Math Library

Generated by Doxygen 1.8.13

Contents

1 Namespace Index **1**

1.1 Namespace List 1

2 Class Index **3**

2.1 Class List 3

3 File Index **5**

3.1 File List 5

4 Namespace Documentation **7**

4.1 math Namespace Reference 7

5 Class Documentation **9**

5.1 math::Vector2 Class Reference 9

5.1.1 Detailed Description 10

5.1.2 Constructor & Destructor Documentation 10

5.1.2.1 Vector2() [1/2] 10

5.1.2.2 Vector2() [2/2] 10

5.1.2.3 ~Vector2() 11

5.1.3 Member Function Documentation 11

5.1.3.1 dot() 11

5.1.3.2 magnitude() 11

5.1.3.3 magnitudeSquared() 12

5.1.3.4 normal() 12

5.1.3.5 normalise() 12

5.1.3.6 operator∗() . . . 12
5.1.3.7 operator∗=() . . . 13
5.1.3.8 operator+() . . . 13
5.1.3.9 operator+=() . . . 13
5.1.3.10 operator-() . . . 13
5.1.3.11 operator-=() . . . 14
5.1.3.12 operator/() . . . 14
5.1.3.13 operator/=() . . . 14
5.1.3.14 operator=() . . . 15
5.1.3.15 operator==() . . . 15
5.1.4 Member Data Documentation . . . 15
5.1.4.1 x . . . 15
5.1.4.2 y . . . 16
5.2 math::Vector3 Class Reference . . . 16
5.2.1 Detailed Description . . . 17
5.2.2 Constructor & Destructor Documentation . . . 17
5.2.2.1 Vector3() [1/2] . . . 17
5.2.2.2 Vector3() [2/2] . . . 17
5.2.2.3 ∼Vector3() . . . 18
5.2.3 Member Function Documentation . . . 18
5.2.3.1 cross() . . . 18
5.2.3.2 dot() . . . 18
5.2.3.3 magnitude() . . . 19
5.2.3.4 magnitudeSquared() . . . 19
5.2.3.5 normal() . . . 19
5.2.3.6 normalise() . . . 19
5.2.3.7 operator∗() . . . 19
5.2.3.8 operator∗=() . . . 20
5.2.3.9 operator+() . . . 20
5.2.3.10 operator+=() . . . 20
5.2.3.11 operator-() . . . 21
5.2.3.12 operator-=() . . . 21
5.2.3.13 operator/() . . . 21
5.2.3.14 operator/=() . . . 22
5.2.3.15 operator=() . . . 22
5.2.3.16 operator==() . . . 22
5.2.4 Member Data Documentation . . . 23
5.2.4.1 x . . . 23
5.2.4.2 y . . . 23
5.2.4.3 z . . . 23

