Math Library Documentation

Classes:

Vector 2 – Contains the functions using Vector 2 maths

Vector 3 – Contains the functions using Vector 3 maths

Vector 4 – Contains the functions using Vector 4 maths

Matrix 3 – Contains the functions using Matrix 3 maths

Matrix 4 – Contains the functions using Matrix 4 maths

Maths – Contains various functions that have no place anywhere else but are useful

Conversions – Contains Binary, Decimal and Hexadecimal conversion

Bitset Array – Contains definition and functionality for the Bitset array, used for storing binary information

Vector 2 Class – Vec2.h

Vec2()
Default constructor, creates a vector of (0.f, 0.f)

Vec2(float *a_fpV2)
Constructor that will create a vector of (a_fpV2[1], a_fpV2[2])

Vec2(const float a_fX, const float a_fY)
Constructor that will create a vector of (a_fX, a_fY)

operator float * ()
Casts the vector to a float array

operator const float * () const
Casts the vector to a float array constant

void **operator=**(const**Vec2**&a_Vec1)
Sets the vector to the vector of a_Vec1

bool **operator==**(const**Vec2**&a_Vec1)
Compares the vector to a vector of a_Vec1 and returns true if the vectors are the same

Vec2 operator+(const**Vec2**&a_Vec1)
Returns the result of the vector components added to the components of a_Vec1

Vec2 operator+(const float a_fAdd)
Returns the result of the float a_fAdd added to the components of the vector

void **operator+=**(const**Vec2**&a_Vec1)
Adds the components of a_Vec1 to the components of the original vector

void **operator+=**(const float a_fAdd)
Adds the float a_fAdd to the components of the original vector

Vec2 operator-(const**Vec2**&a_Vec1)
Returns the result of the vector components subtracted from the components of a_Vec1

Vec2 operator-(const float a_fSub)
Returns the result of the float a_fSub subtracted from the components of the vector

void **operator-=**(const**Vec2**&a_Vec1)
Subtracts the components of a_Vec1 from the components of the original vector

void **operator-=**(const float a_fSub)
Subtracts the float a_fSub from the components of the original vector

Vec2 operator/(const float a_fDiv)
Returns the result of the components of the vector divided by a_fDiv

Vec2 operator/(const**Vec2**&a_Vec1)
Returns the result of the vector components divided by the components of a_Vec1

void **operator/=**(const float a_fDiv)
Divides the components of the original vector by a_fDiv

void **operator/=**(const**Vec2**&a_Vec1)
Divides the components of the original vector by the components of a_Vec1

Vec2 operator* (const float a_fMul)
Returns the result of the components of the original vector multiplied by a_fMul

Vec2 operator* (const**Vec2**&a_Vec1)
Returns the result of the components of the original vector multiplied by the components of a_Vec1

void **operator* =**(const float a_fMul)
Multiplies the components of the original vector by a_fMul

void **operator* =**(const**Vec2**&a_Vec1)
Multiplies the components of the original vector by the components of a_Vec1

float **GetMagnitude**()
Returns the distance between a point and the origin (0.f, 0.f)

Vec2 NormaliseVec2()
Normalises the original vector returning values between 0 and 1

Vec2 LinearInterpolation(const**Vec2**&a_pVec1, float a_fDest)

Vec2 RotateVec2(const**Vec2**&a_pVec1, float a_fRotationInDegrees)
Rotates the point of the original vector around a_pVec1 by a_fRotationInDegrees

float **DotProduct**(const**Vec2**&a_pVec1)
Returns a float of the product of two vectors

float **GetAngle**(const**Vec2**&a_pVec1)
Gets the angle between the original vector as a point and the point of a_pVec1

Vector 3 Class – Vec3.h

Vec3()
Default constructor, creates a vector of (0.f, 0.f, 0.f)

Vec3(float a_fX, float a_fY, float a_fZ)
Constructor that will create a vector of (a_fX, a_fY, a_fZ)

operator float *()
Casts the vector to a float array

operator const float *() const
Casts the vector to a float array consant

float **GetMagnitude**()
Returns the distance between a point and the origin (0.f, 0.f, 0.f)

Vec3 NormaliseVec3()
Normalises the original vector components returning values between 0 and 1

Vec3 LinearInterpolation(const**Vec3**&a_pVec1, float a_fDest)

float **DotProduct**(const**Vec3**&a_pVec1)
Returns a float of the product of two vectors

float **GetAngle**(const**Vec3**&a_pVec1)
Gets the angle between the original vector as a point and the point of a_pVec1

void **operator=**(const**Vec3**&a_Vec1)
Sets the vector to the vector of a_Vec1

bool **operator==**(const**Vec3**&a_Vec1)
Compares the vector to a vector of a_Vec1 and returns true if the vectors are the same

