**Math Library Documentation**

Library: MathLib.dll (dynamic version) or MathLib.lib(static version)

File Dependencies:

Vector3D.h, Vector4D.h, Matrix3D.h, Matrix4D.h, MiscMath.h

Vector3D.cpp, Vector4D.cpp, Matrix3D.cpp, Matrix4D.cpp, MiscMath.cpp

Disclaimer:

MathLib is in beta, many ideal functionalities are missing or do not function as intended. NOT ALL FUNCTIONS ARE INCLUDED IN THE TABLE DOCUMENTATIONS.

Library Overview:

Math Library that performs limited math functionality for the following:

\* 3 Dimensional Vectors

\* 4 Dimensional Vectors

\* 3 x 3 Dimension Matrices

\* 4 x 4 Dimension Matrices

\* Miscellaneous Math

MathLib was developed as a requirement for the first year game programming course at The Academy of Interactive Entertainment(US)

Class Overview:

*3 Dimensional Vectors*

The Vector3D class contains the following features…

\* Addition and Subtraction (vector by vector)

\* Multiplication (vector by Vector, vector by Scalar)

\* Division (vector by scalar)

\* Normalization of a vector

\* Dot Product of two vectors

\* Cross Product of two vectors

\* Euler Angle between two vectors

\* Linear Interpolation for two vectors at a given time

*4 Dimensional Vectors*

The Vector4D class contains the following features…

\* Normalization of a vector

\* Construction from RGBA (Red, Green, Blue, Alpha) hexadecimal color value (format: 0xFFFFFFFF)

*3 x 3 Dimension Matrices*

The Matrix3D class contains the following features…

\* Addition and Subtraction (matrix by matrix)

\* Multiplication (matrix by matrix, matrix by scalar, Matrix by Vector)

\* Conversion to Zero Matrix

\* Conversion to Identity Matrix

\* Rotation Matrix Creation (by providing an angle)

\* Translation Matrix Creation (by providing a translation vector as a homogenous coordinate (x,y,1) )

\* Scaling Matrix Creation (by providing a scaling vector as a homogenous coordinate (x,y,1) )

\* Point Transformation (by providing an angle, a translation vector, and a scaling vector)

*4 x 4 Dimension Matrices*

The Matrix4D class contains the following features…

\* Addition and Subtraction (matrix by matrix)

\* Multiplication (matrix by matrix, matrix by scalar, Matrix by Vector)

\* Conversion to Zero Matrix

\* Conversion to Identity Matrix

\* X,Y,or Z Rotation Matrix Creation (by providing an angle)

\* Translation Matrix Creation (by providing a translation vector as a homogenous coordinate (x,y,z,1) )

\* Scaling Matrix Creation (by providing a scaling vector as a homogenous coordinate (x,y,z,1) )

\* Point Transformation (by providing an angle, a translation vector, and a scaling vector)

\* Vector Transformation(by providing an angle and a scaling vector)

*Miscellaneous Math*

The MiscMath class contains the following features…

\* Conversion from Degrees to Radians

\* Conversion from Radians to Degrees

\* Finding the next power of two.

\* Linear Interpolation.

Class Details and Testing

*Vector3D*

Vector3D objects are stored with three floating point values (x,y,z) they are used primarily as homogenous coordinates (x,y,1).

Functions/Methods

KEY: v3 = Vector3D, x y z = x,y,z elements of a Vector3D object, ret = return

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ret. Type** | **Name** | **Args Types** | **This** | **Input** | **Result** |
| v3 | operator + | v3 | 1 2 3 | 4 4 4 | ret 5 6 7 |
| v3 | operator - | v3 | 1 2 3 | 4 4 4 | ret -3 -2 -1 |
| v3 | operator \* | float | 1 2 3 | 2 | ret 2 4 6 |
| v3 | operator / | float | 1 2 3 | 2 | ret .5 1 1.5 |
| float | Magnitude |  | 1 2 3 |  | ret 3.741... |
| void | Normalize |  | 1 2 3 |  | this = 267… .534 .801 |
| float | Dot | v3 | 1 2 3 | 4 4 4 | ret 24 |
| v3 | Cross | v3 | 1 2 3 | 4 4 4 | ret -4 8 -4 |
| float | Euler | v3 | 1 2 3 | 4 4 4 | ret 22.20.. degrees (in radians) |
| v3 | Interpolate | v3, float t | 1 2 3 | 4 4 4, .5 | 2.5 3 3.5 |

*Vector4D*

Vector4D Objects are stored as four floating point variables (x,y,z,w) and are used primarily as homogenous coordinates (x,y,z,1).

