関係図:



7.16.3.10 map() [1/2]

```
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 4332 行目に定義があります。

7.16.3.11 map() [2/2]

```
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 4341 行目に定義があります。

7.16.3.12 read()

```
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 4445 行目に定義があります。

7.16.3.13 send()

```
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]
```

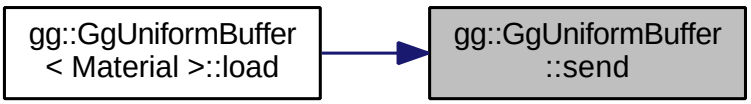
ユニフォームバッファオブジェクトを確保してユニフォームバッファオブジェクトのブロックごとのメンバを同じデータで埋める.

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 4390 行目に定義があります。

被呼び出し関係図:



7.16.3.14 unbind()

```
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 4325 行目に定義があります。

7.16.3.15 unmap()

```
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 4347 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

7.17 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.

7.17.1 詳解

三角形の頂点データ.

gg.h の 4640 行目に定義があります。

7.17.2 構築子と解体子

7.17.2.1 GgVertex() [1/4]

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

法線.

コンストラクタ.

gg.h の 4646 行目に定義があります。

7.17.2.2 GgVertex() [2/4]

```
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 4651 行目に定義があります。

7.17.2.3 GgVertex() [3/4]

```
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 4662 行目に定義があります。

7.17.2.4 GgVertex() [4/4]

```
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 4670 行目に定義があります。

7.17.3 メンバ詳解

7.17.3.1 normal

```
GgVector gg::GgVertex::normal
```

位置.

gg.h の 4643 行目に定義があります。

7.17.3.2 position

GgVector gg::GgVertex::position

gg.h の 4642 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

7.18 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
  光源強度の環境光成分.
- GgVector diffuse
  光源強度の拡散反射光成分.
- GgVector specular
  光源強度の鏡面反射光成分.
- GgVector position
  光源の位置.

7.18.1 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 5246 行目に定義があります。

7.18.2 メンバ詳解

7.18.2.1 ambient

GgVector gg::GgSimpleShader::Light::ambient

光源強度の環境光成分.

gg.h の 5248 行目に定義があります。

7.18.2.2 diffuse

```
GgVector gg::GgSimpleShader::Light::diffuse
```

光源強度の拡散反射光成分.

gg.h の 5249 行目に定義があります。

7.18.2.3 position

```
GgVector gg::GgSimpleShader::Light::position
```

光源の位置.

gg.h の 5251 行目に定義があります。

7.18.2.4 specular

```
GgVector gg::GgSimpleShader::Light::specular
```

光源強度の鏡面反射光成分.

gg.h の 5250 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:

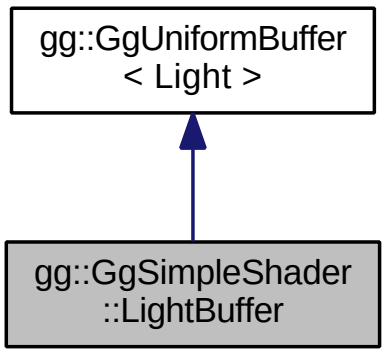
- gg.h

7.19 gg::GgSimpleShader::LightBuffer クラス

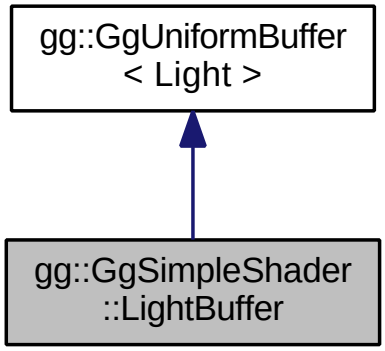
三角形に単純な陰影付けを行うシェーダが参照する光源データのユニフォームバッファオブジェクト.

```
#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

- 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.
- virtual ~LightBuffer ()
  デストラクタ.
- void loadAmbient (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 loadAmbient (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 loadDiffuse (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 loadDiffuse (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 loadSpecular (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 loadSpecular (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 loadColor (const Light &color, 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 loadPosition (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 loadPosition (const GgVector &position, GLint first=0, GLsizei count=1) const
  光源の位置を設定する.
- void loadPosition (const GLfloat *position, GLint first=0, GLsizei count=1) const
  光源の位置を設定する.
- void loadPosition (const GgVector *position, GLint first=0, GLsizei count=1) const
  光源の位置を設定する.
- void load (const Light *light, GLint first=0, GLsizei count=1) const
  光源の色と位置を設定する.
- void load (const Light &light, GLint first=0, GLsizei count=1) const
  光源の色と位置を設定する.
- void select (GLint i=0) const
  光源を選択する.

7.19.1 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 5257 行目に定義があります。

7.19.2 構築子と解体子

7.19.2.1 LightBuffer() [1/2]

```
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.
usage	バッファの使い方のパターン, glBufferData() の第 4 引数の usage に渡される.

gg.h の 5266 行目に定義があります。

7.19.2.2 LightBuffer() [2/2]

```
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.
usage	バッファの使い方のパターン, glBufferData() の第 4 引数の usage に渡される.

gg.h の 5273 行目に定義があります。

7.19.2.3 ～LightBuffer()

```
virtual gg::GgSimpleShader::LightBuffer::~LightBuffer ( )  [inline], [virtual]
```

デストラクタ.

gg.h の 5277 行目に定義があります。

7.19.3 関数詳解

7.19.3.1 load() [1/2]

```
void gg::GgSimpleShader::LightBuffer::load (
            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 5389 行目に定義があります。

呼び出し関係図:

7.19.3.2 load() [2/2]

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

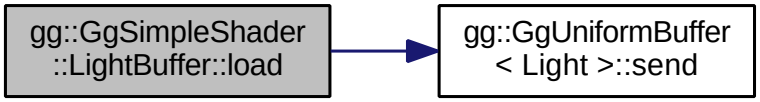
光源の色と位置を設定する.