Vec3 operator+(const**Vec3**&a_Vec1)
Returns the result of the vector components added to the components of a_Vec1

Vec3 operator+(const float a_fAdd)
Returns the result of the float a_fAdd added to the components of the vector

void **operator+=**(const**Vec3**&a_Vec1)
Adds the components of a_Vec1 to the components of the original vector

void **operator+=**(const float a_fAdd)
Adds the float a_fAdd to the components of the original vector

Vec3 operator-(const**Vec3**&a_Vec1)
Returns the result of the vector components subtracted from the components of a_Vec1

Vec3 operator-(const float a_fSub)
Returns the result of the float a_fSub subtracted from the components of the vector

void **operator-=**(const**Vec3**&a_Vec1)
Subtracts the components of a_Vec1 from the components of the original vector

void **operator-=**(const float a_fSub)
Subtracts the float a_fSub from the components of the original vector

Vec3 operator/(const float a_fDiv)
Returns the result of the components of the vector divided by a_fDiv

Vec3 operator/(const**Vec3**&a_Vec1)
Returns the result of the vector components divided by the components of a_Vec1

void **operator/=**(const float a_fDiv)
Divides the components of the original vector by a_fDiv

void **operator/=**(const**Vec3**&a_Vec1)
Divides the components of the original vector by the components of a_Vec1

Vec3 operator* (const float a_fMul)
Returns the result of the components of the original vector multiplied by a_fMul

Vec3 operator* (const**Vec3**&a_Vec1)
Returns the result of the components of the original vector multiplied by the components of a_Vec1

void **operator* =**(const float a_fMul)
Multiplies the components of the original vector by a_fMul

void **operator* =**(const**Vec3**&a_Vec1)
Multiplies the components of the original vector by the components of a_Vec1

Vector 4 Class – Vec4.h

Vec4()
Default constructor, creates a vector of (0.f, 0.f, 0.f, 0.f)

Vec4(float a_fX, float a_fY, float a_fZ, float a_fW)
Constructor that will create a vector of (a_fX, a_fY, a_fZ, a_fW)

Vec4(unsigned int a_uiHex)
Creates a vector based on a hexadecimal value

float **GetMagnitude**()
Returns the distance between a point and the origin (0.f, 0.f, 0.f, 0.f)

Vec4 NormaliseVec4()
Normalises the original vector returning values between 0 and 1

void **operator=**(const**Vec4**&a_Vec1)
Sets the vector to the vector of a_Vec1

bool **operator==**(const**Vec4**&a_Vec1)
Compares the vector to a vector of a_Vec1 and returns true if the vectors are the same

Vec4 operator+(const**Vec4**&a_Vec1)
Returns the result of the vector components added to the components of a_Vec1

Vec4& **operator+=**(const**Vec4**&a_Vec1)
Adds the components of a_Vec1 to the components of the original vector

Vec4 operator-(const**Vec4**&a_Vec1)
Returns the result of the vector components subtracted from the components of a_Vec1

Vec4& **operator-=**(const**Vec4**&a_Vec1)
Subtracts the components of a_Vec1 from the components of the original vector

Vec4 operator/(const float a_fDiv)
Returns the result of the components of the vector divided by a_fDiv

Vec4 operator/(const**Vec4**&a_Vec1)
Returns the result of the vector components divided by the components of a_Vec1

Vec4& **operator/=**(const float a_fDiv)
Divides the components of the original vector by a_fDiv

Vec4& **operator/=**(const**Vec4**&a_Vec1)
Divides the components of the original vector by the components of a_Vec1

Vec4 operator* (const float a_fMul)
Returns the result of the components of the original vector multiplied by a_fMul

Vec4 operator* (const**Vec4**&a_Vec1)
Returns the result of the components of the original vector multiplied by the components of a_Vec1

Vec4& **operator*=**(const float a_fMul)
Multiplies the components of the original vector by a_fMul

Vec4& **operator*=**(const**Vec4**&a_Vec1)
Multiplies the components of the original vector by the components of a_Vec1

Matrix 3 Class – Matrix3.h

Matrix3()
Creates an empty Matrix with 9 elements of 0

Matrix3(float a_fMatrixArray[3][3])
Creates a Matrix that has the elements of a_fMatrixArray

