

ゲームグラフィックス特論

構築: Doxygen 1.9.1

| | |
|--------------------------------------------------|-----------|
| 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 ggsample01 | 3 |
| 2.1 ゲーム グラフィックス特論A 第1回 宿題 | 3 |
| 2.2 宿題プログラムの作成に必要な環境 | 3 |
| 2.3 補足 | 3 |
| 2.4 宿題プログラム用補助プログラムについて | 4 |
| 2.4.1 補助プログラムのドキュメント | 4 |
| 2.4.2 補助プログラムの使い方 | 4 |
| 2.4.3 Oculus Rift を使う場合 | 5 |
| 2.4.4 Dear ImGui を使う場合 | 5 |
| 2.4.4.1 imconfig.h の変更点 | 6 |
| 3 名前空間索引 | 7 |
| 3.1 名前空間一覧 | 7 |
| 4 階層索引 | 9 |
| 4.1 クラス階層 | 9 |
| 5 クラス索引 | 11 |
| 5.1 クラス一覧 | 11 |
| 6 ファイル索引 | 13 |
| 6.1 ファイル一覧 | 13 |
| 7 名前空間詳解 | 15 |
| 7.1 gg 名前空間 | 15 |
| 7.1.1 詳解 | 21 |
| 7.1.2 型定義詳解 | 21 |
| 7.1.2.1 GgVector | 21 |
| 7.1.3 列挙型詳解 | 21 |
| 7.1.3.1 BindingPoints | 21 |
| 7.1.4 関数詳解 | 21 |
| 7.1.4.1 ggError() | 21 |
| 7.1.4.2 ggFBOError() | 22 |
| 7.1.4.3 ggArraysObj() | 22 |
| 7.1.4.4 ggConjugate() | 23 |
| 7.1.4.5 ggCreateComputeShader() | 23 |
| 7.1.4.6 ggCreateNormalMap() | 24 |
| 7.1.4.7 ggCreateShader() | 25 |
| 7.1.4.8 ggCross() [1/2] | 26 |
| 7.1.4.9 ggCross() [2/2] | 26 |
| 7.1.4.10 ggDistance3() | 26 |
| 7.1.4.11 ggDistance4() | 27 |

| | |
|-----------------------------------------------|----|
| 7.1.4.12 ggDot3() [1/2] | 28 |
| 7.1.4.13 ggDot3() [2/2] | 28 |
| 7.1.4.14 ggDot4() [1/2] | 29 |
| 7.1.4.15 ggDot4() [2/2] | 29 |
| 7.1.4.16 ggElementsMesh() | 30 |
| 7.1.4.17 ggElementsObj() | 31 |
| 7.1.4.18 ggElementsSphere() | 31 |
| 7.1.4.19 ggEllipse() | 33 |
| 7.1.4.20 ggEulerQuaternion() [1/2] | 33 |
| 7.1.4.21 ggEulerQuaternion() [2/2] | 34 |
| 7.1.4.22 ggFrustum() | 35 |
| 7.1.4.23 ggIdentity() | 36 |
| 7.1.4.24 ggIdentityQuaternion() | 36 |
| 7.1.4.25 ggInit() | 37 |
| 7.1.4.26 ggInvert() [1/2] | 37 |
| 7.1.4.27 ggInvert() [2/2] | 38 |
| 7.1.4.28 ggLength3() [1/2] | 38 |
| 7.1.4.29 ggLength3() [2/2] | 39 |
| 7.1.4.30 ggLength4() [1/2] | 40 |
| 7.1.4.31 ggLength4() [2/2] | 40 |
| 7.1.4.32 ggLoadComputeShader() | 41 |
| 7.1.4.33 ggLoadHeight() | 41 |
| 7.1.4.34 ggLoadImage() | 42 |
| 7.1.4.35 ggLoadShader() | 43 |
| 7.1.4.36 ggLoadSimpleObj() [1/2] | 44 |
| 7.1.4.37 ggLoadSimpleObj() [2/2] | 44 |
| 7.1.4.38 ggLoadTexture() | 45 |
| 7.1.4.39 ggLookat() [1/3] | 46 |
| 7.1.4.40 ggLookat() [2/3] | 47 |
| 7.1.4.41 ggLookat() [3/3] | 47 |
| 7.1.4.42 ggMatrixQuaternion() [1/2] | 48 |
| 7.1.4.43 ggMatrixQuaternion() [2/2] | 49 |
| 7.1.4.44 ggNorm() | 50 |
| 7.1.4.45 ggNormal() | 50 |
| 7.1.4.46 ggNormalize() | 51 |
| 7.1.4.47 ggNormalize3() [1/3] | 51 |
| 7.1.4.48 ggNormalize3() [2/3] | 52 |
| 7.1.4.49 ggNormalize3() [3/3] | 53 |
| 7.1.4.50 ggNormalize4() [1/3] | 53 |
| 7.1.4.51 ggNormalize4() [2/3] | 54 |
| 7.1.4.52 ggNormalize4() [3/3] | 55 |
| 7.1.4.53 ggOrthogonal() | 55 |

| | |
|----------------------------------------|----|
| 7.1.4.54 ggPerspective() | 56 |
| 7.1.4.55 ggPointsCube() | 57 |
| 7.1.4.56 ggPointsSphere() | 57 |
| 7.1.4.57 ggQuaternion() [1/2] | 58 |
| 7.1.4.58 ggQuaternion() [2/2] | 58 |
| 7.1.4.59 ggQuaternionMatrix() | 59 |
| 7.1.4.60 ggQuaternionTransposeMatrix() | 60 |
| 7.1.4.61 ggReadImage() | 61 |
| 7.1.4.62 ggRectangle() | 62 |
| 7.1.4.63 ggRotate() [1/5] | 62 |
| 7.1.4.64 ggRotate() [2/5] | 63 |
| 7.1.4.65 ggRotate() [3/5] | 63 |
| 7.1.4.66 ggRotate() [4/5] | 64 |
| 7.1.4.67 ggRotate() [5/5] | 65 |
| 7.1.4.68 ggRotateQuaternion() [1/3] | 65 |
| 7.1.4.69 ggRotateQuaternion() [2/3] | 66 |
| 7.1.4.70 ggRotateQuaternion() [3/3] | 67 |
| 7.1.4.71 ggRotateX() | 68 |
| 7.1.4.72 ggRotateY() | 68 |
| 7.1.4.73 ggRotateZ() | 69 |
| 7.1.4.74 ggSaveColor() | 69 |
| 7.1.4.75 ggSaveDepth() | 70 |
| 7.1.4.76 ggSaveTga() | 71 |
| 7.1.4.77 ggScale() [1/3] | 71 |
| 7.1.4.78 ggScale() [2/3] | 72 |
| 7.1.4.79 ggScale() [3/3] | 73 |
| 7.1.4.80 ggSlerp() [1/4] | 73 |
| 7.1.4.81 ggSlerp() [2/4] | 74 |
| 7.1.4.82 ggSlerp() [3/4] | 75 |
| 7.1.4.83 ggSlerp() [4/4] | 76 |
| 7.1.4.84 ggTranslate() [1/3] | 77 |
| 7.1.4.85 ggTranslate() [2/3] | 78 |
| 7.1.4.86 ggTranslate() [3/3] | 79 |
| 7.1.4.87 ggTranspose() | 79 |
| 7.1.4.88 operator*() [1/3] | 80 |
| 7.1.4.89 operator*() [2/3] | 80 |
| 7.1.4.90 operator*() [3/3] | 81 |
| 7.1.4.91 operator*=() [1/2] | 81 |
| 7.1.4.92 operator*=() [2/2] | 82 |
| 7.1.4.93 operator+() [1/3] | 82 |
| 7.1.4.94 operator+() [2/3] | 82 |
| 7.1.4.95 operator+() [3/3] | 83 |

| | |
|----------------------------------------------------|-----------|
| 7.1.4.96 <code>operator+=()</code> [1/2] | 83 |
| 7.1.4.97 <code>operator+=()</code> [2/2] | 84 |
| 7.1.4.98 <code>operator-()</code> [1/3] | 84 |
| 7.1.4.99 <code>operator-()</code> [2/3] | 85 |
| 7.1.4.100 <code>operator-()</code> [3/3] | 85 |
| 7.1.4.101 <code>operator-()</code> [1/2] | 85 |
| 7.1.4.102 <code>operator-()</code> [2/2] | 86 |
| 7.1.4.103 <code>operator/()</code> [1/3] | 86 |
| 7.1.4.104 <code>operator/()</code> [2/3] | 87 |
| 7.1.4.105 <code>operator/()</code> [3/3] | 87 |
| 7.1.4.106 <code>operator/=()</code> [1/2] | 88 |
| 7.1.4.107 <code>operator/=()</code> [2/2] | 88 |
| 7.1.5 変数詳解 | 88 |
| 7.1.5.1 <code>ggBufferAlignment</code> | 88 |
| 8 クラス詳解 | 89 |
| 8.1 <code>GgApp</code> クラス | 89 |
| 8.1.1 詳解 | 89 |
| 8.1.2 関数詳解 | 89 |
| 8.1.2.1 <code>main()</code> | 89 |
| 8.2 <code>gg::GgBuffer< T ></code> クラステンプレート | 90 |
| 8.2.1 詳解 | 90 |
| 8.2.2 構築子と解体子 | 91 |
| 8.2.2.1 <code>GgBuffer()</code> [1/2] | 91 |
| 8.2.2.2 <code>~GgBuffer()</code> | 91 |
| 8.2.2.3 <code>GgBuffer()</code> [2/2] | 91 |
| 8.2.3 関数詳解 | 92 |
| 8.2.3.1 <code>bind()</code> | 92 |
| 8.2.3.2 <code>copy()</code> | 92 |
| 8.2.3.3 <code>getBuffer()</code> | 92 |
| 8.2.3.4 <code>getCount()</code> | 93 |
| 8.2.3.5 <code>getStride()</code> | 93 |
| 8.2.3.6 <code>getTarget()</code> | 93 |
| 8.2.3.7 <code>map()</code> [1/2] | 94 |
| 8.2.3.8 <code>map()</code> [2/2] | 94 |
| 8.2.3.9 <code>operator=()</code> | 94 |
| 8.2.3.10 <code>read()</code> | 95 |
| 8.2.3.11 <code>send()</code> | 95 |
| 8.2.3.12 <code>unbind()</code> | 95 |
| 8.2.3.13 <code>unmap()</code> | 96 |
| 8.3 <code>gg::GgColorTexture</code> クラス | 96 |
| 8.3.1 詳解 | 96 |

| | |
|--------------------------------|-----|
| 8.3.2 構築子と解体子 | 96 |
| 8.3.2.1 GgColorTexture() [1/3] | 97 |
| 8.3.2.2 GgColorTexture() [2/3] | 97 |
| 8.3.2.3 GgColorTexture() [3/3] | 98 |
| 8.3.2.4 ~GgColorTexture() | 98 |
| 8.3.3 関数詳解 | 98 |
| 8.3.3.1 load() [1/2] | 99 |
| 8.3.3.2 load() [2/2] | 99 |
| 8.4 gg::GgElements クラス | 100 |
| 8.4.1 詳解 | 101 |
| 8.4.2 構築子と解体子 | 101 |
| 8.4.2.1 GgElements() [1/2] | 102 |
| 8.4.2.2 GgElements() [2/2] | 102 |
| 8.4.2.3 ~GgElements() | 102 |
| 8.4.3 関数詳解 | 103 |
| 8.4.3.1 draw() | 103 |
| 8.4.3.2 getIndexBuffer() | 103 |
| 8.4.3.3 getIndexCount() | 104 |
| 8.4.3.4 load() | 104 |
| 8.4.3.5 send() | 104 |
| 8.5 gg::GgMatrix クラス | 105 |
| 8.5.1 詳解 | 109 |
| 8.5.2 構築子と解体子 | 109 |
| 8.5.2.1 GgMatrix() [1/3] | 109 |
| 8.5.2.2 GgMatrix() [2/3] | 110 |
| 8.5.2.3 GgMatrix() [3/3] | 110 |
| 8.5.2.4 ~GgMatrix() | 111 |
| 8.5.3 関数詳解 | 111 |
| 8.5.3.1 add() [1/2] | 111 |
| 8.5.3.2 add() [2/2] | 111 |
| 8.5.3.3 divide() [1/2] | 112 |
| 8.5.3.4 divide() [2/2] | 113 |
| 8.5.3.5 frustum() | 114 |
| 8.5.3.6 get() [1/2] | 115 |
| 8.5.3.7 get() [2/2] | 116 |
| 8.5.3.8 invert() | 116 |
| 8.5.3.9 load() [1/2] | 117 |
| 8.5.3.10 load() [2/2] | 117 |
| 8.5.3.11 loadAdd() [1/2] | 118 |
| 8.5.3.12 loadAdd() [2/2] | 119 |
| 8.5.3.13 loadDivide() [1/2] | 119 |
| 8.5.3.14 loadDivide() [2/2] | 120 |

| | |
|--------------------------------|-----|
| 8.5.3.15 loadFrustum() | 121 |
| 8.5.3.16 loadIdentity() | 121 |
| 8.5.3.17 loadInvert() [1/2] | 122 |
| 8.5.3.18 loadInvert() [2/2] | 122 |
| 8.5.3.19 loadLookat() [1/3] | 123 |
| 8.5.3.20 loadLookat() [2/3] | 124 |
| 8.5.3.21 loadLookat() [3/3] | 125 |
| 8.5.3.22 loadMultiply() [1/2] | 126 |
| 8.5.3.23 loadMultiply() [2/2] | 127 |
| 8.5.3.24 loadNormal() [1/2] | 128 |
| 8.5.3.25 loadNormal() [2/2] | 128 |
| 8.5.3.26 loadOrthogonal() | 129 |
| 8.5.3.27 loadPerspective() | 130 |
| 8.5.3.28 loadRotate() [1/5] | 131 |
| 8.5.3.29 loadRotate() [2/5] | 131 |
| 8.5.3.30 loadRotate() [3/5] | 132 |
| 8.5.3.31 loadRotate() [4/5] | 133 |
| 8.5.3.32 loadRotate() [5/5] | 134 |
| 8.5.3.33 loadRotateX() | 134 |
| 8.5.3.34 loadRotateY() | 135 |
| 8.5.3.35 loadRotateZ() | 136 |
| 8.5.3.36 loadScale() [1/3] | 136 |
| 8.5.3.37 loadScale() [2/3] | 137 |
| 8.5.3.38 loadScale() [3/3] | 138 |
| 8.5.3.39 loadSubtract() [1/2] | 138 |
| 8.5.3.40 loadSubtract() [2/2] | 139 |
| 8.5.3.41 loadTranslate() [1/3] | 140 |
| 8.5.3.42 loadTranslate() [2/3] | 140 |
| 8.5.3.43 loadTranslate() [3/3] | 141 |
| 8.5.3.44 loadTranspose() [1/2] | 142 |
| 8.5.3.45 loadTranspose() [2/2] | 142 |
| 8.5.3.46 lookat() [1/3] | 143 |
| 8.5.3.47 lookat() [2/3] | 144 |
| 8.5.3.48 lookat() [3/3] | 144 |
| 8.5.3.49 multiply() [1/2] | 145 |
| 8.5.3.50 multiply() [2/2] | 146 |
| 8.5.3.51 normal() | 146 |
| 8.5.3.52 operator*() [1/3] | 147 |
| 8.5.3.53 operator*() [2/3] | 148 |
| 8.5.3.54 operator*() [3/3] | 148 |
| 8.5.3.55 operator*=(() [1/2] | 148 |
| 8.5.3.56 operator*=(() [2/2] | 149 |

| | |
|---------------------------------------|-----|
| 8.5.3.57 operator+() [1/2] | 149 |
| 8.5.3.58 operator+() [2/2] | 150 |
| 8.5.3.59 operator+=() [1/2] | 150 |
| 8.5.3.60 operator+=() [2/2] | 151 |
| 8.5.3.61 operator-() [1/2] | 151 |
| 8.5.3.62 operator-() [2/2] | 152 |
| 8.5.3.63 operator-() [1/2] | 152 |
| 8.5.3.64 operator-() [2/2] | 153 |
| 8.5.3.65 operator/() [1/2] | 153 |
| 8.5.3.66 operator/() [2/2] | 154 |
| 8.5.3.67 operator/=() [1/2] | 154 |
| 8.5.3.68 operator/=() [2/2] | 155 |
| 8.5.3.69 operator=() [1/2] | 155 |
| 8.5.3.70 operator=() [2/2] | 156 |
| 8.5.3.71 orthogonal() | 156 |
| 8.5.3.72 perspective() | 157 |
| 8.5.3.73 projection() [1/4] | 158 |
| 8.5.3.74 projection() [2/4] | 158 |
| 8.5.3.75 projection() [3/4] | 158 |
| 8.5.3.76 projection() [4/4] | 159 |
| 8.5.3.77 rotate() [1/5] | 159 |
| 8.5.3.78 rotate() [2/5] | 160 |
| 8.5.3.79 rotate() [3/5] | 160 |
| 8.5.3.80 rotate() [4/5] | 161 |
| 8.5.3.81 rotate() [5/5] | 161 |
| 8.5.3.82 rotateX() | 162 |
| 8.5.3.83 rotateY() | 163 |
| 8.5.3.84 rotateZ() | 164 |
| 8.5.3.85 scale() [1/3] | 164 |
| 8.5.3.86 scale() [2/3] | 165 |
| 8.5.3.87 scale() [3/3] | 165 |
| 8.5.3.88 subtract() [1/2] | 166 |
| 8.5.3.89 subtract() [2/2] | 167 |
| 8.5.3.90 translate() [1/3] | 168 |
| 8.5.3.91 translate() [2/3] | 168 |
| 8.5.3.92 translate() [3/3] | 169 |
| 8.5.3.93 transpose() | 170 |
| 8.5.4 フレンドと関連関数の詳解 | 171 |
| 8.5.4.1 GgQuaternion | 171 |
| 8.6 gg::GgNormalTexture クラス | 171 |
| 8.6.1 詳解 | 172 |
| 8.6.2 構築子と解体子 | 172 |

| | |
|------------------------------------------------|-----|
| 8.6.2.1 GgNormalTexture() [1/3] | 172 |
| 8.6.2.2 GgNormalTexture() [2/3] | 172 |
| 8.6.2.3 GgNormalTexture() [3/3] | 173 |
| 8.6.2.4 ~GgNormalTexture() | 173 |
| 8.6.3 関数詳解 | 173 |
| 8.6.3.1 load() [1/2] | 174 |
| 8.6.3.2 load() [2/2] | 174 |
| 8.7 gg::GgPoints クラス | 175 |
| 8.7.1 詳解 | 176 |
| 8.7.2 構築子と解体子 | 176 |
| 8.7.2.1 GgPoints() [1/2] | 177 |
| 8.7.2.2 GgPoints() [2/2] | 177 |
| 8.7.2.3 ~GgPoints() | 177 |
| 8.7.3 関数詳解 | 177 |
| 8.7.3.1 draw() | 177 |
| 8.7.3.2 getBuffer() | 178 |
| 8.7.3.3 getCount() | 178 |
| 8.7.3.4 load() | 178 |
| 8.7.3.5 send() | 179 |
| 8.8 gg::GgPointShader クラス | 179 |
| 8.8.1 詳解 | 181 |
| 8.8.2 構築子と解体子 | 181 |
| 8.8.2.1 GgPointShader() [1/2] | 181 |
| 8.8.2.2 GgPointShader() [2/2] | 181 |
| 8.8.2.3 ~GgPointShader() | 181 |
| 8.8.3 関数詳解 | 182 |
| 8.8.3.1 get() | 182 |
| 8.8.3.2 load() | 182 |
| 8.8.3.3 loadMatrix() [1/2] | 183 |
| 8.8.3.4 loadMatrix() [2/2] | 183 |
| 8.8.3.5 loadModelviewMatrix() [1/2] | 184 |
| 8.8.3.6 loadModelviewMatrix() [2/2] | 184 |
| 8.8.3.7 loadProjectionMatrix() [1/2] | 185 |
| 8.8.3.8 loadProjectionMatrix() [2/2] | 185 |
| 8.8.3.9 unuse() | 186 |
| 8.8.3.10 use() [1/5] | 186 |
| 8.8.3.11 use() [2/5] | 186 |
| 8.8.3.12 use() [3/5] | 187 |
| 8.8.3.13 use() [4/5] | 187 |
| 8.8.3.14 use() [5/5] | 187 |
| 8.9 gg::GgQuaternion クラス | 188 |
| 8.9.1 詳解 | 193 |

| | |
|-------------------------------------|-----|
| 8.9.2 構築子と解体子 | 193 |
| 8.9.2.1 GgQuaternion() [1/5] | 193 |
| 8.9.2.2 GgQuaternion() [2/5] | 193 |
| 8.9.2.3 GgQuaternion() [3/5] | 194 |
| 8.9.2.4 GgQuaternion() [4/5] | 194 |
| 8.9.2.5 GgQuaternion() [5/5] | 195 |
| 8.9.2.6 ~GgQuaternion() | 195 |
| 8.9.3 関数詳解 | 196 |
| 8.9.3.1 add() [1/4] | 196 |
| 8.9.3.2 add() [2/4] | 196 |
| 8.9.3.3 add() [3/4] | 197 |
| 8.9.3.4 add() [4/4] | 197 |
| 8.9.3.5 conjugate() | 198 |
| 8.9.3.6 divide() [1/4] | 199 |
| 8.9.3.7 divide() [2/4] | 200 |
| 8.9.3.8 divide() [3/4] | 200 |
| 8.9.3.9 divide() [4/4] | 201 |
| 8.9.3.10 euler() [1/2] | 202 |
| 8.9.3.11 euler() [2/2] | 202 |
| 8.9.3.12 get() | 203 |
| 8.9.3.13 getConjugateMatrix() [1/3] | 204 |
| 8.9.3.14 getConjugateMatrix() [2/3] | 204 |
| 8.9.3.15 getConjugateMatrix() [3/3] | 205 |
| 8.9.3.16 getMatrix() [1/3] | 205 |
| 8.9.3.17 getMatrix() [2/3] | 206 |
| 8.9.3.18 getMatrix() [3/3] | 206 |
| 8.9.3.19 invert() | 207 |
| 8.9.3.20 load() [1/4] | 208 |
| 8.9.3.21 load() [2/4] | 208 |
| 8.9.3.22 load() [3/4] | 208 |
| 8.9.3.23 load() [4/4] | 209 |
| 8.9.3.24 loadAdd() [1/4] | 210 |
| 8.9.3.25 loadAdd() [2/4] | 211 |
| 8.9.3.26 loadAdd() [3/4] | 211 |
| 8.9.3.27 loadAdd() [4/4] | 212 |
| 8.9.3.28 loadConjugate() | 213 |
| 8.9.3.29 loadConjugate() [2/2] | 213 |
| 8.9.3.30 loadDivide() [1/4] | 214 |
| 8.9.3.31 loadDivide() [2/4] | 215 |
| 8.9.3.32 loadDivide() [3/4] | 215 |
| 8.9.3.33 loadDivide() [4/4] | 216 |
| 8.9.3.34 loadEuler() | 217 |

| | |
|------------------------------------------|-----|
| 8.9.3.35 loadEuler() [2/2] | 217 |
| 8.9.3.36 loadIdentity() | 218 |
| 8.9.3.37 loadInvert() [1/2] | 219 |
| 8.9.3.38 loadInvert() [2/2] | 220 |
| 8.9.3.39 loadMatrix() [1/2] | 221 |
| 8.9.3.40 loadMatrix() [2/2] | 222 |
| 8.9.3.41 loadMultiply() [1/4] | 223 |
| 8.9.3.42 loadMultiply() [2/4] | 224 |
| 8.9.3.43 loadMultiply() [3/4] | 224 |
| 8.9.3.44 loadMultiply() [4/4] | 225 |
| 8.9.3.45 loadNormalize() [1/2] | 226 |
| 8.9.3.46 loadNormalize() [2/2] | 226 |
| 8.9.3.47 loadRotate() [1/3] | 227 |
| 8.9.3.48 loadRotate() [2/3] | 227 |
| 8.9.3.49 loadRotate() [3/3] | 228 |
| 8.9.3.50 loadRotateX() | 229 |
| 8.9.3.51 loadRotateY() | 230 |
| 8.9.3.52 loadRotateZ() | 230 |
| 8.9.3.53 loadSlerp() [1/4] | 231 |
| 8.9.3.54 loadSlerp() [2/4] | 232 |
| 8.9.3.55 loadSlerp() [3/4] | 232 |
| 8.9.3.56 loadSlerp() [4/4] | 233 |
| 8.9.3.57 loadSubtract() [1/4] | 234 |
| 8.9.3.58 loadSubtract() [2/4] | 234 |
| 8.9.3.59 loadSubtract() [3/4] | 235 |
| 8.9.3.60 loadSubtract() [4/4] | 235 |
| 8.9.3.61 multiply() [1/4] | 236 |
| 8.9.3.62 multiply() [2/4] | 237 |
| 8.9.3.63 multiply() [3/4] | 237 |
| 8.9.3.64 multiply() [4/4] | 237 |
| 8.9.3.65 norm() | 238 |
| 8.9.3.66 normalize() | 239 |
| 8.9.3.67 operator*() [1/3] | 239 |
| 8.9.3.68 operator*() [2/3] | 240 |
| 8.9.3.69 operator*() [3/3] | 240 |
| 8.9.3.70 operator*=(()) [1/3] | 240 |
| 8.9.3.71 operator*=(()) [2/3] | 241 |
| 8.9.3.72 operator*=(()) [3/3] | 241 |
| 8.9.3.73 operator+() [1/3] | 242 |
| 8.9.3.74 operator+() [2/3] | 242 |
| 8.9.3.75 operator+() [3/3] | 242 |
| 8.9.3.76 operator+=(()) [1/3] | 243 |

| | |
|----------------------------------------|-----|
| 8.9.3.77 operator+=() [2/3] | 243 |
| 8.9.3.78 operator+=() [3/3] | 244 |
| 8.9.3.79 operator-() [1/3] | 244 |
| 8.9.3.80 operator-() [2/3] | 244 |
| 8.9.3.81 operator-() [3/3] | 245 |
| 8.9.3.82 operator-=(() [1/3] | 245 |
| 8.9.3.83 operator-=(() [2/3] | 246 |
| 8.9.3.84 operator-=(() [3/3] | 246 |
| 8.9.3.85 operator/() [1/3] | 247 |
| 8.9.3.86 operator/() [2/3] | 247 |
| 8.9.3.87 operator/() [3/3] | 247 |
| 8.9.3.88 operator/=(() [1/3] | 248 |
| 8.9.3.89 operator/=(() [2/3] | 248 |
| 8.9.3.90 operator/=(() [3/3] | 249 |
| 8.9.3.91 operator=() [1/2] | 249 |
| 8.9.3.92 operator=() [2/2] | 250 |
| 8.9.3.93 rotate() [1/3] | 250 |
| 8.9.3.94 rotate() [2/3] | 251 |
| 8.9.3.95 rotate() [3/3] | 251 |
| 8.9.3.96 rotateX() | 252 |
| 8.9.3.97 rotateY() | 253 |
| 8.9.3.98 rotateZ() | 253 |
| 8.9.3.99 slerp() [1/2] | 254 |
| 8.9.3.100 slerp() [2/2] | 254 |
| 8.9.3.101 subtract() [1/4] | 255 |
| 8.9.3.102 subtract() [2/4] | 255 |
| 8.9.3.103 subtract() [3/4] | 256 |
| 8.9.3.104 subtract() [4/4] | 257 |
| 8.10 gg::GgShader クラス | 258 |
| 8.10.1 詳解 | 258 |
| 8.10.2 構築子と解体子 | 258 |
| 8.10.2.1 GgShader() [1/2] | 258 |
| 8.10.2.2 ~GgShader() | 259 |
| 8.10.2.3 GgShader() [2/2] | 259 |
| 8.10.3 関数詳解 | 259 |
| 8.10.3.1 get() | 259 |
| 8.10.3.2 operator=() | 260 |
| 8.10.3.3 unuse() | 260 |
| 8.10.3.4 use() | 260 |
| 8.11 gg::GgShape クラス | 260 |
| 8.11.1 詳解 | 261 |
| 8.11.2 構築子と解体子 | 261 |

| | |
|------------------------------------------------|-----|
| 8.11.2.1 GgShape() [1/2] | 261 |
| 8.11.2.2 ~GgShape() | 262 |
| 8.11.2.3 GgShape() [2/2] | 262 |
| 8.11.3 関数詳解 | 262 |
| 8.11.3.1 draw() | 262 |
| 8.11.3.2 get() | 263 |
| 8.11.3.3 getMode() | 263 |
| 8.11.3.4 operator=() | 264 |
| 8.11.3.5 setMode() | 264 |
| 8.12 gg::GgSimpleObj クラス | 264 |
| 8.12.1 詳解 | 264 |
| 8.12.2 構築子と解体子 | 265 |
| 8.12.2.1 GgSimpleObj() | 265 |
| 8.12.2.2 ~GgSimpleObj() | 265 |
| 8.12.3 関数詳解 | 265 |
| 8.12.3.1 draw() | 265 |
| 8.12.3.2 get() | 266 |
| 8.13 gg::GgSimpleShader クラス | 266 |
| 8.13.1 詳解 | 269 |
| 8.13.2 構築子と解体子 | 269 |
| 8.13.2.1 GgSimpleShader() [1/3] | 269 |
| 8.13.2.2 GgSimpleShader() [2/3] | 269 |
| 8.13.2.3 GgSimpleShader() [3/3] | 270 |
| 8.13.2.4 ~GgSimpleShader() | 270 |
| 8.13.3 関数詳解 | 270 |
| 8.13.3.1 load() | 270 |
| 8.13.3.2 loadMatrix() [1/4] | 271 |
| 8.13.3.3 loadMatrix() [2/4] | 272 |
| 8.13.3.4 loadMatrix() [3/4] | 272 |
| 8.13.3.5 loadMatrix() [4/4] | 273 |
| 8.13.3.6 loadModelviewMatrix() [1/4] | 273 |
| 8.13.3.7 loadModelviewMatrix() [2/4] | 274 |
| 8.13.3.8 loadModelviewMatrix() [3/4] | 274 |
| 8.13.3.9 loadModelviewMatrix() [4/4] | 274 |
| 8.13.3.10 operator=() | 275 |
| 8.13.3.11 use() [1/13] | 275 |
| 8.13.3.12 use() [2/13] | 275 |
| 8.13.3.13 use() [3/13] | 276 |
| 8.13.3.14 use() [4/13] | 276 |
| 8.13.3.15 use() [5/13] | 277 |
| 8.13.3.16 use() [6/13] | 277 |
| 8.13.3.17 use() [7/13] | 278 |

| | |
|--------------------------------------|-----|
| 8.13.3.18 use() [8/13] | 278 |
| 8.13.3.19 use() [9/13] | 279 |
| 8.13.3.20 use() [10/13] | 279 |
| 8.13.3.21 use() [11/13] | 279 |
| 8.13.3.22 use() [12/13] | 280 |
| 8.13.3.23 use() [13/13] | 280 |
| 8.14 gg::GgTexture クラス | 281 |
| 8.14.1 詳解 | 281 |
| 8.14.2 構築子と解体子 | 281 |
| 8.14.2.1 GgTexture() [1/2] | 282 |
| 8.14.2.2 ~GgTexture() | 282 |
| 8.14.2.3 GgTexture() [2/2] | 282 |
| 8.14.3 関数詳解 | 282 |
| 8.14.3.1 bind() | 283 |
| 8.14.3.2 getHeight() | 283 |
| 8.14.3.3 getSize() [1/2] | 283 |
| 8.14.3.4 getSize() [2/2] | 283 |
| 8.14.3.5 getTexture() | 284 |
| 8.14.3.6 getWidth() | 284 |
| 8.14.3.7 operator=() | 285 |
| 8.14.3.8 unbind() | 285 |
| 8.15 gg::GgTrackball クラス | 286 |
| 8.15.1 詳解 | 287 |
| 8.15.2 構築子と解体子 | 287 |
| 8.15.2.1 GgTrackball() | 288 |
| 8.15.2.2 ~GgTrackball() | 288 |
| 8.15.3 関数詳解 | 288 |
| 8.15.3.1 begin() | 288 |
| 8.15.3.2 end() | 289 |
| 8.15.3.3 get() | 289 |
| 8.15.3.4 getMatrix() | 290 |
| 8.15.3.5 getQuaternion() | 290 |
| 8.15.3.6 getScale() [1/3] | 290 |
| 8.15.3.7 getScale() [2/3] | 290 |
| 8.15.3.8 getScale() [3/3] | 291 |
| 8.15.3.9 getStart() [1/3] | 291 |
| 8.15.3.10 getStart() [2/3] | 291 |
| 8.15.3.11 getStart() [3/3] | 292 |
| 8.15.3.12 motion() | 292 |
| 8.15.3.13 operator=() | 293 |
| 8.15.3.14 region() [1/2] | 293 |
| 8.15.3.15 region() [2/2] | 294 |

| | |
|-----------------------------------------|-----|
| 8.15.3.16 reset() | 294 |
| 8.15.3.17 rotate() | 295 |
| 8.16 gg::GgTriangles クラス | 295 |
| 8.16.1 詳解 | 297 |
| 8.16.2 構築子と解体子 | 297 |
| 8.16.2.1 GgTriangles() [1/2] | 297 |
| 8.16.2.2 GgTriangles() [2/2] | 297 |
| 8.16.2.3 ~GgTriangles() | 298 |
| 8.16.3 関数詳解 | 298 |
| 8.16.3.1 draw() | 298 |
| 8.16.3.2 getBuffer() | 299 |
| 8.16.3.3 getCount() | 299 |
| 8.16.3.4 load() | 299 |
| 8.16.3.5 send() | 300 |
| 8.17 gg::GgUniformBuffer< T > クラステンプレート | 300 |
| 8.17.1 詳解 | 301 |
| 8.17.2 構築子と解体子 | 301 |
| 8.17.2.1 GgUniformBuffer() [1/3] | 301 |
| 8.17.2.2 GgUniformBuffer() [2/3] | 301 |
| 8.17.2.3 GgUniformBuffer() [3/3] | 302 |
| 8.17.2.4 ~GgUniformBuffer() | 302 |
| 8.17.3 関数詳解 | 302 |
| 8.17.3.1 bind() | 303 |
| 8.17.3.2 copy() | 303 |
| 8.17.3.3 fill() | 303 |
| 8.17.3.4 getBuffer() | 304 |
| 8.17.3.5 getCount() | 304 |
| 8.17.3.6 getStride() | 304 |
| 8.17.3.7 getTarget() | 305 |
| 8.17.3.8 load() [1/2] | 305 |
| 8.17.3.9 load() [2/2] | 305 |
| 8.17.3.10 map() [1/2] | 306 |
| 8.17.3.11 map() [2/2] | 306 |
| 8.17.3.12 read() | 306 |
| 8.17.3.13 send() | 307 |
| 8.17.3.14 unbind() | 307 |
| 8.17.3.15 unmap() | 308 |
| 8.18 gg::GgVertex 構造体 | 308 |
| 8.18.1 詳解 | 308 |
| 8.18.2 構築子と解体子 | 308 |
| 8.18.2.1 GgVertex() [1/4] | 309 |
| 8.18.2.2 GgVertex() [2/4] | 309 |

| | |
|----------------------------------------------------|-----|
| 8.18.2.3 GgVertex() [3/4] | 309 |
| 8.18.2.4 GgVertex() [4/4] | 310 |
| 8.18.3 メンバ詳解 | 310 |
| 8.18.3.1 normal | 310 |
| 8.18.3.2 position | 310 |
| 8.19 gg::GgSimpleShader::Light 構造体 | 311 |
| 8.19.1 詳解 | 311 |
| 8.19.2 メンバ詳解 | 311 |
| 8.19.2.1 ambient | 311 |
| 8.19.2.2 diffuse | 311 |
| 8.19.2.3 position | 312 |
| 8.19.2.4 specular | 312 |
| 8.20 gg::GgSimpleShader::LightBuffer クラス | 312 |
| 8.20.1 詳解 | 314 |
| 8.20.2 構築子と解体子 | 314 |
| 8.20.2.1 LightBuffer() [1/2] | 314 |
| 8.20.2.2 LightBuffer() [2/2] | 314 |
| 8.20.2.3 ~LightBuffer() | 315 |
| 8.20.3 関数詳解 | 315 |
| 8.20.3.1 load() [1/2] | 315 |
| 8.20.3.2 load() [2/2] | 315 |
| 8.20.3.3 loadAmbient() [1/3] | 316 |
| 8.20.3.4 loadAmbient() [2/3] | 316 |
| 8.20.3.5 loadAmbient() [3/3] | 317 |
| 8.20.3.6 loadColor() | 317 |
| 8.20.3.7 loadDiffuse() [1/3] | 318 |
| 8.20.3.8 loadDiffuse() [2/3] | 318 |
| 8.20.3.9 loadDiffuse() [3/3] | 318 |
| 8.20.3.10 loadPosition() [1/4] | 319 |
| 8.20.3.11 loadPosition() [2/4] | 319 |
| 8.20.3.12 loadPosition() [3/4] | 320 |
| 8.20.3.13 loadPosition() [4/4] | 320 |
| 8.20.3.14 loadSpecular() [1/3] | 321 |
| 8.20.3.15 loadSpecular() [2/3] | 321 |
| 8.20.3.16 loadSpecular() [3/3] | 321 |
| 8.20.3.17 select() | 322 |
| 8.21 gg::GgSimpleShader::Material 構造体 | 322 |
| 8.21.1 詳解 | 323 |
| 8.21.2 メンバ詳解 | 323 |
| 8.21.2.1 ambient | 323 |
| 8.21.2.2 diffuse | 323 |
| 8.21.2.3 shininess | 323 |

| | |
|--------------------------------------------------------------------|-----|
| 8.21.2.4 <code>specular</code> | 323 |
| 8.22 <code>gg::GgSimpleShader::MaterialBuffer</code> クラス | 324 |
| 8.22.1 詳解 | 325 |
| 8.22.2 構築子と解体子 | 325 |
| 8.22.2.1 <code>MaterialBuffer()</code> [1/2] | 325 |
| 8.22.2.2 <code>MaterialBuffer()</code> [2/2] | 326 |
| 8.22.2.3 ~ <code>MaterialBuffer()</code> | 326 |
| 8.22.3 関数詳解 | 326 |
| 8.22.3.1 <code>load()</code> [1/2] | 326 |
| 8.22.3.2 <code>load()</code> [2/2] | 327 |
| 8.22.3.3 <code>loadAmbient()</code> [1/2] | 327 |
| 8.22.3.4 <code>loadAmbient()</code> [2/2] | 327 |
| 8.22.3.5 <code>loadAmbientAndDiffuse()</code> [1/2] | 328 |
| 8.22.3.6 <code>loadAmbientAndDiffuse()</code> [2/2] | 328 |
| 8.22.3.7 <code>loadDiffuse()</code> [1/2] | 329 |
| 8.22.3.8 <code>loadDiffuse()</code> [2/2] | 329 |
| 8.22.3.9 <code>loadShininess()</code> [1/2] | 330 |
| 8.22.3.10 <code>loadShininess()</code> [2/2] | 330 |
| 8.22.3.11 <code>loadSpecular()</code> [1/2] | 331 |
| 8.22.3.12 <code>loadSpecular()</code> [2/2] | 331 |
| 8.22.3.13 <code>select()</code> | 331 |
| 8.23 <code>Window</code> クラス | 332 |
| 8.23.1 詳解 | 334 |
| 8.23.2 構築子と解体子 | 335 |
| 8.23.2.1 <code>Window()</code> [1/2] | 335 |
| 8.23.2.2 <code>Window()</code> [2/2] | 335 |
| 8.23.2.3 ~ <code>Window()</code> | 336 |
| 8.23.3 関数詳解 | 336 |
| 8.23.3.1 <code>get()</code> | 336 |
| 8.23.3.2 <code>getAltArrowX()</code> | 336 |
| 8.23.3.3 <code>getAltArrowY()</code> | 336 |
| 8.23.3.4 <code>getAltArrow()</code> | 336 |
| 8.23.3.5 <code>getArrow()</code> [1/2] | 337 |
| 8.23.3.6 <code>getArrow()</code> [2/2] | 337 |
| 8.23.3.7 <code>getArrowX()</code> | 338 |
| 8.23.3.8 <code>getArrowY()</code> | 338 |
| 8.23.3.9 <code>getAspect()</code> | 338 |
| 8.23.3.10 <code>getControlArrow()</code> | 339 |
| 8.23.3.11 <code>getControlArrowX()</code> | 339 |
| 8.23.3.12 <code>getControlArrowY()</code> | 339 |
| 8.23.3.13 <code>getHeight()</code> | 340 |
| 8.23.3.14 <code>getKey()</code> | 340 |

| | |
|----------------------------------|------------|
| 8.23.3.15 getMouse() [1/3] | 340 |
| 8.23.3.16 getMouse() [2/3] | 340 |
| 8.23.3.17 getMouse() [3/3] | 341 |
| 8.23.3.18 getMouseX() | 341 |
| 8.23.3.19 getMouseY() | 341 |
| 8.23.3.20 getRotation() | 342 |
| 8.23.3.21 getRotationMatrix() | 342 |
| 8.23.3.22 getScrollMatrix() | 342 |
| 8.23.3.23 getShiftArrow() | 343 |
| 8.23.3.24 getShiftArrowX() | 343 |
| 8.23.3.25 getShiftArrowY() | 343 |
| 8.23.3.26 getSize() [1/2] | 344 |
| 8.23.3.27 getSize() [2/2] | 344 |
| 8.23.3.28 getTranslation() | 344 |
| 8.23.3.29 getTranslationMatrix() | 345 |
| 8.23.3.30 getUserPointer() | 345 |
| 8.23.3.31 getWheel() [1/3] | 345 |
| 8.23.3.32 getWheel() [2/3] | 345 |
| 8.23.3.33 getWheel() [3/3] | 346 |
| 8.23.3.34 getWheelX() | 346 |
| 8.23.3.35 getWheelY() | 346 |
| 8.23.3.36 getWidth() | 347 |
| 8.23.3.37 init() | 347 |
| 8.23.3.38 operator bool() | 347 |
| 8.23.3.39 operator=() | 348 |
| 8.23.3.40 reset() | 348 |
| 8.23.3.41 resetRotation() | 348 |
| 8.23.3.42 resetTranslation() | 348 |
| 8.23.3.43 restoreViewport() | 348 |
| 8.23.3.44 selectInterface() | 348 |
| 8.23.3.45 setClose() | 349 |
| 8.23.3.46 setKeyboardFunc() | 349 |
| 8.23.3.47 setMenubarHeight() | 349 |
| 8.23.3.48 setMouseFunc() | 350 |
| 8.23.3.49 setResizeFunc() | 350 |
| 8.23.3.50 setUserPointer() | 350 |
| 8.23.3.51 setVelocity() | 351 |
| 8.23.3.52 setWheelFunc() | 351 |
| 8.23.3.53 shouldClose() | 351 |
| 8.23.3.54 swapBuffers() | 352 |
| 9 ファイル詳解 | 353 |

| | |
|--------------------------------------|-----|
| 9.1 gg.cpp ファイル | 353 |
| 9.1.1 詳解 | 353 |
| 9.2 gg.h ファイル | 354 |
| 9.2.1 詳解 | 360 |
| 9.3 GgApp.h ファイル | 360 |
| 9.3.1 マクロ定義詳解 | 361 |
| 9.3.1.1 GG_USE_IMGUI | 361 |
| 9.4 ggsample01.cpp ファイル | 361 |
| 9.4.1 マクロ定義詳解 | 362 |
| 9.4.1.1 PROJECT_NAME | 362 |
| 9.5 main.cpp ファイル | 362 |
| 9.5.1 マクロ定義詳解 | 363 |
| 9.5.1.1 HEADER_STR | 363 |
| 9.5.2 関数詳解 | 363 |
| 9.5.2.1 main() | 364 |
| 9.6 README.md ファイル | 364 |
| 9.7 Window.h ファイル | 364 |
| 9.7.1 詳解 | 365 |
| 9.7.2 マクロ定義詳解 | 365 |
| 9.7.2.1 GG_BUTTON_COUNT | 366 |
| 9.7.2.2 GG_INTERFACE_COUNT | 366 |
| Index | 367 |

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

Copyright (c) 2011-2021 Kohe Tokoi. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies or substantial portions of the Software.