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 5380 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:

7.19.3.3 loadAmbient() [1/2]

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

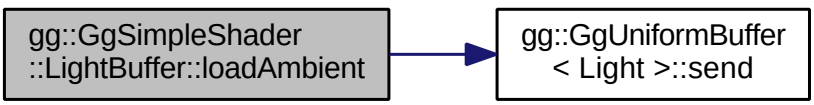
光源の強度の環境光成分を設定する.

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 5292 行目に定義があります。

呼び出し関係図:



7.19.3.4 loadAmbient() [2/2]

```
void gg::GgSimpleShader::LightBuffer::loadAmbient (
            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.
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。

7.19.3.5 loadColor()

```
void gg::GgSimpleShader::LightBuffer::loadColor (
            const Light & color,
            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

color	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 5380 行目に定義があります。

7.19.3.6 loadDiffuse() [1/2]

```
void gg::GgSimpleShader::LightBuffer::loadDiffuse (
            const GLfloat * diffuse,
```

```
            GLint first = 0,
            GLsizei count = 1 ) const  [inline]
```

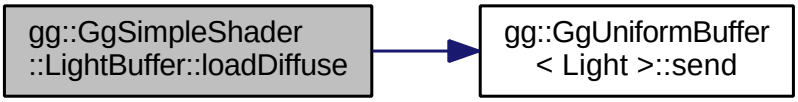
光源の強度の拡散反射光成分を設定する.

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 5311 行目に定義があります。

呼び出し関係図:



7.19.3.7 loadDiffuse() [2/2]

```
void gg::GgSimpleShader::LightBuffer::loadDiffuse (
            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 5325 行目に定義があります。

7.19.3.8 loadPosition() [1/4]

```
void gg::GgSimpleShader::LightBuffer::loadPosition (
            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 5434 行目に定義があります。

7.19.3.9 loadPosition() [2/4]

```
void gg::GgSimpleShader::LightBuffer::loadPosition (
              const GgVector * position,
              GLint first = 0,
              GLsizei count = 1 ) const  [inline]
```

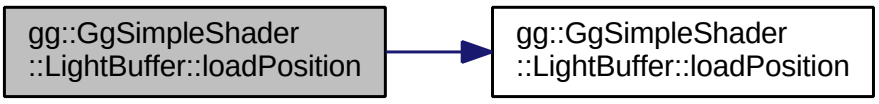
光源の位置を設定する.

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 5371 行目に定義があります。

呼び出し関係図:



7.19.3.10 loadPosition() [3/4]

```
void gg::GgSimpleShader::LightBuffer::loadPosition (
              const GLfloat * position,
              GLint first = 0,
              GLsizei count = 1 ) const  [inline]
```

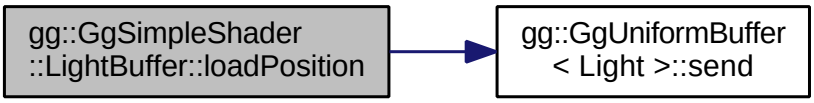
光源の位置を設定する.

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 5361 行目に定義があります。

呼び出し関係図:



7.19.3.11 loadPosition() [4/4]

```
void gg::GgSimpleShader::LightBuffer::loadPosition (
             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 5408 行目に定義があります。

被呼び出し関係図:

7.19.3.12 loadSpecular() [1/2]

```
void gg::GgSimpleShader::LightBuffer::loadSpecular (
             const GLfloat * specular,
             GLint first = 0,
             GLsizei count = 1 ) const  [inline]
```

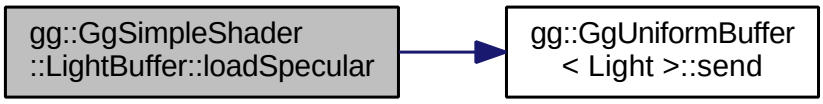
光源の強度の鏡面反射光成分を設定する.

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 5330 行目に定義があります。

呼び出し関係図:



7.19.3.13 loadSpecular() [2/2]

```
void gg::GgSimpleShader::LightBuffer::loadSpecular (
             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 5354 行目に定義があります。

7.19.3.14 select()

```
void gg::GgSimpleShader::LightBuffer::select (
             GLint i = 0 ) const  [inline]
```

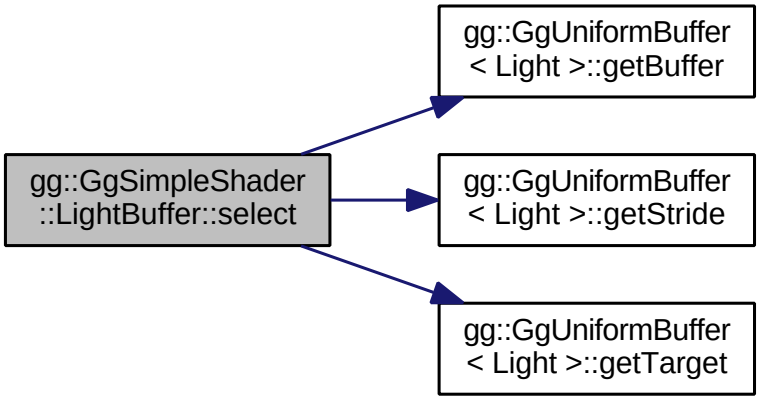
光源を選択する.

引数

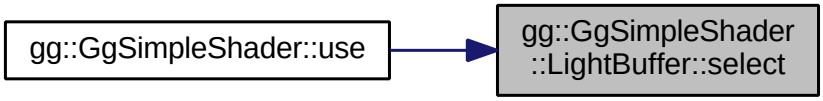
i	光源データの uniform block のインデックス.

gg.h の 5396 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:



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

7.20 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
  環境光に対する反射係数.
- GgVector diffuse
  拡散反射係数.
- GgVector specular
  鏡面反射係数.
- GLfloat shininess
  輝き係数.