6 File Documentation **25**

6.1 src/MathLibrary.hpp File Reference . . . 25

6.2 src/Vector2.hpp File Reference . . . 25

6.3 src/Vector3.hpp File Reference . . . 25

6.4 test/main.cpp File Reference . . . 26

6.4.1 Function Documentation . . . 26

6.4.1.1 main() . . . 26

6.5 test/src/Vector2Tests.cpp File Reference . . . 26

6.5.1 Function Documentation . . . 27

6.5.1.1 TEST() [1/17] . . . 27

6.5.1.2 TEST() [2/17] . . . 27

6.5.1.3 TEST() [3/17] . . . 27

6.5.1.4 TEST() [4/17] . . . 28

6.5.1.5 TEST() [5/17] . . . 28

6.5.1.6 TEST() [6/17] . . . 28

6.5.1.7 TEST() [7/17] . . . 28

6.5.1.8 TEST() [8/17] . . . 28

6.5.1.9 TEST() [9/17] . . . 28

6.5.1.10 TEST() [10/17] . . . 29

6.5.1.11 TEST() [11/17] . . . 29

6.5.1.12 TEST() [12/17] . . . 29

6.5.1.13 TEST() [13/17] . . . 29

6.5.1.14 TEST() [14/17] . . . 29

6.5.1.15 TEST() [15/17] . . . 29

6.5.1.16 TEST() [16/17] . . . 30

6.5.1.17 TEST() [17/17] . . . 30

6.6 test/src/Vector3Tests.cpp File Reference . . . 30

6.6.1 Function Documentation . . . 30

6.6.1.1 TEST() [1/18] . . . 31

6.6.1.2 TEST() [2/18] . . . 31

6.6.1.3 TEST() [3/18] . . . 31
6.6.1.4 TEST() [4/18] . . . 31
6.6.1.5 TEST() [5/18] . . . 31
6.6.1.6 TEST() [6/18] . . . 31
6.6.1.7 TEST() [7/18] . . . 32
6.6.1.8 TEST() [8/18] . . . 32
6.6.1.9 TEST() [9/18] . . . 32
6.6.1.10 TEST() [10/18] . . . 32
6.6.1.11 TEST() [11/18] . . . 32
6.6.1.12 TEST() [12/18] . . . 32
6.6.1.13 TEST() [13/18] . . . 33
6.6.1.14 TEST() [14/18] . . . 33
6.6.1.15 TEST() [15/18] . . . 33
6.6.1.16 TEST() [16/18] . . . 33
6.6.1.17 TEST() [17/18] . . . 33
6.6.1.18 TEST() [18/18] . . . 33

Index **35**

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

math 7

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

math::Vector2
: 2D Vector class containing 2D vector operations. Operations available: Vec+Vec, Vec+=Vec, Vec-Vec, Vec-=Vec, Vec=Vec, Vec==Vec, Vec∗d, Vec∗=d, Vec/d, Vec/=d, Vec.doc(Vec), Vec.↩ normal(), Vec.normalise(), Vec.magnitude(), Vec.magnitudeSquare() 9

math::Vector3
: Vector3 structure. Operations available: Vec+Vec, Vec+=Vec, Vec-Vec, Vec-=Vec, Vec=Vec, Vec==Vec, Vec∗d, Vec∗=d, Vec/d, Vec/=d, Vec.doc(Vec), Vec.cross(Vec), Vec.normal(), Vec.↩ normalise(), Vec.magnitude(), Vec.magnitudeSquare() 16

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

src/MathLibrary.hpp 25
src/Vector2.hpp 25
src/Vector3.hpp 25
test/main.cpp 26
test/src/Vector2Tests.cpp 26
test/src/Vector3Tests.cpp 30

Chapter 4

Namespace Documentation

4.1 math Namespace Reference

Classes

- class Vector2

  2D Vector class containing 2D vector operations. Operations available: Vec+Vec, Vec+=Vec, Vec-Vec, Vec-=Vec, Vec=Vec, Vec==Vec, Vec∗d, Vec∗=d, Vec/d, Vec/=d, Vec.doc(Vec), Vec.normal(), Vec.normalise(), Vec.magnitude(), Vec.magnitudeSquare()
- class Vector3

  Vector3 structure. Operations available: Vec+Vec, Vec+=Vec, Vec-Vec, Vec-=Vec, Vec=Vec, Vec==Vec, Vec∗d, Vec∗=d, Vec/d, Vec/=d, Vec.doc(Vec), Vec.cross(Vec), Vec.normal(), Vec.normalise(), Vec.magnitude(), Vec.↩ magnitudeSquare()

Chapter 5

Class Documentation

5.1 math::Vector2 Class Reference

2D Vector class containing 2D vector operations. Operations available: Vec+Vec, Vec+=Vec, Vec-Vec, Vec-=Vec, Vec=Vec, Vec==Vec, Vec∗d, Vec∗=d, Vec/d, Vec/=d, Vec.doc(Vec), Vec.normal(), Vec.normalise(), Vec.magnitude(), Vec.magnitudeSquare()

`#include <Vector2.hpp>`

Public Member Functions

- Vector2 ()

  Constructs the object. Empty construcor sets x=0 and y=0.
- Vector2 (double x, double y)