Matrix3(const float &a_f11, const float &a_f12, const float &a_f13,
const float &a_f21, const float &a_f22, const float &a_f23,
const float &a_f31, const float &a_f32, const float &a_f33)
Creates a Matrix using the function arguments

operator float * ()
Casts the Matrix to a float pointer

operator const float * () const
Casts the matrix to a constant float pointer

Matrix3 IdentityMatrix()
Returns an Identity Matrix

Matrix3 OrthoView()
Returns an Orthogonal View Matrix

Matrix3 ZeroMatrix()
Returns a Matrix with all elements set to 0.

void RotateMatrixX(float a_fRotation)
Creates a Matrix rotated by a_fRotation on the X axis

void RotateMatrixY(float a_fRotation)
Creates a Matrix rotated by a_fRotation on the Y axis

void RotateMatrixZ(float a_fRotation)
Creates a Matrix rotated by a_fRotation on the Z axis

Vec3 TransformPoint(**Vec3**a_pVec3)
Transforms the Matrix by a_pVec3

void **operator=**(const**Matrix3**&a_Mat1)
Sets the elements of a Matrix to those of Matrix a_Mat1

bool **operator==**(const**Matrix3**&a_Mat1)
Compares the elements of a Matrix to those of Matrix a_Mat1 and returns true if they are the same

bool **operator!=**(const**Matrix3**&a_Mat1)

Compares the elements of a Matrix to those of Matrix a_Mat1 and returns true if they are different

Matrix3 operator+(const**Matrix3**&a_Mat1)
Adds the elements of 2 Matrices and returns the result as a new Matrix

Matrix3 operator+(const**Vec3**&a_Vec3)
Adds the values of a Vector 3 a_Vec3 to the bottom row of elements of a Matrix and returns the result as a new Matrix

Matrix3 operator+=(const**Matrix3**&a_Mat1)
Adds the elements of a Matrix a_Mat1 to the elements of another Matrix

Matrix3 operator+=(const**Vec3**&a_Vec3)
Adds the values of a Vector 3 a_Vec3 to the bottom row of elements of a Matrix

Matrix3 operator-(const**Matrix3**&a_Mat1)
Subtracts the elements of Matrix a_Mat1 from the elements of a Matrix and returns the result as a new Matrix

Matrix3 operator-(const**Vec3**&a_Vec3)
Subtracts the values of a Vector 3 a_Vec3 from the bottom row of element of a Matrix and returns the result as a new Matrix

Matrix3 operator-=(const**Matrix3**&a_Mat1)
Subtracts the elements of Matrix a_Mat1 from the elements of a Matrix

Matrix3 operator-=(const**Vec3**&a_Vec3)
Subtracts the values of a Vector 3 a_Vec3 from the bottom row of element of a Matrix

Matrix3 operator* (const**Matrix3**&a_Mat1)
Multiplies the elements of a Matrix by the elements of Matrix a_Mat1 returns the result as a new Matrix

Matrix3& **operator*=**(const**Matrix3**&a_Mat1)
Multiplies the elements of a Matrix by the elements of Matrix a_Mat1

Matrix3 operator* (const float &a_Float)
Multiplies the elements of a Matrix by the elements of a float a_Float and returns the result as a new Matrix

Matrix3& **operator*=**(const float &a_Float)
Multiplies the elements of a Matrix by the elements of a float a_Float

Vec3 operator* (const**Vec3**&a_Vec3)
Multiplies the elements of a Matrix by the values of a Vector 3 a_Vec3 and returns te result as a new Vector 3

Matrix 4 Class – Matrix4.h

Matrix4()
Creates an empty Matrix with 16 elements of 0

Matrix4(float a_fMatrixArray[4][4])
Creates a Matrix that has the elements of a_fMatrixArray

Matrix4(const float &a_f11, const float &a_f12, const float &a_f13, const float &a_f14,
const float &a_f21, const float &a_f22, const float &a_f23, const float &a_f24,
const float &a_f31, const float &a_f32, const float &a_f33, const float &a_f34,
const float &a_f41, const float &a_f42, const float &a_f43, const float &a_f4)
Creates a Matrix using the function arguements

operator float * ()
Casts the Matrix to a float pointer

operator const float * () const
Casts the Matrix to a constant float pointer

void RotateMatrixX(float a_fRotation)
Creates a Matrix rotated by a_fRotation on the X axis

void RotateMatrixY(float a_fRotation)
Creates a Matrix rotated by a_fRotation on the Y axis

void RotateMatrixZ(float a_fRotation)
Creates a Matrix rotated by a_fRotation on the Z axis