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL KOHE TOKOI BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Chapter 2

ggsample01

2.1 ゲームグラフィックス特論A 第1回 宿題

OpenGL の開発環境を整備してください。

- 宿題のひな形は [GitHub](#) にあります (宿題のひな形で使っている [補助プログラムの解説](#))。
- 詳しくは [講義のスライド](#) を参照してください。

2.2 宿題プログラムの作成に必要な環境

- Linux Mint 18.3 / Windows 10 (Visual Studio 2017) / macOS 10.15 (Xcode 11) 以降に対応しています。
- OpenGL 4.1 以降が実行できる環境 (対応した GPU を搭載したビデオカード や CPU) が必要です。
- macOS では M1 Mac (Apple Silicon) に対応した Universal Binary を作成するようにしています。

2.3 補足

このプログラムを実行すると、次のような図形が表示されます。

- 今回はソースプログラムを修正していないので送る必要はありません。
- ひな形プログラムがコンパイル／実行できなかつたら知らせてください。
- fork 推奨ですが解答をプレリクで受け取る気力はないと思います。

2.4 宿題プログラム用補助プログラムについて

ゲームグラフィックス特論 A / B で課す宿題プログラムでは、専用の補助プログラムを用意しています。これは以下の 3 つのファイルで構成されています。

- `gg.h` / `gg.cpp`
 - GLFW での利用を想定した OpenGL のローダとユーティリティ
- `Window.h`
 - ウィンドウやマウス関連のユーザインターフェースを管理する GLFW のラッパー

GLFW は OpenGL や、その後継の Vulkan を使用したアプリケーションを作成するための、非常にコンパクトなフレームワークです。本当はこれだけで簡単にアプリケーションが作れるのですが、授業内容とはあまり関係のない処理を分離するために、屋上屋ながら**この授業専用の**フレームワークを用意しました。なお、`gg.h` / `gg.cpp` には OpenGL の拡張機能を使用可能にする機能を含んでいますので、別に GLEW や glad、GL3Wなどを導入する必要はありません。

また、この `Window.h` には Dear ImGui をサポートする機能を含んでいます。このプログラム (ggsample01) には、その使い方のサンプルコードを示すために Dear ImGui のソースプログラムも含めています。日本語メニューの表示のために M+ FONTS の `Mplus1-Regular.ttf` もリポジトリに含めています。

このほか、この `Window.h` には Oculus Rift (DK1, DK2, CV1, S) をサポートする機能を組み込んでいます。これを使って、C++ だけで VR アプリケーション () が作れます。

2.4.1 補助プログラムのドキュメント

Doxygen で生成したドキュメントの [HTML 版](#) を `html` フォルダに、[PDF 版](#) を `pdf` フォルダに置いています。

2.4.2 補助プログラムの使い方

補助プログラムを使用するには、最小限、GLFW が使える環境が必要です。ゲームグラフィックス特論 A / B の宿題のリポジトリには Windows 用、macOS 用、および Linux 用にコンパイルしたライブラリファイル一式を含めていますので、宿題のために別に用意する必要はありません。`gg.h`, `gg.cpp`, `Window.h` だけを使うときは、それぞれの環境で GLFW をインストールしておいてください。この補助プログラムを使用した最小のプログラムは、多分こんな感じになります。このソースファイルと同じところに `gg.h`, `gg.cpp`, `Window.h` を置き、`gg.cpp` と一緒にコンパイルして、GLFW のライブラリファイルをリンクしてください。

```
#include "Window.h"
int main()
{
    Window::init();
    Window window;
    while (window)
    {
        // // ここで OpenGL による描画を行う
        //
    }
}
```

使用する OpenGL のバージョンは、`Window::init(major, minor)` の `major` と `minor` で指定できます。`major` を 0 にするか省略すると、OpenGL のバージョンの指定を行いません。その場合は、macOS 以外では恐らく OpenGL のハードウェアもしくはドライバで対応可能な最大のバージョンが使用されます。なお、3.2 以降を指定したときは macOS 対応の都合で `forward compatible` プロファイルと `core` プロファイルを有効にします。macOS の場合は `Window::init(3, 2)` もしくは `Window::init(4, 1)` を指定してください。

```
// for macOS
Window::init(4, 1);
```

2.4.3 Oculus Rift を使う場合

#include "Window.h" の前に #define USE_OOCULUS_RIFT を置いてください。DK1 / DK2 用か CV1 / S 用かは、使用する LibOVR のバージョンが 1.0 以前か以降かで判断しています。ただし DK1 / DK2 用 (LibOVR 0.8) のサポートは、今後は継続しない可能性があります。

```
// ウィンドウ関連の処理
#define USE_OOCULUS_RIFT
#include "Window.h"
```

あるいは、Window.h の中に #define USE_OOCULUS_RIFT を置いてください。

```
// Oculus Rift を使うなら
#define USE_OOCULUS_RIFT
```

実際の使い方は、「Oculus Rift に図形を表示するプログラムを C++ で作る」を参考にしてください。この記事では以前の補助プログラムを使って解説していますが、Window クラスの使い方は変わりません (以前の補助プログラムでは GgApplication クラス内に置いていました)。

2.4.4 Dear ImGui を使う場合

すべての #include "Window.h" の前に、#define USE_IMGUI を置いてください。

```
// ウィンドウ関連の処理
#define USE_IMGUI
#include "Window.h"
```

あるいは、Window.h の中に #define USE_IMGUI を置いてください。

```
// Dear ImGui を使うなら
#define USE_IMGUI
```

そして、OpenGL の描画ループの中で ImGui::NewFrame(); と ImGui::Render(); の間に Dear ImGui の API を置いてください。Dear ImGui のウィンドウの実際のレンダリング (ImGui_ImplOpenGL3_RenderDrawData(); の呼び出し) は window.swapbuffers() の内で行っているので、Dear ImGui の API と OpenGL の API は描画ループの中で混在していても構いません。

なお、Dear ImGui を有効にした場合は、Dear ImGui がマウスを使っているとき (io.WantCaptureMouse == true) に、Window クラスが保持しているマウスカーソルの位置を更新しないようにしています。また、Dear ImGui がキーボードを使っているとき (io.WantCaptureKeyboard == true) には、Window クラスはキーボードのイベントを処理しないようにしています。

```
// ウィンドウ関連の処理
#define USE_IMGUI
#include "Window.h"
int main()
{
    // ウィンドウ関連の初期設定
    Window::init(4, 1);
    // ウィンドウを作成する
    Window window("Window Title", 1280, 720);
    // ImGui の初期設定
    ImGui::StyleColorsDark();
    // 背景色を指定する
    glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
    // ウィンドウが開いている間繰り返す
    while (window)
    {
        // ImGui のフレームを準備する
        ImGui::NewFrame();
        // ImGui のフレームに一つ目の ImGui のウィンドウを描く
        ImGui::Begin("Control panel");
        ImGui::Text("Frame rate: %6.2f fps", ImGui::GetIO().Framerate);
        if (ImGui::Button("Quit")) window.setClose();
        ImGui::End();
        // ImGui のフレームに描画する
        ImGui::Render();
        // ウィンドウを消去する
        glClear(GL_COLOR_BUFFER_BIT);
        //
        // ここで OpenGL による描画を行う
        //
        // カラーバッファを入れ替えてイベントを取り出す
        window.swapBuffers();
    }
}
```

このソースファイルと Dear ImGui に含まれる以下のファイル、および [gg.h](#), [gg.cpp](#), [Window.h](#) を同じところに置き、[gg.cpp](#) と以下のうちの *.cpp ファイルと一緒にコンパイルして、GLFW のライブラリファイルをリンクしてください。

```
imconfig.h
imgui.h
imgui_impl_glfw.h
imgui_impl_opengl3.h
imgui_internal.h
imstb_rectpack.h
imstb_textedit.h
imstb_truetype.h
```

```
imgui.cpp
imgui_draw.cpp
imgui_impl_glfw.cpp
imgui_impl_opengl3.cpp
imgui_tables.cpp
imgui_widgets.cpp
```

2.4.4.1 imconfig.h の変更点

Dear ImGui は使用している OpenGL のローダを自動判別するのですが、この授業オリジナルの [gg.h](#) / [gg.cpp](#) は見つけてくれません。そこで、この中の imconfig.h の**最後**に、以下の定義を追加しています。

```
// My custom OpenGL loader
#define IMGUI_IMPL_OPENGL_LOADER_CUSTOM "../gg.h"
```

Chapter 3

名前空間索引

3.1 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 15

Chapter 4

階層索引

4.1 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。

| | |
|------------------------------------|-----|
| std::array | 105 |
| gg::GgMatrix | 105 |
| GgApp | 89 |
| gg::GgBuffer< T > | 90 |
| gg::GgColorTexture | 96 |
| gg::GgNormalTexture | 171 |
| gg::GgPointShader | 179 |
| gg::GgSimpleShader | 266 |
| gg::GgShader | 258 |
| gg::GgShape | 260 |
| gg::GgPoints | 175 |
| gg::GgTriangles | 295 |
| gg::GgElements | 100 |
| gg::GgSimpleObj | 264 |
| gg::GgTexture | 281 |
| gg::GgUniformBuffer< T > | 300 |
| gg::GgUniformBuffer< Light > | 300 |
| gg::GgSimpleShader::LightBuffer | 312 |
| gg::GgUniformBuffer< Material > | 300 |
| gg::GgSimpleShader::MaterialBuffer | 324 |
| GgVector | |
| gg::GgQuaternion | 188 |
| gg::GgTrackball | 286 |
| gg::GgVertex | 308 |
| gg::GgSimpleShader::Light | 311 |
| gg::GgSimpleShader::Material | 322 |
| Window | 332 |

Chapter 5

クラス索引

5.1 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。

| | |
|---------------------------------------------|-----|
| GgApp | 89 |
| <code>gg::GgBuffer< T ></code> | |
| バッファオブジェクト | 90 |
| <code>gg::GgColorTexture</code> | |
| カラーマップ | 96 |
| <code>gg::GgElements</code> | |
| 三角形で表した形状データ (Elements 形式) | 100 |
| <code>gg::GgMatrix</code> | |
| 変換行列 | 105 |
| <code>gg::GgNormalTexture</code> | |
| 法線マップ | 171 |
| <code>gg::GgPoints</code> | |
| 点 | 175 |
| <code>gg::GgPointShader</code> | |
| 点のシェーダ | 179 |
| <code>gg::GgQuaternion</code> | |
| 四元数 | 188 |
| <code>gg::GgShader</code> | |
| シェーダの基底クラス | 258 |
| <code>gg::GgShape</code> | |
| 形状データの基底クラス | 260 |
| <code>gg::GgSimpleObj</code> | |
| Wavefront OBJ 形式のファイル (Arrays 形式) | 264 |
| <code>gg::GgSimpleShader</code> | |
| 三角形に単純な陰影付けを行うシェーダ | 266 |
| <code>gg::GgTexture</code> | |
| テクスチャ | 281 |
| <code>gg::GgTrackball</code> | |
| 簡易トラックボール処理 | 286 |
| <code>gg::GgTriangles</code> | |
| 三角形で表した形状データ (Arrays 形式) | 295 |
| <code>gg::GgUniformBuffer< T ></code> | |
| ユニフォームバッファオブジェクト | 300 |
| <code>gg::GgVertex</code> | |
| 三角形の頂点データ | 308 |

| | | |
|----------------------------------------------------|---------------------------------------------------------|-----|
| 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 | 311 |
| 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 | 312 |
| 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 | 322 |
| 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 | 324 |
| Window | ウィンドウ関連の処理 | 332 |

Chapter 6

ファイル索引

6.1 ファイル一覧

ファイル一覧です。

| | | |
|--------------------------------|------------------------------------|-----|
| gg.cpp | ゲームグラフィックス特論の宿題用補助プログラム GLFW3 版の定義 | 353 |
| gg.h | ゲームグラフィックス特論の宿題用補助プログラム GLFW3 版の宣言 | 354 |
| GgApp.h | | 360 |
| ggsample01.cpp | | 361 |
| main.cpp | | 362 |
| Window.h | ゲームグラフィックス特論の宿題用補助プログラムのウィンドウ関連の処理 | 364 |

Chapter 7

名前空間詳解

7.1 gg 名前空間

ゲームグラフィックス特論の宿題用補助プログラムの名前空間

クラス

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

- class `GgPointShader`
点のシェーダ.
- class `GgSimpleShader`
三角形に単純な陰影付けを行うシェーダ.
- class `GgSimpleObj`
Wavefront OBJ 形式のファイル (*Arrays* 形式).

型定義

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

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 `std::string` &name="", unsigned int line=0)
OpenGL のエラーをチェックする.
- void `_ggFBOError` (const `std::string` &name="", unsigned int line=0)
FBO のエラーをチェックする.
- bool `ggSaveTga` (const `std::string` &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 `std::string` &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 `std::string` &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 `std::string` &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 `std::string` &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 `std::string` &name, `GLfloat` 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 `std::string` &vsr, const `std::string` &fsr="", const `std::string` &gsr="", `GLint` nvarying=0, const `char` *const *varyings=nullptr, const `std::string` &vtext="vertex shader", const `std::string` &ftext="fragment shader", const `std::string` >ext="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 std::string &vert, const std::string &frag="", const std::string &geom="", 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 std::string &csrc, const std::string &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 std::string &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.
- `void ggCross (GLfloat *c, const GLfloat *a, const GLfloat *b)`
3要素の外積.
- `GLfloat ggDot3 (const GLfloat *a, const GLfloat *b)`
3要素の内積.
- `GLfloat ggLength3 (const GLfloat *a)`
3要素の長さ.
- `void ggNormalize3 (GLfloat *a)`
3要素の正規化.
- `GLfloat ggDot4 (const GLfloat *a, const GLfloat *b)`
4要素の内積
- `GLfloat ggLength4 (const GLfloat *a)`
4要素の長さ.
- `void ggNormalize4 (GLfloat *a)`
4要素の正規化.
- `GgVector operator+ (const GgVector &a, const GgVector &b)`
*GgVector*型の和を返す.
- `GgVector & operator+= (GgVector &a, const GgVector &b)`
*GgVector*型を加算する.
- `GgVector operator- (const GgVector &a, const GgVector &b)`
*GgVector*型の差を返す.
- `GgVector & operator-= (GgVector &a, const GgVector &b)`
*GgVector*型を減算する.
- `GgVector operator* (const GgVector &a, const GgVector &b)`
*GgVector*型の積を返す.
- `GgVector & operator*= (GgVector &a, const GgVector &b)`
*GgVector*型を乗算する.
- `GgVector operator/ (const GgVector &a, const GgVector &b)`
*GgVector*型の商を返す.
- `GgVector & operator/= (GgVector &a, const GgVector &b)`
*GgVector*型を除算する.
- `GgVector operator+ (const GgVector &a, GLfloat b)`
*GgVector*型の各要素にスカラーを足した和を返す.
- `GgVector operator+ (GLfloat a, const GgVector &b)`
*GgVector*型の各要素にスカラーを足した和を返す.
- `GgVector & operator+= (GgVector &a, GLfloat b)`
*GgVector*型の各要素にスカラーを足す.
- `GgVector operator- (const GgVector &a, GLfloat b)`
*GgVector*型の各要素からスカラーを引いた差を返す.
- `GgVector operator- (GLfloat a, const GgVector &b)`
スカラーから*GgVector*型の各要素を引いた差を返す.
- `GgVector & operator-= (GgVector &a, GLfloat b)`
*GgVector*型の各要素からスカラーを引く.
- `GgVector operator* (const GgVector &a, GLfloat b)`

- **GgVector operator*** (GLfloat a, const **GgVector** &b)

 - GgVector* 型の各要素にスカラーを乗じた積を返す.

- **GgVector & operator*= **(GgVector &a, GLfloat b)****

 - GgVector* 型の各要素にスカラーを乗じる.

- **GgVector operator/ (const GgVector &a, GLfloat b)**

 - GgVector* 型の各要素をスカラーで割った商を返す.

- **GgVector operator/ (GLfloat a, const GgVector &b)**

 - スカラーを *GgVector* 型の各要素で割った商を返す.

- **GgVector & operator/= (GgVector &a, GLfloat b)**

 - GgVector* 型の各要素をスカラーで割る.

- **GLfloat ggDot3 (const GgVector &a, const GgVector &b)**

 - GgVector* 型の 3 要素の内積.

- **GgVector ggCross (const GgVector &a, const GgVector &b)**

 - GgVector* 型の 3 要素の外積.

- **GLfloat ggLength3 (const GgVector &a)**

 - GgVector* 型の 3 要素の長さ.

- **GLfloat ggDistance3 (const GgVector &a, const GgVector &b)**

 - GgVector* 型の 3 要素の距離.

- **bool ggNormalize3 (GgVector *a)**

 - GgVector* 型の 3 要素の正規化.

- **GgVector ggNormalize3 (const GgVector &a)**

 - GgVector* 型の 3 要素の正規化.

- **GLfloat ggDot4 (const GgVector &a, const GgVector &b)**

 - GgVector* の 4 要素の内積.

- **GLfloat ggLength4 (const GgVector &a)**

 - GgVector* 型の長さ.

- **GLfloat ggDistance4 (const GgVector &a, const GgVector &b)**

 - GgVector* 型の距離.

- **bool ggNormalize4 (GgVector *a)**

 - GgVector* 型の正規化.

- **GgVector ggNormalize4 (const GgVector &a)**

 - GgVector* 型の正規化.

- **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 std::string &name, bool normalize=false)
Wavefront OBJ ファイルを読み込む (*Arrays* 形式)
- `GgElements * ggElementsObj` (const std::string &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 std::string &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 std::string &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.

7.1.1 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

7.1.2 型定義詳解

7.1.2.1 GgVector

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

4要素の单精度実数の配列。

gg.h の 1341 行目に定義があります。

7.1.3 列挙型詳解

7.1.3.1 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 1327 行目に定義があります。

7.1.4 関数詳解

7.1.4.1 ggError()

```
void gg::ggError (
    const std::string & name = "",
    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

| | |
|-------------|---------------------------------------------------------|
| <i>name</i> | エラー発生時に標準エラー出力に出力する文字列, <code>nullptr</code> なら何も出力しない. |
| <i>line</i> | 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 2564 行目に定義がります。

7.1.4.2 `ggFBOError()`

```
void gg::ggFBOError (
    const std::string & name = "",
    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

| | |
|-------------|---------------------------------------------------------|
| <i>name</i> | エラー発生時に標準エラー出力に出力する文字列, <code>nullptr</code> なら何も出力しない. |
| <i>line</i> | 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 2608 行目に定義がります。

7.1.4.3 `ggArraysObj()`

```
gg::GgTriangles * gg::ggArraysObj (
    const std::string & name,
    bool normalize = false )
```

Wavefront OBJ ファイルを読み込む (Arrays 形式)

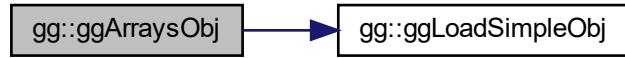
三角形分割された Wavefront OBJ ファイルを読み込んで GgArrays 形式の三角形データを生成する.

引数

| | |
|------------------|------------------------------|
| <i>name</i> | ファイル名. |
| <i>normalize</i> | <code>true</code> なら大きさを正規化. |

gg.cpp の 5196 行目に定義がります。

呼び出し関係図:



7.1.4.4 ggConjugate()

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

共役四元数を返す。

引数

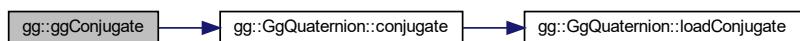
| | |
|----------|---------------------|
| <i>q</i> | GgQuaternion 型の四元数。 |
|----------|---------------------|

戻り値

四元数 *q* の共役四元数。

gg.h の 4207 行目に定義があります。

呼び出し関係図:



7.1.4.5 ggCreateComputeShader()

```
GLuint gg::ggCreateComputeShader (
    const std::string & csrc,
    const std::string & 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

| | |
|--------------|----------------------------------|
| <i>csrc</i> | コンピュートシェーダのソースプログラムの文字列. |
| <i>ctext</i> | 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 4201 行目に定義があります。

7.1.4.6 ggCreateNormalMap()

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

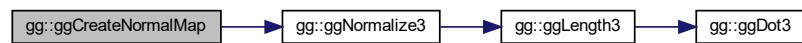
グレースケール画像(8bit)から法線マップのデータを作成する.

引数

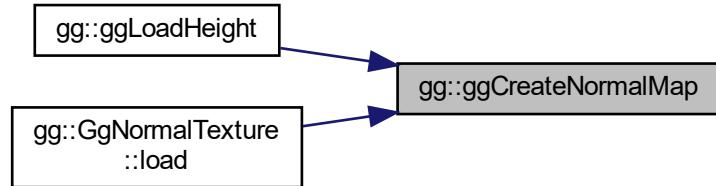
| | |
|-----------------|----------------------------------------------------------|
| <i>hmap</i> | グレースケール画像のデータ. |
| <i>width</i> | 高さマップのグレースケール画像 <i>hmap</i> の横の画素数. |
| <i>height</i> | 高さマップのグレースケール画像 <i>hmap</i> の縦の画素数. |
| <i>format</i> | データの書式(GL_RED, GL_RG, GL_RGB, GL_BGR, GL_RGBA, GL_BGRA).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

gg.cpp の 3023 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:



7.1.4.7 ggCreateShader()

```

GLuint gg::ggCreateShader (
    const std::string & vsrc,
    const std::string & fsrc = "",
    const std::string & gsrc = "",
    GLint nvarying = 0,
    const char *const * varyings = nullptr,
    const std::string & vtext = "vertex shader",
    const std::string & ftext = "fragment shader",
    const std::string & 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

| | |
|-----------------|----------------------------------------------------------------|
| <i>vsrc</i> | バーテックスシェーダのソースプログラムの文字列。 |
| <i>fsrc</i> | フラグメントシェーダのソースプログラムの文字列 (<code>nullptr</code> なら不使用)。 |
| <i>gsrc</i> | ジオメトリシェーダのソースプログラムの文字列 (<code>nullptr</code> なら不使用)。 |
| <i>nvarying</i> | フィードバックする <i>varying</i> 変数の数 (0 なら不使用)。 |
| <i>varyings</i> | フィードバックする <i>varying</i> 変数のリスト (<code>nullptr</code> なら不使用)。 |
| <i>vtext</i> | 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。 |
| <i>ftext</i> | 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。 |
| <i>gtext</i> | 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 4030 行目に定義があります。

7.1.4.8 ggCross() [1/2]

```
GgVector gg::ggCross (
    const GgVector & a,
    const GgVector & b ) [inline]
```

GgVector 型の 3 要素の外積.

引数

| | |
|----------|----------------|
| <i>a</i> | GgVector 型の変数. |
| <i>b</i> | GgVector 型の変数. |

戻り値

a と *b* の外積.

覚え書き

戻り値の *w* (第4) 要素は 0.

gg.h の 1981 行目に定義があります。

7.1.4.9 ggCross() [2/2]

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

3 要素の外積.

引数

| | |
|----------|-------------------------------|
| <i>a</i> | GLfloat 型の 3 要素の配列変数. |
| <i>b</i> | GLfloat 型の 3 要素の配列変数. |
| <i>c</i> | 結果を格納する GLfloat 型の 3 要素の配列変数. |

gg.h の 1599 行目に定義があります。

7.1.4.10 ggDistance3()

```
GLfloat gg::ggDistance3 (
    const GgVector & a,
    const GgVector & b ) [inline]
```

GgVector 型の 3 要素の距離.

引数

| | |
|----------|----------------|
| <i>a</i> | GgVector 型の変数. |
| <i>b</i> | GgVector 型の変数. |

戻り値

2 つの GgVector *a*, *b* の 3 要素の距離.

gg.h の 2012 行目に定義があります。

呼び出し関係図:



7.1.4.11 ggDistance4()

```
GLfloat gg::ggDistance4 (
    const GgVector & a,
    const GgVector & b ) [inline]
```

GgVector 型の距離.

引数

| | |
|----------|----------------|
| <i>a</i> | GgVector 型の変数. |
| <i>b</i> | GgVector 型の変数. |

戻り値

2 つの GgVector *a*, *b* の距離.

gg.h の 2085 行目に定義があります。

呼び出し関係図:



7.1.4.12 ggDot3() [1/2]

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

GgVector 型の 3 要素の内積.

引数

| | |
|----------|----------------|
| <i>a</i> | GgVector 型の変数. |
| <i>b</i> | GgVector 型の変数. |

戻り値

a と *b* のそれぞれの 3 要素の内積.

gg.h の 1968 行目に定義があります。

7.1.4.13 ggDot3() [2/2]

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

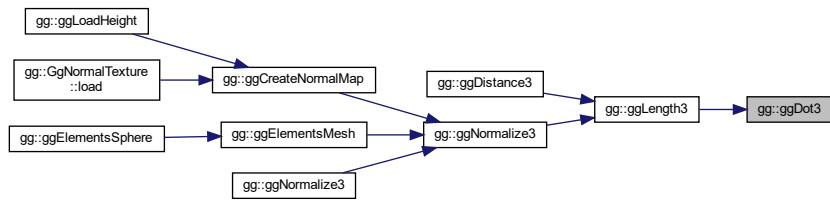
3 要素の内積.

引数

| | |
|----------|-----------------------|
| <i>a</i> | GLfloat 型の 3 要素の配列変数. |
| <i>b</i> | GLfloat 型の 3 要素の配列変数. |

gg.h の 1612 行目に定義があります。

被呼び出し関係図:



7.1.4.14 ggDot4() [1/2]

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

`GgVector` の 4 要素の内積.

引数

| | |
|----------|-----------------------------|
| <i>a</i> | <code>GgVector</code> 型の変数. |
| <i>b</i> | <code>GgVector</code> 型の変数. |

戻り値

`a` と `b` のそれぞれの 4 要素の内積.

`gg.h` の 2062 行目に定義があります。

7.1.4.15 ggDot4() [2/2]

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

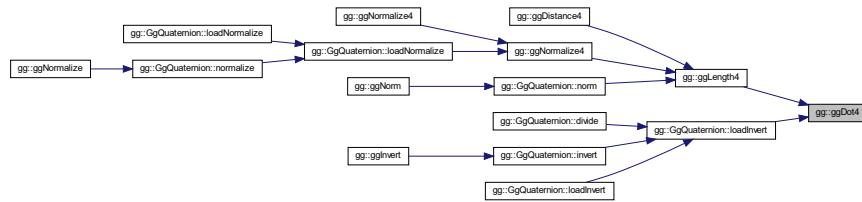
4 要素の内積

引数

| | |
|----------|------------------------------------|
| <i>a</i> | <code>GLfloat</code> 型の 4 要素の配列変数. |
| <i>b</i> | <code>GLfloat</code> 型の 4 要素の配列変数. |

gg.h の 1649 行目に定義がります。

被呼び出し関係図:



7.1.4.16 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).

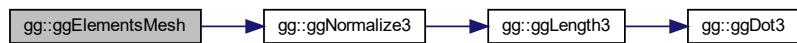
メッシュ状に GgElements 形式の三角形データを生成する.

引数

| | |
|---------------|------------------------------------|
| <i>slices</i> | メッシュの横方向の分割数. |
| <i>stacks</i> | メッシュの縦方向の分割数. |
| <i>pos</i> | メッシュの頂点の位置. |
| <i>norm</i> | メッシュの頂点の法線, nullptr なら頂点の位置から算出する. |

gg.cpp の 5230 行目に定義がります。

呼び出し関係図:



呼び出し関係図:



7.1.4.17 ggElementsObj()

```
gg::GgElements * gg::ggElementsObj (
    const std::string & 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

| | |
|------------------|-----------------|
| <i>name</i> | ファイル名. |
| <i>normalize</i> | true なら大きさを正規化. |

gg.cpp の 5212 行目に定義があります。

呼び出し関係図:



7.1.4.18 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

| | |
|---------------|-------------|
| <i>radius</i> | 球の半径. |
| <i>slices</i> | 球の経度方向の分割数. |
| <i>stacks</i> | 球の緯度方向の分割数. |

gg.cpp の 5326 行目に定義があります。

呼び出し関係図:



7.1.4.19 ggEllipse()

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

橢円状に三角形を生成する。

引数

| | |
|---------------|---------|
| <i>width</i> | 橢円の横幅. |
| <i>height</i> | 橤円の高さ. |
| <i>slices</i> | 橤円の分割数. |

gg.cpp の 5170 行目に定義があります。

7.1.4.20 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

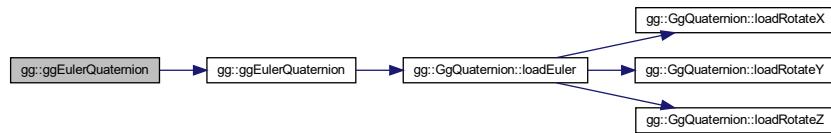
| | |
|----------|-------------------------------------------------------|
| <i>e</i> | オイラー角を表す GLfloat 型の 3 要素の配列変数 (heading, pitch, roll). |
|----------|-------------------------------------------------------|

戻り値

回転を表す四元数。

gg.h の 4142 行目に定義があります。

呼び出し関係図:



7.1.4.21 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

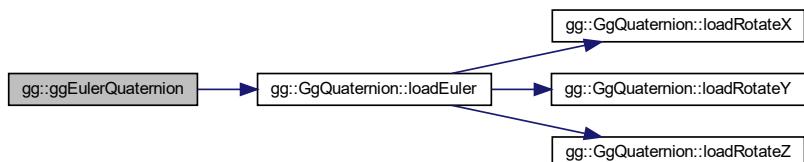
| | |
|----------------|------------|
| <i>heading</i> | y 軸中心の回転角。 |
| <i>pitch</i> | x 軸中心の回転角。 |
| <i>roll</i> | z 軸中心の回転角。 |

戻り値

回転を表す四元数。

gg.h の 4133 行目に定義があります。

呼び出し関係図:



呼び出し関係図:



7.1.4.22 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

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

戻り値

求めた透視透視投影変換行列.

gg.h の 3131 行目に定義があります。

呼び出し関係図:



7.1.4.23 `ggIdentity()`

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

単位行列を返す。

戻り値

単位行列。

gg.h の 2913 行目に定義があります。

呼び出し関係図:



7.1.4.24 `ggIdentityQuaternion()`

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

単位四元数を返す。

戻り値

単位四元数。

gg.h の 4054 行目に定義があります。

呼び出し関係図:



7.1.4.25 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1309行目に定義があります。

呼び出し関係図:



7.1.4.26 ggInvert() [1/2]

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

逆行列を返す。

引数

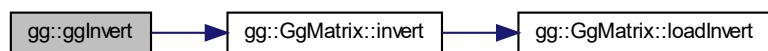
| | |
|----------|---------|
| <i>m</i> | 元の変換行列。 |
|----------|---------|

戻り値

m の逆行列。

gg.hの3167行目に定義があります。

呼び出し関係図:



7.1.4.27 ggInvert() [2/2]

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

四元数の逆元を求める。

引数

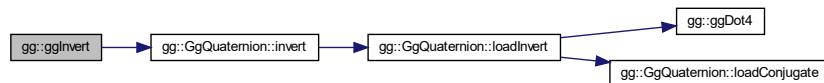
| | |
|----------|---------------------|
| <i>q</i> | GgQuaternion 型の四元数。 |
|----------|---------------------|

戻り値

四元数 *q* の逆元。

gg.h の 4215 行目に定義があります。

呼び出し関係図:



7.1.4.28 ggLength3() [1/2]

```
GLfloat gg::ggLength3 (
    const GgVector & a )  [inline]
```

GgVector 型の 3 要素の長さ。

引数

| | |
|----------|----------------|
| <i>a</i> | GgVector 型の変数。 |
|----------|----------------|

戻り値

a の長さ。

gg.h の 2000 行目に定義があります。

呼び出し関係図:



7.1.4.29 ggLength3() [2/2]

```
GLfloat gg::ggLength3 (
    const GLfloat * a ) [inline]
```

3要素の長さ.

引数

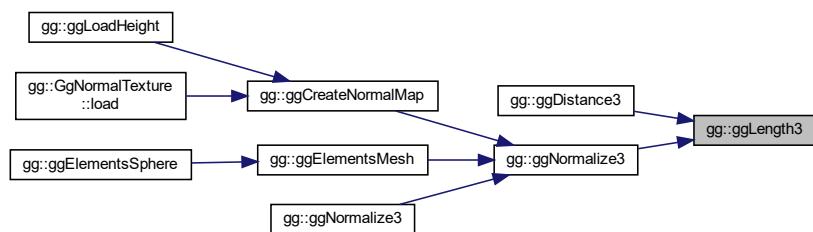
| | |
|---|-----------------------|
| a | GLfloat 型の 3 要素の配列変数. |
|---|-----------------------|

gg.h の 1622 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:



7.1.4.30 `ggLength4()` [1/2]

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

`GgVector` 型の長さ.

引数

| | |
|----------------|-----------------------------|
| <code>a</code> | <code>GgVector</code> 型の変数. |
|----------------|-----------------------------|

戻り値

`a` の長さ.

`gg.h` の 2073 行目に定義があります。

呼び出し関係図:



7.1.4.31 `ggLength4()` [2/2]

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

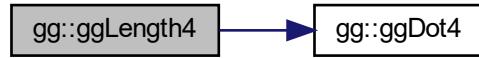
4 要素の長さ.

引数

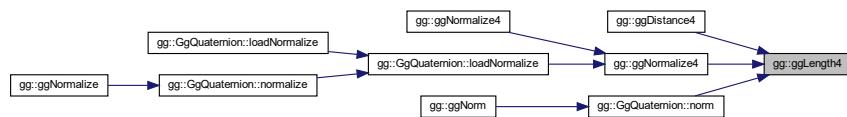
| | |
|----------------|------------------------------------|
| <code>a</code> | <code>GLfloat</code> 型の 4 要素の配列変数. |
|----------------|------------------------------------|

`gg.h` の 1659 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:



7.1.4.32 ggLoadComputeShader()