Functions/Methods

KEY: v4 = Vector4D, x y z w = x,y,z,w elements of a Vector4D object, ret = return

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ret. Type** | **Name** | **Arg Types** | **This** | **Input** | **Result** |
| v4 | Normalize |  | *2,2,2,1* |  | *.5547 .5547 .5547 .2275* |
| *v4* | *ConstructFromColor* | *int hexcolor* |  | *0xFF0000FF* | *ret 1 0 0 1* |

*Matrix3D*

Matrix3D Objects are stored as 9 floating point variables which can be accessed individually, through a 2D 4x4 Array, or through a one dimensional 9 element array. Matrices are used in tangent with homogenous coordinates (Vector3Ds) to perform transformations on coordinate values. Of course Matrices are also used to manipulate

one another.

Functions/Methods

KEY:

m3 = Matrix3D

v3 = Vector3D

1 2 3

4 5 6

7 8 9 = Representation of Matrix3D Object’s element values

ret = return

n = cos(theta), rounding issue causes n to equal a number very close to but not equal to 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ret.Type | Name | Arg Types | This/RHS | Input | Result |
| m3 | operator+ | m3 | 1 2 3  4 5 6  7 8 9 | 1 1 1  1 1 1  1 1 1 | 2 3 4  5 6 7  8 9 10 |
| m3 | operator- | m3 | 1 2 3  4 5 6  7 8 9 | 1 1 1  1 1 1  1 1 1 | 0 1 2  3 4 5  6 7 8 |
| m3 | operator\* | float | 1 2 3  4 5 6  7 8 9 | 2 | 2 4 6  8 10 12  14 16 18 |
| m3 | operator\* | m3 | 1 2 3  4 5 6  7 8 9 | 1 1 1  1 1 1  1 1 1 | 6 6 6  15 15 15  24 24 24 |
| v3 | operator\* | m3 | 1 2 3 | 1 2 3  4 5 6  7 8 9 | 14 32 50 |
| void | SetZero |  | 1 2 3  4 5 6  7 8 9 |  | 0 0 0  0 0 0  0 0 0 |
| void | SetIdentity |  | 1 2 3  4 5 6  7 8 9 |  | 1 0 0  0 1 0  0 0 1 |
| m3 | CreateRotation | float angle | Does not matter. | PI / 2 | n -1 0  -1 n 0  0 0 1 |
| m3 | CreateTranslation | v3 trans | “ “ | 1 1 1 | 1 0 1  0 1 0  0 0 1 |
| m3 | CreateScaling | v3 scale | “ “ | 1 5 1 | 2 0 0  0 2 0  0 0 1 |
| void | TransformPoint | float angle,  v3 trans, v3 scale, | 1 3 1 | PI/2,  1 5,  2 2 | -5 7 1 |

*Matrix 4D*

Matrix4D Objects are created, stored, and accessed the same way as Matrix3D Objects. They have 16 elements instead of 9. Matrix4D objects use the same functions as Matrix3Ds except they replace Vector3D arguments with Vector4D arguments.

Functions unique to Matrix4D are provided in the table below.

m4 = Matrix4D

v4 = Vector4D

1 2 3 5

6 6 7 8

9 10 11 12

13 14 15 16 = Representation of Matrix4D Object’s element values

ret = return

n = cos(theta), rounding issue causes n to equal a number very close to but not equal to 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ret Type | Name | Arg Types | This/RHS | Input | Result |
| m4 | CreateRotationX | float |  | PI/2 | 1 0 0 0  0 n -1 0  0 1 n 0  0 0 0 1 |
| m4 | CreateRotationY | float |  | PI/2 | n 0 1 0  0 1 0 0  -1 0 n 0  0 0 0 1 |
| m4 | CreateRotationZ | float |  | PI/2 | n -1 0 0  1 n 0 0  0 0 1 0  0 0 0 1 |
| void | TransformVector | float angle x,  float angle y,  float angle z,  v4 scale | 1 2 5 1 | PI/2,  0,  0,  2 2 2 | 2 -10 6 1 |
| m4 | CreateOrtho | float rs,  float ls,  float top,  float bottom,  float far,  float near |  |  |  |