7.20.1 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 5407 行目に定義があります。

7.20.2 メンバ詳解

7.20.2.1 ambient

```
GgVector gg::GgSimpleShader::Material::ambient
```

環境光に対する反射係数.

gg.h の 5409 行目に定義があります。

7.20.2.2 diffuse

```
GgVector gg::GgSimpleShader::Material::diffuse
```

拡散反射係数.

gg.h の 5410 行目に定義があります。

7.20.2.3 shininess

```
GLfloat gg::GgSimpleShader::Material::shininess
```

輝き係数.

gg.h の 5412 行目に定義があります。

7.20.2.4 specular

```
GgVector gg::GgSimpleShader::Material::specular
```

鏡面反射係数.

gg.h の 5411 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:

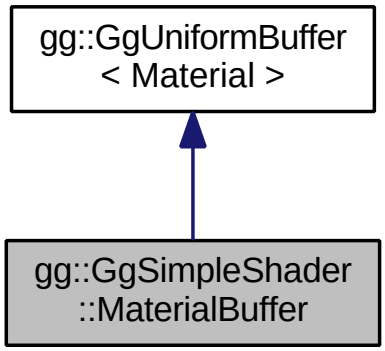
- gg.h

7.21 gg::GgSimpleShader::MaterialBuffer クラス

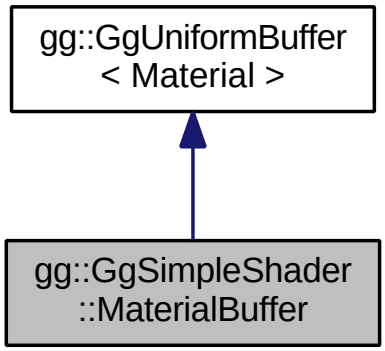
三角形に単純な陰影付けを行うシェーダが参照する材質データのユニフォームバッファオブジェクト.

```
#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

- 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.
- virtual ~MaterialBuffer ()
  デストラクタ
- void loadAmbient (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 loadAmbient (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 loadDiffuse (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 loadDiffuse (const GLfloat *diffuse, GLint first=0, GLsizei count=1) const
  拡散反射係数を設定する.

- void loadAmbientAndDiffuse (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 loadAmbientAndDiffuse (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 loadSpecular (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 loadSpecular (const GLfloat *specular, GLint first=0, GLsizei count=1) const
  鏡面反射係数を設定する.
- void loadShininess (GLfloat shininess, GLint first=0, GLsizei count=1) const
  輝き係数を設定する.
- void loadShininess (const GLfloat *shininess, GLint first=0, GLsizei count=1) const
  輝き係数を設定する.
- void load (const Material *material, GLint first=0, GLsizei count=1) const
  材質を設定する.
- void load (const Material &material, GLint first=0, GLsizei count=1) const
  材質を設定する.
- void select (GLint i=0) const
  材質を選択する.

7.21.1 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 5418 行目に定義があります。

7.21.2 構築子と解体子

7.21.2.1 MaterialBuffer() [1/2]

```
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.
usage	バッファの使い方のパターン, glBufferData() の第 4 引数の usage に渡される.

gg.h の 5427 行目に定義があります。

7.21.2.2 MaterialBuffer() [2/2]

```
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.
usage	バッファの使い方のパターン, glBufferData() の第 4 引数の usage に渡される.

gg.h の 5434 行目に定義があります。

7.21.2.3 ~MaterialBuffer()

```
virtual gg::GgSimpleShader::MaterialBuffer::~MaterialBuffer ( )  [inline], [virtual]
```

デストラクタ

gg.h の 5438 行目に定義があります。

7.21.3 関数詳解

7.21.3.1 load() [1/2]

```
void gg::GgSimpleShader::MaterialBuffer::load (
            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 5537 行目に定義があります。

呼び出し関係図:

7.21.3.2 load() [2/2]

```
void gg::GgSimpleShader::MaterialBuffer::load (
            const Material * material,
            GLint first = 0,
            GLsizei count = 1 ) const  [inline]
```

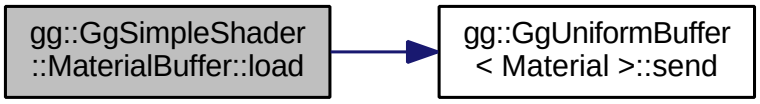
材質を設定する.

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 5528 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:

7.21.3.3 loadAmbient() [1/2]

```
void gg::GgSimpleShader::MaterialBuffer::loadAmbient (
            const GLfloat * ambient,
            GLint first = 0,
            GLsizei count = 1 ) const  [inline]
```

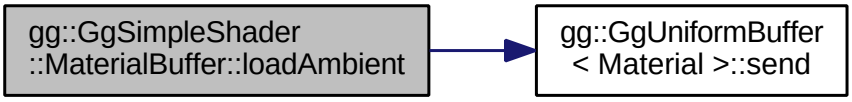
環境光に対する反射係数を設定する.

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.
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 5453 行目に定義があります。

呼び出し関係図:



7.21.3.4 loadAmbient() [2/2]

```
void gg::GgSimpleShader::MaterialBuffer::loadAmbient (
            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 5460 行目に定義があります。

7.21.3.5 loadAmbientAndDiffuse() [1/2]

```
void gg::GgSimpleShader::MaterialBuffer::loadAmbientAndDiffuse (
            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 5544 行目に定義があります。

7.21.3.6 loadAmbientAndDiffuse() [2/2]

```
void gg::GgSimpleShader::MaterialBuffer::loadAmbientAndDiffuse (
            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 5518 行目に定義があります。

7.21.3.7 loadDiffuse() [1/2]

```
void gg::GgSimpleShader::MaterialBuffer::loadDiffuse (
            const GLfloat * diffuse,
            GLint first = 0,
            GLsizei count = 1 ) const  [inline]
```

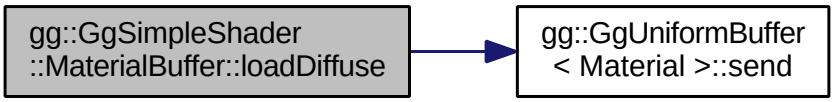
拡散反射係数を設定する.

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 5472 行目に定義があります。

呼び出し関係図:



7.21.3.8 loadDiffuse() [2/2]

