| LCOV - code coverage report |
| --- |
|  |
| | Current view: | [top level](http://docs.google.com/index.html) - [LibGMC/src](http://docs.google.com/index.html) - GMCorrector.cpp (source / [functions](http://docs.google.com/GMCorrector.cpp.func.html)) |  |  | Hit | Total | Coverage | | --- | --- | --- | --- | --- | --- | --- | | Test: | EBM9K Converter Coverage Report |  | Lines: | 72 | 78 | 92.3 % | | Date: | 2012-03-23 |  | Functions: | 13 | 16 | 81.2 % | | Legend: | Lines: hit not hit | Branches: + taken - not taken # not executed |  | Branches: | 22 | 30 | 73.3 % | |  | |
|  |

|  |
| --- |
| Branch data Line data Source code  1 : :   2 : : #include <math.h>  3 : : #include "GMCorrector.h"  4 : : #include "GMCMap.h"  5 : : #include "print.h"  6 : : #include "Logger.h"  7 : : #include "ErrorHandler.h"  8 : :   9 : : namespace {  10 : : inline double hornerEvaluate( size\_t maxIndex  11 : : , size\_t minIndex  12 : : , size\_t ratio  13 : : , double multiplier  14 : : , const double \*polynomial )  15 : 2184 : {  16 : :   17 : 910 : double result = 0.0L;  18 : :   19 [ + + ][ + + ]: 2548 : for( register size\_t index = maxIndex; index >= minIndex; )  [ + + ][ + + ]  [ + - ]  20 : : {  21 : 2548 : result += polynomial[ index ];  22 : 2548 : result \*= multiplier;  23 : :   24 : : //This is not part of horner's algo  25 : : //but derived from formula supplied by NFT  26 [ + - ][ + - ]: 2548 : if( ratio > index )  [ + - ][ + - ]  [ + + ]  27 : 2366 : break;  28 : :   29 : 2366 : index -= ratio;  30 : 2366 : ratio--;  31 : 910 : }  32 : :   33 : : return result;  34 : : }  35 : :   36 : : }  37 : :   38 : : namespace nft {  39 : :   40 : : DoublePosition  41 : : GMCorrector::calculate(const LongPosition & inPosition  42 : : , const GMCMap \* inGmcData  43 : : , const sgs\_gmc\_param\_t & inGmcParam  44 : : , GMCorrector::GMCMode inMode) const  45 : 201 : {  46 : : //Check validity  47 : : //only in debug mode  48 : : #ifdef DEBUG  49 : : if( eGMCMap == inMode )  50 : : {  51 : : if( NULL == inGmcData )  52 : : {  53 : : e9k\_debug("No data for GMC although param is set to MAP\n");  54 : : }  55 : : }  56 : : #endif  57 : 201 : DoublePosition retPos;  58 [ + + ]: 201 : if( eGMCMap == inMode )  59 : : {  60 : 110 : return mapModeCorrection( inPosition, \*inGmcData, inGmcParam );  61 : : }  62 : 91 : return polynomialModeCorrection( inPosition, inGmcParam );  63 : : }  64 : :   65 : :   66 : : GMCorrector::GMCorrector()  67 : 2162 : :mXValueToBeChecked(),  68 : 2162 : mYValueToBeChecked() {  69 : :   70 : : }  71 : :   72 : :   73 : 2162 : GMCorrector::~GMCorrector(){  74 : :   75 : : }  76 : :   77 : :   78 : 0 : GMCorrector::GMCorrector(const GMCorrector & source){  79 : :   80 : : }  81 : :   82 : : GMCorrector &  83 : 0 : GMCorrector::operator =(const GMCorrector & source) {  84 : 0 : return \*this;  85 : : }  86 : :   87 : : DoublePosition  88 : : GMCorrector::mapModeCorrection(  89 : : const LongPosition & inPosition  90 : : , const GMCMap & gmcData  91 : : , const sgs\_gmc\_param\_t & gmcParam) const  92 : 110 : {  93 : 110 : double originX = (gmcParam.map).pos\_x ;  94 : 110 : double meshSizeX = (gmcParam.map).mesh\_size\_x ;  95 : 110 : double originY = (gmcParam.map).pos\_y ;  96 : 110 : double meshSizeY = (gmcParam.map).mesh\_size\_y ;  97 : :   98 : : uint32\_t iX = static\_cast<uint32\_t>(  99 : 110 : (inPosition.getX() - originX) / meshSizeX );  100 : :   101 : : uint32\_t iY = static\_cast<uint32\_t>(  102 : 110 : (inPosition.getY() - originY) / meshSizeY );  103 : :   104 : : double sX = calculateDisplacement(   105 : 110 : inPosition.getX(), iX, originX, meshSizeX );  106 : :   107 : : double sY = calculateDisplacement(   108 : 110 : inPosition.getY(), iY, originY, meshSizeY );  109 : :   110 : 110 : double correctedPosX = 0.0;  111 : 110 : double correctedPosY = 0.0;  112 : :   113 : :   114 : : //X corrections  115 : : {  116 : 110 : const RowMajorDoubleTableReader \*pTableX = gmcData.getReaderX();  117 : 110 : correctedPosX = calculateMapCorrectedValue(   118 : : iY, iX, sX, sY, pTableX );  119 : 110 : mXValueToBeChecked += correctedPosX;  120 : : }  121 : :   122 : : //Y corrections  123 : : {  124 : 110 : const RowMajorDoubleTableReader \*pTableY = gmcData.getReaderY();  125 : 110 : correctedPosY = calculateMapCorrectedValue(   