```
GLuint gg::ggLoadComputeShader (
    const std::string & 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

| | |
|-------------|----------------------|
| <i>comp</i> | 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 4242 行目に定義があります。

7.1.4.33 ggLoadHeight()

```
GLuint gg::ggLoadHeight (
    const std::string & name,
    GLfloat 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

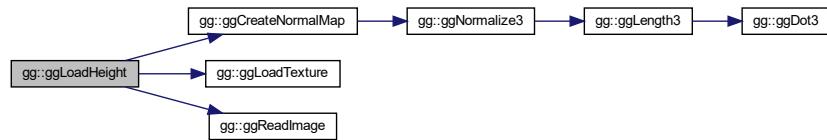
| | |
|-----------------|--------------------------------------------------|
| <i>name</i> | 読み込むファイル名. |
| <i>nz</i> | 法線の z 成分の割合. |
| <i>pWidth</i> | 読みだした画像ファイルの横の画素数の格納先のポインタ (nullptr なら格納しない). ++ |
| <i>pHeight</i> | 読みだした画像ファイルの縦の画素数の格納先のポインタ (nullptr なら格納しない). |
| <i>internal</i> | glTexImage2D() 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 3110 行目に定義があります。

呼び出し関係図:

7.1.4.34 **ggLoadImage()**

```
GLuint gg::ggLoadImage (
    const std::string & 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

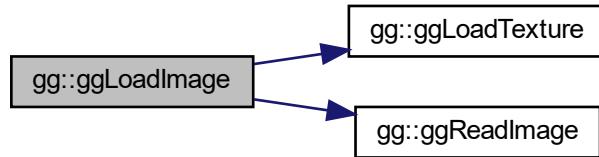
| | |
|-----------------|--------------------------------------------------------|
| <i>name</i> | 読み込むファイル名. |
| <i>pWidth</i> | 読みだした画像ファイルの横の画素数の格納先のポインタ (nullptr なら格納しない). ++ |
| <i>pHeight</i> | 読みだした画像ファイルの縦の画素数の格納先のポインタ (nullptr なら格納しない). |
| <i>internal</i> | glTexImage2D() に指定するテクスチャの内部フォーマット, 0 なら外部フォーマットに合わせる. |
| <i>wrap</i> | 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 2961 行目に定義があります。

呼び出し関係図:



7.1.4.35 ggLoadShader()

```

GLuint gg::ggLoadShader (
    const std::string & vert,
    const std::string & frag = "",
    const std::string & geom = "",
    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

| | |
|-----------------|--------------------------------------------------|
| <i>vert</i> | バーテックスシェーダのソースファイル名. |
| <i>frag</i> | 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). |
| <i>geom</i> | ジオメトリシェーダのソースファイル名 (nullptr なら不使用). |
| <i>nvarying</i> | フィードバックする <i>varying</i> 変数の数 (0 なら不使用). |
| <i>varyings</i> | フィードバックする <i>varying</i> 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 4168 行目に定義があります。

7.1.4.36 `ggLoadSimpleObj()` [1/2]

```
bool gg::ggLoadSimpleObj (
    const std::string & 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

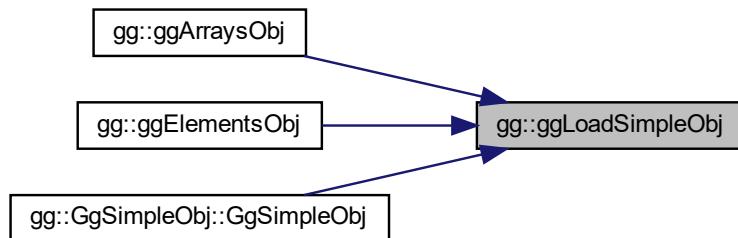
| | |
|------------------------|------------------------------------------------------------------|
| <code>name</code> | 読み込む Wavefront OBJ ファイル名. |
| <code>group</code> | 読み込んだデータのポリゴングループごとの最初の三角形の番号と三角形数・材質番号. |
| <code>material</code> | 読み込んだデータのポリゴングループごとの <code>GgSimpleShader::Material</code> 型の材質. |
| <code>vert</code> | 読み込んだデータの頂点属性. |
| <code>normalize</code> | <code>true</code> 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 3782 行目に定義があります。

被呼び出し関係図:



7.1.4.37 `ggLoadSimpleObj()` [2/2]

```
bool gg::ggLoadSimpleObj (
    const std::string & 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

| | |
|------------------|------------------------------------------|
| <i>name</i> | 読み込む Wavefront OBJ ファイル名. |
| <i>group</i> | 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. |
| <i>material</i> | 読み込んだデータのポリゴングループごとの材質. |
| <i>vert</i> | 読み込んだデータの頂点属性. |
| <i>face</i> | 読み込んだデータの三角形の頂点インデックス. |
| <i>normalize</i> | 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 3871 行目に定義があります。

7.1.4.38 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

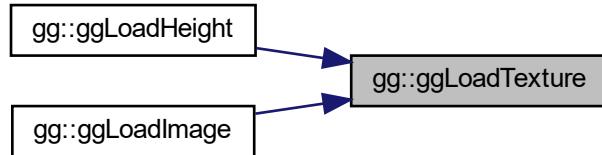
| | |
|-----------------|----------------------------------------------|
| <i>image</i> | テクスチャとして読み込むデータ, nullptr ならテクスチャメモリの確保のみを行う. |
| <i>width</i> | テクスチャとして読み込むデータ <i>image</i> の横の画素数. |
| <i>height</i> | テクスチャとして読み込むデータ <i>image</i> の縦の画素数. |
| <i>format</i> | <i>image</i> のフォーマット. |
| <i>type</i> | <i>image</i> のデータ型. |
| <i>internal</i> | テクスチャの内部フォーマット. |
| <i>wrap</i> | 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 2921 行目に定義があります。

呼び出し関係図:



7.1.4.39 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

| | |
|----------|------------------------------|
| <i>e</i> | 視点の位置を格納した GgVector 型の変数. |
| <i>t</i> | 目標点の位置を格納した GgVector 型の変数. |
| <i>u</i> | 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 3097 行目に定義があります。

呼び出し関係図:



7.1.4.40 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

| | |
|----------|--------------------------------------------------|
| <i>e</i> | 視点の位置を格納した <code>GLfloat</code> 型の 3 要素の配列変数。 |
| <i>t</i> | 目標点の位置を格納した <code>GLfloat</code> 型の 3 要素の配列変数。 |
| <i>u</i> | 上方向のベクトルを格納した <code>GLfloat</code> 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 3082 行目に定義があります。

呼び出し関係図:



7.1.4.41 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

| | |
|-----------------|-----------------|
| <code>ex</code> | 視点の位置の x 座標値. |
| <code>ey</code> | 視点の位置の y 座標値. |
| <code>ez</code> | 視点の位置の z 座標値. |
| <code>tx</code> | 目標点の位置の x 座標値. |
| <code>ty</code> | 目標点の位置の y 座標値. |
| <code>tz</code> | 目標点の位置の z 座標値. |
| <code>ux</code> | 上方向のベクトルの x 成分. |
| <code>uy</code> | 上方向のベクトルの y 成分. |
| <code>uz</code> | 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 3067 行目に定義があります。

呼び出し関係図:



7.1.4.42 `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

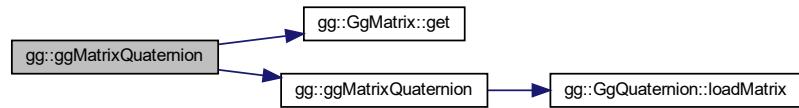
| | |
|----------------|-------------------------------|
| <code>m</code> | <code>GgMatrix</code> 型の変換行列. |
|----------------|-------------------------------|

戻り値

`m` による回転の変換に相当する四元数.

`gg.h` の 4072 行目に定義があります。

呼び出し関係図:



7.1.4.43 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

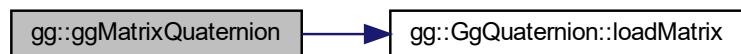
| | |
|----------------|------------------------|
| <code>a</code> | GLfloat 型の 16 要素の配列変数. |
|----------------|------------------------|

戻り値

`a` による回転の変換に相当する四元数.

`gg.h` の 4063 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:



7.1.4.44 ggNorm()

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

四元数のノルムを返す。

引数

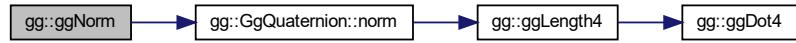
| | |
|----------|---------------------|
| <i>q</i> | GgQuaternion 型の四元数。 |
|----------|---------------------|

戻り値

四元数 *q* のノルム。

gg.h の 4191 行目に定義があります。

呼び出し関係図:



7.1.4.45 ggNormal()

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

法線変換行列を返す。

引数

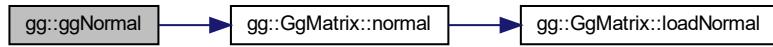
| | |
|----------|---------|
| <i>m</i> | 元の変換行列。 |
|----------|---------|

戻り値

m の法線変換行列。

gg.h の 3175 行目に定義があります。

呼び出し関係図:



7.1.4.46 ggNormalize()

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

正規化した四元数を返す.

引数

| | |
|----------|---------------------|
| <i>q</i> | 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 4199 行目に定義があります。

呼び出し関係図:



7.1.4.47 ggNormalize3() [1/3]

```
GgVector gg::ggNormalize3 (
    const GgVector & a ) [inline]
```

GgVector 型の 3 要素の正規化.

引数

| | |
|----------|----------------|
| <i>a</i> | GgVector 型の変数. |
|----------|----------------|

戻り値

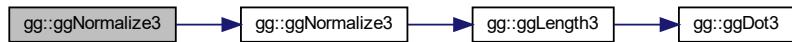
`a` の 3 要素を正規化したもの.

覚え書き

戻り値の `w` (第4) 要素は 0 になる. 正規化できなければ元の値を返す.

`gg.h` の 2048 行目に定義があります。

呼び出し関係図:



7.1.4.48 `ggNormalize3()` [2/3]

```
bool gg::ggNormalize3 (
    GgVector * a ) [inline]
```

`GgVector` 型の 3 要素の正規化.

引数

| | |
|----------------|----------------------------------|
| <code>a</code> | <code>GgVector</code> 型の変数のポインタ. |
|----------------|----------------------------------|

戻り値

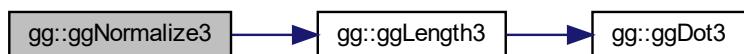
正規化できなかったら `false`.

覚え書き

`a` の `w` (第4) 要素は 0 になる.

`gg.h` の 2024 行目に定義があります。

呼び出し関係図:



7.1.4.49 ggNormalize3() [3/3]

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

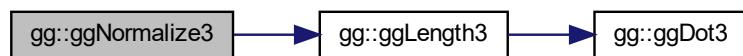
3要素の正規化。

引数

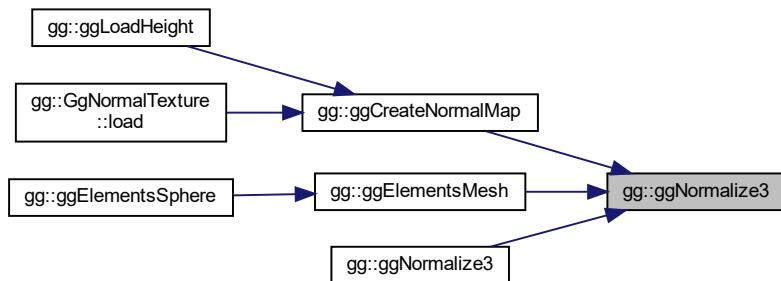
| | |
|---|-----------------------|
| a | GLfloat 型の 3 要素の配列変数。 |
|---|-----------------------|

gg.h の 1632 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:



7.1.4.50 ggNormalize4() [1/3]

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

GgVector 型の正規化。

引数

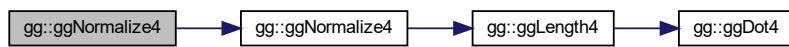
| | |
|---|----------------|
| a | GgVector 型の変数. |
|---|----------------|

戻り値

a の 4 要素を正規化したもの.

gg.h の 2119 行目に定義があります。

呼び出し関係図:



7.1.4.51 ggNormalize4() [2/3]

```
bool gg::ggNormalize4 (
    GgVector * a ) [inline]
```

GgVector 型の正規化.

引数

| | |
|---|---------------------|
| a | GgVector 型の変数のポインタ. |
|---|---------------------|

戻り値

正規化できなかったら false.

gg.h の 2096 行目に定義があります。

呼び出し関係図:



7.1.4.52 ggNormalize4() [3/3]

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

4 要素の正規化.

引数

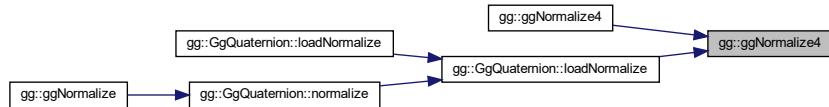
| | |
|---|-----------------------|
| a | GLfloat 型の 4 要素の配列変数. |
|---|-----------------------|

gg.h の 1669 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:



7.1.4.53 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. |
|------|--------------|

引数

| | |
|---------------|---------------|
| <i>right</i> | ウィンドウの右端の位置. |
| <i>bottom</i> | ウィンドウの下端の位置. |
| <i>top</i> | ウィンドウの上端の位置. |
| <i>zNear</i> | 視点から前方面までの位置. |
| <i>zFar</i> | 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 3115 行目に定義があります。

呼び出し関係図:



7.1.4.54 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

| | |
|---------------|---------------|
| <i>fovy</i> | y 方向の画角. |
| <i>aspect</i> | 縦横比. |
| <i>zNear</i> | 視点から前方面までの位置. |
| <i>zFar</i> | 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 3147 行目に定義がります。

呼び出し関係図:



7.1.4.55 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

| | |
|---------------|-------------------|
| <i>countv</i> | 生成する点の数。 |
| <i>length</i> | 点群を生成する立方体の一辺の長さ。 |
| <i>cx</i> | 点群の中心の x 座標。 |
| <i>cy</i> | 点群の中心の y 座標。 |
| <i>cz</i> | 点群の中心の z 座標。 |

gg.cpp の 5088 行目に定義がります。

7.1.4.56 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

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

gg.cpp の 5118 行目に定義があります。

7.1.4.57 ggQuaternion() [1/2]

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

四元数を返す.

引数

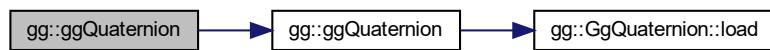
| | |
|----------|------------------------------------------|
| <i>a</i> | GLfloat 型の GLfloat 型の 4 要素の配列変数に格納した四元数. |
|----------|------------------------------------------|

戻り値

四元数.

gg.h の 4047 行目に定義があります。

呼び出し関係図:



7.1.4.58 ggQuaternion() [2/2]

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

四元数を返す.

引数

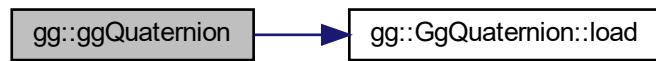
| | |
|----------|-------------------|
| <i>x</i> | 四元数の <i>x</i> 要素. |
| <i>y</i> | 四元数の <i>y</i> 要素. |
| <i>z</i> | 四元数の <i>z</i> 要素. |
| <i>w</i> | 四元数の <i>w</i> 要素. |

戻り値

四元数.

gg.h の 4038 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:



7.1.4.59 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

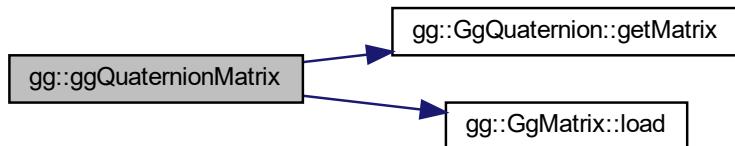
| | |
|----------|--------|
| <i>q</i> | 元の四元数. |
|----------|--------|

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 4080 行目に定義があります。

呼び出し関係図:



7.1.4.60 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

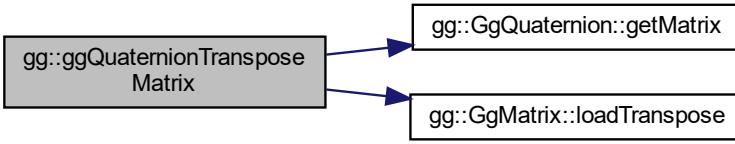
| | |
|-----|--------|
| q | 元の四元数. |
|-----|--------|

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 4091 行目に定義があります。

呼び出し関係図:



7.1.4.61 **ggReadImage()**

```
bool gg::ggReadImage (
    const std::string & 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

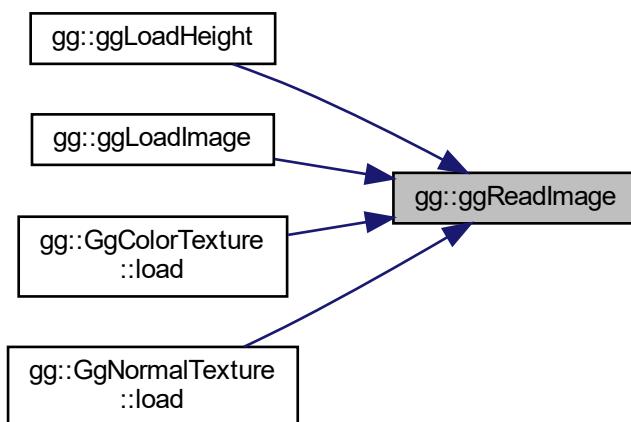
| | |
|----------------|--------------------------------------------------------------------------------------|
| <i>name</i> | 読み込むファイル名。 |
| <i>image</i> | 読み込んだデータを格納する vector。 |
| <i>pWidth</i> | 読み込んだ画像の横の画素数の格納先のポインタ, <code>nullptr</code> なら格納しない。 |
| <i>pHeight</i> | 読み込んだ画像の縦の画素数の格納先のポインタ, <code>nullptr</code> なら格納しない。 |
| <i>pFormat</i> | 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, <code>nullptr</code> 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 2801 行目に定義があります。

被呼び出し関係図:



7.1.4.62 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

| | |
|---------------|--------|
| <i>width</i> | 矩形の横幅. |
| <i>height</i> | 矩形の高さ. |

gg.cpp の 5149 行目に定義があります。

7.1.4.63 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

| | |
|----------|--------------------------------|
| <i>r</i> | 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 3050 行目に定義があります。

呼び出し関係図:



7.1.4.64 **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 3032 行目に定義があります。

呼び出し関係図:

7.1.4.65 **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 3041 行目に定義があります。

呼び出し関係図:



7.1.4.66 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

| | |
|----------|-----------------------------------|
| <i>r</i> | 回転軸のベクトルを表す GLfloat 型の 3 要素の配列変数. |
| <i>a</i> | 回転角. |

戻り値

r を軸に *a* だけ回転する変換行列.

gg.h の 3022 行目に定義があります。

呼び出し関係図:



7.1.4.67 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 3012 行目に定義があります。

呼び出し関係図:



7.1.4.68 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

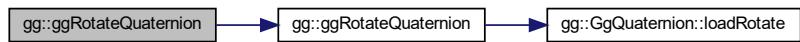
| | |
|-----|--------------------------------|
| v | 軸ベクトルを表す GLfloat 型の 4 要素の配列変数。 |
|-----|--------------------------------|

戻り値

回転を表す四元数。

gg.h の 4123 行目に定義があります。

呼び出し関係図:



7.1.4.69 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

| | |
|----------|--------------------------------|
| <i>v</i> | 軸ベクトルを表す GLfloat 型の 3 要素の配列変数。 |
| <i>a</i> | 回転角。 |

戻り値

回転を表す四元数。

gg.h の 4115 行目に定義があります。

呼び出し関係図:



7.1.4.70 **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

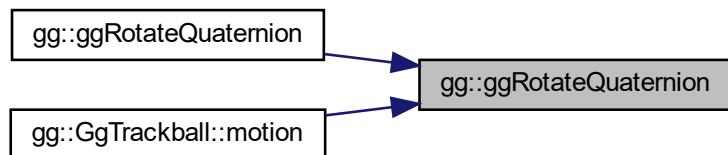
回転を表す四元数.

gg.h の 4105 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:



7.1.4.71 `ggRotateX()`

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

x 軸中心の回転の変換行列を返す。

引数

| | |
|----------------|------|
| <code>a</code> | 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 2982 行目に定義があります。

呼び出し関係図:



7.1.4.72 `ggRotateY()`

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

y 軸中心の回転の変換行列を返す。

引数

| | |
|----------------|------|
| <code>a</code> | 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 2991 行目に定義があります。

呼び出し関係図:



7.1.4.73 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 3000 行目に定義があります。

呼び出し関係図:



7.1.4.74 ggSaveColor()

```
bool gg::ggSaveColor (   
    const std::string & name )
```

カラーバッファの内容を TGA ファイルに保存する。

引数

| | |
|-------------|------------|
| <i>name</i> | 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 2745 行目に定義がります。

呼び出し関係図:



7.1.4.75 `ggSaveDepth()`

```
bool gg::ggSaveDepth (
    const std::string & name )
```

デプスバッファの内容を TGA ファイルに保存する。

引数

| | |
|-------------|------------|
| <i>name</i> | 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 2771 行目に定義がります。

呼び出し関係図:



7.1.4.76 ggSaveTga()

```
bool gg::ggSaveTga (
    const std::string & 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

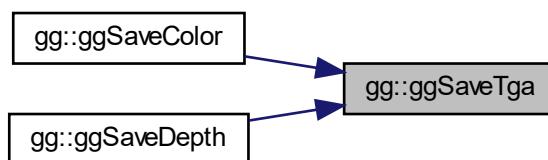
| | |
|---------------|---------------|
| <i>name</i> | 保存するファイル名。 |
| <i>buffer</i> | 画像データを格納した配列。 |
| <i>width</i> | 画像の横の画素数。 |
| <i>height</i> | 画像の縦の画素数。 |
| <i>depth</i> | 1画素のバイト数。 |

戻り値

保存に成功すれば `true`, 失敗すれば `false`。

gg.cpp の 2655 行目に定義があります。

被呼び出し関係図:



7.1.4.77 ggScale() [1/3]

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

拡大縮小の変換行列を返す。

引数

| | |
|----------------|----------------------------------|
| <code>s</code> | 拡大率の <code>GgVector</code> 型の変数. |
|----------------|----------------------------------|

戻り値

拡大縮小の変換行列.

`gg.h` の 2973 行目に定義がります。

呼び出し関係図:



7.1.4.78 `ggScale()` [2/3]

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

拡大縮小の変換行列を返す.

引数

| | |
|----------------|------------------------------------------------------------------|
| <code>s</code> | 拡大率の <code>GLfloat</code> 型の 3 要素の配列変数 (<code>x, y, z</code>). |
|----------------|------------------------------------------------------------------|

戻り値

拡大縮小の変換行列.

`gg.h` の 2964 行目に定義がります。

呼び出し関係図:



7.1.4.79 **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

| | |
|----------|------------------------|
| <i>x</i> | <i>x</i> 方向の拡大率。 |
| <i>y</i> | <i>y</i> 方向の拡大率。 |
| <i>z</i> | <i>z</i> 方向の拡大率。 |
| <i>w</i> | 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 2955 行目に定義があります。

呼び出し関係図:

7.1.4.80 **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

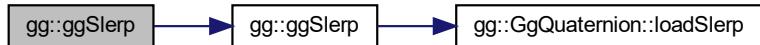
| | |
|----------|----------------------------------|
| <i>q</i> | <code>GgQuaternion</code> 型の四元数. |
| <i>r</i> | <code>GgQuaternion</code> 型の四元数. |
| <i>t</i> | 補間パラメータ. |

戻り値

q, r を *t* で内分した四元数.

gg.h の 4163 行目に定義があります。

呼び出し関係図:



7.1.4.81 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

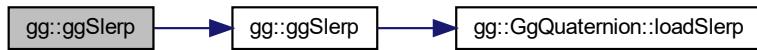
| | |
|----------|---------------------------------------------|
| <i>q</i> | <code>GgQuaternion</code> 型の四元数. |
| <i>a</i> | 四元数を格納した <code>GLfloat</code> 型の 4 要素の配列変数. |
| <i>t</i> | 補間パラメータ. |

戻り値

q, a を *t* で内分した四元数.

gg.h の 4173 行目に定義があります。

呼び出し関係図:



7.1.4.82 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

| | |
|----------|---------------------------------------|
| <i>a</i> | 四元数を格納した GLfloat 型の 4 要素の配列変数。 |
| <i>q</i> | GgQuaternion 型の四元数。 |
| <i>t</i> | 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 4183 行目に定義があります。

呼び出し関係図:



7.1.4.83 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

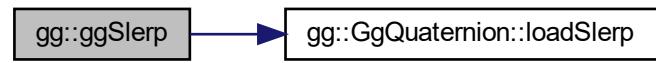
| | |
|----------|--------------------------------|
| <i>a</i> | 四元数を格納した GLfloat 型の 4 要素の配列変数. |
| <i>b</i> | 四元数を格納した GLfloat 型の 4 要素の配列変数. |
| <i>t</i> | 補間パラメータ. |

戻り値

a, b を *t* で内分した四元数.

gg.h の 4152 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:



7.1.4.84 ggTranslate() [1/3]

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

平行移動の変換行列を返す.

引数

| | |
|----------|---------------------|
| <i>t</i> | 移動量の GgVector 型の変数. |
|----------|---------------------|

戻り値

平行移動の変換行列.

gg.h の 2943 行目に定義があります。

呼び出し関係図:



7.1.4.85 ggTranslate() [2/3]

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

平行移動の変換行列を返す.

引数

| | |
|----------|--------------------------------------|
| <i>t</i> | 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 2934 行目に定義があります。

呼び出し関係図:



7.1.4.86 **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

| | |
|----------|------------------------|
| <i>x</i> | <i>x</i> 方向の移動量. |
| <i>y</i> | <i>y</i> 方向の移動量. |
| <i>z</i> | <i>z</i> 方向の移動量. |
| <i>w</i> | 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 2925 行目に定義があります。

呼び出し関係図:

7.1.4.87 **ggTranspose()**

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

転置行列を返す.

引数

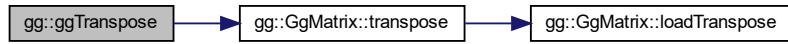
| | |
|----------|---------|
| <i>m</i> | 元の変換行列. |
|----------|---------|

戻り値

m の転置行列.

gg.h の 3159 行目に定義があります。

呼び出し関係図:



7.1.4.88 operator*() [1/3]

```
GgVector gg::operator* (
    const GgVector & a,
    const GgVector & b ) [inline]
```

GgVector 型の積を返す.

引数

| | |
|-----|----------------|
| a | GgVector 型の変数. |
| b | GgVector 型の変数. |

戻り値

a の各要素と b の各要素の要素ごとの積.

gg.h の 1746 行目に定義があります。

7.1.4.89 operator*() [2/3]

```
GgVector gg::operator* (
    const GgVector & a,
    GLfloat b ) [inline]
```

GgVector 型の各要素にスカラーを乗じた積を返す.

引数

| | |
|-----|----------------|
| a | GgVector 型の変数. |
| b | GLfloat 型の変数. |

戻り値

`a` の各要素に `b` を乗じた積.

`gg.h` の 1886 行目に定義があります。

7.1.4.90 `operator*()` [3/3]

```
GgVector gg::operator* (
    GLfloat a,
    const GgVector & b )  [inline]
```

`GgVector` 型の各要素にスカラーを乗じた積を返す.

引数

| | |
|----------------|-----------------------------|
| <code>b</code> | <code>GLfloat</code> 型の変数. |
| <code>a</code> | <code>GgVector</code> 型の変数. |

戻り値

`a` に `b` の各要素を乗じた積.

`gg.h` の 1898 行目に定義があります。

7.1.4.91 `operator*=(())` [1/2]

```
GgVector& gg::operator*=
( GgVector & a,
  const GgVector & b )  [inline]
```

`GgVector` 型を乗算する.

引数

| | |
|----------------|-----------------------------|
| <code>a</code> | <code>GgVector</code> 型の変数. |
| <code>b</code> | <code>GgVector</code> 型の変数. |

戻り値

`a` の各要素に `b` の各要素をそれぞれ乗算した `a` の参照.

`gg.h` の 1758 行目に定義があります。

7.1.4.92 `operator*=()` [2/2]

```
GgVector& gg::operator*=(  
    GgVector & a,  
    GLfloat b ) [inline]
```

`GgVector` 型の各要素にスカラーを乗じる。

引数

| | |
|----------|-----------------------------|
| <i>a</i> | <code>GgVector</code> 型の変数。 |
| <i>b</i> | <code>GLfloat</code> 型の変数。 |

戻り値

a の各要素に *b* を乗じた *a* の参照。

`gg.h` の 1910 行目に定義があります。

7.1.4.93 `operator+()` [1/3]

```
GgVector gg::operator+ (  
    const GgVector & a,  
    const GgVector & b ) [inline]
```

`GgVector` 型の和を返す。

引数

| | |
|----------|-----------------------------|
| <i>a</i> | <code>GgVector</code> 型の変数。 |
| <i>b</i> | <code>GgVector</code> 型の変数。 |

戻り値

a の各要素と *b* の各要素の要素ごとの和。

`gg.h` の 1688 行目に定義があります。

7.1.4.94 `operator+()` [2/3]

```
GgVector gg::operator+ (  
    const GgVector & a,  
    GLfloat b ) [inline]
```

`GgVector` 型の各要素にスカラーを足した和を返す。

引数

| | |
|----------|----------------|
| <i>a</i> | GgVector 型の変数. |
| <i>b</i> | GLfloat 型の変数. |

戻り値

a の各要素に *b* を足した和.

gg.h の 1804 行目に定義があります。

7.1.4.95 operator+() [3/3]

```
GgVector gg::operator+ (
    GLfloat a,
    const GgVector & b ) [inline]
```

GgVector 型の各要素にスカラーを足した和を返す.

引数

| | |
|----------|----------------|
| <i>b</i> | GLfloat 型の変数. |
| <i>a</i> | GgVector 型の変数. |

戻り値

a に *b* の各要素を足した和.

gg.h の 1816 行目に定義があります。

7.1.4.96 operator+=() [1/2]

```
GgVector& gg::operator+= (
    GgVector & a,
    const GgVector & b ) [inline]
```

GgVector 型を加算する.

引数

| | |
|----------|----------------|
| <i>a</i> | GgVector 型の変数. |
| <i>b</i> | GgVector 型の変数. |

戻り値

`a` の各要素に `b` の各要素をそれぞれ加算した `a` の参照.

`gg.h` の 1700 行目に定義があります。

7.1.4.97 `operator+=()` [2/2]

```
GgVector& gg::operator+= (
    GgVector & a,
    GLfloat b ) [inline]
```

`GgVector` 型の各要素にスカラーを足す.

引数

| | |
|----------------|-----------------------------|
| <code>a</code> | <code>GgVector</code> 型の変数. |
| <code>b</code> | <code>GLfloat</code> 型の変数. |

戻り値

`a` の各要素に `b` を足した `a` の参照.

`gg.h` の 1828 行目に定義があります。

7.1.4.98 `operator-()` [1/3]

```
GgVector gg::operator- (
    const GgVector & a,
    const GgVector & b ) [inline]
```

`GgVector` 型の差を返す.

引数

| | |
|----------------|-----------------------------|
| <code>a</code> | <code>GgVector</code> 型の変数. |
| <code>b</code> | <code>GgVector</code> 型の変数. |

戻り値

`a` の各要素から `b` の各要素を要素ごとに引いた結果.

`gg.h` の 1717 行目に定義があります。

7.1.4.99 **operator-()** [2/3]

```
GgVector gg::operator- (
    const GgVector & a,
    GLfloat b ) [inline]
```

GgVector 型の各要素からスカラーを引いた差を返す.

引数

| | |
|----------|----------------|
| <i>a</i> | GgVector 型の変数. |
| <i>b</i> | GLfloat 型の変数. |

戻り値

a の各要素から *b* を引いた差.

gg.h の 1845 行目に定義があります。

7.1.4.100 **operator-()** [3/3]

```
GgVector gg::operator- (
    GLfloat a,
    const GgVector & b ) [inline]
```

スカラーから GgVector 型の各要素を引いた差を返す.

引数

| | |
|----------|----------------|
| <i>b</i> | GLfloat 型の変数. |
| <i>a</i> | GgVector 型の変数. |

戻り値

a から *b* の各要素を引いた差.

gg.h の 1857 行目に定義があります。

7.1.4.101 **operator-=()** [1/2]

```
GgVector& gg::operator-= (
    GgVector & a,
    const GgVector & b ) [inline]
```

GgVector 型を減算する.

引数

| | |
|----------|----------------|
| <i>a</i> | GgVector 型の変数. |
| <i>b</i> | GgVector 型の変数. |

戻り値

a の各要素から *b* の各要素をそれぞれ減算した *a* の参照.

gg.h の 1729 行目に定義があります。

7.1.4.102 operator-() [2/2]

```
GgVector& gg::operator- (
    GgVector & a,
    GLfloat b ) [inline]
```

GgVector 型の各要素からスカラーを引く.

引数

| | |
|----------|----------------|
| <i>a</i> | GgVector 型の変数. |
| <i>b</i> | GLfloat 型の変数. |

戻り値

a の各要素から *b* を引いた *a* の参照.

gg.h の 1869 行目に定義があります。

7.1.4.103 operator/() [1/3]

```
GgVector gg::operator/ (
    const GgVector & a,
    const GgVector & b ) [inline]
```

GgVector 型の商を返す.

引数

| | |
|----------|----------------|
| <i>a</i> | GgVector 型の変数. |
| <i>b</i> | GgVector 型の変数. |

戻り値

`a` の各要素を `b` の各要素で要素ごとに割った結果.

`gg.h` の 1775 行目に定義があります。

7.1.4.104 `operator/()` [2/3]

```
GgVector gg::operator/ (
    const GgVector & a,
    GLfloat b ) [inline]
```

`GgVector` 型の各要素をスカラーで割った商を返す.

引数

| | |
|----------------|-----------------------------|
| <code>a</code> | <code>GgVector</code> 型の変数. |
| <code>b</code> | <code>GLfloat</code> 型の変数. |

戻り値

`a` の各要素を `b` で割った商.

`gg.h` の 1927 行目に定義があります。

7.1.4.105 `operator/()` [3/3]

```
GgVector gg::operator/ (
    GLfloat a,
    const GgVector & b ) [inline]
```

スカラーを `GgVector` 型の各要素で割った商を返す.

引数

| | |
|----------------|-----------------------------|
| <code>b</code> | <code>GLfloat</code> 型の変数. |
| <code>a</code> | <code>GgVector</code> 型の変数. |

戻り値

`a` を `b` の各要素で割った商.

`gg.h` の 1939 行目に定義があります。

7.1.4.106 operator/() [1/2]

```
GgVector& gg::operator/= (
    GgVector & a,
    const GgVector & b ) [inline]
```

GgVector 型を除算する.

引数

| | |
|----------|----------------|
| <i>a</i> | GgVector 型の変数. |
| <i>b</i> | GgVector 型の変数. |

戻り値

a の各要素を *b* の各要素でそれぞれ割った *a* の参照.

gg.h の 1787 行目に定義があります。

7.1.4.107 operator/() [2/2]

```
GgVector& gg::operator/= (
    GgVector & a,
    GLfloat b ) [inline]
```

GgVector 型の各要素をスカラーで割る.

引数

| | |
|----------|----------------|
| <i>a</i> | GgVector 型の変数. |
| <i>b</i> | GLfloat 型の変数. |

戻り値

a の各要素を *b* で割った *a* の参照.

gg.h の 1951 行目に定義があります。

7.1.5 変数詳解

7.1.5.1 ggBufferAlignment

```
GLint gg::ggBufferAlignment [extern]
```

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 8

クラス詳解

8.1 GgApp クラス

```
#include <GgApp.h>
```

公開メンバ関数

- int [main](#) (int argc, const char *const *argv)

8.1.1 詳解

GgApp.h の 15 行目に定義があります。

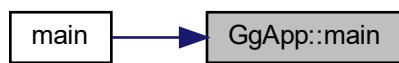
8.1.2 関数詳解

8.1.2.1 main()

```
int GgApp::main (
    int argc,
    const char *const * argv )
```

ggsample01.cpp の 26 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:

- [GgApp.h](#)
- [ggsample01.cpp](#)

8.2 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`
コピー構造関数は使用禁止.
- `GgBuffer< T > & operator= (const GgBuffer< T > &o)=delete`
代入演算子は使用禁止.
- `const 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.
- `const 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.
- `const 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.

8.2.1 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 4639 行目に定義があります。

8.2.2 構築子と解体子

8.2.2.1 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

| | |
|---------------|----------------------------------------------|
| <i>target</i> | バッファオブジェクトのターゲット. |
| <i>data</i> | 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). |
| <i>count</i> | データの数. |
| <i>stride</i> | データの間隔. |
| <i>usage</i> | バッファオブジェクトの使い方. |

gg.h の 4651 行目に定義があります。

8.2.2.2 ~GgBuffer()

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

デストラクタ.

gg.h の 4651 行目に定義があります。

8.2.2.3 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.

8.2.3 関数詳解

8.2.3.1 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 4721 行目に定義があります。

8.2.3.2 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

| | |
|-------------------|-------------------------------------------------|
| <i>src_buffer</i> | 複写元のバッファオブジェクト名。 |
| <i>src_first</i> | 複写元 (<i>src_buffer</i>) の先頭のデータの位置。 |
| <i>dst_first</i> | 複写先 (getBuffer()) の先頭のデータの位置。 |
| <i>count</i> | 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 4795 行目に定義があります。

8.2.3.3 getBuffer()

```
template<typename T >
const 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 4715 行目に定義があります。

8.2.3.4 getCount()

```
template<typename T>
const 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 4708 行目に定義があります。

8.2.3.5 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 4701 行目に定義があります。

8.2.3.6 getTarget()

```
template<typename T>
const 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 4694 行目に定義があります。

8.2.3.7 `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 4734 行目に定義があります。

8.2.3.8 `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

| | |
|--------------|--------------------------------|
| <i>first</i> | 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。 |
| <i>count</i> | 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 4744 行目に定義があります。

8.2.3.9 `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。

8.2.3.10 `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

| | |
|--------------|------------------------------------|
| <i>data</i> | 抽出先の領域の先頭のポインタ. |
| <i>first</i> | 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. |
| <i>count</i> | 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 4779 行目に定義があります。

8.2.3.11 `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

| | |
|--------------|-------------------------------|
| <i>data</i> | 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. |
| <i>first</i> | 転送先のバッファオブジェクトの先頭の要素番号. |
| <i>count</i> | 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 4764 行目に定義があります。

8.2.3.12 `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 4727 行目に定義があります。

8.2.3.13 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 4755 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](#)

8.3 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\(const std::string &name, GLenum internal=0, GLenum wrap=GL_CLAMP_TO_EDGE\)](#)
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\(const std::string &name, GLenum internal=0, GLenum wrap=GL_CLAMP_TO_EDGE\)](#)
テクスチャを作成してファイルからデータを読み込む.