```
void gg::GgSimpleShader::MaterialBuffer::loadDiffuse (
            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 5489 行目に定義があります。

7.21.3.9 loadShininess() [1/2]

```
void gg::GgSimpleShader::MaterialBuffer::loadShininess (
            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 5634 行目に定義があります。

7.21.3.10 loadShininess() [2/2]

```
void gg::GgSimpleShader::MaterialBuffer::loadShininess (
            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 5613 行目に定義があります。

7.21.3.11 loadSpecular() [1/2]

```
void gg::GgSimpleShader::MaterialBuffer::loadSpecular (
            const GLfloat * specular,
            GLint first = 0,
            GLsizei count = 1 ) const  [inline]
```

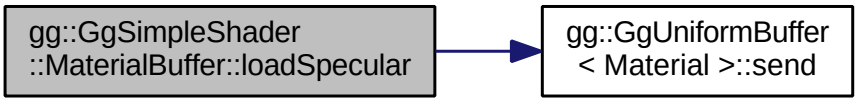
鏡面反射係数を設定する.

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 5506 行目に定義があります。

呼び出し関係図:



7.21.3.12 loadSpecular() [2/2]

```
void gg::GgSimpleShader::MaterialBuffer::loadSpecular (
            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 5587 行目に定義があります。

7.21.3.13 select()

```
void gg::GgSimpleShader::MaterialBuffer::select (
            GLint i = 0 ) const  [inline]
```

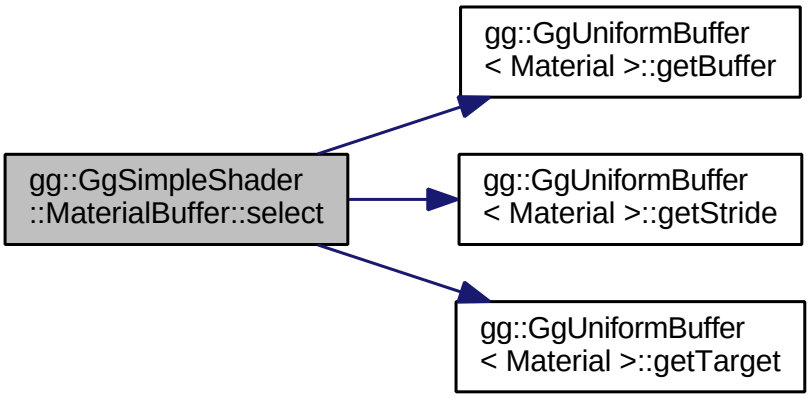
材質を選択する.

引数

i	材質データの uniform block のインデックス.

gg.h の 5544 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

7.22 Window クラス

ウィンドウ関連の処理.

```
#include <Window.h>
```

公開メンバ関数

- Window (const char *title="GLFW Window", int width=640, int height=480, int fullscreen=0, GLFWwindow *share=nullptr)
  コンストラクタ.
- Window (const Window &w)=delete
  コピーコンストラクタは使用禁止.
- Window & operator= (const Window &w)=delete
  代入演算子は使用禁止.
- virtual ~Window ()
  デストラクタ.
- bool begin ()
  Oculus Rift による描画開始.
- void select (int eye, GLfloat *screen, GLfloat *position, GLfloat *orientation)
  Oculus Rift の描画する目の指定.
- void timewarp (const GgMatrix &projection)
  Time Warp 処理に使う投影変換行列の成分の設定 *(DK1, DK2)*.
- void commit (int eye)
  図形の描画を完了する *(CV1* 以降*)*.
- void submit (bool mirror=true)
  フレームを転送する.
- GLFWwindow * get () const

ウィンドウの識別子のポインタを取得する.

- void setClose (int flag=GLFW_TRUE) const
  ウィンドウのクローズフラグを設定する.
- bool shouldClose () const
  ウィンドウを閉じるべきかどうか調べる.
- operator bool ()
  イベントを取得してループを継続すべきかどうか調べる.
- void swapBuffers ()
  カラーバッファを入れ替える.
- GLsizei getWidth () const
  ウィンドウの横幅を得る.
- GLsizei getHeight () const
  ウィンドウの高さを得る.
- const GLsizei * getSize () const
  ウィンドウのサイズを得る.
- void getSize (GLsizei *size) const
  ウィンドウのサイズを得る.
- GLfloat getAspect () const
  ウィンドウのアスペクト比を得る.
- void resetViewport ()
  ビューポートをウィンドウ全体に設定する.
- bool getKey (int key)
  キーが押されているかどうかを判定する.
- GLfloat getArrow (int direction=0, int mods=0) const
  矢印キーの現在の値を得る.
- GLfloat getArrowX (int mods=0) const
  矢印キーの現在の *X* 値を得る.
- GLfloat getArrowY (int mods=0) const
  矢印キーの現在の *Y* 値を得る.
- void getArrow (GLfloat *arrow, int mods=0) const
  矢印キーの現在の値を得る.
- GLfloat getShiftArrowX () const
  SHIFT キーを押しながら矢印キーを押したときの現在の *X* 値を得る.
- GLfloat getShiftArrowY () const
  SHIFT キーを押しながら矢印キーを押したときの現在の *Y* 値を得る.
- void getShiftArrow (GLfloat *shift_arrow) const
  SHIFT キーを押しながら矢印キーを押したときの現在の値を得る.
- GLfloat getControlArrowX () const
  CTRL キーを押しながら矢印キーを押したときの現在の *X* 値を得る.
- GLfloat getControlArrowY () const
  CTRL キーを押しながら矢印キーを押したときの現在の *Y* 値を得る.
- void getControlArrow (GLfloat *control_arrow) const
  CTRL キーを押しながら矢印キーを押したときの現在の値を得る.
- GLfloat getAltArrowX () const
  ALT キーを押しながら矢印キーを押したときの現在の *X* 値を得る.
- GLfloat getAltArrowY () const
  ALT キーを押しながら矢印キーを押したときの現在の *Y* 値を得る.
- void getAltlArrow (GLfloat *alt_arrow) const
  ALT キーを押しながら矢印キーを押したときの現在の値を得る.
- const GLfloat * getMouse () const
  マウスカーソルの現在位置を得る.

- void getMouse (GLfloat *position) const
  マウスカーソルの現在位置を得る.
- const GLfloat getMouse (int direction) const
  マウスカーソルの現在位置を得る.
- GLfloat getMouseX () const
  マウスカーソルの現在位置の X 座標を得る.
- GLfloat getMouseY () const
  マウスカーソルの現在位置の Y 座標を得る.
- const GLfloat * getWheel () const
  マウスホイールの回転量を得る.
- void getWheel (GLfloat *rotation) const
  マウスホイールの回転量を得る.
- GLfloat getWheel (int direction) const
  マウスホイールの回転量を得る.
- const GLfloat getWheelX () const
  マウスホイールの X 方向の回転量を得る.
- const GLfloat getWheelY () const
  マウスホイールの Y 方向の回転量を得る.
- GgMatrix getTranslation (int button=GLFW_MOUSE_BUTTON_1) const
  トラックボール処理を考慮したマウスによる平行移動の変換行列を得る.
- GgMatrix getTrackball (int button=GLFW_MOUSE_BUTTON_1) const
  トラックボールの回転変換行列を得る.
- void * getUserPointer () const
  ユーザーポインタを取り出す.
- void setUserPointer (void *pointer)
  任意のユーザポインタを保存する.
- void setResizeFunc (void(*func)(const Window *window, int width, int height))
  ユーザ定義の *resize* 関数を設定する.
- void setKeyboardFunc (void(*func)(const Window *window, int key, int scancode, int action, int mods))
  ユーザ定義の *keyboard* 関数を設定する.
- void setMouseFunc (void(*func)(const Window *window, int button, int action, int mods))
  ユーザ定義の *mouse* 関数を設定する.
- void setResizeFunc (void(*func)(const Window *window, double x, double y))
  ユーザ定義の *wheel* 関数を設定する.

静的公開メンバ関数

- static void init (int major=0, int minor=1)
  初期化, 最初に一度だけ実行する.

公開変数類

- const int eyeCount = ovrEye_Count
  視点の数.

7.22.1 詳解

ウィンドウ関連の処理.

GLFW を使って OpenGL のウィンドウを操作するラッパークラス.

Window.h の 74 行目に定義があります。

7.22.2 構築子と解体子

7.22.2.1 Window() [1/2]