126 : : iY, iX, sX, sY, pTableY );  127 : 110 : mYValueToBeChecked += correctedPosY;  128 : : }  129 : 110 : Logger::instance()->log(LogMessageTable::eSHOT\_GMC\_MAP\_MODE,  130 : : correctedPosX, correctedPosY);  131 : :   132 : 110 : return DoublePosition( correctedPosX, correctedPosY );  133 : :   134 : : }  135 : :   136 : : double  137 : : GMCorrector::calculateDisplacement(  138 : : int64\_t position  139 : : , int64\_t iPos  140 : : , double origin  141 : : , double meshSize) const  142 : 660 : {  143 : 220 : double posWRTOrigin = static\_cast<double>(position) - origin;  144 : 220 : double posWRTLastMesh = posWRTOrigin - iPos \* meshSize;  145 : :   146 : 0 : return posWRTLastMesh / meshSize;  147 : : }  148 : :   149 : : double  150 : : GMCorrector::calculateMapCorrectedValue(  151 : : int64\_t column  152 : : , int64\_t row  153 : : , double sX  154 : : , double sY  155 : : , const RowMajorDoubleTableReader \* pReader) const  156 : 440 : {  157 : 220 : double d1 = pReader->get( column , row );  158 : 220 : double d2 = pReader->get( column, row + 1 );  159 : 220 : double d3 = pReader->get( column + 1, row );  160 : 220 : double d4 = pReader->get( column + 1, row + 1 );  161 : :   162 : 0 : return d1 + sX \* ( d2 - d1 ) + sY \* ( d3 - d1 )   163 : : + sX \* sY \* ( d1 - d2 - d3 + d4 );  164 : : }  165 : :   166 : : DoublePosition  167 : : GMCorrector::polynomialModeCorrection(  168 : : const LongPosition & sfPosition,  169 : : const sgs\_gmc\_param\_t & gmcParam) const  170 : 201 : {  171 : : double correctedPosX = calculatePolycorrectedValue(   172 : 91 : sfPosition, gmcParam.poly\_x);  173 : 91 : correctedPosX += gmcParam.poly\_x[0];  174 : 91 : mXValueToBeChecked += correctedPosX;  175 : :   176 : : double correctedPosY = calculatePolycorrectedValue(   177 : 91 : sfPosition, gmcParam.poly\_y);  178 : 91 : correctedPosY += gmcParam.poly\_y[0];  179 : 91 : mYValueToBeChecked += correctedPosY;  180 : :   181 : 91 : Logger::instance()->log(LogMessageTable::eSHOT\_GMC\_POLY\_MODE,  182 : : correctedPosX, correctedPosY);  183 : :   184 : 0 : return DoublePosition( correctedPosX , correctedPosY );  185 : :   186 : : }  187 : :   188 : : double  189 : : GMCorrector::calculatePolycorrectedValue(const LongPosition & position,  190 : 182 : const double \*polynomial) const {  191 : 182 : double result = 0.0;  192 : : //Calculate component with only X  193 : 182 : result += hornerEvaluate( 10, 1, 4, position.getX(), polynomial);  194 : :   195 : : //Calculate component with only Y  196 : 182 : result += hornerEvaluate( 14, 2, 5, position.getY(), polynomial);  197 : :   198 : : //Calculate component with xy   199 : :   200 : 182 : double product = position.getX() \* position.getY();  201 : :   202 : 182 : result += product  203 : : \* ( hornerEvaluate( 11, 7, 4, position.getX(), polynomial)  204 : : + hornerEvaluate( 13, 8, 5, position.getY(), polynomial) )  205 : : + hornerEvaluate( 12, 4, 8, product, polynomial );  206 : :   207 : 182 : return result;  208 : : }  209 : :   210 : : void GMCorrector::initialize() const throw()  211 : 142 : {  212 : 142 : mXValueToBeChecked = 0;  213 [ + - ]: 154 : mYValueToBeChecked = 0;  214 : : }  215 : :   216 : : void GMCorrector::check( const sgs\_gmc\_param\_t & gmcParam,   217 : : const LongPosition & position ) const throw(DPL::Error \*)  218 : 154 : {  219 [ + - ]: 154 : double maxCorrect = fabs( gmcParam.max\_allowable\_value );  220 [ + + ][ - + ]: 154 : if( ( fabs( mXValueToBeChecked ) > maxCorrect )   221 : : || ( fabs( mYValueToBeChecked ) > maxCorrect ) )  222 : : {  223 : 12 : Logger::instance()->log(LogMessageTable::eSHOT\_ERROR\_GMC,  224 : : mXValueToBeChecked, mYValueToBeChecked, position.getX(),  225 : : position.getY(), maxCorrect);  226 : :   227 : 12 : ErrorHandler \*pErr = ErrorHandler::instance();  228 : 12 : pErr->errorAdd(NULL, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  229 : : ErrorHandler::eErrGMCValueBeyondLimit,  230 : : pErr->getErrorMsg((ErrorHandler::eErrGMCValueBeyondLimit),  231 : : mXValueToBeChecked, mYValueToBeChecked, position.getX(),  232 : : position.getY(), maxCorrect));  233 : : }  234 : 142 : initialize();  235 : : }  236 : :   237 : 1215 : } // namespace nft |

|  |
| --- |
| Generated by: [LCOV version 1.9](http://ltp.sourceforge.net/coverage/lcov.php) |