**gfl::core 算術クラス**

**【特徴】**

デフォルト行列の順番を AOS から SOA に変更しています。

（デフォルトをAOSにすることも可能です）

gfl::core 算術クラスは nn::math クラスを継承しています。

ネイティブクラスは他プラットフォーム対応のため typedef しています。

gfl::core::VectorStruct? == nn::math::VEC?\_

gfl::core::VectorNative? == nn::math::VEC?

gfl::core::MatrixStruct?? == nn::math::MTX??\_

gfl::core::MatrixNative?? == nn::math::MTX??

gfl::core::QuaternionStruct == nn::math::QUAT\_

gfl::core::QuaternionNative == nn::math::QUAT

デフォルトのベクトル、行列、クォータニオンクラスは

**gfl::core::Vector　gfl::core::Matrix　(gfl::core::Matrix44)　gfl::core::Quaternion**

になります。

基本的にこちらのクラスを使用するようにしてください。

（デフォルトベクトルが Vector4 のプラットフォームもあるので）

nn::math ネイティブクラスを汎用的に記述したクラスを作成したので(一部未実装ですが)

**VisualStudio、GCC、PS3やXbox360でもコンパイル、実行が可能**です。

（gfl::math も nn::math を継承（または置換）していますが、

他プラットフォームでコンパイル出来ません。）

算術のテストをPCでやると作業効率が上がると思います。

（PCでのテスト方法は　home/xy\_tools/CopyProject/ReadMe.txt を参照してください。）

テンプレートを使用していないので、

**コード補完が nn::math レベルまで完全に動作します。**

（共通のメソッドを別ファイルにしてクラス内にインクルードしています）

gfl::core 算術クラスの**メソッドは、基本的にネイティブクラスを引数として受け取る**ようになっています。

これは gfl::math のクラスもそのまま受け取れるということを意味しています。

ただ、gfl の描画命令等が gfl::math を受け取るようになっているので

それらの命令を呼ぶときには引数の明示的なキャストが必要になります。

（クラス内変数が gfl::core と gfl::math は同じなのでキャストをしても安全です。）

各パラメータのアクセスは**ゲッター、セッターかGFL\_VECTOR\_X 等のマクロ**(gfl\_MathBase.hに定義)を使用してください。

プラットフォームが変わった場合に変更をしなくて済みます。

**【注意点】**

変更に伴う注意点です。

**１）ToSOA**等の変換命令は必要なくなります。

gfl::core::Matrix の SetRotateZYX と nn::math::MTX34 の SetRotateXyz

と同じ結果になりますので、角度を取り出して回しなおす必要なく

そのまま SetRotateMatrix すれば大丈夫です。

**２）**SOAにしたために、**matrix->GetTrans()** の返り値が

ポインタから実体に変わっています。

この部分はプログラムの書き換えが必要になります。

Gflib xy\_project は変更しましたが、**xy\_tools** で使用されていれば

書き換えが必要になりますので、申し訳ありませんが**各自**で変更をお願いします。

**３）**直接行列の成分にアクセスしている場合には、場合分けが必要になります。

#if GFL\_DEFAULT\_AOS

f32 sin\_y = con\_mat->Get( 2, 0 );

#else

f32 sin\_y = con\_mat->Get( 0, 2 );

#endif

現在の Matrix がAOS か SOA は **GFL\_DEFAULT\_AOS** の真偽で判断してください。

**４）OpenGL** を使用されている方への注意点です。

AOSとSOAの違いを吸収したGL命令を用意したので

下記のものに変更してください。

これをやらないとポリゴンが壊れます。

**glLoadMatrixf(f32\*) => GlLoadMatrix(const Matrix&)**

**glMultMatrixf(f32\*) => GlMultMatrix(const Matrix&)**

**５）**基底クラスと挙動が違うところがあります。

Quaternion の引き算は単純な成分の引き算でなく、Quaternionの差分になります。

つまり q0 \* (q1 - q0) = q1 になります。

補完の計算にはこちらのほうが都合がいいので、意図的にこうしてあります。

以上。