```
Window::Window (
            const char * title = "GLFW Window",
            int width = 640,
            int height = 480,
            int fullscreen = 0,
            GLFWwindow * share = nullptr )  [inline]
```

コンストラクタ.

引数

title	ウィンドウタイトルの文字列.
width	開くウィンドウの幅.
height	開くウィンドウの高さ.
fullscreen	フルスクリーン表示を行うディスプレイ番号, 0 ならフルスクリーン表示を行わない.
share	共有するコンテキスト, nullptr ならコンテキストを共有しない.

Window.h の 517 行目に定義があります。

7.22.2.2 Window() [2/2]

```
Window::Window (
            const Window & w )  [delete]
```

コピーコンストラクタは使用禁止.

7.22.2.3 ~Window()

```
virtual Window::~Window ( )  [inline], [virtual]
```

デストラクタ.

Window.h の 785 行目に定義があります。

7.22.3 関数詳解

7.22.3.1 begin()

```
bool Window::begin ( )  [inline]
```

Oculus Rift による描画開始.

戻り値

描画可能なら true.

Window.h の 868 行目に定義があります。

7.22.3.2 commit()

```
void Window::commit (
            int eye )  [inline]
```

図形の描画を完了する (CV1 以降).

引数

eye	表示する目.

Window.h の 1020 行目に定義があります。

7.22.3.3 get()

```
GLFWwindow* Window::get ( ) const  [inline]
```

ウィンドウの識別子のポインタを取得する.

戻り値

GLFWwindow 型のウィンドウ識別子のポインタ.

Window.h の 1104 行目に定義があります。

7.22.3.4 getAltArrowX()

```
GLfloat Window::getAltArrowX ( ) const  [inline]
```

ALT キーを押しながら矢印キーを押したときの現在の X 値を得る.

戻り値

ALT キーを押しながら矢印キーを押したときの現在の X 値.

Window.h の 1307 行目に定義があります。

7.22.3.5 getAltArrowY()

```
GLfloat Window::getAltArrowY ( ) const  [inline]
```

ALT キーを押しながら矢印キーを押したときの現在の Y 値を得る.

戻り値

ALT キーを押しながら矢印キーを押したときの現在の Y 値.

Window.h の 1314 行目に定義があります。

7.22.3.6 getAltlArrow()

```
void Window::getAltlArrow (
            GLfloat * alt_arrow ) const  [inline]
```

ALT キーを押しながら矢印キーを押したときの現在の値を得る.

引数

alt_arrow	ALT キーを押しながら矢印キーを押したときの値を格納する GLfloat 型の 2 要素の配列.

Window.h の 1321 行目に定義があります。

7.22.3.7 getArrow() [1/2]

```
void Window::getArrow (
            GLfloat * arrow,
            int mods = 0 ) const  [inline]
```

矢印キーの現在の値を得る.

引数

arrow	矢印キーの値を格納する GLfloat[2] の配列.
mods	修飾キーの状態 (0:なし, 1, SHIFT, 2: CTRL, 3: ALT).

Window.h の 1255 行目に定義があります。

7.22.3.8 getArrow() [2/2]