8.3.1 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 4464 行目に定義があります。

8.3.2 構築子と解体子

8.3.2.1 GgColorTexture() [1/3]

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

コンストラクタ.

gg.h の 4472 行目に定義があります。

8.3.2.2 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

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

gg.h の 4484 行目に定義があります。

呼び出し関係図:



8.3.2.3 GgColorTexture() [3/3]

```
gg::GgColorTexture::GgColorTexture (
    const std::string & name,
    GLenum internal = 0,
    GLenum wrap = GL_CLAMP_TO_EDGE ) [inline]
```

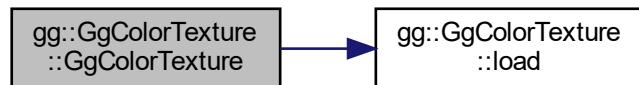
ファイルからデータを読み込んでテクスチャを作成するコンストラクタ.

引数

| | |
|-----------------|-----------------------------------------------------------------|
| <i>name</i> | 読み込むファイル名. |
| <i>internal</i> | 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. |
| <i>wrap</i> | 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 4501 行目に定義があります。

呼び出し関係図:



8.3.2.4 ~GgColorTexture()

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

デストラクタ.

gg.h の 4507 行目に定義があります。

8.3.3 関数詳解

8.3.3.1 **load()** [1/2]

```
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

| | |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>image</i> | テクスチャとして用いる画像データ, <code>nullptr</code> ならデータを読み込まない。 |
| <i>width</i> | テクスチャの横の画素数。 |
| <i>height</i> | テクスチャの縦の画素数。 |
| <i>format</i> | 読み込む画像のフォーマット。 |
| <i>type</i> | 読み込む画像のデータ型。 |
| <i>internal</i> | <code>glTexImage2D()</code> に指定するテクスチャの内部フォーマット。 |
| <i>wrap</i> | テクスチャのラッピングモード (<code>GL_CLAMP_TO_EDGE</code> , <code>GL_CLAMP_TO_BORDER</code> , <code>GL_REPEAT</code> , <code>GL_MIRRORED_REPEAT</code>). |

gg.h の 4519 行目に定義があります。

被呼び出し関係図:

8.3.3.2 **load()** [2/2]

```
void gg::GgColorTexture::load (
    const std::string & name,
    GLenum internal = 0,
    GLenum wrap = GL_CLAMP_TO_EDGE )
```

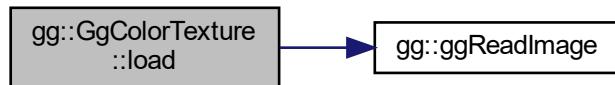
テクスチャを作成してファイルからデータを読み込む。

引数

| | |
|-----------------|---------------------------------------------------------------------------------------|
| <i>name</i> | 読み込むファイル名. |
| <i>internal</i> | 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. |
| <i>wrap</i> | 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 3154 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:

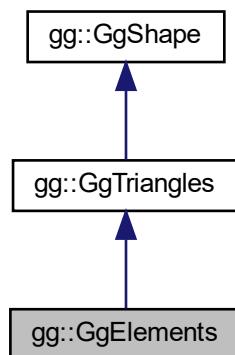
- [gg.h](#)
- [gg.cpp](#)

8.4 gg::GgElements クラス

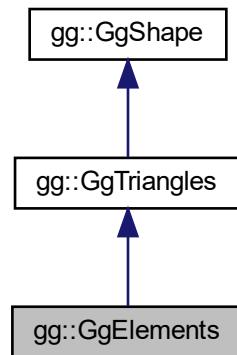
三角形で表した形状データ (Elements 形式).

```
#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.
- `const GLsizei & getIndexCount () const`
データの数を取り出す.
- `const 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.

8.4.1 詳解

三角形で表した形状データ (Elements 形式).

gg.h の 5352 行目に定義があります。

8.4.2 構築子と解体子

8.4.2.1 GgElements() [1/2]

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

コンストラクタ.

引数

| | |
|-------------|--------------|
| <i>mode</i> | 描画する基本図形の種類. |
|-------------|--------------|

gg.h の 5362 行目に定義がります。

8.4.2.2 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

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

gg.h の 5374 行目に定義がります。

8.4.2.3 ~GgElements()

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

デストラクタ.

gg.h の 5388 行目に定義がります。

8.4.3 関数詳解

8.4.3.1 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

| | |
|--------------|--------------------------|
| <i>first</i> | 描画を開始する最初の三角形番号. |
| <i>count</i> | 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 5075 行目に定義があります。

呼び出し関係図:



8.4.3.2 getIndexBuffer()

```
const 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 5401 行目に定義があります。

8.4.3.3 getIndexCount()

```
const 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 5394 行目に定義があります。

8.4.3.4 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

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

gg.h の 5432 行目に定義があります。

8.4.3.5 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

| | |
|---------------|--------------------------------|
| <i>vert</i> | 頂点属性が格納されてている領域の先頭のポインタ. |
| <i>firstv</i> | 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. |
| <i>countv</i> | 頂点のデータの数 (頂点数). |
| <i>face</i> | 三角形の頂点インデックスデータ. |
| <i>firstf</i> | 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. |
| <i>countf</i> | 三角形の頂点数. |

gg.h の 5413 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:

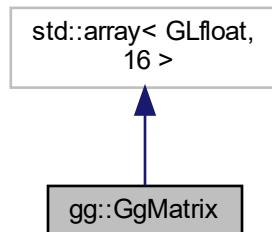
- [gg.h](#)
- [gg.cpp](#)

8.5 gg::GgMatrix クラス

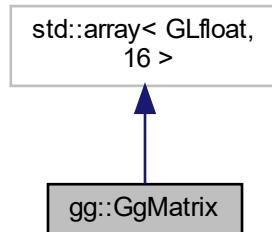
変換行列.

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

gg::GgMatrix の継承関係図



gg::GgMatrix 連携図



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.

フレンド

- class **GgQuaternion**

8.5.1 詳解

変換行列.

gg.h の 2129 行目に定義があります。

8.5.2 構築子と解体子

8.5.2.1 GgMatrix() [1/3]

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

コンストラクタ.

gg.h の 2143 行目に定義があります。

8.5.2.2 GgMatrix() [2/3]

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

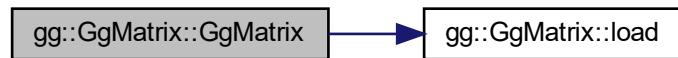
コンストラクタ。

引数

| | |
|----------|------------------------|
| <i>a</i> | GLfloat 型の 16 要素の配列変数。 |
|----------|------------------------|

gg.h の 2149 行目に定義がります。

呼び出し関係図:



8.5.2.3 GgMatrix() [3/3]

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

コピーコンストラクタ。

引数

| | |
|----------|----------------|
| <i>m</i> | GgMatrix 型の変数。 |
|----------|----------------|

gg.h の 2156 行目に定義がります。

呼び出し関係図:



8.5.2.4 ~GgMatrix()

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

デストラクタ.

gg.h の 2162 行目に定義があります。

8.5.3 関数詳解

8.5.3.1 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

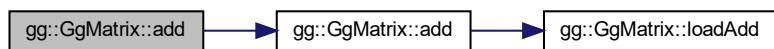
| | |
|----------------|-----------------------------|
| <code>m</code> | <code>GgMatrix</code> 型の変数. |
|----------------|-----------------------------|

戻り値

変換行列に `m` を加えた `GgMatrix` 型の値.

gg.h の 2261 行目に定義があります。

呼び出し関係図:



8.5.3.2 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

| | |
|----------------|------------------------|
| <code>a</code> | 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 2252 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:



8.5.3.3 `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

| | |
|----------------|-----------------------------|
| <code>m</code> | <code>GgMatrix</code> 型の変数. |
|----------------|-----------------------------|

戻り値

変換行列を m で割った [GgMatrix](#) 型の値.

gg.h の 2315 行目に定義があります。

呼び出し関係図:



8.5.3.4 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

| | |
|----------|------------------------|
| <i>a</i> | GLfloat 型の 16 要素の配列変数. |
|----------|------------------------|

戻り値

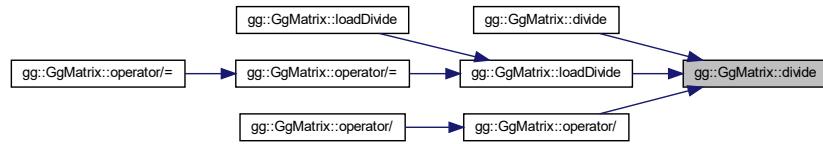
変換行列を a で割った [GgMatrix](#) 型の値.

gg.h の 2304 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:



8.5.3.5 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

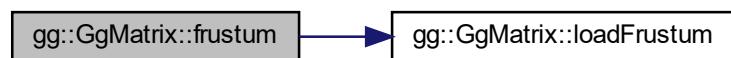
| | |
|---------------|---------------|
| <i>left</i> | ウィンドウの左端の位置. |
| <i>right</i> | ウィンドウの右端の位置. |
| <i>bottom</i> | ウィンドウの下端の位置. |
| <i>top</i> | ウィンドウの上端の位置. |
| <i>zNear</i> | 視点から前方面までの位置. |
| <i>zFar</i> | 視点から後方面までの位置. |

戻り値

透視投影変換行列を乗じた変換行列.

gg.h の 2805 行目に定義があります。

呼び出し関係図:



8.5.3.6 `get()` [1/2]

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

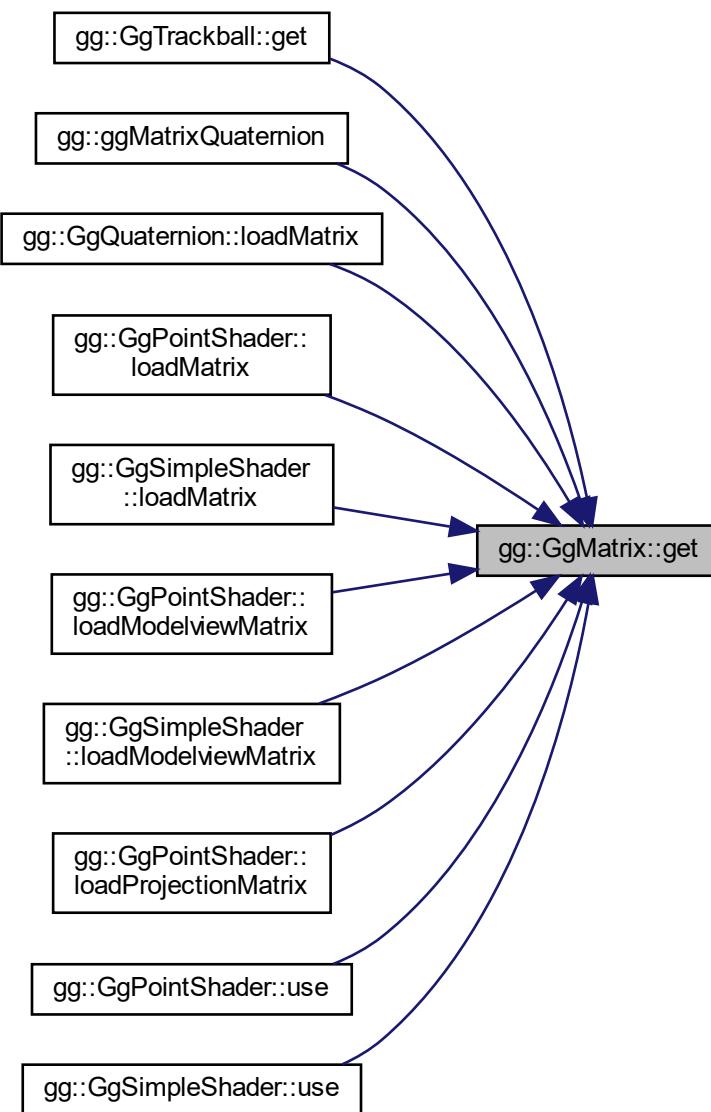
変換行列を取り出す。

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 2898 行目に定義があります。

被呼び出し関係図:



8.5.3.7 `get()` [2/2]

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

変換行列を取り出す。

引数

| | |
|----------|----------------------------------|
| <i>a</i> | 変換行列を格納する GLfloat 型の 16 要素の配列変数。 |
|----------|----------------------------------|

gg.h の 2905 行目に定義があります。

8.5.3.8 `invert()`

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

逆行列を返す。

戻り値

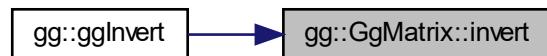
逆行列。

gg.h の 2840 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:



8.5.3.9 **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

| | |
|----------|----------------|
| <i>m</i> | GgMatrix 型の変数。 |
|----------|----------------|

戻り値

m を代入した GgMatrix 型の値。

gg.h の 2178 行目に定義があります。

呼び出し関係図:

8.5.3.10 **load()** [2/2]

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

配列変数の値を格納する。

引数

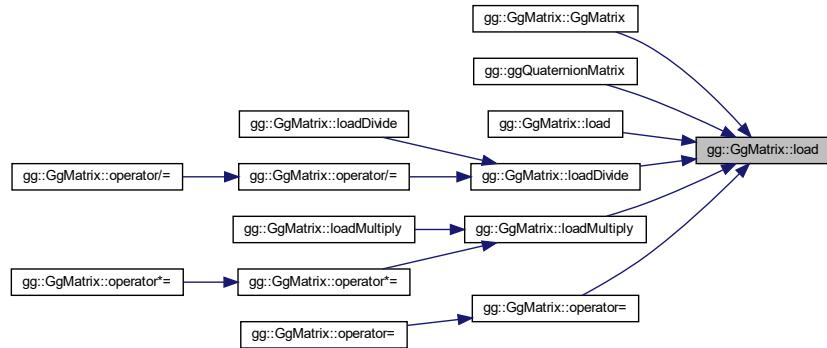
| | |
|----------|------------------------|
| <i>a</i> | GLfloat 型の 16 要素の配列変数。 |
|----------|------------------------|

戻り値

a を代入した GgMatrix 型の値。

gg.h の 2169 行目に定義があります。

被呼び出し関係図:



8.5.3.11 loadAdd() [1/2]

```
GgMatrix& gg::GgMatrix::loadAdd (
```

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 2195 行目に定義があります。

呼び出し関係図:



8.5.3.12 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

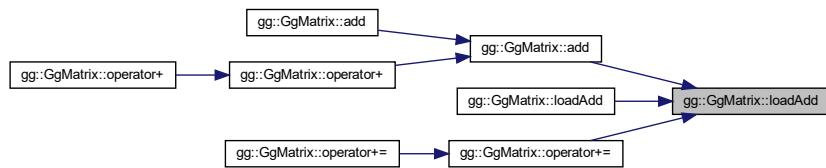
| | |
|----------|------------------------|
| <i>a</i> | GLfloat 型の 16 要素の配列変数。 |
|----------|------------------------|

戻り値

変換行列に *a* を加えた GgMatrix 型の値。

gg.h の 2186 行目に定義があります。

被呼び出し関係図:



8.5.3.13 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

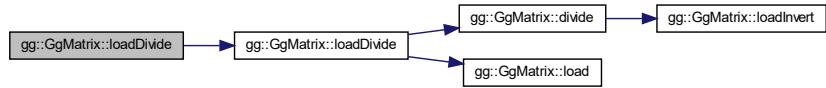
| | |
|----------|----------------|
| <i>m</i> | GgMatrix 型の変数。 |
|----------|----------------|

戻り値

変換行列に *m* を乗じた GgMatrix 型の値。

gg.h の 2244 行目に定義があります。

呼び出し関係図:



8.5.3.14 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

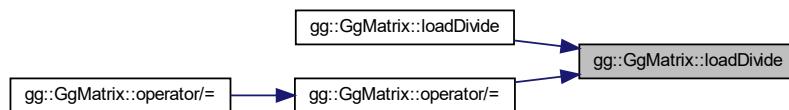
変換行列に a を乗じた `GgMatrix` 型の値。

`gg.h` の 2236 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:



8.5.3.15 **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

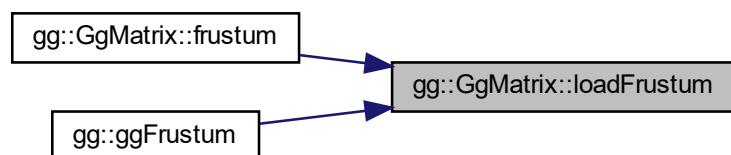
| | |
|---------------|---------------|
| <i>left</i> | ウィンドウの左端の位置。 |
| <i>right</i> | ウィンドウの右端の位置。 |
| <i>bottom</i> | ウィンドウの下端の位置。 |
| <i>top</i> | ウィンドウの上端の位置。 |
| <i>zNear</i> | 視点から前方面までの位置。 |
| <i>zFar</i> | 視点から後方面までの位置。 |

戻り値

設定した透視投影変換行列。

gg.cpp の 4631 行目に定義があります。

被呼び出し関係図:

8.5.3.16 **loadIdentity()**

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

単位行列を格納する。

gg.cpp の 4284 行目に定義がります。

呼び出し関係図:



8.5.3.17 loadInvert() [1/2]

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

逆行列を格納する。

引数

| | |
|----------|------------------|
| <i>m</i> | GgMatrix 型の変換行列。 |
|----------|------------------|

戻り値

設定した *m* の逆行列。

gg.h の 2591 行目に定義がります。

呼び出し関係図:



8.5.3.18 loadInvert() [2/2]

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

逆行列を格納する。

引数

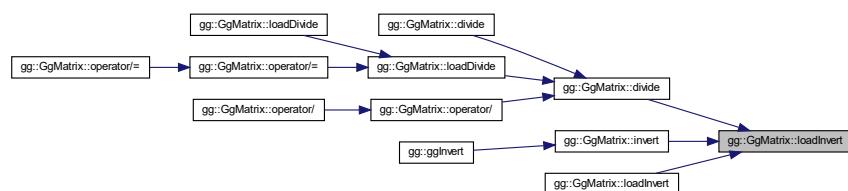
| | |
|---|------------------------|
| a | GLfloat 型の 16 要素の変換行列. |
|---|------------------------|

戻り値

設定した a の逆行列.

gg.cpp の 4441 行目に定義があります。

被呼び出し関係図:



8.5.3.19 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 2532 行目に定義があります。

呼び出し関係図:



8.5.3.20 `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

| | |
|----------|----------------|
| <i>e</i> | 視点の位置の配列変数. |
| <i>t</i> | 目標点の位置の配列変数. |
| <i>u</i> | 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 2522 行目に定義があります。

呼び出し関係図:



8.5.3.21 `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

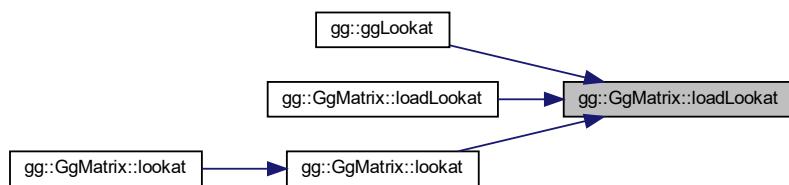
| | |
|-----------------|-----------------|
| <code>ex</code> | 視点の位置の x 座標値. |
| <code>ey</code> | 視点の位置の y 座標値. |
| <code>ez</code> | 視点の位置の z 座標値. |
| <code>tx</code> | 目標点の位置の x 座標値. |
| <code>ty</code> | 目標点の位置の y 座標値. |
| <code>tz</code> | 目標点の位置の z 座標値. |
| <code>ux</code> | 上方向のベクトルの x 成分. |
| <code>uy</code> | 上方向のベクトルの y 成分. |
| <code>uz</code> | 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 4543 行目に定義があります。

被呼び出し関係図:



8.5.3.22 `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

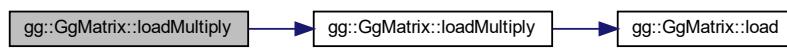
| | |
|----------------|----------------|
| <code>m</code> | GgMatrix 型の変数. |
|----------------|----------------|

戻り値

変換行列に `m` を掛けた GgMatrix 型の値.

gg.h の 2228 行目に定義があります。

呼び出し関係図:



8.5.3.23 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

| | |
|----------------|------------------------|
| <code>a</code> | GLfloat 型の 16 要素の配列変数. |
|----------------|------------------------|

戻り値

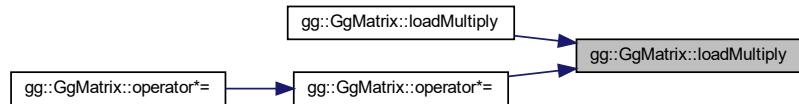
変換行列に `a` を掛けた GgMatrix 型の値.

gg.h の 2220 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:



8.5.3.24 `loadNormal()` [1/2]

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

法線変換行列を格納する。

引数

| | |
|----------------|-------------------------------|
| <code>m</code> | <code>GgMatrix</code> 型の変換行列。 |
|----------------|-------------------------------|

戻り値

設定した `m` の法線変換行列。

`gg.h` の 2604 行目に定義があります。

呼び出し関係図:



8.5.3.25 `loadNormal()` [2/2]

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

法線変換行列を格納する。

引数

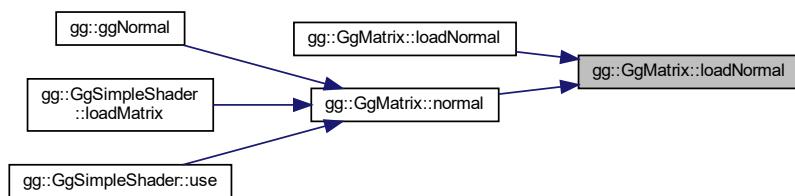
| | |
|---|------------------------|
| a | GLfloat 型の 16 要素の変換行列. |
|---|------------------------|

戻り値

設定した m の法線変換行列.

gg.cpp の 4523 行目に定義があります。

被呼び出し関係図:



8.5.3.26 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

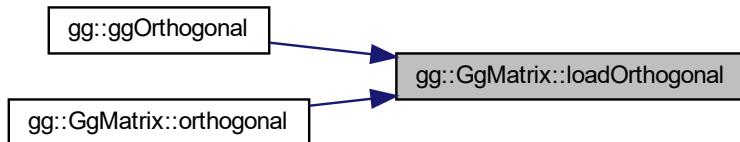
| | |
|---------------|---------------|
| <i>left</i> | ウィンドウの左端の位置. |
| <i>right</i> | ウィンドウの右端の位置. |
| <i>bottom</i> | ウィンドウの下端の位置. |
| <i>top</i> | ウィンドウの上端の位置. |
| <i>zNear</i> | 視点から前方面までの位置. |
| <i>zFar</i> | 視点から後方面までの位置. |

戻り値

設定した直交投影変換行列.

gg.cpp の 4601 行目に定義があります。

被呼び出し関係図:



8.5.3.27 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

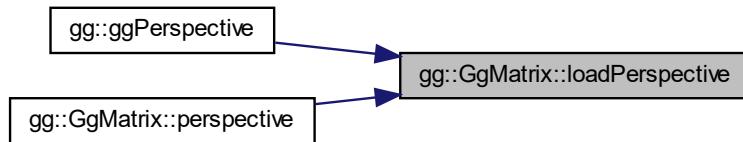
| | |
|---------------|-----------------|
| <i>fovy</i> | <i>y</i> 方向の画角. |
| <i>aspect</i> | 縦横比. |
| <i>zNear</i> | 視点から前方面までの位置. |
| <i>zFar</i> | 視点から後方面までの位置. |

戻り値

設定した透視投影変換行列.

gg.cpp の 4661 行目に定義があります。

呼び出し関係図:



8.5.3.28 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

| | |
|----------|------------------------------------|
| <i>r</i> | 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 2497 行目に定義があります。

呼び出し関係図:



8.5.3.29 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

| | |
|----------|--------------------------------|
| <i>r</i> | 回転軸の方向ベクトルを格納した GgVector 型の変数. |
| <i>a</i> | 回転角. |

戻り値

設定した変換行列.

gg.h の 2481 行目に定義があります。

呼び出し関係図:



8.5.3.30 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

| | |
|----------|--------------------------------------------------------|
| <i>r</i> | 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 2489 行目に定義があります。

呼び出し関係図:



8.5.3.31 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 2472 行目に定義があります。

呼び出し関係図:



8.5.3.32 `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

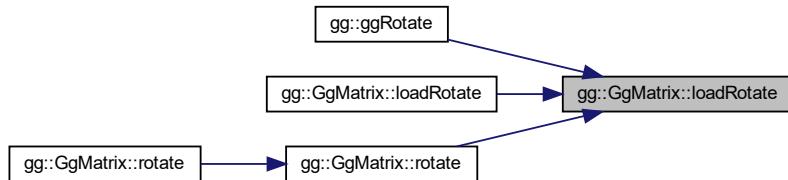
| | |
|----------|------------|
| <i>x</i> | 回転軸の x 成分. |
| <i>y</i> | 回転軸の y 成分. |
| <i>z</i> | 回転軸の z 成分. |
| <i>a</i> | 回転角. |

戻り値

設定した変換行列.

`gg.cpp` の 4377 行目に定義があります。

被呼び出し関係図:



8.5.3.33 `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

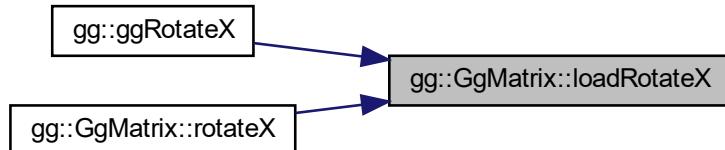
| | |
|----------|------|
| <i>a</i> | 回転角. |
|----------|------|

戻り値

設定した変換行列.

gg.cpp の 4329 行目に定義があります。

被呼び出し関係図:



8.5.3.34 `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

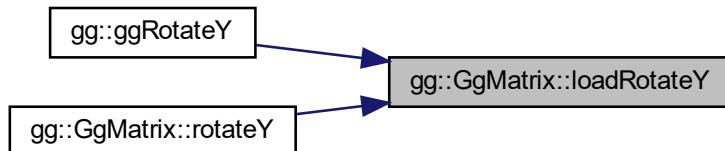
| | |
|----------------|------|
| <code>a</code> | 回転角. |
|----------------|------|

戻り値

設定した変換行列.

gg.cpp の 4345 行目に定義があります。

被呼び出し関係図:



8.5.3.35 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

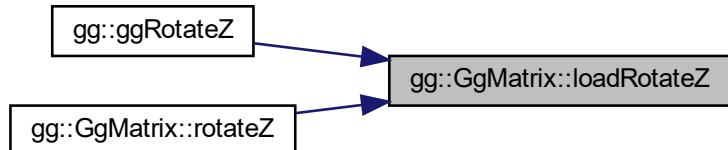
| | |
|----------|------|
| <i>a</i> | 回転角。 |
|----------|------|

戻り値

設定した変換行列。

gg.cpp の 4361 行目に定義があります。

被呼び出し関係図:



8.5.3.36 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

| | |
|----------|---------------------|
| <i>s</i> | 拡大率の GgVector 型の変数。 |
|----------|---------------------|

戻り値

設定した変換行列.

gg.h の 2440 行目に定義があります。

呼び出し関係図:



8.5.3.37 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

| | |
|----------------|------------------------------|
| <code>s</code> | 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 2432 行目に定義があります。

呼び出し関係図:



8.5.3.38 `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

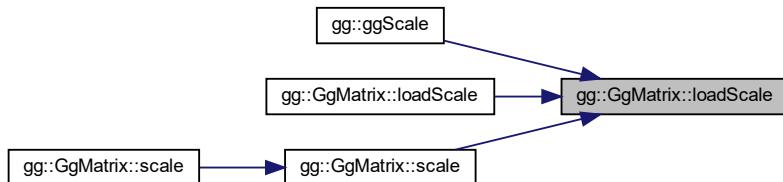
| | |
|----------|---------------------------------|
| <i>x</i> | <i>x</i> 方向の拡大率. |
| <i>y</i> | <i>y</i> 方向の拡大率. |
| <i>z</i> | <i>z</i> 方向の拡大率. |
| <i>w</i> | <i>w</i> 拡大率のスケールファクタ (= 1.0f). |

戻り値

設定した変換行列.

gg.cpp の 4313 行目に定義があります。

被呼び出し関係図:



8.5.3.39 `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

| | |
|----------|-----------------------------|
| <i>m</i> | <code>GgMatrix</code> 型の変数. |
|----------|-----------------------------|

戻り値

変換行列に m を引いた [GgMatrix](#) 型の値.

gg.h の 2212 行目に定義があります。

呼び出し関係図:



8.5.3.40 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

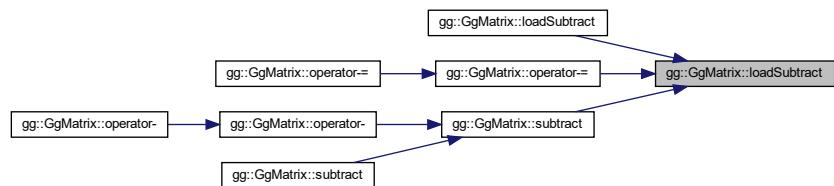
| | |
|---|------------------------|
| a | GLfloat 型の 16 要素の配列変数. |
|---|------------------------|

戻り値

変換行列に a を引いた [GgMatrix](#) 型の値.

gg.h の 2203 行目に定義があります。

被呼び出し関係図:



8.5.3.41 `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

| | |
|----------------|---------------------|
| <code>t</code> | 移動量の GgVector 型の変数。 |
|----------------|---------------------|

戻り値

設定した変換行列。

gg.h の 2416 行目に定義があります。

呼び出し関係図:



8.5.3.42 `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

| | |
|----------------|------------------------------|
| <code>t</code> | 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 2408 行目に定義があります。

呼び出し関係図:



8.5.3.43 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

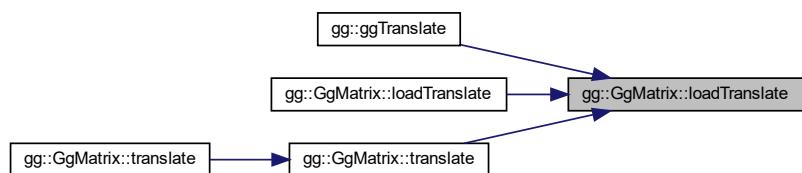
| | |
|----------|---------------------------------|
| <i>x</i> | <i>x</i> 方向の移動量. |
| <i>y</i> | <i>y</i> 方向の移動量. |
| <i>z</i> | <i>z</i> 方向の移動量. |
| <i>w</i> | <i>w</i> 移動量のスケールファクタ (= 1.0f). |

戻り値

設定した変換行列.

gg.cpp の 4297 行目に定義があります。

被呼び出し関係図:



8.5.3.44 `loadTranspose()` [1/2]

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

転置行列を格納する。

引数

| | |
|----------------|-------------------------------|
| <code>m</code> | <code>GgMatrix</code> 型の変換行列。 |
|----------------|-------------------------------|

戻り値

設定した `m` の転置行列。

`gg.h` の 2578 行目に定義があります。

呼び出し関係図:



8.5.3.45 `loadTranspose()` [2/2]

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

転置行列を格納する。

引数

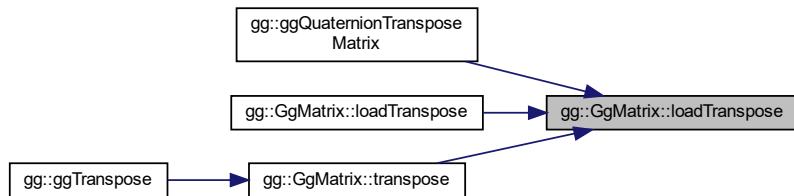
| | |
|----------------|-------------------------------------|
| <code>a</code> | <code>GLfloat</code> 型の 16 要素の変換行列。 |
|----------------|-------------------------------------|

戻り値

設定した `a` の転置行列。

`gg.cpp` の 4416 行目に定義があります。

被呼び出し関係図:



8.5.3.46 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

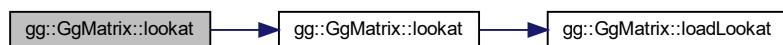
| | |
|----------|------------------------------|
| <i>e</i> | 視点の位置を格納した GgVector 型の変数. |
| <i>t</i> | 目標点の位置を格納した GgVector 型の変数. |
| <i>u</i> | 上方向のベクトルを格納した GgVector 型の変数. |

戻り値

ビュー変換行列を乗じた変換行列.

gg.h の 2774 行目に定義があります。

呼び出し関係図:



8.5.3.47 `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

| | |
|----------|--------------------------------------------------|
| <i>e</i> | 視点の位置を格納した <code>GLfloat</code> 型の 3 要素の配列変数。 |
| <i>t</i> | 目標点の位置を格納した <code>GLfloat</code> 型の 3 要素の配列変数。 |
| <i>u</i> | 上方向のベクトルを格納した <code>GLfloat</code> 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 2764 行目に定義があります。

呼び出し関係図:



8.5.3.48 `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

| | |
|-----------|---------------|
| <i>ex</i> | 視点の位置の x 座標値。 |
|-----------|---------------|

引数

| | |
|-----------|-----------------|
| <i>ey</i> | 視点の位置の y 座標値. |
| <i>ez</i> | 視点の位置の z 座標値. |
| <i>tx</i> | 目標点の位置の x 座標値. |
| <i>ty</i> | 目標点の位置の y 座標値. |
| <i>tz</i> | 目標点の位置の z 座標値. |
| <i>ux</i> | 上方向のベクトルの x 成分. |
| <i>uy</i> | 上方向のベクトルの y 成分. |
| <i>uz</i> | 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 2749 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:



8.5.3.49 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

| | |
|----------------|-----------------------------|
| <code>m</code> | <code>GgMatrix</code> 型の変数. |
|----------------|-----------------------------|

戻り値

変換行列に `m` を掛けた `GgMatrix` 型の値.

`gg.h` の 2296 行目に定義がります。

8.5.3.50 `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

| | |
|----------------|-------------------------------------|
| <code>a</code> | <code>GLfloat</code> 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 2286 行目に定義がります。

8.5.3.51 `normal()`

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

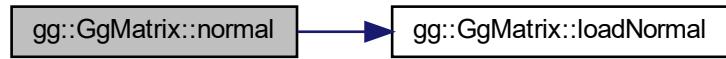
法線変換行列を返す.

戻り値

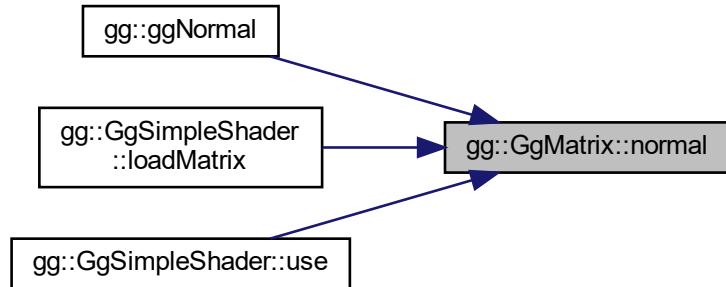
法線変換行列.