Vec3 TransformPoint(**Vec3**a_pVec3)
Transforms the Matrix by a_pVec3

void **operator=**(const **Matrix4**&a_Mat1)
Sets the elements of a Matrix to those of Matrix a_Mat1

bool **operator==**(const **Matrix4**&a_Mat1)
Compares the elements of a Matrix to those of Matrix a_Mat1 and returns true if they are the same

bool **operator!=**(const **Matrix4**&a_Mat1)
Compares the elements of a Matrix to those of Matrix a_Mat1 and returns true if they are different

Matrix4 operator+(const **Matrix4**&a_Mat1)
Adds the elements of 2 Matrices and returns the result as a new Matrix

Matrix4 operator+(const **Vec3**&a_Vec3)
Adds the values of a Vector 3 a_Vec3 to the bottom row of elements of a Matrix and returns the result as a new Matrix

Matrix4 operator+=(const **Matrix4**&a_Mat1)
Adds the elements of a Matrix a_Mat1 to the elements of another Matrix

Matrix4 operator+=(const**Vec3**&a_Vec3)
Adds the values of a Vector 3 a_Vec3 to the bottom row of elements of a Matrix

Matrix4 operator-(const **Matrix4**&a_Mat1)
Subtracts the elements of Matrix a_Mat1 from the elements of a Matrix and returns the result as a new Matrix

Matrix4 operator-(const **Vec3**&a_Vec3)
Subtracts the values of a Vector 3 a_Vec3 from the bottom row of element of a Matrix and returns the result as a new Matrix

Matrix4 operator-=(const **Matrix4**&a_Mat1)
Subtracts the elements of Matrix a_Mat1 from the elements of a Matrix

Matrix4 operator-=(const **Vec3**&a_Vec3)
Subtracts the values of a Vector 3 a_Vec3 from the bottom row of element of a Matrix

Matrix4 operator* (const **Matrix4**&a_Mat1)
Multiplies the elements of a Matrix by the elements of Matrix a_Mat1 returns the result as a new Matrix

Matrix4& **operator*** =(const **Matrix4**&a_Mat1)
Multiplies the elements of a Matrix by the elements of Matrix a_Mat1

Matrix4 operator* (const float &a_Float)
Multiplies the elements of a Matrix by the elements of a float a_Float and returns the result as a new Matrix

Matrix4& **operator*** =(const float &a_Float)
Multiplies the elements of a Matrix by the elements of a float a_Float

Vec4 operator* (const**Vec4**&a_Vec4)
Multiplies the elements of a Matrix by the values of a Vector 4 a_Vec4 and returns te result as a new Vector 4

Maths Class – Maths.h

unsigned int NextPowerOfTwo(unsigned int a_uiInput)
Takes an input of a_uiInput and finds the next power of two using bit shifting, returns an unsigned int

float ScalarInterpolation(const float &a_pF1, const float &a_pF2, float a_fDest)
Returns a float

static double DegToRad(float a_fDegrees)
Converts a float of a_fDegrees into Radians and returns the result as a double

static double RadToDeg(float a_fRadians)
Converts a float of a_ fRadians into Degrees and returns the result as a double

Conversions Class – Conversions.h

Bitset* HexToBin(unsigned int a_uiHex)
Converts a Hexadecimal number to Binary and stores it in a Bitset array

Bitset* DecToBin(int a_iDec)
Converts a Decimal number to Binary and stores it in a Bitset array

unsigned int DecToHex(int a_iDec)
Converts a Decimal number to Hexadecimal and returns the result as an unsigned int

unsigned int BinToHex(**Bitset** a_pBin)
Converts a Binary number from a Bitset array to Hexadecimal and returns the result as an unsigned int

int BinToDec(**Bitset** a_pBin)
Converts a Binary number from a Bitset array to Decimal and returns the result as an int

int HexToDec(float a_fRadians)
Converts a Hexadecimal number to Decimal and returns the result as an int

Bitset Class – Bitset.h

Bitset(unsigned int a_uiSize)
Creates and sets initial size of Bitset array

~Bitset()
Default destructor, deletes Bitset array data

void SetAllBits()
Sets all bits in array to 1

void ClearAllBits()
Sets all bits in array to 0

void SetBit(unsigned int a_uiBit)
Sets bit at location a_uiBit to 1

void ClearBit(unsigned int a_uiBit)
Sets bit at location a_uiBit to 0

bool operator[](unsigned int a_uiBit)
Returns value of the bit at location a_uiBit

int GetSize()
Returns the size of the Bitset array