```
GLfloat Window::getArrow (
```

```
            int direction = 0,
            int mods = 0 ) const  [inline]
```

矢印キーの現在の値を得る.

引数

direction	方向 (0: X, 1:Y).
mods	修飾キーの状態 (0:なし, 1, SHIFT, 2: CTRL, 3: ALT).

戻り値

矢印キーの値.

Window.h の 1231 行目に定義があります。

7.22.3.9 getArrowX()

```
GLfloat Window::getArrowX (
            int mods = 0 ) const  [inline]
```

矢印キーの現在の X 値を得る.

引数

mods	修飾キーの状態 (0:なし, 1, SHIFT, 2: CTRL, 3: ALT).

戻り値

矢印キーの X 値.

Window.h の 1239 行目に定義があります。

7.22.3.10 getArrowY()

```
GLfloat Window::getArrowY (
            int mods = 0 ) const  [inline]
```

矢印キーの現在の Y 値を得る.

引数

mods	修飾キーの状態 (0:なし, 1, SHIFT, 2: CTRL, 3: ALT).

戻り値

矢印キーの Y 値.

Window.h の 1247 行目に定義があります。

7.22.3.11 getAspect()

```
GLfloat Window::getAspect ( ) const  [inline]
```

ウィンドウのアスペクト比を得る.

戻り値

ウィンドウの縦横比.

Window.h の 1206 行目に定義があります。

7.22.3.12 getControlArrow()

```
void Window::getControlArrow (
            GLfloat * control_arrow ) const  [inline]
```

CTRL キーを押しながら矢印キーを押したときの現在の値を得る.

引数

control_arrow	CTRL キーを押しながら矢印キーを押したときの値を格納する GLfloat 型の 2 要素の配列.

Window.h の 1299 行目に定義があります。

7.22.3.13 getControlArrowX()

```
GLfloat Window::getControlArrowX ( ) const  [inline]
```

CTRL キーを押しながら矢印キーを押したときの現在の X 値を得る.

戻り値

CTRL キーを押しながら矢印キーを押したときの現在の X 値.

Window.h の 1285 行目に定義があります。

7.22.3.14 getControlArrowY()

```
GLfloat Window::getControlArrowY ( ) const  [inline]
```

CTRL キーを押しながら矢印キーを押したときの現在の Y 値を得る.

戻り値

CTRL キーを押しながら矢印キーを押したときの現在の Y 値.

Window.h の 1292 行目に定義があります。

7.22.3.15 getHeight()

```
GLsizei Window::getHeight ( ) const  [inline]
```

ウィンドウの高さを得る.

戻り値

ウィンドウの高さ.

Window.h の 1184 行目に定義があります。

7.22.3.16 getKey()

```
bool Window::getKey (
            int key )  [inline]
```

キーが押されているかどうかを判定する.

戻り値

キーが押されていれば true.

Window.h の 1222 行目に定義があります。

7.22.3.17 getMouse() [1/3]

```
const GLfloat* Window::getMouse ( ) const  [inline]
```

マウスカーソルの現在位置を得る.

戻り値

マウスカーソルの現在位置を格納した GLfloat 型の 2 要素の配列.

Window.h の 1329 行目に定義があります。

7.22.3.18 getMouse() [2/3]

```
void Window::getMouse (
            GLfloat * position ) const  [inline]
```

マウスカーソルの現在位置を得る.

引数

position	マウスカーソルの現在位置を格納した GLfloat 型の 2 要素の配列.

Window.h の 1336 行目に定義があります。

7.22.3.19 getMouse() [3/3]

```
const GLfloat Window::getMouse (
            int direction ) const  [inline]
```

マウスカーソルの現在位置を得る.

引数

direction	方向 (0:X, 1:Y).

戻り値

direction 方向のマウスカーソルの現在位置.

Window.h の 1345 行目に定義があります。

7.22.3.20 getMouseX()

```
GLfloat Window::getMouseX ( ) const  [inline]
```

マウスカーソルの現在位置の X 座標を得る.

戻り値

direction 方向のマウスカーソルの X 方向の現在位置.

Window.h の 1352 行目に定義があります。

7.22.3.21 getMouseY()

```
GLfloat Window::getMouseY ( ) const  [inline]
```

マウスカーソルの現在位置の Y 座標を得る.

戻り値

direction 方向のマウスカーソルの Y 方向の現在位置.

Window.h の 1359 行目に定義があります。

7.22.3.22 getShiftArrow()

```
void Window::getShiftArrow (
            GLfloat * shift_arrow ) const  [inline]
```

SHIFT キーを押しながら矢印キーを押したときの現在の値を得る.

引数

shift_arrow	SHIFT キーを押しながら矢印キーを押したときの値を格納する GLfloat 型の 2 要素の配列.

Window.h の 1277 行目に定義があります。

7.22.3.23 getShiftArrowX()

```
GLfloat Window::getShiftArrowX ( ) const  [inline]
```

SHIFT キーを押しながら矢印キーを押したときの現在の X 値を得る.

戻り値

SHIFT キーを押しながら矢印キーを押したときの現在の X 値.

Window.h の 1263 行目に定義があります。

7.22.3.24 getShiftArrowY()

```
GLfloat Window::getShiftArrowY ( ) const  [inline]
```

SHIFT キーを押しながら矢印キーを押したときの現在の Y 値を得る.

戻り値

SHIFT キーを押しながら矢印キーを押したときの現在の Y 値.

Window.h の 1270 行目に定義があります。

7.22.3.25 getSize() [1/2]

```
const GLsizei* Window::getSize ( ) const  [inline]
```

ウィンドウのサイズを得る.

戻り値

ウィンドウの幅と高さを格納した GLsizei 型の 2 要素の配列.

Window.h の 1191 行目に定義があります。

7.22.3.26 getSize() [2/2]

```
void Window::getSize (
            GLsizei * size ) const  [inline]
```

ウィンドウのサイズを得る.

引数

size	ウィンドウの幅と高さを格納した GLsizei 型の 2 要素の配列.

Window.h の 1198 行目に定義があります。

7.22.3.27 getTrackball()