`gg.h` の 2848 行目に定義がります。

呼び出し関係図:



被呼び出し関係図:



8.5.3.52 operator*() [1/3]

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

gg.h の 2381 行目に定義があります。

呼び出し関係図:



8.5.3.53 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

| | |
|----------|------------------------|
| <i>v</i> | 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 2889 行目に定義があります。

8.5.3.54 operator*() [3/3]

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

gg.h の 2377 行目に定義があります。

呼び出し関係図:

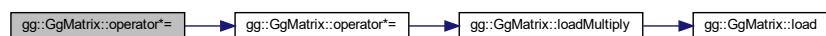


8.5.3.55 operator*=(()) [1/2]

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

gg.h の 2349 行目に定義があります。

呼び出し関係図:



8.5.3.56 operator*=() [2/2]

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

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:

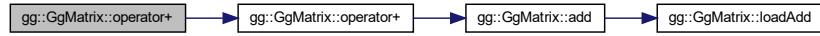


8.5.3.57 operator+() [1/2]

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

gg.h の 2365 行目に定義があります。

呼び出し関係図:



8.5.3.58 operator+() [2/2]

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

gg.h の 2361 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:

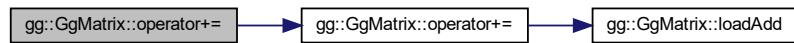


8.5.3.59 operator+=(()) [1/2]

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

gg.h の 2333 行目に定義があります。

呼び出し関係図:



8.5.3.60 operator+=() [2/2]

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

gg.h の 2329 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:



8.5.3.61 operator-() [1/2]

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

gg.h の 2373 行目に定義があります。

呼び出し関係図:

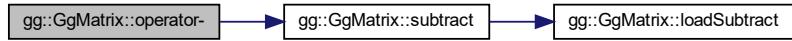


8.5.3.62 operator-() [2/2]

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

gg.h の 2369 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:



8.5.3.63 operator-=() [1/2]

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

gg.h の 2341 行目に定義があります。

呼び出し関係図:



8.5.3.64 operator-=() [2/2]

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

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:



8.5.3.65 operator/() [1/2]

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

gg.h の 2389 行目に定義があります。

呼び出し関係図:

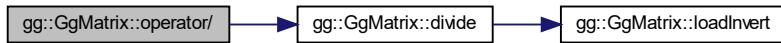


8.5.3.66 operator/() [2/2]

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

gg.h の 2385 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:



8.5.3.67 operator/=(()) [1/2]

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

gg.h の 2357 行目に定義があります。

呼び出し関係図:

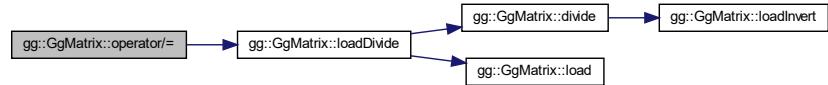


8.5.3.68 operator/() [2/2]

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

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:

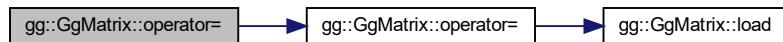


8.5.3.69 operator=() [1/2]

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

gg.h の 2325 行目に定義があります。

呼び出し関係図:



8.5.3.70 operator=() [2/2]

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

gg.h の 2321 行目に定義があります。

呼び出し関係図:



被呼び出し関係図:



8.5.3.71 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

| | |
|---------------|---------------|
| <i>left</i> | ウィンドウの左端の位置。 |
| <i>right</i> | ウィンドウの右端の位置。 |
| <i>bottom</i> | ウィンドウの下端の位置。 |
| <i>top</i> | ウィンドウの上端の位置。 |
| <i>zNear</i> | 視点から前方面までの位置。 |
| <i>zFar</i> | 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 2787 行目に定義があります。

呼び出し関係図:



8.5.3.72 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

| | |
|---------------|---------------|
| <i>fovy</i> | y 方向の画角. |
| <i>aspect</i> | 縦横比. |
| <i>zNear</i> | 視点から前方面までの位置. |
| <i>zFar</i> | 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 2821 行目に定義があります。

呼び出し関係図:



8.5.3.73 `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

| | |
|----------|--------------------------|
| <i>c</i> | 変換結果を格納する GgVector 型の変数。 |
| <i>v</i> | 元のベクトルの GgVector 型の変数。 |

gg.h の 2881 行目に定義があります。

8.5.3.74 `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

| | |
|----------|-------------------------------|
| <i>c</i> | 変換結果を格納する GgVector 型の変数。 |
| <i>v</i> | 元のベクトルの GLfloat 型の 4 要素の配列変数。 |

gg.h の 2873 行目に定義があります。

8.5.3.75 `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

| | |
|----------|---------------------------------|
| <i>c</i> | 変換結果を格納する GLfloat 型の 4 要素の配列変数。 |
| <i>v</i> | 元のベクトルの GgVector 型の変数。 |

gg.h の 2865 行目に定義があります。

8.5.3.76 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

| | |
|----------|---------------------------------|
| <i>c</i> | 変換結果を格納する GLfloat 型の 4 要素の配列変数。 |
| <i>v</i> | 元のベクトルの GLfloat 型の 4 要素の配列変数。 |

gg.h の 2857 行目に定義があります。

8.5.3.77 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

| | |
|----------|-------------------------------------|
| <i>r</i> | 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 2733 行目に定義があります。

呼び出し関係図:



8.5.3.78 `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

| | |
|----------------|---------------------------------------------|
| <code>r</code> | 回転軸の方向ベクトルを格納した <code>GgVector</code> 型の変数. |
| <code>a</code> | 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 2717 行目に定義があります。

呼び出し関係図:



8.5.3.79 `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

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

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 2725 行目に定義があります。

呼び出し関係図:



8.5.3.80 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

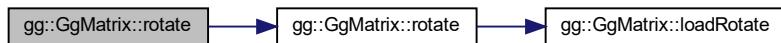
| | |
|-----|------------------------------------------------------------------|
| r | 回転軸の方向ベクトルを格納した <code>GLfloat</code> 型の 3 要素の配列変数 (x, y, z). |
| a | 回転角. |

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 2708 行目に定義があります。

呼び出し関係図:



8.5.3.81 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

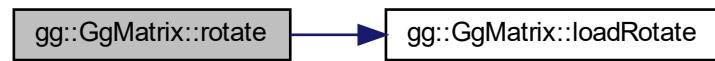
| | |
|----------|------------|
| <i>x</i> | 回転軸の x 成分. |
| <i>y</i> | 回転軸の y 成分. |
| <i>z</i> | 回転軸の z 成分. |
| <i>a</i> | 回転角. |

戻り値

(x, y, z) を軸にさらに *a* 回転した変換行列.

gg.h の 2698 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:



8.5.3.82 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

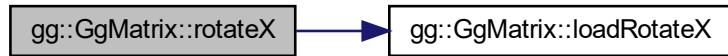
| | |
|----------|------|
| <i>a</i> | 回転角. |
|----------|------|

戻り値

x 軸中心にさらに a 回転した変換行列.

gg.h の 2668 行目に定義があります。

呼び出し関係図:



8.5.3.83 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

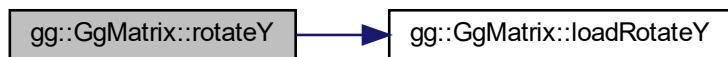
| | |
|-----|------|
| a | 回転角. |
|-----|------|

戻り値

y 軸中心にさらに a 回転した変換行列.

gg.h の 2677 行目に定義があります。

呼び出し関係図:



8.5.3.84 `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

| | |
|----------------|------|
| <code>a</code> | 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 2686 行目に定義があります。

呼び出し関係図:



8.5.3.85 `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

| | |
|----------------|----------------------------------|
| <code>s</code> | 拡大率の <code>GgVector</code> 型の変数. |
|----------------|----------------------------------|

戻り値

拡大縮小した結果の変換行列。

`gg.h` の 2660 行目に定義があります。

呼び出し関係図:



8.5.3.86 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

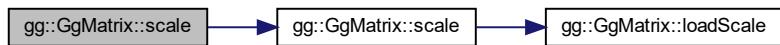
| | |
|----------|--------------------------------------|
| s | 拡大率の GLfloat 型の 3 要素の配列変数 (x, y, z). |
|----------|--------------------------------------|

戻り値

拡大縮小した結果の変換行列.

gg.h の 2652 行目に定義があります。

呼び出し関係図:



8.5.3.87 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

| | |
|----------|---------------------------------|
| <i>x</i> | <i>x</i> 方向の拡大率. |
| <i>y</i> | <i>y</i> 方向の拡大率. |
| <i>z</i> | <i>z</i> 方向の拡大率. |
| <i>w</i> | <i>w</i> 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 2643 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:



8.5.3.88 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

| | |
|----------|-----------------------|
| <i>m</i> | <i>GgMatrix</i> 型の変数. |
|----------|-----------------------|

戻り値

変換行列に m を引いた [GgMatrix](#) 型の値.

gg.h の 2278 行目に定義があります。

呼び出し関係図:



8.5.3.89 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

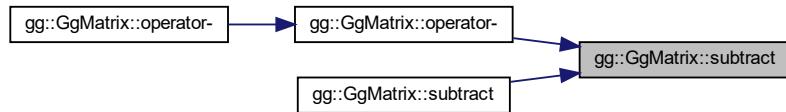
変換行列に a を引いた [GgMatrix](#) 型の値.

gg.h の 2269 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:



8.5.3.90 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

| | |
|----------|---------------------|
| <i>t</i> | 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 2632 行目に定義があります。

呼び出し関係図:



8.5.3.91 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

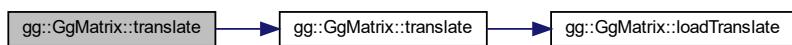
| | |
|----------|--------------------------------------|
| <i>t</i> | 移動量の GLfloat 型の 3 要素の配列変数 (x, y, z). |
|----------|--------------------------------------|

戻り値

平行移動した結果の変換行列.

gg.h の 2624 行目に定義があります。

呼び出し関係図:



8.5.3.92 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

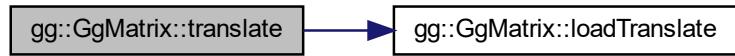
| | |
|----------|---------------------------------|
| <i>x</i> | <i>x</i> 方向の移動量. |
| <i>y</i> | <i>y</i> 方向の移動量. |
| <i>z</i> | <i>z</i> 方向の移動量. |
| <i>w</i> | <i>w</i> 移動量のスケールファクタ (= 1.0f). |

戻り値

平行移動した結果の変換行列.

gg.h の 2615 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:



8.5.3.93 transpose()

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

転置行列を返す。

戻り値

転置行列。

gg.h の 2832 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:



8.5.4 フレンドと関連関数の詳解

8.5.4.1 GgQuaternion

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

gg.h の 2138 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](#)

8.6 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 \(const GLubyte *image, GLsizei width, GLsizei height, GLenum format=GL_RED, GLfloat nz=1.0f, GLenum internal=GL_RGBA\)](#)
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 \(const std::string &name, GLfloat nz=1.0f, GLenum internal=GL_RGBA\)](#)
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 \(const GLubyte *hmap, GLsizei width, GLsizei height, GLenum format=GL_RED, GLfloat nz=1.0f, GLenum internal=GL_RGBA\)](#)
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 \(const std::string &name, GLfloat nz=1.0f, GLenum internal=GL_RGBA\)](#)
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.

8.6.1 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 4545 行目に定義があります。

8.6.2 構築子と解体子

8.6.2.1 GgNormalTexture() [1/3]

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

コンストラクタ.

gg.h の 4553 行目に定義があります。

8.6.2.2 GgNormalTexture() [2/3]

```
gg::GgNormalTexture::GgNormalTexture (
    const GLubyte * image,
    GLsizei width,
    GLsizei height,
    GLenum format = GL_RED,
    GLfloat 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

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

gg.h の 4564 行目に定義があります。

呼び出し関係図:



8.6.2.3 GgNormalTexture() [3/3]

```
gg::GgNormalTexture::GgNormalTexture (
    const std::string & name,
    GLfloat nz = 1.0f,
    GLenum internal = GL_RGBA ) [inline]
```

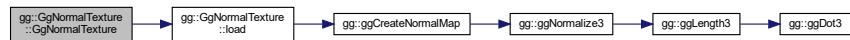
ファイルからデータを読み込んで法線マップのテクスチャを作成するコンストラクタ.

引数

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

gg.h の 4581 行目に定義があります。

呼び出し関係図:



8.6.2.4 ~GgNormalTexture()

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

デストラクタ.

gg.h の 4592 行目に定義があります。

8.6.3 関数詳解

8.6.3.1 `load()` [1/2]

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

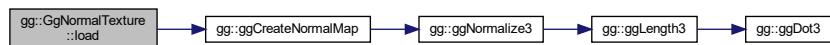
メモリ上のデータから法線マップのテクスチャを作成する。

引数

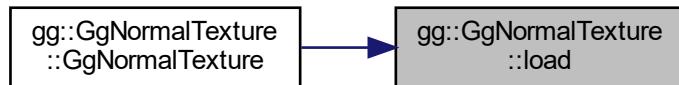
| | |
|-----------------|-----------------------------------------------------------|
| <i>hmap</i> | テクスチャとして用いる画像データ, <code>nullptr</code> ならデータを読み込まない. |
| <i>width</i> | テクスチャとして用いる画像データの横幅. |
| <i>height</i> | テクスチャとして用いる画像データの高さ. |
| <i>format</i> | テクスチャとして用いる画像データのフォーマット (GL_RED, GL_RG, GL_RGB,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

`gg.h` の 4603 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:



8.6.3.2 `load()` [2/2]

```
void gg::GgNormalTexture::load (
    const std::string & name,
    GLfloat nz = 1.0f,
    GLenum internal = GL_RGBA )
```

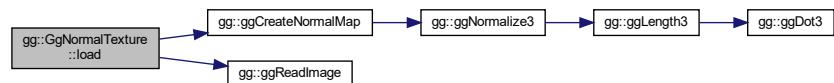
ファイルからデータを読み込んで法線マップのテクスチャを作成する。

引数

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

gg.cpp の 3203 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:

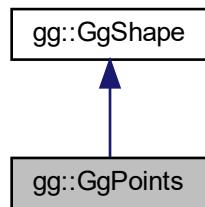
- [gg.h](#)
- [gg.cpp](#)

8.7 gg::GgPoints クラス

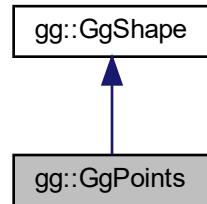
点.

```
#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 count, GLenum mode=GL_POINTS, GLenum usage=GL_STATIC_DRAW)`
コンストラクタ.
- `virtual ~GgPoints ()`
デストラクタ.
- `const GLsizei & getCount () const`
データの数を取り出す.
- `const 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.

8.7.1 詳解

点.

gg.h の 5155 行目に定義があります。

8.7.2 構築子と解体子

8.7.2.1 GgPoints() [1/2]

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

コンストラクタ.

gg.h の 5164 行目に定義があります。

8.7.2.2 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

| | |
|---------------|------------------------------------------|
| <i>pos</i> | 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). |
| <i>countv</i> | 頂点数. |
| <i>mode</i> | 描画する基本図形の種類. |
| <i>usage</i> | バッファオブジェクトの使い方. |

gg.h の 5174 行目に定義があります。

8.7.2.3 ~GgPoints()

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

デストラクタ.

gg.h の 5186 行目に定義があります。

8.7.3 関数詳解

8.7.3.1 draw()

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

点の描画.

引数

| | |
|--------------|-----------------------|
| <i>first</i> | 描画を開始する最初の点の番号. |
| <i>count</i> | 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 5032 行目に定義があります。

呼び出し関係図:



8.7.3.2 `getBuffer()`

```
const 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 5199 行目に定義があります。

8.7.3.3 `getCount()`

```
const 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 5192 行目に定義があります。

8.7.3.4 `load()`

```
void gg::GgPoints::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。

引数

| | |
|--------------|------------------------------|
| <i>pos</i> | 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. |
| <i>count</i> | 頂点のデータの数 (頂点数). |
| <i>usage</i> | バッファオブジェクトの使い方. |

gg.cpp の 5019 行目に定義がります。

8.7.3.5 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

| | |
|--------------|------------------------------------|
| <i>pos</i> | 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. |
| <i>first</i> | 転送先のバッファオブジェクトの先頭の要素番号. |
| <i>count</i> | 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 5208 行目に定義がります。

このクラス詳解は次のファイルから抽出されました:

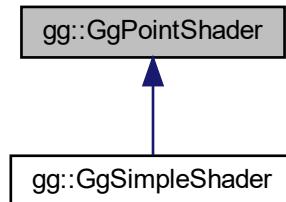
- [gg.h](#)
- [gg.cpp](#)

8.8 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 std::string &vert, const std::string &frag="", const std::string &geom="", GLint nvarying=0, const char *const *varyings=nullptr)**
コンストラクタ.
- **virtual ~GgPointShader ()**
デストラクタ.
- **bool load (const std::string &vert, const std::string &frag="", const std::string &geom="", GLint nvarying=0, const char *const *varyings=nullptr)**
シェーダのソースファイルを読み込む.
- **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.

8.8.1 詳解

点のシェーダ.

gg.h の 5637 行目に定義があります。

8.8.2 構築子と解体子

8.8.2.1 GgPointShader() [1/2]

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

コンストラクタ.

gg.h の 5651 行目に定義があります。

8.8.2.2 GgPointShader() [2/2]

```
gg::GgPointShader::GgPointShader (
    const std::string & vert,
    const std::string & frag = "",
    const std::string & geom = "",
    GLint nvarying = 0,
    const char *const * varyings = nullptr ) [inline]
```

コンストラクタ.

引数

| | |
|-----------------|-----------------------------------|
| <i>vert</i> | パーティクルシェーダのソースファイル名. |
| <i>frag</i> | 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). |
| <i>geom</i> | ジオメトリシェーダのソースファイル名 (0 なら不使用). |
| <i>nvarying</i> | フィードバックする varying 変数の数 (0 なら不使用). |
| <i>varyings</i> | フィードバックする varying 変数のリスト. |

gg.h の 5663 行目に定義があります。

8.8.2.3 ~GgPointShader()

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

デストラクタ.

gg.h の 5676 行目に定義があります。

8.8.3 関数詳解

8.8.3.1 `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 5803 行目に定義があります。

8.8.3.2 `load()`

```
bool gg::GgPointShader::load (
    const std::string & vert,
    const std::string & frag = "",
    const std::string & geom = "",
    GLint nvarying = 0,
    const char *const * varyings = nullptr ) [inline]
```

シェーダのソースファイルを読み込む。

引数

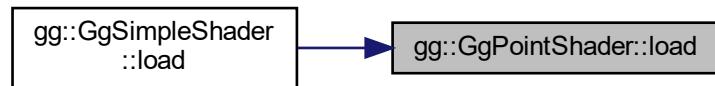
| | |
|-----------------|------------------------------------------|
| <i>vert</i> | バーテックスシェーダのソースファイル名。 |
| <i>frag</i> | フラグメントシェーダのソースファイル名 (0 なら不使用)。 |
| <i>geom</i> | ジオメトリシェーダのソースファイル名 (0 なら不使用)。 |
| <i>nvarying</i> | フィードバックする <i>varying</i> 変数の数 (0 なら不使用)。 |
| <i>varyings</i> | フィードバックする <i>varying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

戻り値

プログラムオブジェクトが作成できれば `true`。

gg.h の 5687 行目に定義があります。

被呼び出し関係図:



8.8.3.3 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。

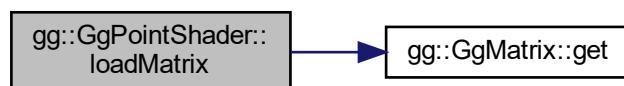
引数

| | |
|-----------|-------------------------------------|
| <i>mp</i> | <code>GgMatrix</code> 型の投影変換行列。 |
| <i>mv</i> | <code>GgMatrix</code> 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 5752 行目に定義があります。

呼び出し関係図:



8.8.3.4 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

| | |
|-----------|----------------------------------------|
| <i>mp</i> | GLfloat 型の 16 要素の配列変数に格納された投影変換行列. |
| <i>mv</i> | 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 5743 行目に定義があります。

8.8.3.5 `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

| | |
|-----------|------------------------|
| <i>mv</i> | 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 5735 行目に定義があります。

呼び出し関係図:



8.8.3.6 `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

| | |
|-----------|----------------------------------------|
| <i>mv</i> | 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 5728 行目に定義があります。

8.8.3.7 loadProjectionMatrix() [1/2]

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

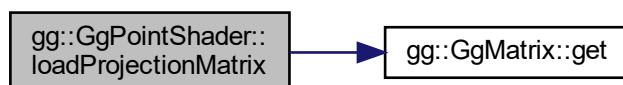
投影変換行列を設定する。

引数

| | |
|-----------|--------------------|
| <i>mp</i> | GgMatrix 型の投影変換行列. |
|-----------|--------------------|

gg.h の 5721 行目に定義があります。

呼び出し関係図:



8.8.3.8 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

| | |
|-----------|------------------------------------|
| <i>mp</i> | 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 5714 行目に定義があります。

8.8.3.9 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 5796 行目に定義があります。

8.8.3.10 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 5758 行目に定義があります。

8.8.3.11 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

| | |
|-----------|------------------------------------|
| <i>mp</i> | GgMatrix 型の投影変換行列。 |
|-----------|------------------------------------|

gg.h の 5773 行目に定義があります。

呼び出し関係図:



8.8.3.12 use() [3/5]

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

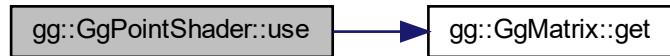
投影変換行列とモデルビューを設定してシェーダプログラムの使用を開始する。

引数

| | |
|-----------|------------------------|
| <i>mp</i> | GgMatrix 型の投影変換行列。 |
| <i>mv</i> | GgMatrix 型のモデルビュー変換行列。 |

gg.h の 5790 行目に定義があります。

呼び出し関係図:



8.8.3.13 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

| | |
|-----------|------------------------------------|
| <i>mp</i> | 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 5765 行目に定義があります。

8.8.3.14 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

| | |
|-----------|----------------------------------------|
| <i>mp</i> | GLfloat 型の 16 要素の配列変数に格納された投影変換行列。 |
| <i>mv</i> | 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 5781 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:

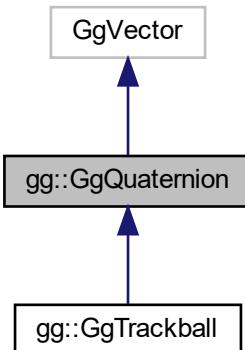
- [gg.h](#)

8.9 gg::GgQuaternion クラス

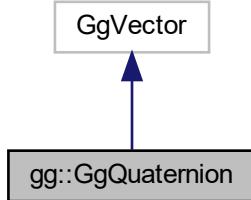
四元数。

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

gg::GgQuaternion の継承関係図



gg::GgQuaternion 連携図



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 \(const GLfloat *a\)](#)
コンストラクタ.
- [GgQuaternion \(const GgVector &v\)](#)
コンストラクタ.
- [GgQuaternion \(const GgQuaternion &q\)](#)
コピー構造関数.
- [~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 \(const GLfloat *a\)](#)
四元数を格納する.
- [GgQuaternion & load \(const GgVector &v\)](#)
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 \(const GLfloat *a\)](#)
四元数に別の四元数を加算した結果を格納する.
- [GgQuaternion & loadAdd \(const GgVector &v\)](#)
四元数に別の四元数を加算した結果を格納する.
- [GgQuaternion & loadAdd \(const GgQuaternion &q\)](#)
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 \(const GLfloat *a\)](#)

- 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 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 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 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 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 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 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 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 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 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 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 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 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 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 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 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 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 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 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 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 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 GgVector &v)`
- `GgQuaternion & operator+= (const GLfloat *a)`
- `GgQuaternion & operator+= (const GgVector &v)`
- `GgQuaternion & operator+= (const GgQuaternion &q)`
- `GgQuaternion & operator-= (const GLfloat *a)`
- `GgQuaternion & operator-= (const GgVector &v)`
- `GgQuaternion & operator-= (const GgQuaternion &q)`
- `GgQuaternion & operator*=(const GLfloat *a)`
- `GgQuaternion & operator*=(const GgVector &v)`
- `GgQuaternion & operator*=(const GgQuaternion &q)`
- `GgQuaternion & operator/=(const GLfloat *a)`
- `GgQuaternion & operator/=(const GgVector &v)`
- `GgQuaternion & operator/=(const GgQuaternion &q)`
- `GgQuaternion operator+ (const GLfloat *a) const`
- `GgQuaternion operator+ (const GgVector &v) const`
- `GgQuaternion operator+ (const GgQuaternion &q) const`
- `GgQuaternion operator- (const GLfloat *a) const`
- `GgQuaternion operator- (const GgVector &v) const`
- `GgQuaternion operator- (const GgQuaternion &q) const`
- `GgQuaternion operator* (const GLfloat *a) const`
- `GgQuaternion operator* (const GgVector &v) const`
- `GgQuaternion operator* (const GgQuaternion &q) const`
- `GgQuaternion operator/ (const GLfloat *a) const`
- `GgQuaternion operator/ (const GgVector &v) 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 軸を中心に角度 a 回転する四元数を格納する.
- `GgQuaternion & loadRotateY (GLfloat a)`
 y 軸を中心に角度 a 回転する四元数を格納する.
- `GgQuaternion & loadRotateZ (GLfloat a)`
 z 軸を中心に角度 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.
- `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.

8.9.1 詳解

四元数.

gg.h の 3183 行目に定義があります。

8.9.2 構築子と解体子

8.9.2.1 GgQuaternion() [1/5]

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

コンストラクタ.

gg.h の 3200 行目に定義があります。

8.9.2.2 GgQuaternion() [2/5]

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

コンストラクタ.

引数

| | |
|----------------|-------------------------|
| <code>x</code> | 四元数の <code>x</code> 要素. |
| <code>y</code> | 四元数の <code>y</code> 要素. |
| <code>z</code> | 四元数の <code>z</code> 要素. |
| <code>w</code> | 四元数の <code>w</code> 要素. |

gg.h の 3209 行目に定義があります。

呼び出し関係図:



8.9.2.3 GgQuaternion() [3/5]

```
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 3216 行目に定義があります。

呼び出し関係図:



8.9.2.4 GgQuaternion() [4/5]

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

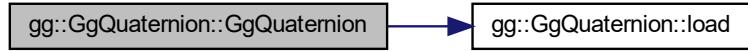
コンストラクタ.

引数

| | |
|---|-------------------------|
| v | 四元数を格納した GgVector 型の変数. |
|---|-------------------------|

gg.h の 3223 行目に定義がります。

呼び出し関係図:



8.9.2.5 GgQuaternion() [5/5]

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

コピー構造関数。

引数

| | |
|----------|---------------------|
| <i>q</i> | GgQuaternion 型の四元数。 |
|----------|---------------------|

gg.h の 3230 行目に定義がります。

呼び出し関係図:



8.9.2.6 ~GgQuaternion()

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

デストラクタ。

gg.h の 3236 行目に定義がります。

8.9.3 関数詳解

8.9.3.1 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

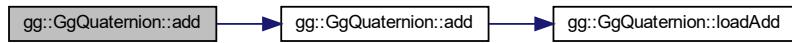
| | |
|----------|---------------------|
| <i>q</i> | GgQuaternion 型の四元数。 |
|----------|---------------------|

戻り値

q を加えた四元数。

gg.h の 3469 行目に定義があります。

呼び出し関係図:



8.9.3.2 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

| | |
|----------|-------------------------|
| <i>v</i> | 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 3461 行目に定義がります。

呼び出し関係図:



8.9.3.3 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

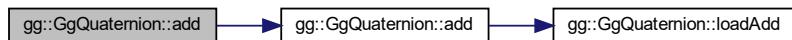
| | |
|----------|--------------------------------|
| <i>a</i> | 四元数を格納した GLfloat 型の 4 要素の配列変数。 |
|----------|--------------------------------|

戻り値

a を加えた四元数。

gg.h の 3453 行目に定義がります。

呼び出し関係図:



8.9.3.4 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

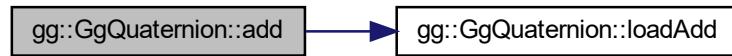
| | |
|----------|----------------------|
| <i>x</i> | 加える四元数の <i>x</i> 要素. |
| <i>y</i> | 加える四元数の <i>y</i> 要素. |
| <i>z</i> | 加える四元数の <i>z</i> 要素. |
| <i>w</i> | 加える四元数の <i>w</i> 要素. |

戻り値

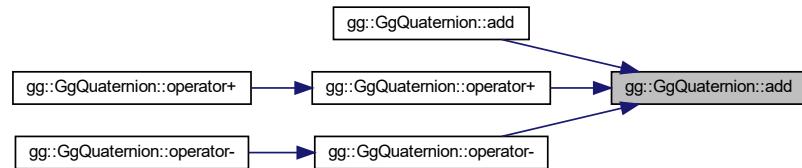
(x, y, z, w) を加えた四元数.

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:



8.9.3.5 conjugate()

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

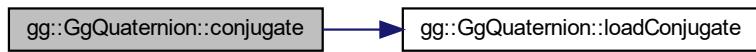
共役四元数に変換する.

戻り値

共役四元数.

gg.h の 3957 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:



8.9.3.6 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

| | |
|----------|---------------------|
| <i>q</i> | GgQuaternion 型の四元数. |
|----------|---------------------|

戻り値

q で割った四元数.

gg.h の 3582 行目に定義があります。

呼び出し関係図:



8.9.3.7 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 3574 行目に定義があります。

呼び出し関係図:



8.9.3.8 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 3563 行目に定義があります。

呼び出し関係図:



8.9.3.9 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

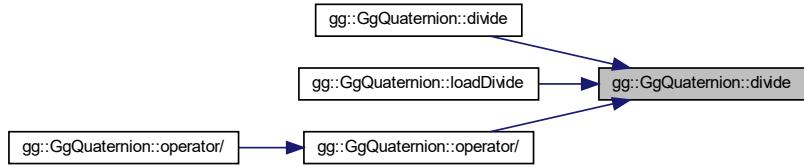
| | |
|---|--------------|
| x | 割る四元数の x 要素。 |
| y | 割る四元数の y 要素。 |
| z | 割る四元数の z 要素。 |
| w | 割る四元数の w 要素。 |

戻り値

(x, y, z, w) を割った四元数。

gg.h の 3554 行目に定義があります。

被呼び出し関係図:



8.9.3.10 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

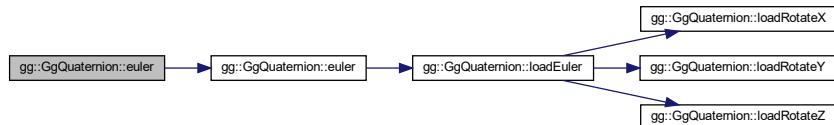
| | |
|----------|-------------------------------------------------------|
| <i>e</i> | オイラー角を表す GLfloat 型の 3 要素の配列変数 (heading, pitch, roll). |
|----------|-------------------------------------------------------|

戻り値

回転した四元数.

gg.h の 3839 行目に定義があります。

呼び出し関係図:



8.9.3.11 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

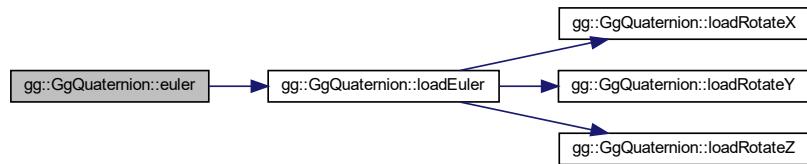
| | |
|----------------|------------|
| <i>heading</i> | y 軸中心の回転角. |
| <i>pitch</i> | x 軸中心の回転角. |
| <i>roll</i> | z 軸中心の回転角. |

戻り値

回転した四元数.

gg.h の 3830 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:



8.9.3.12 get()

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

四元数を取り出す.

引数

| | |
|----------|--------------------------------|
| <i>a</i> | 四元数を格納する GLfloat 型の 4 要素の配列変数. |
|----------|--------------------------------|

gg.h の 3975 行目に定義がります。

8.9.3.13 `getConjugateMatrix()` [1/3]

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

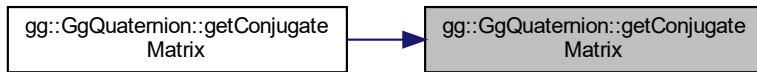
四元数の共役が表す回転の変換行列を取り出す。

戻り値

回転の変換を表す `GgMatrix` 型の変換行列。

gg.h の 4024 行目に定義がります。

被呼び出し関係図:



8.9.3.14 `getConjugateMatrix()` [2/3]

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

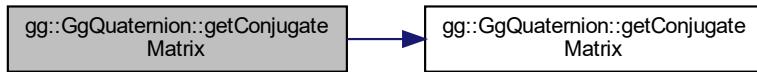
四元数の共役が表す回転の変換行列を `m` に求める。

引数

| | |
|----------------|------------------------------------------|
| <code>m</code> | 回転の変換行列を格納する <code>GgMatrix</code> 型の変数。 |
|----------------|------------------------------------------|

gg.h の 4017 行目に定義がります。

呼び出し関係図:



8.9.3.15 getConjugateMatrix() [3/3]

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

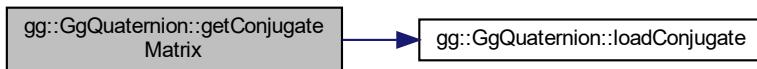
四元数の共役が表す回転の変換行列を a に求める.

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 4008 行目に定義があります。

呼び出し関係図:



8.9.3.16 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 3999 行目に定義があります。

呼び出し関係図:



8.9.3.17 `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

| | |
|----------------|------------------------------------------|
| <code>m</code> | 回転の変換行列を格納する <code>GgMatrix</code> 型の変数. |
|----------------|------------------------------------------|

gg.h の 3992 行目に定義があります。

呼び出し関係図:



8.9.3.18 `getMatrix()` [3/3]

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

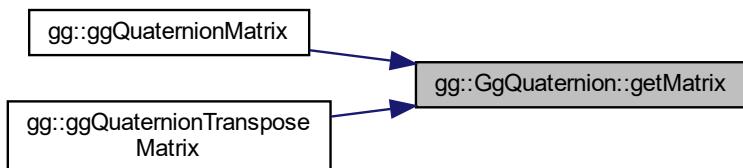
四元数が表す回転の変換行列を `a` に求める.

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 3985 行目に定義があります。

呼び出し関係図:



8.9.3.19 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 3966 行目に定義があります。

呼び出し関係図:



呼び出し関係図:



8.9.3.20 `load()` [1/4]

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

四元数を格納する。

引数

| | |
|----------|----------------------------------|
| <i>q</i> | <code>GgQuaternion</code> 型の四元数。 |
|----------|----------------------------------|

戻り値

設定した四元数。

`gg.h` の 3282 行目に定義があります。

8.9.3.21 `load()` [2/4]

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

四元数を格納する。

引数

| | |
|----------|--------------------------------------|
| <i>v</i> | 四元数を格納した <code>GgVector</code> 型の変数。 |
|----------|--------------------------------------|

戻り値

設定した四元数。

`gg.h` の 3273 行目に定義があります。

8.9.3.22 `load()` [3/4]

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

四元数を格納する。

引数

| | |
|----------|---------------------------------------------|
| <i>a</i> | 四元数を格納した <code>GLfloat</code> 型の 4 要素の配列変数。 |
|----------|---------------------------------------------|

戻り値

設定した四元数.

gg.h の 3265 行目に定義があります。

呼び出し関係図:



8.9.3.23 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

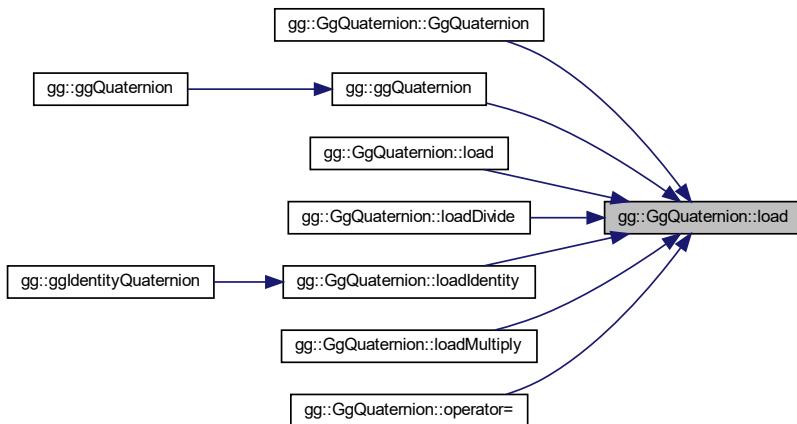
| | |
|----------|------------|
| <i>x</i> | 四元数の x 要素. |
| <i>y</i> | 四元数の y 要素. |
| <i>z</i> | 四元数の z 要素. |
| <i>w</i> | 四元数の w 要素. |

戻り値

設定した四元数.

gg.h の 3253 行目に定義があります。

被呼び出し関係図:



8.9.3.24 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

| | |
|----------|---------------------|
| <i>q</i> | GgQuaternion 型の四元数. |
|----------|---------------------|

戻り値

q を加えた四元数.

gg.h の 3322 行目に定義があります。

呼び出し関係図:



8.9.3.25 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 3314 行目に定義があります。

呼び出し関係図:



8.9.3.26 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 3306 行目に定義があります。

呼び出し関係図:



8.9.3.27 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

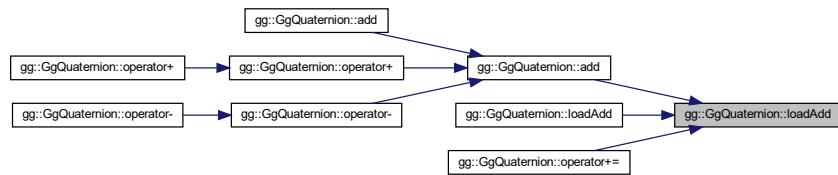
| | |
|---|---------------|
| x | 加える四元数の x 要素. |
| y | 加える四元数の y 要素. |
| z | 加える四元数の z 要素. |
| w | 加える四元数の w 要素. |

戻り値

(x, y, z, w) を加えた四元数.

gg.h の 3294 行目に定義があります。

被呼び出し関係図:



8.9.3.28 loadConjugate() [1/2]

```
GgQuaternion& gg::GgQuaternion::loadConjugate (
```

引数に指定した四元数の共役四元数を格納する。

引数

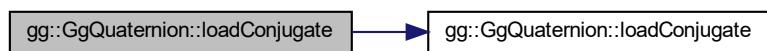
q GgQuaternion 型の四元数.

戻り値

共役四元数.

gg.h の 3906 行目に定義があります。

呼び出し関係図:



8.9.3.29 loadConjugate() [2/2]

```
gg:::GgQuaternion & gg:::GgQuaternion::loadConjugate (
```

引数に指定した四元数の共役四元数を格納する。

引数

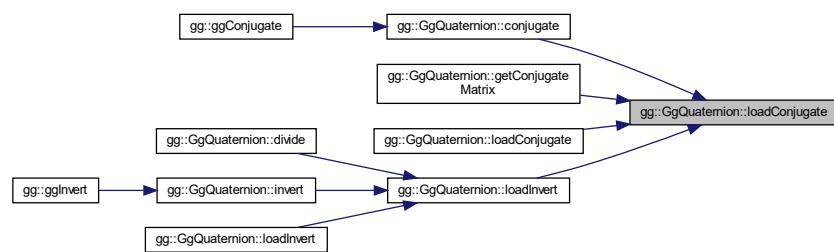
| | |
|----------------|---------------------------------------------|
| <code>a</code> | 四元数を格納した <code>GLfloat</code> 型の 4 要素の配列変数. |
|----------------|---------------------------------------------|

戻り値

共役四元数.

`gg.cpp` の 4871 行目に定義があります。

被呼び出し関係図:



8.9.3.30 `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

| | |
|----------------|----------------------------------|
| <code>q</code> | <code>GgQuaternion</code> 型の四元数. |
|----------------|----------------------------------|

戻り値

`q` で割った四元数.

`gg.h` の 3433 行目に定義があります。

呼び出し関係図:



8.9.3.31 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 3425 行目に定義があります。

呼び出し関係図:



8.9.3.32 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

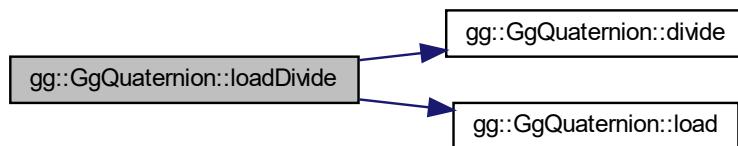
| | |
|---|--------------------------------|
| a | 四元数を格納した GLfloat 型の 4 要素の配列変数. |
|---|--------------------------------|

戻り値

a で割った四元数.

gg.h の 3417 行目に定義があります。

呼び出し関係図:



8.9.3.33 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

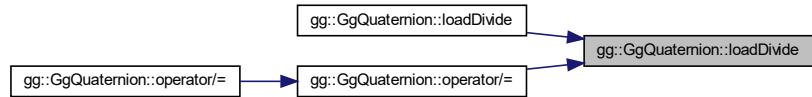
| | |
|---|--------------|
| x | 割る四元数の x 要素. |
| y | 割る四元数の y 要素. |
| z | 割る四元数の z 要素. |
| w | 割る四元数の w 要素. |

戻り値

(x, y, z, w) を割った四元数.

gg.h の 3408 行目に定義があります。

被呼び出し関係図:



8.9.3.34 loadEuler() [1/2]

```
gg::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 3820 行目に定義があります。

呼び出し関係図:



8.9.3.35 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

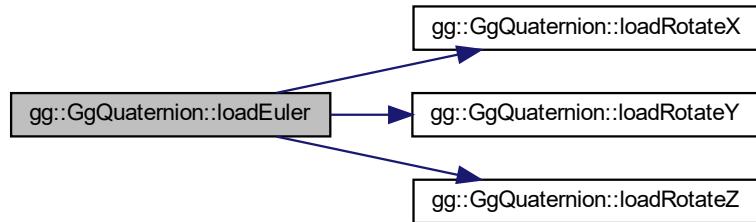
| | |
|----------------|------------|
| <i>heading</i> | y 軸中心の回転角. |
| <i>pitch</i> | x 軸中心の回転角. |
| <i>roll</i> | z 軸中心の回転角. |

戻り値

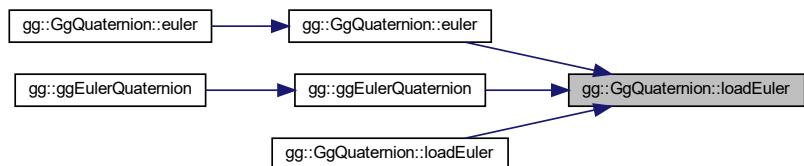
格納した回転を表す四元数.

gg.cpp の 4840 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:



8.9.3.36 loadIdentity()

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

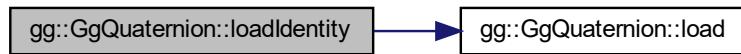
単位元を格納する.

戻り値

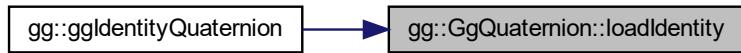
格納された単位元.

gg.h の 3712 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:



8.9.3.37 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

| | |
|----------|---------------------|
| <i>q</i> | GgQuaternion 型の四元数. |
|----------|---------------------|

戻り値

四元数の逆元.

gg.h の 3919 行目に定義があります。

呼び出し関係図:



8.9.3.38 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

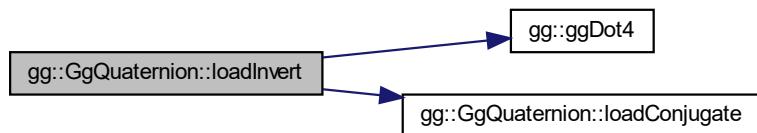
| | |
|----------|--------------------------------|
| <i>a</i> | 四元数を格納した GLfloat 型の 4 要素の配列変数。 |
|----------|--------------------------------|

戻り値

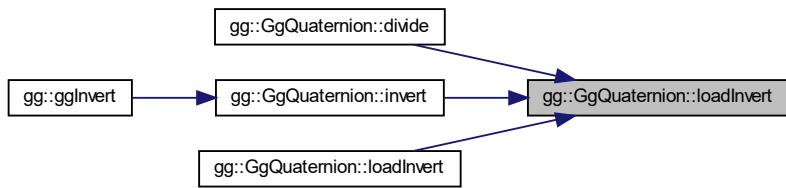
四元数の逆元。

gg.cpp の 4885 行目に定義があります。

呼び出し関係図:



呼び出し関係図:



8.9.3.39 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

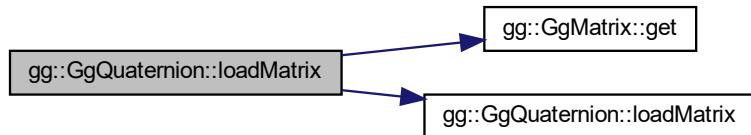
| | |
|-----|------------------|
| m | Ggmatrix 型の変換行列。 |
|-----|------------------|

戻り値

m による回転の変換に相当する四元数。

gg.h の 3705 行目に定義があります。

呼び出し関係図:



8.9.3.40 `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

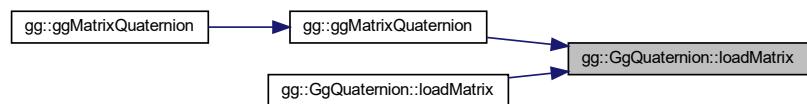
| | |
|---|------------------------|
| a | GLfloat 型の 16 要素の変換行列. |
|---|------------------------|

戻り値

a による回転の変換に相当する四元数.

gg.h の 3696 行目に定義があります。

呼び出し関係図:



8.9.3.41 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 3397 行目に定義があります。

呼び出し関係図:



8.9.3.42 `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

| | |
|----------------|--------------------------------------|
| <code>v</code> | 四元数を格納した <code>GgVector</code> 型の変数。 |
|----------------|--------------------------------------|

戻り値

`v` を乗じた四元数。

`gg.h` の 3389 行目に定義があります。

呼び出し関係図:



8.9.3.43 `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

| | |
|----------------|---------------------------------------------|
| <code>a</code> | 四元数を格納した <code>GLfloat</code> 型の 4 要素の配列変数。 |
|----------------|---------------------------------------------|

戻り値

`a` を乗じた四元数。

`gg.h` の 3381 行目に定義があります。

呼び出し関係図:



8.9.3.44 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

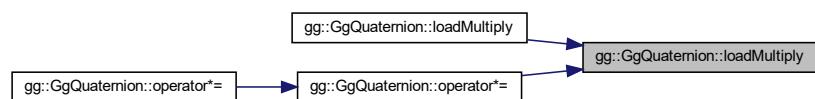
| | |
|----------|----------------------|
| <i>x</i> | 掛ける四元数の <i>x</i> 要素. |
| <i>y</i> | 掛ける四元数の <i>y</i> 要素. |
| <i>z</i> | 掛ける四元数の <i>z</i> 要素. |
| <i>w</i> | 掛ける四元数の <i>w</i> 要素. |

戻り値

(x, y, z, w) を掛けた四元数.

gg.h の 3372 行目に定義があります。

被呼び出し関係図:



8.9.3.45 `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

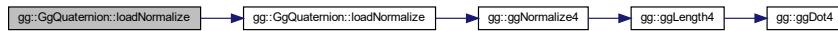
| | |
|----------|----------------------------------|
| <i>q</i> | <code>GgQuaternion</code> 型の四元数。 |
|----------|----------------------------------|

戻り値

正規化された四元数。

`gg.h` の 3893 行目に定義があります。

呼び出し関係図:



8.9.3.46 `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

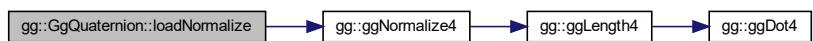
| | |
|----------|---------------------------------------------|
| <i>a</i> | 四元数を格納した <code>GLfloat</code> 型の 4 要素の配列変数。 |
|----------|---------------------------------------------|

戻り値

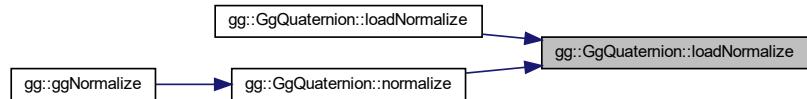
正規化された四元数。

`gg.cpp` の 4856 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:



8.9.3.47 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 3737 行目に定義があります。

呼び出し関係図:



8.9.3.48 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

| | |
|----------|--------------------------------|
| <i>v</i> | 軸ベクトルを表す GLfloat 型の 3 要素の配列変数. |
| <i>a</i> | 回転角. |

戻り値

格納された回転を表す四元数.

gg.h の 3729 行目に定義があります。

呼び出し関係図:



8.9.3.49 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

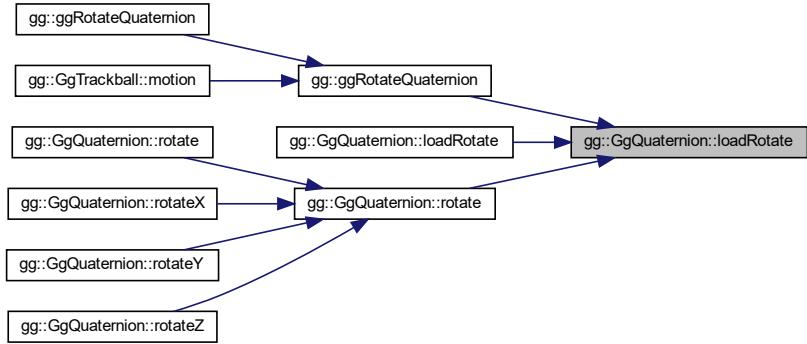
| | |
|----------|---------------------|
| <i>x</i> | 軸ベクトルの <i>x</i> 成分. |
| <i>y</i> | 軸ベクトルの <i>y</i> 成分. |
| <i>z</i> | 軸ベクトルの <i>z</i> 成分. |
| <i>a</i> | 回転角. |

戻り値

格納された回転を表す四元数.

gg.cpp の 4774 行目に定義があります。

被呼び出し関係図:



8.9.3.50 `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

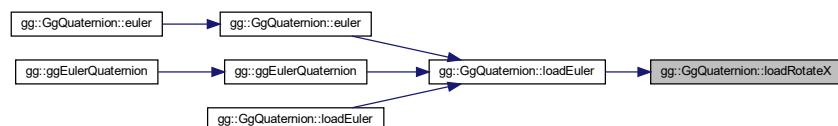
| | |
|-----|------|
| a | 回転角. |
|-----|------|

戻り値

格納された回転を表す四元数。

`gg.cpp` の 4798 行目に定義があります。

被呼び出し関係図:



8.9.3.51 `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

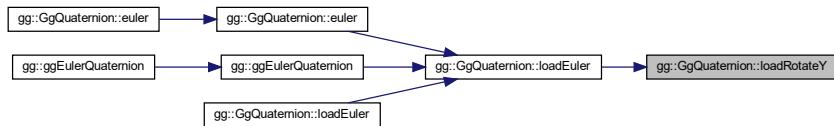
| | |
|-----|------|
| a | 回転角。 |
|-----|------|

戻り値

格納された回転を表す四元数。

`gg.cpp` の 4812 行目に定義があります。

被呼び出し関係図:



8.9.3.52 `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

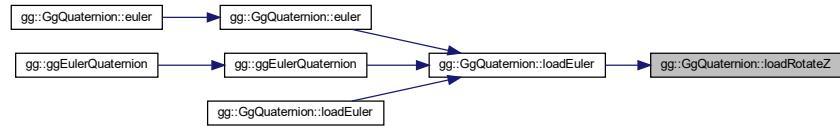
| | |
|-----|------|
| a | 回転角。 |
|-----|------|

戻り値

格納された回転を表す四元数。

`gg.cpp` の 4826 行目に定義があります。

被呼び出し関係図:



8.9.3.53 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

| | |
|----------|----------------------------|
| <i>q</i> | <i>GgQuaternion</i> 型の四元数。 |
| <i>r</i> | <i>GgQuaternion</i> 型の四元数。 |
| <i>t</i> | 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 3860 行目に定義があります。

呼び出し関係図:



8.9.3.54 `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

| | |
|----------|---------------------------------------------|
| <i>q</i> | <code>GgQuaternion</code> 型の四元数。 |
| <i>a</i> | 四元数を格納した <code>GLfloat</code> 型の 4 要素の配列変数。 |
| <i>t</i> | 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 3870 行目に定義があります。

呼び出し関係図:



8.9.3.55 `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

| | |
|----------|---------------------------------------------|
| <i>a</i> | 四元数を格納した <code>GLfloat</code> 型の 4 要素の配列変数。 |
| <i>q</i> | <code>GgQuaternion</code> 型の四元数。 |
| <i>t</i> | 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 3880 行目に定義があります。

呼び出し関係図:



8.9.3.56 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

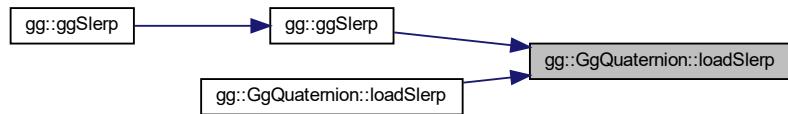
| | |
|----------|--------------------------------|
| <i>a</i> | 四元数を格納した GLfloat 型の 4 要素の配列変数. |
| <i>b</i> | 四元数を格納した GLfloat 型の 4 要素の配列変数. |
| <i>t</i> | 補間パラメータ. |

戻り値

格納した a, b を t で内分した四元数.

gg.h の 3849 行目に定義があります。

被呼び出し関係図:



8.9.3.57 `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

| | |
|----------------|----------------------------------|
| <code>q</code> | <code>GgQuaternion</code> 型の四元数。 |
|----------------|----------------------------------|

戻り値

`q` を引いた四元数。

`gg.h` の 3361 行目に定義があります。

呼び出し関係図:



8.9.3.58 `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

| | |
|----------------|--------------------------------------|
| <code>v</code> | 四元数を格納した <code>GgVector</code> 型の変数。 |
|----------------|--------------------------------------|

戻り値

`v` を引いた四元数。

`gg.h` の 3353 行目に定義があります。

呼び出し関係図:



8.9.3.59 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

| | |
|----------|--------------------------------|
| <i>a</i> | 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 3345 行目に定義があります。

呼び出し関係図:



8.9.3.60 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

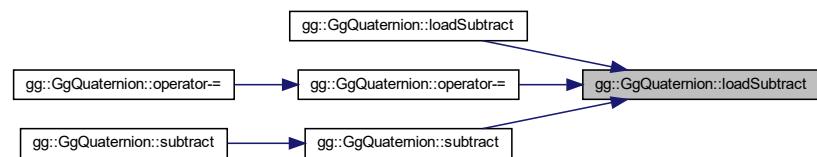
| | |
|----------|---------------------|
| <i>x</i> | 引く四元数の <i>x</i> 要素. |
| <i>y</i> | 引く四元数の <i>y</i> 要素. |
| <i>z</i> | 引く四元数の <i>z</i> 要素. |
| <i>w</i> | 引く四元数の <i>w</i> 要素. |

戻り値

(x, y, z, w) を引いた四元数.

gg.h の 3333 行目に定義があります。

被呼び出し関係図:



8.9.3.61 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

| | |
|----------|---------------------|
| <i>q</i> | GgQuaternion 型の四元数. |
|----------|---------------------|

戻り値

q を掛けた四元数.

gg.h の 3543 行目に定義があります。

8.9.3.62 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 3535 行目に定義がります。

8.9.3.63 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 3525 行目に定義がります。

8.9.3.64 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

| | |
|----------|----------------------|
| <i>x</i> | 掛ける四元数の <i>x</i> 要素. |
| <i>y</i> | 掛ける四元数の <i>y</i> 要素. |
| <i>z</i> | 掛ける四元数の <i>z</i> 要素. |
| <i>w</i> | 掛ける四元数の <i>w</i> 要素. |

戻り値

(x, y, z, w) を掛けた四元数.

gg.h の 3516 行目に定義があります。

8.9.3.65 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 3242 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:



8.9.3.66 normalize()

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

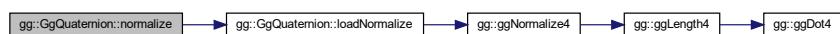
正規化する。

戻り値

正規化された四元数。

gg.h の 3948 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:



8.9.3.67 operator*() [1/3]

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

gg.h の 3676 行目に定義があります。

呼び出し関係図:



8.9.3.68 operator*() [2/3]

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

gg.h の 3672 行目に定義があります。

8.9.3.69 operator*() [3/3]

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

gg.h の 3668 行目に定義があります。

被呼び出し関係図:

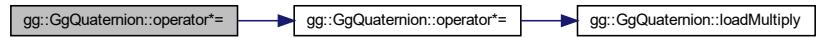


8.9.3.70 operator*=(()) [1/3]

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

gg.h の 3628 行目に定義があります。

呼び出し関係図:



8.9.3.71 operator*=() [2/3]

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

gg.h の 3624 行目に定義があります。

呼び出し関係図:



8.9.3.72 operator*=() [3/3]

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

gg.h の 3620 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:

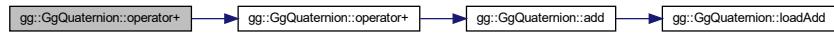


8.9.3.73 operator+() [1/3]

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

gg.h の 3652 行目に定義がります。

呼び出し関係図:

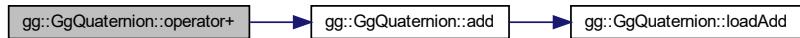


8.9.3.74 operator+() [2/3]

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

gg.h の 3648 行目に定義がります。

呼び出し関係図:

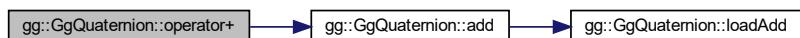


8.9.3.75 operator+() [3/3]

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

gg.h の 3644 行目に定義がります。

呼び出し関係図:



被呼び出し関係図:



8.9.3.76 operator+=() [1/3]

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

gg.h の 3604 行目に定義があります。

呼び出し関係図:



8.9.3.77 operator+=() [2/3]

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

gg.h の 3600 行目に定義があります。

呼び出し関係図:

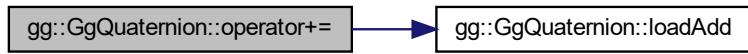


8.9.3.78 operator+=() [3/3]

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

gg.h の 3596 行目に定義があります。

呼び出し関係図:



8.9.3.79 operator-() [1/3]

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

gg.h の 3664 行目に定義があります。

呼び出し関係図:



8.9.3.80 operator-() [2/3]

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

gg.h の 3660 行目に定義があります。

呼び出し関係図:

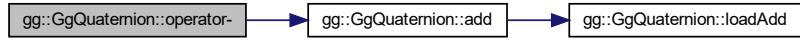


8.9.3.81 operator-() [3/3]

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

gg.h の 3656 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:

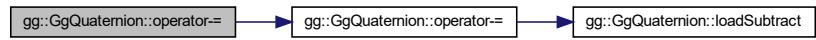


8.9.3.82 operator-=() [1/3]

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

gg.h の 3616 行目に定義があります。

呼び出し関係図:



8.9.3.83 operator-() [2/3]

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

gg.h の 3612 行目に定義があります。

呼び出し関係図:



8.9.3.84 operator-() [3/3]

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

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:

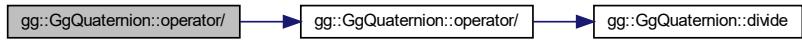


8.9.3.85 operator/() [1/3]

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

gg.h の 3688 行目に定義があります。

呼び出し関係図:

**8.9.3.86 operator/() [2/3]**

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

gg.h の 3684 行目に定義があります。

呼び出し関係図:

**8.9.3.87 operator/() [3/3]**

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

gg.h の 3680 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:



8.9.3.88 operator/() [1/3]

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

gg.h の 3640 行目に定義があります。

呼び出し関係図:



8.9.3.89 operator/() [2/3]

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

gg.h の 3636 行目に定義があります。

呼び出し関係図:



8.9.3.90 operator/() [3/3]

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

gg.h の 3632 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:

**8.9.3.91 operator=() [1/2]**

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

gg.h の 3592 行目に定義があります。

呼び出し関係図:



8.9.3.92 operator=() [2/2]

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

gg.h の 3588 行目に定義があります。

呼び出し関係図:



8.9.3.93 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

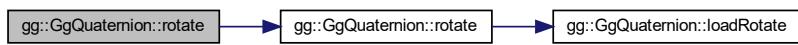
| | |
|---|--------------------------------|
| v | 軸ベクトルを表す GLfloat 型の 4 要素の配列変数。 |
|---|--------------------------------|

戻り値

回転した四元数。

gg.h の 3781 行目に定義があります。

呼び出し関係図:



8.9.3.94 **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

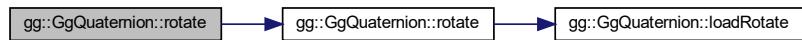
| | |
|-----|--------------------------------|
| v | 軸ベクトルを表す GLfloat 型の 3 要素の配列変数. |
| a | 回転角. |

戻り値

回転した四元数.

gg.h の 3773 行目に定義があります。

呼び出し関係図:

8.9.3.95 **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

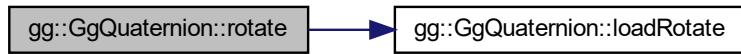
| | |
|-----|--------------|
| x | 軸ベクトルの x 成分. |
| y | 軸ベクトルの y 成分. |
| z | 軸ベクトルの z 成分. |
| a | 回転角. |

戻り値

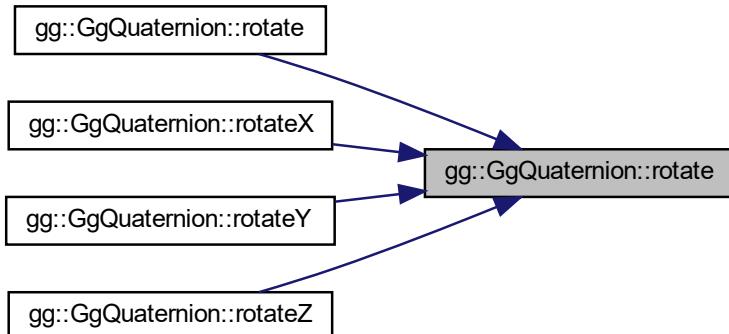
回転した四元数.

gg.h の 3763 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:



8.9.3.96 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

| | |
|----------|------|
| <i>a</i> | 回転角. |
|----------|------|

戻り値

回転した四元数.

gg.h の 3789 行目に定義があります。

呼び出し関係図:



8.9.3.97 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

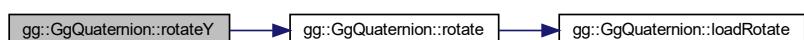
| | |
|---|------|
| a | 回転角. |
|---|------|

戻り値

回転した四元数.

gg.h の 3797 行目に定義があります。

呼び出し関係図:



8.9.3.98 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

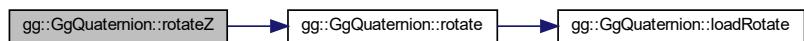
| | |
|---|------|
| a | 回転角. |
|---|------|

戻り値

回転した四元数.

gg.h の 3805 行目に定義があります。

呼び出し関係図:



8.9.3.99 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 3939 行目に定義があります。

8.9.3.100 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

| | |
|----------|--------------------------------|
| <i>a</i> | 四元数を格納した GLfloat 型の 4 要素の配列変数. |
| <i>t</i> | 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 3928 行目に定義があります。

8.9.3.101 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

| | |
|----------|---------------------|
| <i>q</i> | GgQuaternion 型の四元数. |
|----------|---------------------|

戻り値

q を引いた四元数.

gg.h の 3505 行目に定義があります。

呼び出し関係図:



8.9.3.102 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

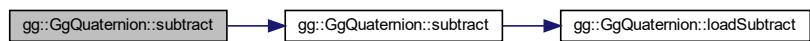
| | |
|----------|-------------------------|
| <i>v</i> | 四元数を格納した GgVector 型の変数. |
|----------|-------------------------|

戻り値

v を引いた四元数.

gg.h の 3497 行目に定義がります。

呼び出し関係図:



8.9.3.103 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

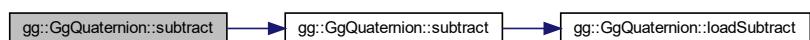
| | |
|----------|--------------------------------|
| <i>a</i> | 四元数を格納した GLfloat 型の 4 要素の配列変数. |
|----------|--------------------------------|

戻り値

a を引いた四元数.

gg.h の 3489 行目に定義がります。

呼び出し関係図:



8.9.3.104 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

| | |
|----------|---------------------|
| <i>x</i> | 引く四元数の <i>x</i> 要素。 |
| <i>y</i> | 引く四元数の <i>y</i> 要素。 |
| <i>z</i> | 引く四元数の <i>z</i> 要素。 |
| <i>w</i> | 引く四元数の <i>w</i> 要素。 |

戻り値

(x, y, z, w) を引いた四元数。

gg.h の 3480 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](#)

8.10 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 std::string &vert, const std::string &frag="", const std::string &geom="", int nvarying=0, const char *const *varyings=nullptr)

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。

8.10.1 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 5576 行目に定義があります。

8.10.2 構築子と解体子

8.10.2.1 GgShader() [1/2]

```
gg::GgShader::GgShader (
    const std::string & vert,
    const std::string & frag = "",
    const std::string & geom = "",
    int nvarying = 0,
    const char *const * varyings = nullptr ) [inline]
```

コンストラクタ。

引数

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

gg.h の 5589 行目に定義があります。

8.10.2.2 ~GgShader()

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

デストラクタ.

gg.h の 5601 行目に定義があります。

8.10.2.3 GgShader() [2/2]

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

コピー構造操作子は使用禁止.

8.10.3 関数詳解

8.10.3.1 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 5628 行目に定義があります。

8.10.3.2 operator=()

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

代入演算子は使用禁止。

8.10.3.3 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 5621 行目に定義があります。

8.10.3.4 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 5615 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:

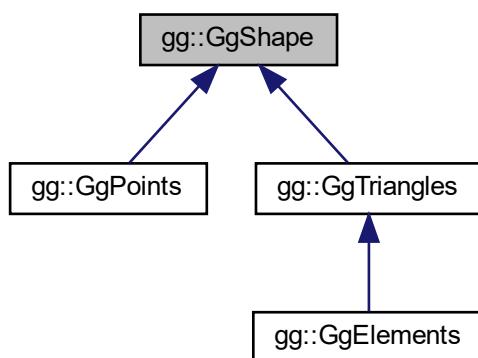
- [gg.h](#)

8.11 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.
- **const 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.
- **const 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.

8.11.1 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 5090 行目に定義があります。

8.11.2 構築子と解体子

8.11.2.1 GgShape() [1/2]

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

コンストラクタ.

引数

| | |
|-------------|----------|
| <i>mode</i> | 基本図形の種類. |
|-------------|----------|

gg.h の 5102 行目に定義があります。

8.11.2.2 ~GgShape()

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

デストラクタ。

gg.h の 5110 行目に定義があります。

8.11.2.3 GgShape() [2/2]

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

コピー構造関数は使用禁止。

8.11.3 関数詳解

8.11.3.1 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。

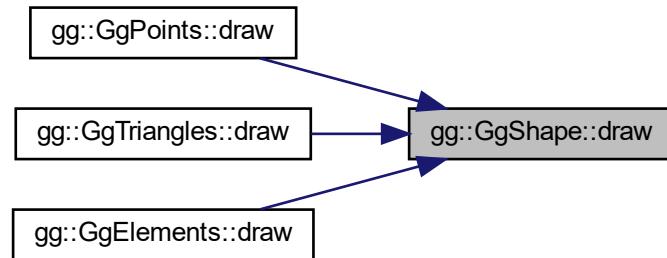
引数

| | |
|--------------|-------------------------------|
| <i>first</i> | 描画する最初のアイテム。 |
| <i>count</i> | 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 5146 行目に定義があります。

被呼び出し関係図:



8.11.3.2 get()

const 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 5124 行目に定義があります。

被呼び出し関係図:



8.11.3.3 getMode()

const 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 5138 行目に定義があります。

8.11.3.4 operator=()

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

代入演算子は使用禁止。

8.11.3.5 setMode()

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

基本図形の設定。

引数

| | |
|-------------|----------|
| <i>mode</i> | 基本図形の種類。 |
|-------------|----------|

gg.h の 5131 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](#)

8.12 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 std::string &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。

8.12.1 詳解

Wavefront OBJ 形式のファイル (Arrays 形式)。

gg.h の 6538 行目に定義があります。

8.12.2 構築子と解体子

8.12.2.1 GgSimpleObj()

```
gg::GgSimpleObj::GgSimpleObj (
    const std::string & name,
    bool normalize = false )
```

コンストラクタ。

引数

| | |
|------------------|-----------------------------------|
| <i>name</i> | 三角形分割された Alias OBJ 形式のファイルのファイル名。 |
| <i>normalize</i> | true なら図形のサイズを [-1, 1] に正規化する。 |

gg.cpp の 5814 行目に定義があります。

呼び出し関係図:



8.12.2.2 ~GgSimpleObj()

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

デストラクタ。

gg.h の 6557 行目に定義があります。

8.12.3 関数詳解

8.12.3.1 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

| | |
|--------------|---------------------------------|
| <i>first</i> | 描画する最初のパーティクル番号. |
| <i>count</i> | 描画するパーティクルの数, 0 なら全部のパーティクルを描く. |

gg.cpp の 5842 行目に定義があります。

8.12.3.2 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 6563 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:

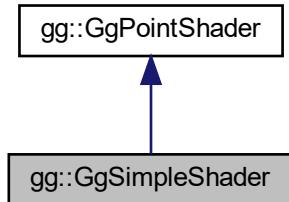
- `gg.h`
- `gg.cpp`

8.13 gg::GgSimpleShader クラス

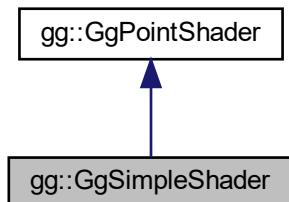
三角形に単純な陰影付けを行うシェーダ.

```
#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 \(const std::string &vert, const std::string &frag="", const std::string &geom="", GLint nvarying=0, const char *const *varyings=nullptr\)](#)
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)`
代入演算子.
- `bool load (const std::string &vert, const std::string &frag="", const std::string &geom="", GLint nvarying=0, const char *const *varyings=nullptr)`
シェーダのソースプログラムの文字列からプログラムオブジェクトを作成する.
- `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.

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.

8.13.1 詳解

三角形に単純な陰影付けを行うシェーダ.

gg.h の 5812 行目に定義があります。

8.13.2 構築子と解体子

8.13.2.1 GgSimpleShader() [1/3]

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

コンストラクタ.

gg.h の 5827 行目に定義があります。

8.13.2.2 GgSimpleShader() [2/3]

```
gg::GgSimpleShader::GgSimpleShader (
    const std::string & vert,
    const std::string & frag = "",
    const std::string & geom = "",
    GLint nvarying = 0,
    const char *const * varyings = nullptr ) [inline]
```

コンストラクタ.

引数

| | |
|-----------------|----------------------------------|
| <i>vert</i> | パーティクルシェーダのソースファイル名. |
| <i>frag</i> | 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). |
| <i>geom</i> | ジオメトリシェーダのソースファイル名(0 なら不使用). |
| <i>nvarying</i> | フィードバックする varying 変数の数(0 なら不使用). |
| <i>varyings</i> | フィードバックする varying 変数のリスト. |

gg.h の 5841 行目に定義があります。

8.13.2.3 GgSimpleShader() [3/3]

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

コピー コンストラクタ.

gg.h の 5853 行目に定義があります。

8.13.2.4 ~GgSimpleShader()

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

デストラクタ.

gg.h の 5862 行目に定義があります。

8.13.3 関数詳解

8.13.3.1 load()

```
bool gg::GgSimpleShader::load (
    const std::string & vert,
    const std::string & frag = "",
    const std::string & geom = "",
    GLint nvarying = 0,
    const char *const * varyings = nullptr )
```

シェーダのソースプログラムの文字列からプログラムオブジェクトを作成する.

引数

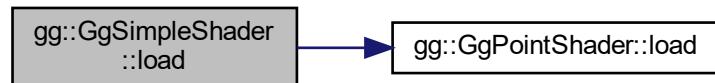
| | |
|-----------------|------------------------------------------------------|
| <i>vert</i> | バーテックスシェーダのソースプログラムの文字列. |
| <i>frag</i> | フラグメントシェーダのソースプログラムの文字列. |
| <i>geom</i> | ジオメトリシェーダのソースプログラムの文字列. |
| <i>nvarying</i> | Transform Feedback に使う <i>varying</i> 変数の数. |
| <i>varyings</i> | Transform Feedback に使う <i>varying</i> 変数の変数名の文字列の配列. |

戻り値

プログラムオブジェクトの作成に成功したら true.

gg.cpp の 5797 行目に定義があります。

呼び出し関係図:



8.13.3.2 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

| | |
|-----------|-------------------------------------|
| <i>mp</i> | <code>GgMatrix</code> 型の投影変換行列。 |
| <i>mv</i> | <code>GgMatrix</code> 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 5956 行目に定義があります。

呼び出し関係図:



8.13.3.3 `loadMatrix()` [2/4]

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

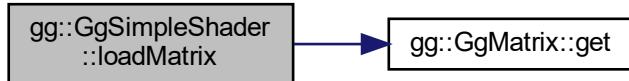
投影変換行列とモデルビュー変換行列と法線変換行列を設定する。

引数

| | |
|-----------|--------------------------------------------|
| <i>mp</i> | <code>GgMatrix</code> 型の投影変換行列。 |
| <i>mv</i> | <code>GgMatrix</code> 型のモデルビュー変換行列。 |
| <i>mn</i> | <code>GgMatrix</code> 型のモデルビュー変換行列の法線変換行列。 |

`gg.h` の 5940 行目に定義があります。

呼び出し関係図:



8.13.3.4 `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

| | |
|-----------|-----------------------------------------------------|
| <i>mp</i> | <code>GLfloat</code> 型の 16 要素の配列変数に格納された投影変換行列。 |
| <i>mv</i> | <code>GLfloat</code> 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 5948 行目に定義があります。

8.13.3.5 **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

| | |
|-----------|-----------------------------------------------|
| <i>mp</i> | GLfloat 型の 16 要素の配列変数に格納された投影変換行列. |
| <i>mv</i> | 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. |
| <i>mn</i> | 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 5930 行目に定義があります。

8.13.3.6 **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。

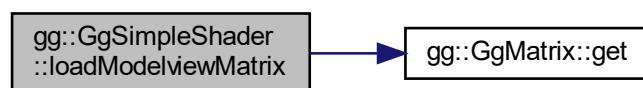
引数

| | |
|-----------|------------------------|
| <i>mv</i> | 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 5921 行目に定義があります。

呼び出し関係図:



8.13.3.7 `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

| | |
|-----------------|--------------------------------------------|
| <code>mv</code> | <code>GgMatrix</code> 型のモデルビュー変換行列。 |
| <code>mn</code> | <code>GgMatrix</code> 型のモデルビュー変換行列の法線変換行列。 |

gg.h の 5907 行目に定義があります。

呼び出し関係図:



8.13.3.8 `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

| | |
|-----------------|-----------------------------------------------------|
| <code>mv</code> | <code>GLfloat</code> 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 5914 行目に定義があります。

8.13.3.9 `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

| | |
|-----------|-----------------------------------------------|
| <i>mv</i> | 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。 |
| <i>mn</i> | 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 5898 行目に定義があります。

8.13.3.10 operator=()

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

代入演算子。

gg.h の 5867 行目に定義があります。

8.13.3.11 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 6341 行目に定義があります。

8.13.3.12 use() [2/13]

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

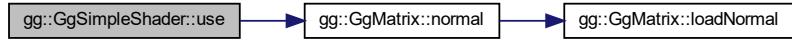
投影変換行列とモデルビュー変換行列を設定しモデルビュー変換行列から求めた法線変換行列を設定してシェーダプログラムの使用を開始する。

引数

| | |
|-----------|------------------------|
| <i>mp</i> | GgMatrix 型の投影変換行列。 |
| <i>mv</i> | GgMatrix 型のモデルビュー変換行列。 |

gg.h の 6380 行目に定義がります。

呼び出し関係図:



8.13.3.13 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

| | |
|-----------|--------------------------------------------|
| <i>mp</i> | <code>GgMatrix</code> 型の投影変換行列。 |
| <i>mv</i> | <code>GgMatrix</code> 型のモデルビュー変換行列。 |
| <i>mn</i> | <code>GgMatrix</code> 型のモデルビュー変換行列の法線変換行列。 |

gg.h の 6364 行目に定義がります。

呼び出し関係図:



8.13.3.14 use() [4/13]

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

```
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

| | |
|--------------|--------------------------------------------|
| <i>mp</i> | <code>GgMatrix</code> 型の投影変換行列. |
| <i>mv</i> | <code>GgMatrix</code> 型のモデルビュー変換行列. |
| <i>mn</i> | <code>GgMatrix</code> 型のモデルビュー変換行列の法線変換行列. |
| <i>light</i> | 光源の特性の <code>gg::LightBuffer</code> 構造体. |
| <i>i</i> | 光源データの uniform block のインデックス. |

gg.h の 6432 行目に定義があります。

8.13.3.15 `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

| | |
|--------------|------------------------------------------|
| <i>mp</i> | <code>GgMatrix</code> 型の投影変換行列. |
| <i>mv</i> | <code>GgMatrix</code> 型のモデルビュー変換行列. |
| <i>light</i> | 光源の特性の <code>gg::LightBuffer</code> 構造体. |
| <i>i</i> | 光源データの uniform block のインデックス. |

gg.h の 6463 行目に定義があります。

8.13.3.16 `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

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

gg.h の 6490 行目に定義があります。

8.13.3.17 `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

| | |
|-----------|----------------------------------------|
| <i>mp</i> | GLfloat 型の 16 要素の配列変数に格納された投影変換行列. |
| <i>mv</i> | 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 6372 行目に定義があります。

8.13.3.18 `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

| | |
|-----------|-----------------------------------------------|
| <i>mp</i> | GLfloat 型の 16 要素の配列変数に格納された投影変換行列. |
| <i>mv</i> | 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. |
| <i>mn</i> | 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 6351 行目に定義があります。

8.13.3.19 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

| | |
|--------------|-----------------------------------------------|
| <i>mp</i> | GLfloat 型の 16 要素の配列変数に格納された投影変換行列。 |
| <i>mv</i> | 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。 |
| <i>mn</i> | 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。 |
| <i>light</i> | 光源の特性の gg::LightBuffer 構造体のポインタ。 |
| <i>i</i> | 光源データの uniform block のインデックス。 |

gg.h の 6411 行目に定義があります。

8.13.3.20 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

| | |
|--------------|----------------------------------------|
| <i>mp</i> | GLfloat 型の 16 要素の配列変数に格納された投影変換行列。 |
| <i>mv</i> | 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。 |
| <i>light</i> | 光源の特性の gg::LightBuffer 構造体のポインタ。 |
| <i>i</i> | 光源データの uniform block のインデックス。 |

gg.h の 6448 行目に定義があります。

8.13.3.21 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

| | |
|--------------|------------------------------------|
| <i>mp</i> | GLfloat 型の 16 要素の配列変数に格納された投影変換行列。 |
| <i>light</i> | 光源の特性の gg::LightBuffer 構造体のポインタ。 |
| <i>i</i> | 光源データの uniform block のインデックス。 |

gg.h の 6477 行目に定義があります。

8.13.3.22 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

| | |
|--------------|-------------------------------|
| <i>light</i> | 光源の特性の gg::LightBuffer 構造体。 |
| <i>i</i> | 光源データの uniform block のインデックス。 |

gg.h の 6400 行目に定義があります。

8.13.3.23 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

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

gg.h の 6388 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](#)

8.14 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
コピー構造子は使用禁止.
- [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).
- const 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.
- const 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.
- const GLuint & [getTexture](#) () const
使用しているテクスチャのテクスチャ名を得る.

8.14.1 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 4365 行目に定義があります。

8.14.2 構築子と解体子

8.14.2.1 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

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

gg.h の 4383 行目に定義があります。

8.14.2.2 ~GgTexture()

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

デストラクタ。

gg.h の 4398 行目に定義があります。

8.14.2.3 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。

8.14.3 関数詳解

8.14.3.1 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 4411 行目に定義があります。

8.14.3.2 getHeight()

```
const 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 4431 行目に定義があります。

被呼び出し関係図:



8.14.3.3 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 4446 行目に定義があります。

8.14.3.4 getSize() [2/2]

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

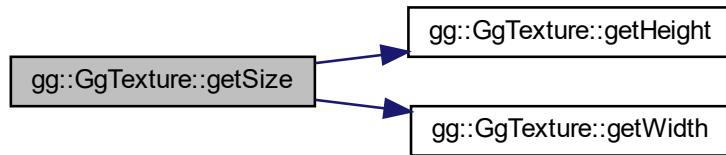
使用しているテクスチャのサイズを取り出す.

引数

| | |
|-------------|---------------------------------------------------|
| size | テクスチャのサイズを格納する <code>GLsizei</code> 型の 2 要素の配列変数. |
|-------------|---------------------------------------------------|

`gg.h` の 4438 行目に定義があります。

呼び出し関係図:



8.14.3.5 `getTexture()`

```
const 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 4453 行目に定義があります。

8.14.3.6 `getWidth()`

```
const 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 4424 行目に定義があります。

被呼び出し関係図:



8.14.3.7 operator=()

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

代入演算子は使用禁止.

8.14.3.8 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 4417 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:

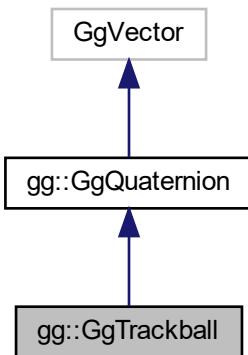
- [gg.h](#)

8.15 gg::GgTrackball クラス

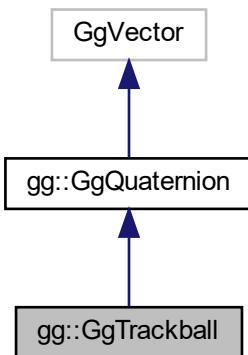
簡易トラックボール処理.