```
GgMatrix Window::getTrackball (
            int button = GLFW_MOUSE_BUTTON_1 ) const  [inline]
```

トラックボールの回転変換行列を得る.

引数

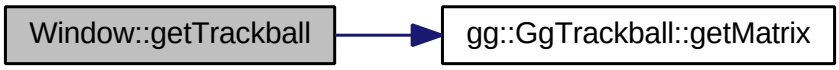
button	回転変換行列を取得するマウスボタン (GLFW_MOUSE_BUTTON_[1,2]).

戻り値

回転を行う GgMarix 型の変換行列.

Window.h の 1410 行目に定義があります。

呼び出し関係図:



7.22.3.28 getTranslation()

```
GgMatrix Window::getTranslation (
            int button = GLFW_MOUSE_BUTTON_1 ) const  [inline]
```

トラックボール処理を考慮したマウスによる平行移動の変換行列を得る.

引数

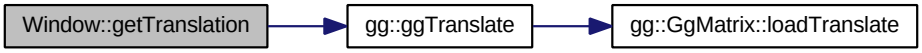
button	平行移動量を取得するマウスボタン (GLFW_MOUSE_BUTTON_[1,2]).

戻り値

平行移動を行う GgMarix 型の変換行列.

Window.h の 1402 行目に定義があります。

呼び出し関係図:



7.22.3.29 getUserPointer()

```
void* Window::getUserPointer ( ) const  [inline]
```

ユーザーポインタを取り出す.

戻り値

保存されているユーザポインタ.

Window.h の 1417 行目に定義があります。

7.22.3.30 getWheel() [1/3]

```
const GLfloat* Window::getWheel ( ) const  [inline]
```

マウスホイールの回転量を得る.

戻り値

マウスホイールの回転量を格納した GLfloat 型の 2 要素の配列.

Window.h の 1366 行目に定義があります。

7.22.3.31 getWheel() [2/3]

```
void Window::getWheel (
            GLfloat * rotation ) const  [inline]
```

マウスホイールの回転量を得る.

引数

rotation	マウスホイールの回転量を格納した GLfloat 型の 2 要素の配列.

Window.h の 1373 行目に定義があります。

7.22.3.32 getWheel() [3/3]

```
GLfloat Window::getWheel (
            int direction ) const  [inline]
```

マウスホイールの回転量を得る.

引数

direction	方向 (0:X, 1:Y).

戻り値

direction 方向のマウスホイールの回転量.

Window.h の 1382 行目に定義があります。

7.22.3.33 getWheelX()

```
const GLfloat Window::getWheelX ( ) const  [inline]
```

マウスホイールの X 方向の回転量を得る.

Window.h の 1388 行目に定義があります。

7.22.3.34 getWheelY()

```
const GLfloat Window::getWheelY ( ) const  [inline]
```

マウスホイールの Y 方向の回転量を得る.

Window.h の 1394 行目に定義があります。

7.22.3.35 getWidth()

```
GLsizei Window::getWidth ( ) const  [inline]
```

ウィンドウの横幅を得る.

戻り値

ウィンドウの横幅.

Window.h の 1177 行目に定義があります。

7.22.3.36 init()

```
static void Window::init (
            int major = 0,
            int minor = 1 )  [inline], [static]
```

初期化, 最初に一度だけ実行する.

引数

major	使用する OpenGL の major 番号, 0 なら無指定.
minor	使用する OpenGL の minor 番号, major 番号が 0 なら無視.

Window.h の 411 行目に定義があります。

7.22.3.37 operator bool()

```
Window::operator bool ( )  [inline]
```

イベントを取得してループを継続すべきかどうか調べる.

戻り値

ループを継続すべきなら true.

Window.h の 1126 行目に定義があります。

呼び出し関係図:

7.22.3.38 operator=()

```
Window& Window::operator= (
            const Window & w )  [delete]
```

代入演算子は使用禁止.

7.22.3.39 resetViewport()

```
void Window::resetViewport ( )  [inline]
```

ビューポートをウィンドウ全体に設定する.

Window.h の 1212 行目に定義があります。

7.22.3.40 select()

```
void Window::select (
            int eye,
            GLfloat * screen,
            GLfloat * position,
            GLfloat * orientation )  [inline]
```

Oculus Rift の描画する目の指定.

引数

eye	表示する目.
screen	HMD の視野の視錐台.
position	HMD の位置.
orientation	HMD の方法の四元数.

Window.h の 933 行目に定義があります。

7.22.3.41 setClose()

```
void Window::setClose (
            int flag = GLFW_TRUE ) const  [inline]
```

ウィンドウのクローズフラグを設定する.

引数

flag	クローズフラグ, 0 (GLFW_FALSE) 以外ならウィンドウを閉じる.

Window.h の 1111 行目に定義があります。

7.22.3.42 setKeyboardFunc()

```
void Window::setKeyboardFunc (
            void(*)(const Window *window, int key, int scancode, int action, int mods) func )
[inline]
```

ユーザ定義の keyboard 関数を設定する.

引数

func	ユーザ定義の keyboard 関数, キーボードの操作時に呼び出される.

Window.h の 1438 行目に定義があります。

7.22.3.43 setMouseFunc()

```
void Window::setMouseFunc (
            void(*)(const Window *window, int button, int action, int mods) func )  [inline]
```

ユーザ定義の mouse 関数を設定する.

引数

func	ユーザ定義の mouse 関数, マウスボタンの操作時に呼び出される.

Window.h の 1445 行目に定義があります。

7.22.3.44 setResizeFunc() [1/2]

```
void Window::setResizeFunc (
            void(*)(const Window *window, double x, double y) func )  [inline]
```

ユーザ定義の wheel 関数を設定する.

引数

func	ユーザ定義の wheel 関数, マウスホイールの操作時に呼び出される.

Window.h の 1452 行目に定義があります。

7.22.3.45 setResizeFunc() [2/2]