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

gg::GgTrackball の継承関係図



gg::GgTrackball 連携図



公開メンバ関数

- [GgTrackball \(const GgQuaternion &q=ggIdentityQuaternion\(\)\)](#)
コンストラクタ.
- [virtual ~GgTrackball \(\)](#)

- デストラクタ.
- `GgTrackball & operator= (const GgQuaternion &q)`
コンストラクタ.
- `void region (GLfloat w, GLfloat 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 (GLfloat x, GLfloat y)`
　　トラックボール処理を開始する.
- `void motion (GLfloat x, GLfloat 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 (GLfloat x, GLfloat y)`
　　トラックボール処理を停止する.
- `void reset (const GgQuaternion &q=ggetIdentityQuaternion())`
　　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.
- `const 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.
- `const 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.

8.15.1 詳解

簡易トラックボール処理.

gg.h の 4223 行目に定義があります。

8.15.2 構築子と解体子

8.15.2.1 GgTrackball()

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

コンストラクタ.

gg.h の 4235 行目に定義があります。

呼び出し関係図:



8.15.2.2 ~GgTrackball()

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

デストラクタ.

gg.h の 4241 行目に定義があります。

8.15.3 関数詳解

8.15.3.1 begin()

```
void gg::GgTrackball::begin (
    GLfloat x,
    GLfloat 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

| | |
|----------|---------------|
| <i>x</i> | 現在のマウスの x 座標. |
| <i>y</i> | 現在のマウスの y 座標. |

gg.cpp の 4940 行目に定義がります。

8.15.3.2 end()

```
void gg::GgTrackball::end (
    GLfloat x,
    GLfloat 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

| | |
|----------|----------------------|
| <i>x</i> | 現在のマウスの <i>x</i> 座標。 |
| <i>y</i> | 現在のマウスの <i>y</i> 座標。 |

gg.cpp の 5004 行目に定義がります。

8.15.3.3 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 4354 行目に定義がります。

呼び出し関係図:



8.15.3.4 `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 4347 行目に定義があります。

8.15.3.5 `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 4340 行目に定義があります。

8.15.3.6 `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 4318 行目に定義があります。

8.15.3.7 `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

| | |
|---------------|----------------------------|
| <i>factor</i> | 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 4332 行目に定義がります。

8.15.3.8 `getScale()` [3/3]

```
const 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

| | |
|------------------|-----------------------|
| <i>direction</i> | 0 なら x 方向, 1 なら y 方向. |
|------------------|-----------------------|

gg.h の 4325 行目に定義がります。

8.15.3.9 `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 4296 行目に定義がります。

8.15.3.10 `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

| | |
|-----------------|----------------------------|
| <i>position</i> | 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 4310 行目に定義があります。

8.15.3.11 `getStart()` [3/3]

```
const 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

| | |
|------------------|-----------------------|
| <i>direction</i> | 0 なら x 方向, 1 なら y 方向. |
|------------------|-----------------------|

gg.h の 4303 行目に定義があります。

8.15.3.12 `motion()`

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

回転の変換行列を計算する。

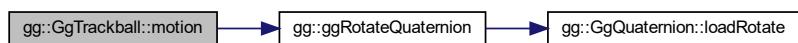
マウスのドラッグ中に呼び出す。

引数

| | |
|----------|---------------|
| <i>x</i> | 現在のマウスの x 座標. |
| <i>y</i> | 現在のマウスの y 座標. |

gg.cpp の 4956 行目に定義があります。

呼び出し関係図:



8.15.3.13 operator=()

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

コンストラクタ.

gg.h の 4247 行目に定義があります。

呼び出し関係図:



8.15.3.14 region() [1/2]

```
void gg::GgTrackball::region ( GLfloat w, GLfloat 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 4927 行目に定義があります。

被呼び出し関係図:



8.15.3.15 `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

| | |
|----------------|--------|
| <code>w</code> | 領域の横幅。 |
| <code>h</code> | 領域の高さ。 |

`gg.h` の 4263 行目に定義があります。

呼び出し関係図:



8.15.3.16 `reset()`

```
void gg::GgTrackball::reset (
    const GgQuaternion & q = ggIdentityQuaternion() )
```

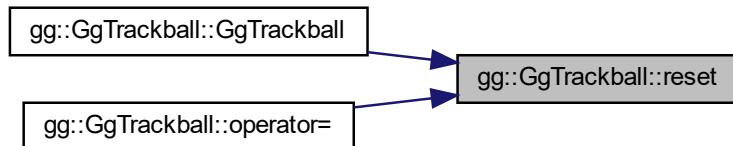
トラックボールをリセットする。

引数

| | |
|---------------------|--|
| トラックボールの回転の初期値の四元数。 | |
|---------------------|--|

`gg.cpp` の 4909 行目に定義があります。

被呼び出し関係図:



8.15.3.17 rotate()

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

ト ラ ッ ク ボ ル の 回 転 角 を 修 正 す る。

引 数

| | |
|----------------|-------------------------|
| <code>q</code> | 修 正 分 の 回 転 角 の 四 元 数 . |
|----------------|-------------------------|

gg.cpp の 4983 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 :

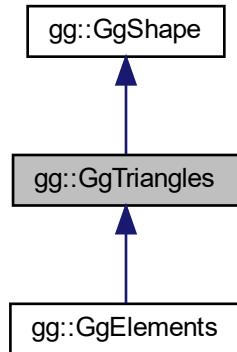
- [gg.h](#)
- [gg.cpp](#)

8.16 gg::GgTriangles クラス

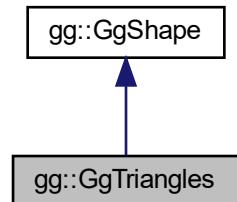
三 角 形 で 表 し た 形 状 デ タ (A r r a y s 形 式) .

```
#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.
- [const GLsizei & getCount \(\) const](#)
データの数を取り出す.
- [const 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.

8.16.1 詳解

三角形で表した形状データ (Arrays 形式).

gg.h の 5278 行目に定義があります。

8.16.2 構築子と解体子

8.16.2.1 GgTriangles() [1/2]

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

コンストラクタ.

引数

| | |
|-------------|--------------|
| <i>mode</i> | 描画する基本図形の種類. |
|-------------|--------------|

gg.h の 5288 行目に定義があります。

8.16.2.2 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

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

gg.h の 5298 行目に定義がります。

8.16.2.3 ~GgTriangles()

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

デストラクタ。

gg.h の 5310 行目に定義がります。

8.16.3 関数詳解

8.16.3.1 draw()

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

三角形の描画。

引数

| | |
|--------------|--------------------------|
| <i>first</i> | 描画を開始する最初の三角形番号。 |
| <i>count</i> | 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 5063 行目に定義がります。

呼び出し関係図:



8.16.3.2 getBuffer()

```
const 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 5323 行目に定義があります。

8.16.3.3 getCount()

```
const 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 5316 行目に定義があります。

8.16.3.4 load()

```
void gg::GgTriangles::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。

引数

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

gg.cpp の 5044 行目に定義があります。

8.16.3.5 `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

| | |
|--------------|----------------------------------|
| <i>vert</i> | 転送元の頂点属性が格納されている領域の先頭のポインタ。 |
| <i>first</i> | 転送先のバッファオブジェクトの先頭の要素番号。 |
| <i>count</i> | 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 5332 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](#)

8.17 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プ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プ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。
- [const 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。
- [const GLsizei & getCount \(\) const](#)
データの数を取り出す。
- [const 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.

8.17.1 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 4820 行目に定義があります。

8.17.2 構築子と解体子

8.17.2.1 GgUniformBuffer() [1/3]

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

コンストラクタ.

gg.h の 4823 行目に定義があります。

8.17.2.2 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

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

gg.h の 4823 行目に定義があります。

8.17.2.3 `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

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

gg.h の 4823 行目に定義があります。

8.17.2.4 `~GgUniformBuffer()`

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

デストラクタ.

gg.h の 4823 行目に定義があります。

8.17.3 関数詳解

8.17.3.1 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 4884 行目に定義があります。

8.17.3.2 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

| | |
|-------------------|-------------------------------------------------|
| <i>src_buffer</i> | 複写元のバッファオブジェクト名。 |
| <i>src_first</i> | 複写元 (buffer) の先頭のデータの位置。 |
| <i>dst_first</i> | 複写先 (getBuffer()) の先頭のデータの位置。 |
| <i>count</i> | 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 5056 行目に定義があります。

8.17.3.3 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

| | |
|---------------|-----------------------------|
| <i>data</i> | 格納するデータ。 |
| <i>offset</i> | 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。 |
| <i>size</i> | 格納するデータの一個あたりのバイト数。 |
| <i>first</i> | 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。 |
| <i>count</i> | 格納するデータの数。 |

gg.h の 4990 行目に定義があります。

8.17.3.4 `getBuffer()`

```
template<typename T >
const 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 4878 行目に定義があります。

8.17.3.5 `getCount()`

```
template<typename T >
const 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 4871 行目に定義があります。

8.17.3.6 `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 4864 行目に定義があります。

8.17.3.7 `getTarget()`

```
template<typename T >
const 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 4857 行目に定義があります。

8.17.3.8 `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

| | |
|--------------------|-----------------|
| <code>data</code> | 格納するデータ。 |
| <code>count</code> | 格納する数。 |
| <code>usage</code> | バッファオブジェクトの使い方。 |

gg.h の 4937 行目に定義があります。

8.17.3.9 `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

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

gg.h の 4921 行目に定義があります。

8.17.3.10 `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 4897 行目に定義があります。

8.17.3.11 `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

| | |
|--------------|--------------------------------|
| <i>first</i> | 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。 |
| <i>count</i> | 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 4906 行目に定義があります。

8.17.3.12 `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

| | |
|---------------|----------------------------------------------|
| <i>data</i> | 抽出先の領域の先頭のポインタ. |
| <i>offset</i> | 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. |
| <i>size</i> | 抽出するデータの一個あたりのバイト数. |
| <i>first</i> | 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. |
| <i>count</i> | 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 5022 行目に定義がります。

8.17.3.13 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

| | |
|---------------|-----------------------------|
| <i>data</i> | データが格納されている領域の先頭のポインタ. |
| <i>offset</i> | 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. |
| <i>size</i> | 格納するデータの一個あたりのバイト数. |
| <i>first</i> | 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. |
| <i>count</i> | 格納するデータの数. |

gg.h の 4955 行目に定義がります。

8.17.3.14 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 4890 行目に定義がります。

8.17.3.15 `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 4912 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](#)

8.18 `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 \(\)](#)
コンストラクタ。
- [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](#)
位置。
- [GgVector normal](#)
法線。

8.18.1 詳解

三角形の頂点データ。

`gg.h` の 5228 行目に定義があります。

8.18.2 構築子と解体子

8.18.2.1 GgVertex() [1/4]

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

コンストラクタ。

gg.h の 5237 行目に定義があります。

8.18.2.2 GgVertex() [2/4]

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

コンストラクタ。

引数

| | |
|-------------|-------------------|
| <i>pos</i> | GgVector 型の位置データ。 |
| <i>norm</i> | GgVector 型の法線データ。 |

gg.h の 5244 行目に定義があります。

8.18.2.3 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

| | |
|-----------|-------------------------|
| <i>px</i> | GgVector 型の位置データの x 成分。 |
| <i>py</i> | GgVector 型の位置データの y 成分。 |
| <i>pz</i> | GgVector 型の位置データの z 成分。 |
| <i>nx</i> | GgVector 型の法線データの x 成分。 |
| <i>ny</i> | GgVector 型の法線データの y 成分。 |
| <i>nz</i> | GgVector 型の法線データの z 成分。 |

gg.h の 5257 行目に定義がります。

8.18.2.4 GgVertex() [4/4]

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

コンストラクタ.

引数

| | |
|-------------|------------------------------------------|
| <i>pos</i> | 3 要素の <code>GLfloat</code> 型の位置データのポインタ. |
| <i>norm</i> | 3 要素の <code>GLfloat</code> 型の法線データのポインタ. |

gg.h の 5269 行目に定義がります。

8.18.3 メンバ詳解

8.18.3.1 normal

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

法線.

gg.h の 5234 行目に定義がります。

8.18.3.2 position

`GgVector` `gg::GgVertex::position`

位置.

gg.h の 5231 行目に定義がります。

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](#)

8.19 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.

8.19.1 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。

gg.h の 5964 行目に定義があります。

8.19.2 メンバ詳解

8.19.2.1 ambient

[GgVector gg::GgSimpleShader::Light::ambient](#)

光源強度の環境光成分.

gg.h の 5966 行目に定義があります。

8.19.2.2 diffuse

[GgVector gg::GgSimpleShader::Light::diffuse](#)

光源強度の拡散反射光成分.

gg.h の 5967 行目に定義があります。

8.19.2.3 position

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

光源の位置.

`gg.h` の 5969 行目に定義があります。

8.19.2.4 specular

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

光源強度の鏡面反射光成分.

`gg.h` の 5968 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:

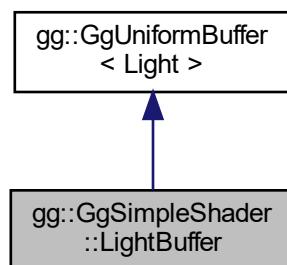
- `gg.h`

8.20 gg::GgSimpleShader::LightBuffer クラス

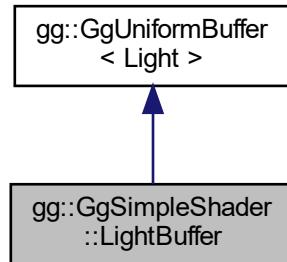
三角形に単純な陰影付けを行うシェーダが参照する光源データのユニフォームバッファオブジェクト.

`#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 loadAmbient (const GgVector &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 GgVector &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 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 GgVector &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 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.

8.20.1 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 5975 行目に定義があります。

8.20.2 構築子と解体子

8.20.2.1 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

| | |
|--------------|-------------------------------------------------------------------|
| <i>light</i> | GgSimpleShader::Light 型の光源データのポインタ. |
| <i>count</i> | バッファ中の GgSimpleShader::Light 型の光源データの数. |
| <i>usage</i> | バッファの使い方のパターン, glBufferData() の第 4 引数の <i>usage</i> に渡される. |

gg.h の 5984 行目に定義があります。

8.20.2.2 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

| | |
|--------------|-----------------------------------------------------|
| <i>light</i> | GgSimpleShader::Light 型の光源データ. |
| <i>count</i> | バッファ中の GgSimpleShader::Light 型の光源データの数. |
| <i>usage</i> | 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 5997 行目に定義があります。

8.20.2.3 ~LightBuffer()

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

デストラクタ.

gg.h の 6007 行目に定義があります。

8.20.3 関数詳解

8.20.3.1 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

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

gg.h の 6151 行目に定義があります。

8.20.3.2 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

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

gg.h の 6142 行目に定義があります。

8.20.3.3 [loadAmbient\(\)](#) [1/3]

```
void gg::GgSimpleShader::LightBuffer::loadAmbient (
    const GgVector & ambient,
    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

| | |
|----------------|-------------------------------------------------|
| <i>ambient</i> | 光源の強度の環境光成分を格納した GgVector 型の変数。 |
| <i>first</i> | 値を設定する光源データの最初の番号, デフォルトは 0. |
| <i>count</i> | 値を設定する光源データの数, デフォルトは 1. |

gg.cpp の 5399 行目に定義があります。

8.20.3.4 [loadAmbient\(\)](#) [2/3]

```
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

| | |
|----------------|--------------------------------------------------------|
| <i>ambient</i> | 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。 |
| <i>first</i> | 値を設定する光源データの最初の番号, デフォルトは 0. |
| <i>count</i> | 値を設定する光源データの数, デフォルトは 1. |

gg.h の 6027 行目に定義がります。

8.20.3.5 loadAmbient() [3/3]

```
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

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

gg.cpp の 5373 行目に定義がります。

8.20.3.6 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

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

gg.cpp の 5526 行目に定義がります。

8.20.3.7 `loadDiffuse()` [1/3]

```
void gg::GgSimpleShader::LightBuffer::loadDiffuse (
    const GgVector & 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。

引数

| | |
|-----------------|------------------------------------------------|
| <i>specular</i> | 光源の強度の拡散反射光成分を格納した <code>GgVector</code> 型の変数. |
| <i>first</i> | 値を設定する光源データの最初の番号, デフォルトは 0. |
| <i>count</i> | 値を設定する光源データの数, デフォルトは 1. |

`gg.cpp` の 5451 行目に定義があります。

8.20.3.8 `loadDiffuse()` [2/3]

```
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

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

`gg.h` の 6061 行目に定義があります。

8.20.3.9 `loadDiffuse()` [3/3]

```
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

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

gg.cpp の 5425 行目に定義があります。

8.20.3.10 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

| | |
|-----------------|--------------------------------|
| <i>position</i> | 光源の位置の同次座標を格納した GgVector 型の変数. |
| <i>first</i> | 値を設定する光源データの最初の番号, デフォルトは 0. |
| <i>count</i> | 値を設定する光源データの数, デフォルトは 1. |

gg.cpp の 5580 行目に定義があります。

8.20.3.11 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

| | |
|-----------------|--------------------------------|
| <i>position</i> | 光源の位置の同次座標を格納した GgVector 型の配列. |
| <i>first</i> | 値を設定する光源データの最初の番号, デフォルトは 0. |
| <i>count</i> | 値を設定する光源データの数, デフォルトは 1. |

gg.h の 6133 行目に定義がります。

8.20.3.12 `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

| | |
|-----------------|----------------------------------------------------|
| <i>position</i> | 光源の位置の同次座標を格納した <code>GLfloat</code> 型の 4 要素の配列変数。 |
| <i>first</i> | 値を設定する光源データの最初の番号, デフォルトは 0. |
| <i>count</i> | 値を設定する光源データの数, デフォルトは 1. |

gg.h の 6123 行目に定義がります。

8.20.3.13 `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

| | |
|--------------|------------------------------|
| <i>x</i> | 光源の位置の x 座標。 |
| <i>y</i> | 光源の位置の y 座標。 |
| <i>z</i> | 光源の位置の z 座標。 |
| <i>w</i> | 光源の位置の w 座標, デフォルトは 1. |
| <i>first</i> | 値を設定する光源データの最初の番号, デフォルトは 0. |
| <i>count</i> | 値を設定する光源データの数, デフォルトは 1. |

gg.cpp の 5554 行目に定義がります。

8.20.3.14 **loadSpecular()** [1/3]

```
void gg::GgSimpleShader::LightBuffer::loadSpecular (
    const GgVector & 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。

引数

| | |
|-----------------|------------------------------------------------|
| <i>specular</i> | 光源の強度の鏡面反射光成分を格納した <code>GgVector</code> 型の変数. |
| <i>first</i> | 値を設定する光源データの最初の番号, デフォルトは 0. |
| <i>count</i> | 値を設定する光源データの数, デフォルトは 1. |

`gg.cpp` の 5503 行目に定義があります。

8.20.3.15 **loadSpecular()** [2/3]

```
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

| | |
|-----------------|-------------------------------------------------------|
| <i>specular</i> | 光源の強度の鏡面反射光成分を格納した <code>GLfloat</code> 型の 4 要素の配列変数. |
| <i>first</i> | 値を設定する光源データの最初の番号, デフォルトは 0. |
| <i>count</i> | 値を設定する光源データの数, デフォルトは 1. |

`gg.h` の 6089 行目に定義があります。

8.20.3.16 **loadSpecular()** [3/3]

```
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

| | |
|--------------|-------------------------------|
| <i>r</i> | 光源の強度の鏡面反射光成分の赤成分. |
| <i>g</i> | 光源の強度の鏡面反射光成分の緑成分. |
| <i>b</i> | 光源の強度の鏡面反射光成分の青成分. |
| <i>a</i> | 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. |
| <i>first</i> | 値を設定する光源データの最初の番号, デフォルトは 0. |
| <i>count</i> | 値を設定する光源データの数, デフォルトは 1. |

gg.cpp の 5477 行目に定義があります。

8.20.3.17 `select()`

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

光源を選択する.

引数

| | |
|----------|-------------------------------|
| <i>i</i> | 光源データの uniform block のインデックス. |
|----------|-------------------------------|

gg.h の 6158 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](#)

8.21 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.

8.21.1 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。

gg.h の 6169 行目に定義があります。

8.21.2 メンバ詳解

8.21.2.1 ambient

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

環境光に対する反射係数。

gg.h の 6171 行目に定義があります。

8.21.2.2 diffuse

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

拡散反射係数。

gg.h の 6172 行目に定義があります。

8.21.2.3 shininess

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

輝き係数。

gg.h の 6174 行目に定義があります。

8.21.2.4 specular

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

鏡面反射係数。

gg.h の 6173 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:

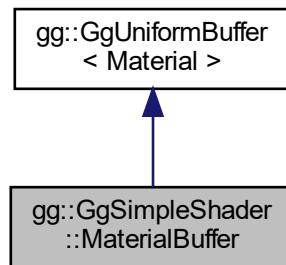
- [gg.h](#)

8.22 gg::GgSimpleShader::MaterialBuffer クラス

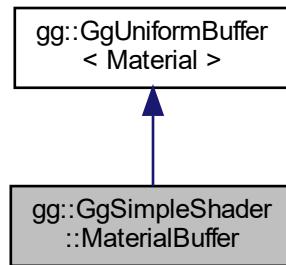
三角形に単純な陰影付けを行うシェーダが参照する材質データのユニフォームバッファオブジェクト。

```
#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.

8.22.1 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 6180 行目に定義があります。

8.22.2 構築子と解体子

8.22.2.1 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

| | |
|-----------------|-------------------------------------------------------------------------|
| <i>material</i> | <code>GgSimpleShader::Material</code> 型の材質データのポインタ. |
| <i>count</i> | バッファ中の <code>GgSimpleShader::Material</code> 型の材質データの数. |
| <i>usage</i> | バッファの使い方のパターン, <code>glBufferData()</code> の第 4 引数の <i>usage</i> に渡される. |

gg.h の 6189 行目に定義があります。

8.22.2.2 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

| | |
|-----------------|-------------------------------------------------------------------------|
| <i>material</i> | <code>GgSimpleShader::Material</code> 型の材質データ。 |
| <i>count</i> | バッファ中の <code>GgSimpleShader::Material</code> 型の材質データの数。 |
| <i>usage</i> | バッファの使い方のパターン, <code>glBufferData()</code> の第 4 引数の <i>usage</i> に渡される。 |

gg.h の 6202 行目に定義があります。

8.22.2.3 ~MaterialBuffer()

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

デストラクタ。

gg.h の 6212 行目に定義があります。

8.22.3 関数詳解

8.22.3.1 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

| | |
|-----------------|---------------------------------------------------|
| <i>material</i> | 光源の特性の <code>GgSimpleShader::Material</code> 構造体。 |
| <i>first</i> | 値を設定する材質データの最初の番号, デフォルトは 0. |
| <i>count</i> | 値を設定する材質データの数, デフォルトは 1. |

gg.h の 6325 行目に定義がります。

8.22.3.2 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

| | |
|-----------------|-----------------------------------------------------------|
| <i>material</i> | 光源の特性の GgSimpleShader::Material 構造体のポインタ。 |
| <i>first</i> | 値を設定する材質データの最初の番号, デフォルトは 0. |
| <i>count</i> | 値を設定する材質データの数, デフォルトは 1. |

gg.h の 6316 行目に定義がります。

8.22.3.3 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

| | |
|----------------|-----------------------------------------------------|
| <i>ambient</i> | 環境光に対する反射係数を格納した <code>GLfloat</code> 型の 4 要素の配列変数。 |
| <i>first</i> | 値を設定する材質データの最初の番号, デフォルトは 0. |
| <i>count</i> | 値を設定する材質データの数, デフォルトは 1. |

gg.h の 6232 行目に定義がります。

8.22.3.4 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

| | |
|--------------|------------------------------|
| <i>r</i> | 環境光に対する反射係数の赤成分。 |
| <i>g</i> | 環境光に対する反射係数の緑成分。 |
| <i>b</i> | 環境光に対する反射係数の青成分。 |
| <i>a</i> | 環境光に対する反射係数の不透明度, デフォルトは 1. |
| <i>first</i> | 値を設定する材質データの最初の番号, デフォルトは 0. |
| <i>count</i> | 値を設定する材質データの数, デフォルトは 1. |

gg.cpp の 5606 行目に定義があります。

8.22.3.5 **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

| | |
|--------------|------------------------------------------------------------|
| <i>color</i> | 環境光に対する反射係数と拡散反射係数を格納した <code>GLfloat</code> 型の 4 要素の配列変数。 |
| <i>first</i> | 値を設定する材質データの最初の番号, デフォルトは 0. |
| <i>count</i> | 値を設定する材質データの数, デフォルトは 1. |

gg.cpp の 5690 行目に定義があります。

8.22.3.6 **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

| | |
|--------------|------------------------------------|
| <i>r</i> | 環境光に対する反射係数と拡散反射係数の赤成分. |
| <i>g</i> | 環境光に対する反射係数と拡散反射係数の緑成分. |
| <i>b</i> | 環境光に対する反射係数と拡散反射係数の青成分. |
| <i>a</i> | 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. |
| <i>first</i> | 値を設定する材質データの最初の番号, デフォルトは 0. |
| <i>count</i> | 値を設定する材質データの数, デフォルトは 1. |

gg.cpp の 5664 行目に定義があります。

8.22.3.7 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

| | |
|----------------|-----------------------------------|
| <i>diffuse</i> | 拡散反射係数を格納した GLfloat 型の 4 要素の配列変数. |
| <i>first</i> | 値を設定する材質データの最初の番号, デフォルトは 0. |
| <i>count</i> | 値を設定する材質データの数, デフォルトは 1. |

gg.h の 6254 行目に定義があります。

8.22.3.8 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

| | |
|----------|-------------|
| <i>r</i> | 拡散反射係数の赤成分. |
| <i>g</i> | 拡散反射係数の緑成分. |

引数

| | |
|--------------|------------------------------|
| <i>b</i> | 拡散反射係数の青成分. |
| <i>a</i> | 拡散反射係数の不透明度, デフォルトは 1. |
| <i>first</i> | 値を設定する材質データの最初の番号, デフォルトは 0. |
| <i>count</i> | 値を設定する材質データの数, デフォルトは 1. |

gg.cpp の 5635 行目に定義がります。

8.22.3.9 `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

| | |
|------------------|------------------------------|
| <i>shininess</i> | 輝き係数. |
| <i>first</i> | 値を設定する材質データの最初の番号, デフォルトは 0. |
| <i>count</i> | 値を設定する材質データの数, デフォルトは 1. |

gg.cpp の 5780 行目に定義がります。

8.22.3.10 `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

| | |
|------------------|------------------------------|
| <i>shininess</i> | 輝き係数. |
| <i>first</i> | 値を設定する材質データの最初の番号, デフォルトは 0. |
| <i>count</i> | 値を設定する材質データの数, デフォルトは 1. |

gg.cpp の 5759 行目に定義がります。

8.22.3.11 **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

| | |
|-----------------|------------------------------------------------|
| <i>specular</i> | 鏡面反射係数を格納した <code>GLfloat</code> 型の 4 要素の配列変数。 |
| <i>first</i> | 値を設定する材質データの最初の番号, デフォルトは 0. |
| <i>count</i> | 値を設定する材質データの数, デフォルトは 1. |

gg.h の 6294 行目に定義があります。

8.22.3.12 **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

| | |
|--------------|------------------------------|
| <i>r</i> | 鏡面反射係数の赤成分。 |
| <i>g</i> | 鏡面反射係数の緑成分。 |
| <i>b</i> | 鏡面反射係数の青成分。 |
| <i>a</i> | 鏡面反射係数の不透明度, デフォルトは 1. |
| <i>first</i> | 値を設定する材質データの最初の番号, デフォルトは 0. |
| <i>count</i> | 値を設定する材質データの数, デフォルトは 1. |

gg.cpp の 5733 行目に定義があります。

8.22.3.13 **select()**

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

材質を選択する。

引数

| | |
|----------|-------------------------------|
| <i>i</i> | 材質データの uniform block のインデックス. |
|----------|-------------------------------|

gg.h の 6332 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](#)

8.23 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 \(const std::string &title="GLFW Window", int width=640, int height=480, int fullscreen=0, GLFWwindow *share=nullptr\)](#)
コンストラクタ.
- [Window \(const Window &w\)=delete](#)
コピー構造子は使用禁止.
- [Window & operator= \(const Window &w\)=delete](#)
代入演算子は使用禁止.
- [virtual ~Window \(\)](#)
デストラクタ.
- [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 **restoreViewport** ()
 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.
- void **selectInterface** (int no)
 インターフェースを選択する
- void **setVelocity** (GLfloat vx, GLfloat vy, GLfloat vz=0.1f)
 マウスの移動速度を設定する
- 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 **getAltArrow** (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.
- const GLfloat * **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スクロールの変換行列を得る.
- GgMatrix **getTranslationMatrix** (int button=GLFW_MOUSE_BUTTON_1) const
マウスによって視点の平行移動の変換行列を得る.
- GgMatrix **getScrollMatrix** (int button=GLFW_MOUSE_BUTTON_1) const
マウスによってオブジェクトの平行移動の変換行列を得る.
- GgQuaternion **getRotation** (int button=GLFW_MOUSE_BUTTON_1) const
トラックボールの回転変換行列を得る.
- GgMatrix **getRotationMatrix** (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 **resetRotation** ()
トラックボール処理をリセットする
- void **resetTranslation** ()
現在位置と平行移動量をリセットする
- void **reset** ()
トラックボール・マウスホイール・矢印キーの値を初期化する
- void * **getUserPointer** () const
ユーザー pointer を取り出す.
- void **setUserPointer** (void *pointer)
任意のユーザ pointer 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 **setWheelFunc** (void(*func)(const Window *window, double x, double y))
ユーザ定義の *wheel* 関数を設定する.
- void **setMenubarHeight** (float menubarheight)
表示領域をメニューバーの高さだけ減らす.

静的公開メンバ関数

- static void **init** (int major=0, int minor=1)
ゲームグラフィックス特論の宿題用補助プログラムの初期化. 最初に一度だけ実行する.

8.23.1 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 92 行目に定義があります。

8.23.2 構築子と解体子

8.23.2.1 Window() [1/2]

```
Window::Window (
    const std::string & 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

| | |
|-------------------|--------------------------------------------------|
| <i>title</i> | ウィンドウタイトルの文字列. |
| <i>width</i> | 開くウィンドウの幅. |
| <i>height</i> | 開くウィンドウの高さ. |
| <i>fullscreen</i> | 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. |
| <i>share</i> | 共有するコンテキスト, <code>nullptr</code> ならコンテキストを共有しない. |

Window.h の 449 行目に定義があります。

呼び出し関係図:



8.23.2.2 Window() [2/2]

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

コピーコンストラクタは使用禁止.

8.23.2.3 ~Window()

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

デストラクタ.

Window.h の 532 行目に定義があります。

8.23.3 関数詳解

8.23.3.1 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 549 行目に定義があります。

8.23.3.2 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 788 行目に定義があります。

8.23.3.3 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 795 行目に定義があります。

8.23.3.4 getAltArrow()

```
void Window::getAltArrow (
    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

| | |
|------------------|---------------------------------------------------|
| <i>alt_arrow</i> | 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 802 行目に定義があります。

8.23.3.5 `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

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

Window.h の 736 行目に定義があります。

8.23.3.6 `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

| | |
|------------------|--------------------------------------------|
| <i>direction</i> | 方向 (0: X, 1:Y)。 |
| <i>mods</i> | 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 711 行目に定義があります。

8.23.3.7 getArrowX()

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

矢印キーの現在の X 値を得る.

引数

| | |
|-------------|--------------------------------------------|
| <i>mods</i> | 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 720 行目に定義があります。

8.23.3.8 getArrowY()

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

矢印キーの現在の Y 値を得る.

引数

| | |
|-------------|--------------------------------------------|
| <i>mods</i> | 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 728 行目に定義があります。

8.23.3.9 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 668 行目に定義があります。

8.23.3.10 `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

| | |
|----------------------------|-----------------------------------------------------------------|
| <code>control_arrow</code> | 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 <code>GLfloat</code> 型の 2 要素の配列。 |
|----------------------------|-----------------------------------------------------------------|

`Window.h` の 780 行目に定義があります。

8.23.3.11 `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 766 行目に定義があります。

8.23.3.12 `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 773 行目に定義があります。

8.23.3.13 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 646 行目に定義があります。

8.23.3.14 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 681 行目に定義があります。

8.23.3.15 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 810 行目に定義があります。

8.23.3.16 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

| | |
|-----------------|---------------------------------------|
| <i>position</i> | 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 818 行目に定義があります。

8.23.3.17 **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

| | |
|------------------|----------------|
| <i>direction</i> | 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 828 行目に定義があります。

8.23.3.18 **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 836 行目に定義があります。

8.23.3.19 **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 844 行目に定義があります。

8.23.3.20 `getRotation()`

```
GgQuaternion Window::getRotation (
    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

| | |
|---------------------|----------------------------------------------|
| <code>button</code> | 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 GgQuaternion 型の四元数。

Window.h の 937 行目に定義があります。

8.23.3.21 `getRotationMatrix()`

```
GgMatrix Window::getRotationMatrix (
    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

| | |
|---------------------|----------------------------------------------|
| <code>button</code> | 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 GgMatrix 型の変換行列。

Window.h の 947 行目に定義があります。

8.23.3.22 `getScrollMatrix()`

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

マウスによってオブジェクトの平行移動の変換行列を得る。

引数

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

戻り値

クリッピング座標系で平行移動を行う GgMatrix 型の変換行列.

Window.h の 920 行目に定義があります。

8.23.3.23 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

| | |
|--------------------|-----------------------------------------------------|
| <i>shift_arrow</i> | 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 758 行目に定義があります。

8.23.3.24 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 744 行目に定義があります。

8.23.3.25 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 751 行目に定義があります。

8.23.3.26 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 653 行目に定義があります。

8.23.3.27 getSize() [2/2]

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

ウィンドウのサイズを得る。

引数

| | |
|-------------------|--------------------------------------------------|
| <code>size</code> | ウィンドウの幅と高さを格納した <code>GLsizei</code> 型の 2 要素の配列。 |
|-------------------|--------------------------------------------------|

`Window.h` の 660 行目に定義があります。

8.23.3.28 getTranslation()

```
const GLfloat* 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スクロールの変換行列を得る。

引数

| | |
|---------------------|------------------------------------------------------------|
| <code>button</code> | 平行移動量を取得するマウスボタン (<code>GLFW_MOUSE_BUTTON_[1,2]</code>)。 |
|---------------------|------------------------------------------------------------|

戻り値

平行移動量を格納した `GLfloat[3]` の配列のポインタ。

`Window.h` の 893 行目に定義があります。

8.23.3.29 getTranslationMatrix()

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

マウスによって視点の平行移動の変換行列を得る。

引数

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

戻り値

視点座標系で平行移動を行う GgMatrix 型の変換行列。

Window.h の 903 行目に定義があります。

8.23.3.30 getUserPointer()

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

ユーザー pointer を取り出す。

戻り値

保存されているユーザ pointer。

Window.h の 992 行目に定義があります。

8.23.3.31 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 852 行目に定義があります。

8.23.3.32 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

| | |
|-----------------|--------------------------------------|
| <i>rotation</i> | 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 860 行目に定義があります。

8.23.3.33 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

| | |
|------------------|----------------|
| <i>direction</i> | 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 870 行目に定義があります。

8.23.3.34 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 877 行目に定義があります。

8.23.3.35 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 884 行目に定義があります。

8.23.3.36 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 639 行目に定義があります。

8.23.3.37 init()

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

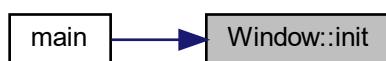
ゲームグラフィックス特論の宿題用補助プログラムの初期化, 最初に一度だけ実行する.

引数

| | |
|--------------|-------------------------------------------|
| <i>major</i> | 使用する OpenGL の major 番号, 0 なら無指定. |
| <i>minor</i> | 使用する OpenGL の minor 番号, major 番号が 0 なら無視. |

Window.h の 391 行目に定義があります。

呼び出し関係図:



8.23.3.38 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 571 行目に定義があります。

8.23.3.39 operator=()

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

代入演算子は使用禁止.

8.23.3.40 reset()

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

トラックボール・マウスホイール・矢印キーの値を初期化する

Window.h の 981 行目に定義があります。

8.23.3.41 resetRotation()

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

トラックボール処理をリセットする

Window.h の 955 行目に定義があります。

8.23.3.42 resetTranslation()

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

現在位置と平行移動量をリセットする

Window.h の 962 行目に定義があります。

8.23.3.43 restoreViewport()

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

ビューポートをウィンドウ全体に設定する.

Window.h の 674 行目に定義があります。

8.23.3.44 selectInterface()

```
void Window::selectInterface (
    int no ) [inline]
```

インターフェースを選択する

引数

| | |
|-----------|------------|
| <i>no</i> | インターフェース番号 |
|-----------|------------|

Window.h の 693 行目に定義があります。

8.23.3.45 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

| | |
|-------------|----------------------------------------|
| <i>flag</i> | 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 556 行目に定義があります。

8.23.3.46 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

| | |
|-------------|---------------------------------------|
| <i>func</i> | 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 1013 行目に定義があります。

8.23.3.47 setMenubarHeight()

```
void Window::setMenubarHeight (
    float menubarheight ) [inline]
```

表示領域をメニューバーの高さだけ減らす。

引数

| | |
|-----------|--|
| メニューバーの高さ | |
|-----------|--|

Window.h の 1034 行目に定義があります。

8.23.3.48 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 1020 行目に定義があります。

8.23.3.49 setResizeFunc()

```
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 1006 行目に定義があります。

8.23.3.50 setUserPointer()

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

任意のユーザポインタを保存する。

引数

| | |
|----------------|--------------|
| <i>pointer</i> | 保存するユーザポインタ. |
|----------------|--------------|

Window.h の 999 行目に定義があります。

8.23.3.51 setVelocity()

```
void Window::setVelocity (
    GLfloat vx,
    GLfloat vy,
    GLfloat vz = 0.1f ) [inline]
```

マウスの移動速度を設定する

引数

| | |
|-----------|------------|
| <i>vx</i> | x 方向の移動速度. |
| <i>vy</i> | y 方向の移動速度. |

Window.h の 702 行目に定義があります。

8.23.3.52 setWheelFunc()

```
void Window::setWheelFunc (
    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

| | |
|-------------|--------------------------------------|
| <i>func</i> | 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 1027 行目に定義があります。

8.23.3.53 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 563 行目に定義があります。

8.23.3.54 swapBuffers()

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

カラー バッファを入れ替える。

Window.h の 623 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](#)

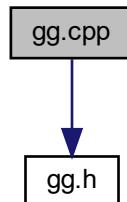
Chapter 9

ファイル詳解

9.1 gg.cpp ファイル

ゲームグラフィックス特論の宿題用補助プログラム GLFW3 版の定義。

```
#include "gg.h"  
gg.cpp の依存先関係図:
```



9.1.1 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 GLFW3 版の定義。

著者

Kohe Tokoi

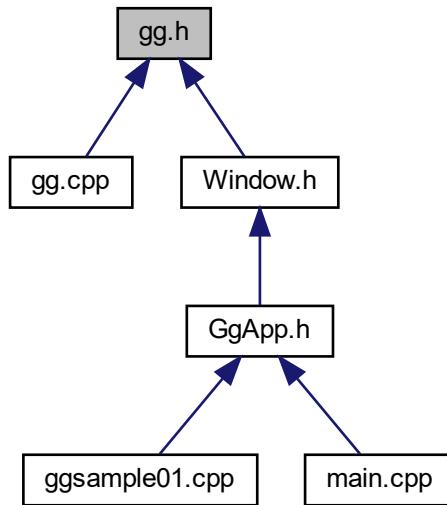
日付

March 31, 2021

9.2 gg.h ファイル

ゲームグラフィックス特論の宿題用補助プログラム GLFW3 版の宣言.

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

型定義

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

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 std::string &name="", unsigned int line=0)**
OpenGL のエラーをチェックする.
- **void gg::ggFBOError (const std::string &name="", unsigned int line=0)**
FBO のエラーをチェックする.
- **bool gg::ggSaveTga (const std::string &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 std::string &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 std::string &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 std::string &name, std::vector< GLubyte > &image, GLsizei *pWidth, GLsizei *pHeight, GLenum *pFormat)`
TGA ファイル (*8/16/24/32bit*) をメモリに読み込む.
- `GLuint gg::ggLoadTexture (const GLvoid *image, GLsizei width, GLsizei height, GLenum format=GL_BGR, GLenum type=GL_UNSIGNED_BYTE, GLenum internal=GL_RGB, GLenum wrap=GL_CLAMP_TO_EDGE)`
テクスチャメモリを確保して画像データをテクスチャとして読み込む.
- `GLuint gg::ggLoadImage (const std::string &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 std::string &name, GLfloat 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 std::string &vsrc, const std::string &fsrc="", const std::string &gsrc="", GLint nvarying=0, const char *const *varyings=nullptr, const std::string &vtext="vertex shader", const std::string &ftext="fragment shader", const std::string >ext="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 std::string &vert, const std::string &frag="", const std::string &geom="", 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 std::string &csrc, const std::string &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 std::string &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.
- `void gg::ggCross (GLfloat *c, const GLfloat *a, const GLfloat *b)`
3要素の外積.
- `GLfloat gg::ggDot3 (const GLfloat *a, const GLfloat *b)`
3要素の内積.
- `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::ggDot4 (const GLfloat *a, const GLfloat *b)`
4要素の内積
- `GLfloat gg::ggLength4 (const GLfloat *a)`
4要素の長さ.
- `void gg::ggNormalize4 (GLfloat *a)`
4要素の正規化.
- `GgVector gg::operator+ (const GgVector &a, const GgVector &b)`
GgVector 型の和を返す.
- `GgVector & gg::operator+= (GgVector &a, const GgVector &b)`
GgVector 型を加算する.
- `GgVector gg::operator- (const GgVector &a, const GgVector &b)`
GgVector 型の差を返す.
- `GgVector & gg::operator-= (GgVector &a, const GgVector &b)`
GgVector 型を減算する.

- GgVector `gg::operator*` (const GgVector &a, const GgVector &b)
GgVector 型の積を返す.
- GgVector & `gg::operator*=(` (GgVector &a, const GgVector &b)
GgVector 型を乗算する.
- GgVector `gg::operator/` (const GgVector &a, const GgVector &b)
GgVector 型の商を返す.
- GgVector & `gg::operator/=` (GgVector &a, const GgVector &b)
GgVector 型を除算する.
- GgVector `gg::operator+` (const GgVector &a, GLfloat b)
GgVector 型の各要素にスカラーを足した和を返す.
- GgVector `gg::operator+` (GLfloat a, const GgVector &b)
GgVector 型の各要素にスカラーを足した和を返す.
- GgVector & `gg::operator+=` (GgVector &a, GLfloat b)
GgVector 型の各要素にスカラーを足す.
- GgVector `gg::operator-` (const GgVector &a, GLfloat b)
GgVector 型の各要素からスカラーを引いた差を返す.
- GgVector `gg::operator-` (GLfloat a, const GgVector &b)
スカラーから *GgVector* 型の各要素を引いた差を返す.
- GgVector & `gg::operator-=` (GgVector &a, GLfloat b)
GgVector 型の各要素からスカラーを引く.
- GgVector `gg::operator*` (const GgVector &a, GLfloat b)
GgVector 型の各要素にスカラーを乗じた積を返す.
- GgVector `gg::operator*` (GLfloat a, const GgVector &b)
GgVector 型の各要素にスカラーを乗じた積を返す.
- GgVector & `gg::operator*=(` (GgVector &a, GLfloat b)
GgVector 型の各要素にスカラーを乗じる.
- GgVector `gg::operator/` (const GgVector &a, GLfloat b)
GgVector 型の各要素をスカラーで割った商を返す.
- GgVector `gg::operator/` (GLfloat a, const GgVector &b)
スカラーを *GgVector* 型の各要素で割った商を返す.
- GgVector & `gg::operator/=` (GgVector &a, GLfloat b)
GgVector 型の各要素をスカラーで割る.
- GLfloat `gg::ggDot3` (const GgVector &a, const GgVector &b)
GgVector 型の 3 要素の内積.
- GgVector `gg::ggCross` (const GgVector &a, const GgVector &b)
GgVector 型の 3 要素の外積.
- GLfloat `gg::ggLength3` (const GgVector &a)
GgVector 型の 3 要素の長さ.
- GLfloat `gg::ggDistance3` (const GgVector &a, const GgVector &b)
GgVector 型の 3 要素の距離.
- bool `gg::ggNormalize3` (GgVector *a)
GgVector 型の 3 要素の正規化.
- GgVector `gg::ggNormalize3` (const GgVector &a)
GgVector 型の 3 要素の正規化.
- GLfloat `gg::ggDot4` (const GgVector &a, const GgVector &b)
GgVector の 4 要素の内積.
- GLfloat `gg::ggLength4` (const GgVector &a)
GgVector 型の長さ.
- GLfloat `gg::ggDistance4` (const GgVector &a, const GgVector &b)
GgVector 型の距離.
- bool `gg::ggNormalize4` (GgVector *a)

- GgVector* 型の正規化.
- **GgVector gg::ggNormalize4** (const GgVector &a)

GgVector 型の正規化.
 - **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)

- 梢円状に三角形を生成する.
- `GgTriangles * gg::ggArraysObj (const std::string &name, bool normalize=false)`
`Wavefront OBJ` ファイルを読み込む (`Arrays` 形式)
 - `GgElements * gg::ggElementsObj (const std::string &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 std::string &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 std::string &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.

9.2.1 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 `GLFW3` 版の宣言.

著者

Kohe Tokoi

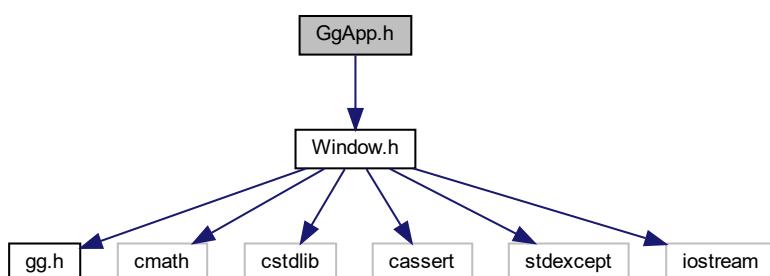
日付

March 31, 2021

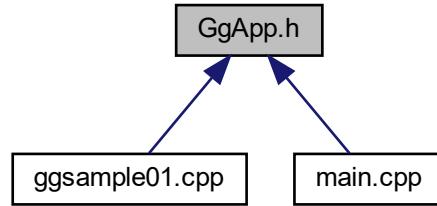
9.3 GgApp.h ファイル

```
#include "Window.h"
```

`GgApp.h` の依存先関係図:



被依存関係図:



クラス

- class `GgApp`

マクロ定義

- `#define GG_USE_IMGUI`

9.3.1 マクロ定義詳解

9.3.1.1 GG_USE_IMGUI

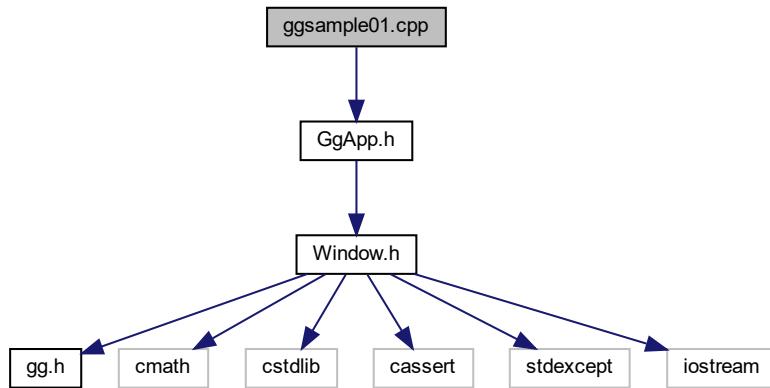
```
#define GG_USE_IMGUI
```

`GgApp.h` の 4 行目に定義があります。

9.4 ggsample01.cpp ファイル

```
#include "GgApp.h"
```

ggsample01.cpp の依存先関係図:



マクロ定義

- `#define PROJECT_NAME "ggsample01"`

9.4.1 マクロ定義詳解

9.4.1.1 PROJECT_NAME

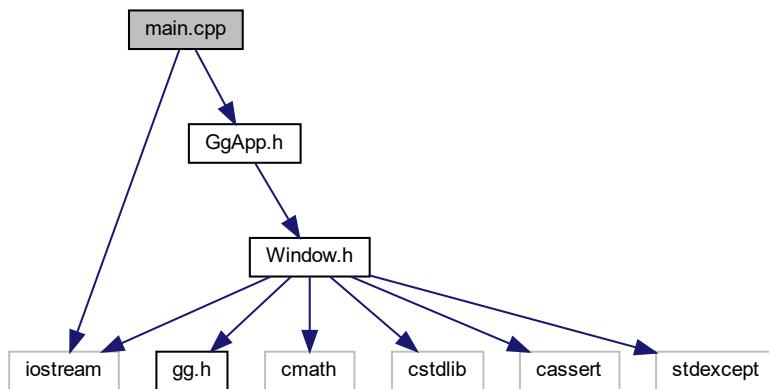
```
#define PROJECT_NAME "ggsample01"
```

ggsample01.cpp の 8 行目に定義があります。

9.5 main.cpp ファイル

```
#include <iostream>
#include "GgApp.h"
```

main.cpp の依存先関係図:



マクロ定義

- `#define HEADER_STR "ゲーム グラフィックス特論"`

関数

- `int main (int argc, const char *const *argv)`

9.5.1 マクロ定義詳解

9.5.1.1 HEADER_STR

```
#define HEADER_STR "ゲーム グラフィックス特論"
```

main.cpp の 15 行目に定義があります。

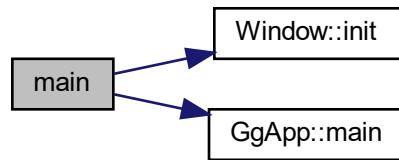
9.5.2 関数詳解

9.5.2.1 main()

```
int main (
    int argc,
    const char *const * argv )
```

main.cpp の 23 行目に定義があります。

呼び出し関係図:

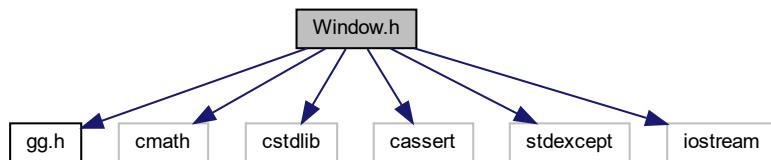


9.6 README.md ファイル

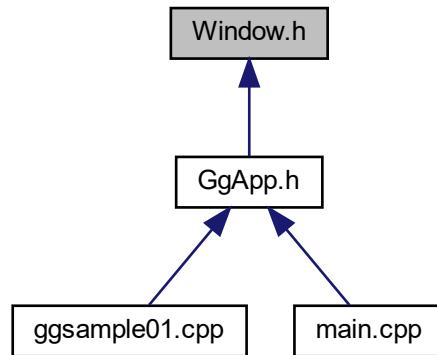
9.7 Window.h ファイル

ゲームグラフィックス特論の宿題用補助プログラムのウィンドウ関連の処理.

```
#include "gg.h"
#include <cmath>
#include <cstdlib>
#include <cassert>
#include <stdexcept>
#include <iostream>
Window.h の依存先関係図:
```



被依存関係図:



クラス

- class `Window`
 ウィンドウ関連の処理.

マクロ定義

- `#define GG_BUTTON_COUNT 3`
- `#define GG_INTERFACE_COUNT 5`

9.7.1 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ウィンドウ関連の処理.

著者

Kohe Tokoi

日付

March 31, 2021

9.7.2 マクロ定義詳解

9.7.2.1 GG_BUTTON_COUNT

```
#define GG_BUTTON_COUNT 3
```

Window.h の 43 行目に定義がります。

9.7.2.2 GG_INTERFACE_COUNT

```
#define GG_INTERFACE_COUNT 5
```

Window.h の 48 行目に定義がります。

Index

```
_ggError
    gg, 21
_~ggFBOError
    gg, 22
~GgBuffer
    gg::GgBuffer< T >, 91
~GgColorTexture
    gg::GgColorTexture, 98
~GgElements
    gg::GgElements, 102
~GgMatrix
    gg::GgMatrix, 111
~GgNormalTexture
    gg::GgNormalTexture, 173
~GgPointShader
    gg::GgPointShader, 181
~GgPoints
    gg::GgPoints, 177
~GgQuaternion
    gg::GgQuaternion, 195
~GgShader
    gg::GgShader, 259
~GgShape
    gg::GgShape, 261
~GgSimpleObj
    gg::GgSimpleObj, 265
~GgSimpleShader
    gg::GgSimpleShader, 270
~GgTexture
    gg::GgTexture, 282
~GgTrackball
    gg::GgTrackball, 288
~GgTriangles
    gg::GgTriangles, 298
~GgUniformBuffer
    gg::GgUniformBuffer< T >, 302
~LightBuffer
    gg::GgSimpleShader::LightBuffer, 315
~MaterialBuffer
    gg::GgSimpleShader::MaterialBuffer, 326
~Window
    Window, 335

add
    gg::GgMatrix, 111
    gg::GgQuaternion, 196, 197
ambient
    gg::GgSimpleShader::Light, 311
    gg::GgSimpleShader::Material, 323
begin
    gg::GgTrackball, 288
bind
    gg::GgBuffer< T >, 92
    gg::GgTexture, 282
    gg::GgUniformBuffer< T >, 302
BindingPoints
    gg, 21
conjugate
    gg::GgQuaternion, 198
copy
    gg::GgBuffer< T >, 92
    gg::GgUniformBuffer< T >, 303
diffuse
    gg::GgSimpleShader::Light, 311
    gg::GgSimpleShader::Material, 323
divide
    gg::GgMatrix, 112, 113
    gg::GgQuaternion, 199–201
draw
    gg::GgElements, 103
    gg::GgPoints, 177
    gg::GgShape, 262
    gg::GgSimpleObj, 265
    gg::GgTriangles, 298
end
    gg::GgTrackball, 289
euler
    gg::GgQuaternion, 202
fill
    gg::GgUniformBuffer< T >, 303
frustum
    gg::GgMatrix, 114
get
    gg::GgMatrix, 114, 115
    gg::GgPointShader, 182
    gg::GgQuaternion, 203
    gg::GgShader, 259
    gg::GgShape, 263
    gg::GgSimpleObj, 266
    gg::GgTrackball, 289
    Window, 336
getAltArrowX
    Window, 336
getAltArrowY
    Window, 336
```

getAltArrow
 Window, 336

getArrow
 Window, 337

getArrowX
 Window, 337

getArrowY
 Window, 338

getAspect
 Window, 338

getBuffer
 `gg::GgBuffer< T >`, 92

`gg::GgPoints`, 178

`gg::GgTriangles`, 298

`gg::GgUniformBuffer< T >`, 304

getConjugateMatrix
 `gg::GgQuaternion`, 204, 205

getControlArrow
 Window, 338

getControlArrowX
 Window, 339

getControlArrowY
 Window, 339

getCount
 `gg::GgBuffer< T >`, 92

`gg::GgPoints`, 178

`gg::GgTriangles`, 299

`gg::GgUniformBuffer< T >`, 304

getHeight
 `gg::GgTexture`, 283

 Window, 339

getIndexBuffer
 `gg::GgElements`, 103

getIndexCount
 `gg::GgElements`, 103

getKey
 Window, 340

getMatrix
 `gg::GgQuaternion`, 205, 206

`gg::GgTrackball`, 289

getMode
 `gg::GgShape`, 263

getMouse
 Window, 340, 341

getMouseX
 Window, 341

getMouseY
 Window, 341

getQuaternion
 `gg::GgTrackball`, 290

getRotation
 Window, 341

getRotationMatrix
 Window, 342

getScale
 `gg::GgTrackball`, 290, 291

getScrollMatrix
 Window, 342

getShiftArrow
 Window, 343

getShiftArrowX
 Window, 343

getShiftArrowY
 Window, 343

getSize
 `gg::GgTexture`, 283

 Window, 343, 344

getStart
 `gg::GgTrackball`, 291, 292

getStride
 `gg::GgBuffer< T >`, 93

`gg::GgUniformBuffer< T >`, 304

getTarget
 `gg::GgBuffer< T >`, 93

`gg::GgUniformBuffer< T >`, 304

getTexture
 `gg::GgTexture`, 284

getTranslation
 Window, 344

getTranslationMatrix
 Window, 344

getUserPointer
 Window, 345

getWheel
 Window, 345, 346

getWheelX
 Window, 346

getWheelY
 Window, 346

getWidth
 `gg::GgTexture`, 284

 Window, 346

gg, 15

- `ggError`, 21
- `ggFBOError`, 22
- `BindingPoints`, 21
- `ggArraysObj`, 22
- `ggBufferAlignment`, 88
- `ggConjugate`, 23
- `ggCreateComputeShader`, 23
- `ggCreateNormalMap`, 24
- `ggCreateShader`, 25
- `ggCross`, 25, 26
- `ggDistance3`, 26
- `ggDistance4`, 27
- `ggDot3`, 28
- `ggDot4`, 29
- `ggElementsMesh`, 30
- `ggElementsObj`, 31
- `ggElementsSphere`, 31
- `ggEllipse`, 33
- `ggEulerQuaternion`, 33, 34
- `ggFrustum`, 35
- `ggIdentity`, 36
- `ggIdentityQuaternion`, 36
- `ggInit`, 36

ggInvert, 37
ggLength3, 38, 39
ggLength4, 40
ggLoadComputeShader, 41
ggLoadHeight, 41
ggLoadImage, 42
ggLoadShader, 43
ggLoadSimpleObj, 43, 44
ggLoadTexture, 45
ggLookat, 46, 47
ggMatrixQuaternion, 48, 49
ggNorm, 49
ggNormal, 50
ggNormalize, 51
ggNormalize3, 51–53
ggNormalize4, 53, 54
ggOrthogonal, 55
ggPerspective, 56
ggPointsCube, 57
ggPointsSphere, 57
ggQuaternion, 58
ggQuaternionMatrix, 59
ggQuaternionTransposeMatrix, 60
ggReadImage, 60
ggRectangle, 61
ggRotate, 62–64
ggRotateQuaternion, 65, 66
ggRotateX, 67
ggRotateY, 68
ggRotateZ, 69
ggSaveColor, 69
ggSaveDepth, 70
ggSaveTga, 71
ggScale, 71–73
ggSlerp, 73–75
ggTranslate, 77, 78
ggTranspose, 79
GgVector, 21
LightBindingPoint, 21
MaterialBindingPoint, 21
operator*, 80, 81
operator*+, 81
operator+, 82, 83
operator+=, 83, 84
operator-, 84, 85
operator-=, 85, 86
operator/, 86, 87
operator/=, 87, 88
gg.cpp, 353
gg.h, 354
gg::GgBuffer< T >, 90
 ~GgBuffer, 91
 bind, 92
 copy, 92
 getBuffer, 92
 getCount, 92
 getStride, 93
 getTarget, 93
 GgBuffer, 91
 map, 93, 94
 operator=, 94
 read, 94
 send, 95
 unbind, 95
 unmap, 95
 gg::GgColorTexture, 96
 ~GgColorTexture, 98
 GgColorTexture, 96, 97
 load, 98, 99
 gg::GgElements, 100
 ~GgElements, 102
 draw, 103
 getIndexBuffer, 103
 getIndexCount, 103
 GgElements, 101, 102
 load, 104
 send, 104
 gg::GgMatrix, 105
 ~GgMatrix, 111
 add, 111
 divide, 112, 113
 frustum, 114
 get, 114, 115
 GgMatrix, 109, 110
 GgQuaternion, 171
 invert, 116
 load, 116, 117
 loadAdd, 118
 loadDivide, 119, 120
 loadFrustum, 120
 loadIdentity, 121
 loadInvert, 122
 loadLookat, 123, 124
 loadMultiply, 125, 127
 loadNormal, 128
 loadOrthogonal, 129
 loadPerspective, 130
 loadRotate, 131–133
 loadRotateX, 134
 loadRotateY, 135
 loadRotateZ, 136
 loadScale, 136, 137
 loadSubtract, 138, 139
 loadTranslate, 139–141
 loadTranspose, 141, 142
 lookat, 143, 144
 multiply, 145, 146
 normal, 146
 operator*, 147, 148
 operator*+, 148
 operator+, 149
 operator+=, 150
 operator-, 151
 operator-=, 152
 operator/, 153
 operator/=, 154

operator=, 155
 orthogonal, 156
 perspective, 157
 projection, 158, 159
 rotate, 159–161
 rotateX, 162
 rotateY, 163
 rotateZ, 163
 scale, 164, 165
 subtract, 166, 167
 translate, 168, 169
 transpose, 170
 gg::GgNormalTexture, 171
 ~GgNormalTexture, 173
 GgNormalTexture, 172, 173
 load, 173, 174
 gg::GgPoints, 175
 ~GgPoints, 177
 draw, 177
 getBuffer, 178
 getCount, 178
 GgPoints, 176, 177
 load, 178
 send, 179
 gg::GgPointShader, 179
 ~GgPointShader, 181
 get, 182
 GgPointShader, 181
 load, 182
 loadMatrix, 183
 loadModelviewMatrix, 184
 loadProjectionMatrix, 185
 unuse, 186
 use, 186, 187
 gg::GgQuaternion, 188
 ~GgQuaternion, 195
 add, 196, 197
 conjugate, 198
 divide, 199–201
 euler, 202
 get, 203
 getConjugateMatrix, 204, 205
 getMatrix, 205, 206
 GgQuaternion, 193–195
 invert, 207
 load, 207–209
 loadAdd, 210–212
 loadConjugate, 213
 loadDivide, 214–216
 loadEuler, 217
 loadIdentity, 218
 loadInvert, 219, 220
 loadMatrix, 221
 loadMultiply, 223–225
 loadNormalize, 225, 226
 loadRotate, 227, 228
 loadRotateX, 229
 loadRotateY, 229
 loadRotateZ, 230
 loadSlerp, 231–233
 loadSubtract, 233–235
 multiply, 236, 237
 norm, 238
 normalize, 238
 operator*, 239, 240
 operator*=, 240, 241
 operator+, 241, 242
 operator+=, 243
 operator-, 244
 operator-=, 245, 246
 operator/, 246, 247
 operator/=, 248
 operator=, 249
 rotate, 250, 251
 rotateX, 252
 rotateY, 253
 rotateZ, 253
 slerp, 254
 subtract, 255, 256
 gg::GgShader, 258
 ~GgShader, 259
 get, 259
 GgShader, 258, 259
 operator=, 259
 unuse, 260
 use, 260
 gg::GgShape, 260
 ~GgShape, 261
 draw, 262
 get, 263
 getMode, 263
 GgShape, 261, 262
 operator=, 263
 setMode, 264
 gg::GgSimpleObj, 264
 ~GgSimpleObj, 265
 draw, 265
 get, 266
 GgSimpleObj, 265
 gg::GgSimpleShader, 266
 ~GgSimpleShader, 270
 GgSimpleShader, 269, 270
 load, 270
 loadMatrix, 271, 272
 loadModelviewMatrix, 273, 274
 operator=, 275
 use, 275–280
 gg::GgSimpleShader::Light, 311
 ambient, 311
 diffuse, 311
 position, 311
 specular, 312
 gg::GgSimpleShader::LightBuffer, 312
 ~LightBuffer, 315
 LightBuffer, 314
 load, 315

loadAmbient, 316, 317
loadColor, 317
loadDiffuse, 317, 318
loadPosition, 319, 320
loadSpecular, 320, 321
select, 322
gg::GgSimpleShader::Material, 322
 ambient, 323
 diffuse, 323
 shininess, 323
 specular, 323
gg::GgSimpleShader::MaterialBuffer, 324
 ~MaterialBuffer, 326
 load, 326, 327
 loadAmbient, 327
 loadAmbientAndDiffuse, 328
 loadDiffuse, 329
 loadShininess, 330
 loadSpecular, 330, 331
 MaterialBuffer, 325, 326
 select, 331
gg::GgTexture, 281
 ~GgTexture, 282
 bind, 282
 getHeight, 283
 getSize, 283
 getTexture, 284
 getWidth, 284
 GgTexture, 281, 282
 operator=, 285
 unbind, 285
gg::GgTrackball, 286
 ~GgTrackball, 288
 begin, 288
 end, 289
 get, 289
 getMatrix, 289
 getQuaternion, 290
 getScale, 290, 291
 getStart, 291, 292
 GgTrackball, 287
 motion, 292
 operator=, 292
 region, 293
 reset, 294
 rotate, 295
gg::GgTriangles, 295
 ~GgTriangles, 298
 draw, 298
 getBuffer, 298
 getCount, 299
 GgTriangles, 297
 load, 299
 send, 299
gg::GgUniformBuffer< T >, 300
 ~GgUniformBuffer, 302
 bind, 302
 copy, 303
 fill, 303
 getBuffer, 304
 getCount, 304
 getStride, 304
 getTarget, 304
 GgUniformBuffer, 301, 302
 load, 305
 map, 306
 read, 306
 send, 307
 unbind, 307
 unmap, 307
gg::GgVertex, 308
 GgVertex, 308–310
 normal, 310
 position, 310
GG_BUTTON_COUNT
 Window.h, 365
GG_INTERFACE_COUNT
 Window.h, 366
GG_USE_IMGUI
 GgApp.h, 361
GgApp, 89
 main, 89
GgApp.h, 360
 GG_USE_IMGUI, 361
ggArraysObj
 gg, 22
GgBuffer
 gg::GgBuffer< T >, 91
ggBufferAlignment
 gg, 88
GgColorTexture
 gg::GgColorTexture, 96, 97
ggConjugate
 gg, 23
ggCreateComputeShader
 gg, 23
ggCreateNormalMap
 gg, 24
ggCreateShader
 gg, 25
ggCross
 gg, 25, 26
ggDistance3
 gg, 26
ggDistance4
 gg, 27
ggDot3
 gg, 28
ggDot4
 gg, 29
GgElements
 gg::GgElements, 101, 102
ggElementsMesh
 gg, 30
ggElementsObj
 gg, 31

ggElementsSphere
 gg, 31
ggEllipse
 gg, 33
ggEulerQuaternion
 gg, 33, 34
ggFrustum
 gg, 35
ggIdentity
 gg, 36
ggIdentityQuaternion
 gg, 36
ggInit
 gg, 36
ggInvert
 gg, 37
ggLength3
 gg, 38, 39
ggLength4
 gg, 40
ggLoadComputeShader
 gg, 41
ggLoadHeight
 gg, 41
ggLoadImage
 gg, 42
ggLoadShader
 gg, 43
ggLoadSimpleObj
 gg, 43, 44
ggLoadTexture
 gg, 45
ggLookat
 gg, 46, 47
GgMatrix
 gg::GgMatrix, 109, 110
ggMatrixQuaternion
 gg, 48, 49
ggNorm
 gg, 49
ggNormal
 gg, 50
ggNormalize
 gg, 51
ggNormalize3
 gg, 51–53
ggNormalize4
 gg, 53, 54
GgNormalTexture
 gg::GgNormalTexture, 172, 173
ggOrthogonal
 gg, 55
ggPerspective
 gg, 56
GgPoints
 gg::GgPoints, 176, 177
ggPointsCube
 gg, 57

GgPointShader
 gg::GgPointShader, 181
ggPointsSphere
 gg, 57
GgQuaternion
 gg::GgMatrix, 171
 gg::GgQuaternion, 193–195
ggQuaternion
 gg, 58
ggQuaternionMatrix
 gg, 59
ggQuaternionTransposeMatrix
 gg, 60
ggReadImage
 gg, 60
ggRectangle
 gg, 61
ggRotate
 gg, 62–64
ggRotateQuaternion
 gg, 65, 66
ggRotateX
 gg, 67
ggRotateY
 gg, 68
ggRotateZ
 gg, 69
ggsample01.cpp, 361
 PROJECT.NAME, 362
ggSaveColor
 gg, 69
ggSaveDepth
 gg, 70
ggSaveTga
 gg, 71
ggScale
 gg, 71–73
GgShader
 gg::GgShader, 258, 259
GgShape
 gg::GgShape, 261, 262
GgSimpleObj
 gg::GgSimpleObj, 265
GgSimpleShader
 gg::GgSimpleShader, 269, 270
ggSlerp
 gg, 73–75
GgTexture
 gg::GgTexture, 281, 282
GgTrackball
 gg::GgTrackball, 287
ggTranslate
 gg, 77, 78
ggTranspose
 gg, 79
GgTriangles
 gg::GgTriangles, 297
GgUniformBuffer

gg::GgUniformBuffer< T >, 301, 302
GgVector
 gg, 21
GgVertex
 gg::GgVertex, 308–310
HEADER_STR
 main.cpp, 363
init
 Window, 347
invert
 gg::GgMatrix, 116
 gg::GgQuaternion, 207
LightBindingPoint
 gg, 21
LightBuffer
 gg::GgSimpleShader::LightBuffer, 314
load
 gg::GgColorTexture, 98, 99
 gg::GgElements, 104
 gg::GgMatrix, 116, 117
 gg::GgNormalTexture, 173, 174
 gg::GgPoints, 178
 gg::GgPointShader, 182
 gg::GgQuaternion, 207–209
 gg::GgSimpleShader, 270
 gg::GgSimpleShader::LightBuffer, 315
 gg::GgSimpleShader::MaterialBuffer, 326, 327
 gg::GgTriangles, 299
 gg::GgUniformBuffer< T >, 305
loadAdd
 gg::GgMatrix, 118
 gg::GgQuaternion, 210–212
loadAmbient
 gg::GgSimpleShader::LightBuffer, 316, 317
 gg::GgSimpleShader::MaterialBuffer, 327
loadAmbientAndDiffuse
 gg::GgSimpleShader::MaterialBuffer, 328
loadColor
 gg::GgSimpleShader::LightBuffer, 317
loadConjugate
 gg::GgQuaternion, 213
loadDiffuse
 gg::GgSimpleShader::LightBuffer, 317, 318
 gg::GgSimpleShader::MaterialBuffer, 329
loadDivide
 gg::GgMatrix, 119, 120
 gg::GgQuaternion, 214–216
loadEuler
 gg::GgQuaternion, 217
loadFrustum
 gg::GgMatrix, 120
loadIdentity
 gg::GgMatrix, 121
 gg::GgQuaternion, 218
loadInvert
 gg::GgMatrix, 122
 gg::GgQuaternion, 219, 220
loadLookat
 gg::GgMatrix, 123, 124
loadMatrix
 gg::GgPointShader, 183
 gg::GgQuaternion, 221
 gg::GgSimpleShader, 271, 272
loadModelviewMatrix
 gg::GgPointShader, 184
 gg::GgSimpleShader, 273, 274
loadMultiply
 gg::GgMatrix, 125, 127
 gg::GgQuaternion, 223–225
loadNormal
 gg::GgMatrix, 128
loadNormalize
 gg::GgQuaternion, 225, 226
loadOrthogonal
 gg::GgMatrix, 129
loadPerspective
 gg::GgMatrix, 130
loadPosition
 gg::GgSimpleShader::LightBuffer, 319, 320
loadProjectionMatrix
 gg::GgPointShader, 185
loadRotate
 gg::GgMatrix, 131–133
 gg::GgQuaternion, 227, 228
loadRotateX
 gg::GgMatrix, 134
 gg::GgQuaternion, 229
loadRotateY
 gg::GgMatrix, 135
 gg::GgQuaternion, 229
loadRotateZ
 gg::GgMatrix, 136
 gg::GgQuaternion, 230
loadScale
 gg::GgMatrix, 136, 137
loadShininess
 gg::GgSimpleShader::MaterialBuffer, 330
loadSlerp
 gg::GgQuaternion, 231–233
loadSpecular
 gg::GgSimpleShader::LightBuffer, 320, 321
 gg::GgSimpleShader::MaterialBuffer, 330, 331
loadSubtract
 gg::GgMatrix, 138, 139
 gg::GgQuaternion, 233–235
loadTranslate
 gg::GgMatrix, 139–141
loadTranspose
 gg::GgMatrix, 141, 142
lookat
 gg::GgMatrix, 143, 144
main
 GgApp, 89
 main.cpp, 363

main.cpp, 362
 HEADER_STR, 363
 main, 363
 map
 gg::GgBuffer< T >, 93, 94
 gg::GgUniformBuffer< T >, 306
 MaterialBindingPoint
 gg, 21
 MaterialBuffer
 gg::GgSimpleShader::MaterialBuffer, 325, 326
 motion
 gg::GgTrackball, 292
 multiply
 gg::GgMatrix, 145, 146
 gg::GgQuaternion, 236, 237
 norm
 gg::GgQuaternion, 238
 normal
 gg::GgMatrix, 146
 gg::GgVertex, 310
 normalize
 gg::GgQuaternion, 238
 operator bool
 Window, 347
 operator*
 gg, 80, 81
 gg::GgMatrix, 147, 148
 gg::GgQuaternion, 239, 240
 operator*=
 gg, 81
 gg::GgMatrix, 148
 gg::GgQuaternion, 240, 241
 operator+
 gg, 82, 83
 gg::GgMatrix, 149
 gg::GgQuaternion, 241, 242
 operator+=
 gg, 83, 84
 gg::GgMatrix, 150
 gg::GgQuaternion, 243
 operator-
 gg, 84, 85
 gg::GgMatrix, 151
 gg::GgQuaternion, 244
 operator-=
 gg, 85, 86
 gg::GgMatrix, 152
 gg::GgQuaternion, 245, 246
 operator/
 gg, 86, 87
 gg::GgMatrix, 153
 gg::GgQuaternion, 246, 247
 operator/=
 gg, 87, 88
 gg::GgMatrix, 154
 gg::GgQuaternion, 248
 operator=

gg::GgBuffer< T >, 94
 gg::GgMatrix, 155
 gg::GgQuaternion, 249
 gg::GgShader, 259
 gg::GgShape, 263
 gg::GgSimpleShader, 275
 gg::GgTexture, 285
 gg::GgTrackball, 292
 Window, 348
 orthogonal
 gg::GgMatrix, 156
 perspective
 gg::GgMatrix, 157
 position
 gg::GgSimpleShader::Light, 311
 gg::GgVertex, 310
 PROJECT_NAME
 ggsample01.cpp, 362
 projection
 gg::GgMatrix, 158, 159
 read
 gg::GgBuffer< T >, 94
 gg::GgUniformBuffer< T >, 306
 README.md, 364
 region
 gg::GgTrackball, 293
 reset
 gg::GgTrackball, 294
 Window, 348
 resetRotation
 Window, 348
 resetTranslation
 Window, 348
 restoreViewport
 Window, 348
 rotate
 gg::GgMatrix, 159–161
 gg::GgQuaternion, 250, 251
 gg::GgTrackball, 295
 rotateX
 gg::GgMatrix, 162
 gg::GgQuaternion, 252
 rotateY
 gg::GgMatrix, 163
 gg::GgQuaternion, 253
 rotateZ
 gg::GgMatrix, 163
 gg::GgQuaternion, 253
 scale
 gg::GgMatrix, 164, 165
 select
 gg::GgSimpleShader::LightBuffer, 322
 gg::GgSimpleShader::MaterialBuffer, 331
 selectInterface
 Window, 348
 send

gg::GgBuffer< T >, 95
gg::GgElements, 104
gg::GgPoints, 179
gg::GgTriangles, 299
gg::GgUniformBuffer< T >, 307
setClose
 Window, 349
setKeyboardFunc
 Window, 349
setMenubarHeight
 Window, 349
setMode
 gg::GgShape, 264
setMouseFunc
 Window, 350
setResizeFunc
 Window, 350
setUserPointer
 Window, 350
setVelocity
 Window, 351
setWheelFunc
 Window, 351
shininess
 gg::GgSimpleShader::Material, 323
shouldClose
 Window, 351
slerp
 gg::GgQuaternion, 254
specular
 gg::GgSimpleShader::Light, 312
 gg::GgSimpleShader::Material, 323
subtract
 gg::GgMatrix, 166, 167
 gg::GgQuaternion, 255, 256
swapBuffers
 Window, 352
translate
 gg::GgMatrix, 168, 169
transpose
 gg::GgMatrix, 170
unbind
 gg::GgBuffer< T >, 95
 gg::GgTexture, 285
 gg::GgUniformBuffer< T >, 307
unmap
 gg::GgBuffer< T >, 95
 gg::GgUniformBuffer< T >, 307
unuse
 gg::GgPointShader, 186
 gg::GgShader, 260
use
 gg::GgPointShader, 186, 187
 gg::GgShader, 260
 gg::GgSimpleShader, 275–280
Window, 332
 ~Window, 335
 get, 336
 getAltArrowX, 336
 getAltArrowY, 336
 getAltArrow, 336
 getArrow, 337
 getArrowX, 337
 getArrowY, 338
 getAspect, 338
 getControlArrow, 338
 getControlArrowX, 339
 getControlArrowY, 339
 getHeight, 339
 getKey, 340
 getMouse, 340, 341
 getMouseX, 341
 getMouseY, 341
 getRotation, 341
 getRotationMatrix, 342
 getScrollMatrix, 342
 getShiftArrow, 343
 getShiftArrowX, 343
 getShiftArrowY, 343
 getSize, 343, 344
 getTranslation, 344
 getTranslationMatrix, 344
 getUserPointer, 345
 getWheel, 345, 346
 getWheelX, 346
 getWheelY, 346
 getWidth, 346
 init, 347
 operator bool, 347
 operator=, 348
 reset, 348
 resetRotation, 348
 resetTranslation, 348
 restoreViewport, 348
 selectInterface, 348
 setClose, 349
 setKeyboardFunc, 349
 setMenubarHeight, 349
 setMouseFunc, 350
 setResizeFunc, 350
 setUserPointer, 350
 setVelocity, 351
 setWheelFunc, 351
 shouldClose, 351
 swapBuffers, 352
 Window, 335
Window.h, 364
 GG_BUTTON_COUNT, 365
 GG_INTERFACE_COUNT, 366