```
void Window::setResizeFunc (
            void(*)(const Window *window, int width, int height) func )  [inline]
```

ユーザ定義の resize 関数を設定する.

引数

func	ユーザ定義の resize 関数, ウィンドウのサイズ変更時に呼び出される.

Window.h の 1431 行目に定義があります。

7.22.3.46 setUserPointer()

```
void Window::setUserPointer (
            void * pointer )  [inline]
```

任意のユーザポインタを保存する.

引数

pointer	保存するユーザポインタ.

Window.h の 1424 行目に定義があります。

7.22.3.47 shouldClose()

```
bool Window::shouldClose ( ) const  [inline]
```

ウィンドウを閉じるべきかどうか調べる.

戻り値

ウィンドウを閉じるべきなら true.

Window.h の 1118 行目に定義があります。

7.22.3.48 submit()

```
void Window::submit (
            bool mirror = true )  [inline]
```

フレームを転送する.

引数

mirror	true ならミラー表示を行う, デフォルトは true.

Window.h の 1037 行目に定義があります。

7.22.3.49 swapBuffers()

```
void Window::swapBuffers ( )  [inline]
```

カラーバッファを入れ替える.

Window.h の 1161 行目に定義があります。

7.22.3.50 timewarp()

```
void Window::timewarp (
            const GgMatrix & projection )  [inline]
```

Time Warp 処理に使う投影変換行列の成分の設定 (DK1, DK2).

引数

projection	投影変換行列.

Window.h の 1007 行目に定義があります。

呼び出し関係図:

7.22.4 メンバ詳解

7.22.4.1 eyeCount

```
const int Window::eyeCount = ovrEye_Count
```

視点の数.

Window.h の 1093 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:

- Window.h

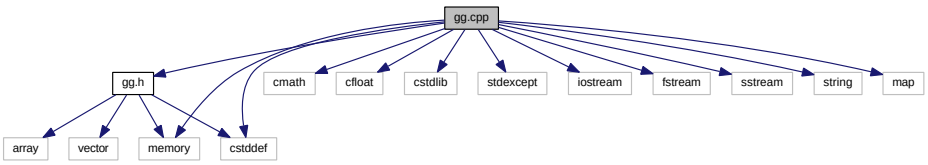
Chapter 8

ファイル詳解

8.1 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>
```

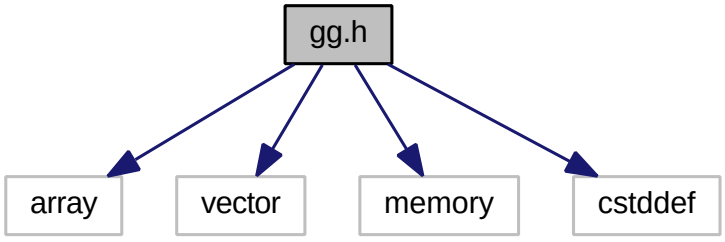
gg.cpp の依存先関係図:



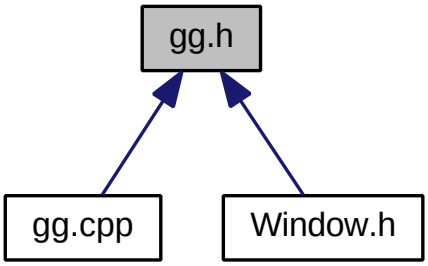
8.2 gg.h ファイル

```
#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()
  OpenGL のエラーの発生を検知したときにソースファイル名と行番号を表示する.
- #define ggFBOError()
  FBO のエラーの発生を検知したときにソースファイル名と行番号を表示する.

型定義

- using gg::GgVector = std::array< GLfloat, 4 >
  4 要素の単精度実数の配列.

列挙型

- enum gg::BindingPoints { gg::LightBindingPoint = 0, gg::MaterialBindingPoint }
  光源と材質の *uniform buffer object* の結合ポイント.

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 *p↩
  Height, 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, 初期化に取得される.

8.2.1 マクロ定義詳解

8.2.1.1 ggError

```
#define ggError( )
```

OpenGL のエラーの発生を検知したときにソースファイル名と行番号を表示する.

このマクロを置いた場所（より前）で OpenGL のエラーが発生していた時に, このマクロを置いたソースファイル名と行番号を出力する. リリースビルド時には無視される.

gg.h の 1344 行目に定義があります。

8.2.1.2 ggFBOError

```
#define ggFBOError( )
```

FBO のエラーの発生を検知したときにソースファイル名と行番号を表示する.

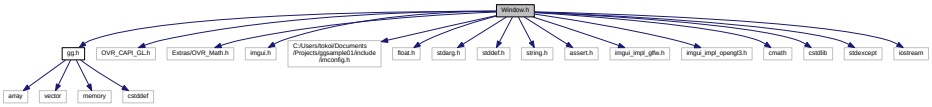
このマクロを置いた場所（より前）で FBO のエラーが発生していた時に, このマクロを置いたソースファイル名と行番号を出力する. リリースビルド時には無視される.

gg.h の 1368 行目に定義があります。

8.3 Window.h ファイル

```
#include "gg.h"
#include <OVR_CAPI_GL.h>
#include <Extras/OVR_Math.h>
#include "imgui.h"
#include "imgui_impl_glfw.h"
#include "imgui_impl_opengl3.h"
#include <cmath>
#include <cstdlib>
#include <stdexcept>
#include <iostream>
```

Window.h の依存先関係図:



クラス

- class Window
  ウィンドウ関連の処理.

マクロ定義

- #define USE_OCULUS_RIFT
- #define USE_IMGUI

8.3.1 マクロ定義詳解

8.3.1.1 USE_IMGUI

```
#define USE_IMGUI
```

Window.h の 54 行目に定義があります。

8.3.1.2 USE_OCULUS_RIFT

```
#define USE_OCULUS_RIFT
```

Window.h の 33 行目に定義があります。