  Constructs the object. Uses the x and y provided to build the 2D vector.
- ∼Vector2 ()

  Destroys the object.
- Vector2 operator+ (const Vector2 &other)

  Overloads + operator. Vec+Vec, sum of the x and y components of the 2D vector.
- void operator+= (const Vector2 &other)

  Overloads += operator. Vec+=Vec, sum of the x and y components.
- Vector2 operator- (const Vector2 &other)

  Overloads - operator. Vec-Vec, subtraction of x and y components.
- void operator-= (const Vector2 &other)

  Overloads -= operator. Vec-=Vec, subtraction of x and y components.
- void operator= (const Vector2 &other)

  Overloads = operator. Vec=Vec, assign x and y components to left hand side 2D vector.
- bool operator== (const Vector2 &other)

  Overloads == operator. Vec==Vec, compares if x and y components are the same.
- Vector2 operator∗ (double value)

  Overloads ∗ operator. Vec∗Vec, multiplies the x and y components of the 2D vectors.
- void operator∗= (double value)

  Overloads ∗= operator. Vec∗=Vec, multiplies the x and y components of the 2D vectors.
- Vector2 operator/ (double value)

  Overloads / operator. Vec/Vec, divide left vector by right vector.

- void operator/= (double value)

  Overloads /= operator. Vec/=Vec, divide left vector by right vector.
- double dot (Vector2 &other)

  Dot product. Vec.dot(Vec), result of the dot product operation. V1 is first vector and V2 second vector, DotProduct↩ Result = V1.x ∗ V2.x + V1.y ∗ V2.y.
- Vector2 normal ()

  Computes the vector mormal. Vec.normal(), divides x and y components by magnitude.
- void normalise ()

  Normalises the vector. Vec.normal(), divides x and y components by magnitude.
- double magnitude ()

  Computes the magnitude. Magnitude = sqrt(x∗x + y∗y).
- double magnitudeSquared ()

  Computes the magnitude squared. MagnitudeSquared = x∗x + y∗y.

Public Attributes

- double x
- double y

5.1.1 Detailed Description

2D Vector class containing 2D vector operations. Operations available: Vec+Vec, Vec+=Vec, Vec-Vec, Vec-=Vec, Vec=Vec, Vec==Vec, Vec∗d, Vec∗=d, Vec/d, Vec/=d, Vec.doc(Vec), Vec.normal(), Vec.normalise(), Vec.magnitude(), Vec.magnitudeSquare()

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Vector2() [1/2]

```
math::Vector2::Vector2 ( )  [inline]
```

Constructs the object. Empty construcor sets x=0 and y=0.

5.1.2.2 Vector2() [2/2]

```
math::Vector2::Vector2 (
            double x,
            double y )  [inline]
```

Constructs the object. Uses the x and y provided to build the 2D vector.

Parameters

in	*x*	x component of the 2D vector
in	*y*	y component of the 2D vector

5.1.2.3 ~Vector2()

```
math::Vector2::~Vector2 ( )  [inline]
```

Destroys the object.

5.1.3 Member Function Documentation

5.1.3.1 dot()

```
double math::Vector2::dot (
            Vector2 & other )  [inline]
```

Dot product. Vec.dot(Vec), result of the dot product operation. V1 is first vector and V2 second vector, DotProduct↩ Result = V1.x ∗ V2.x + V1.y ∗ V2.y.

Parameters

other	The other vector used in the operation

Returns

Result of the dot product.

5.1.3.2 magnitude()

```
double math::Vector2::magnitude ( )  [inline]
```

Computes the magnitude. Magnitude = sqrt(x∗x + y∗y).

Returns

The vector magnitude.

5.1.3.3 magnitudeSquared()

```
double math::Vector2::magnitudeSquared ( )  [inline]
```

Computes the magnitude squared. MagnitudeSquared = x∗x + y∗y.

Returns

The magnitude squared.

5.1.3.4 normal()

```
Vector2 math::Vector2::normal ( )  [inline]
```

Computes the vector mormal. Vec.normal(), divides x and y components by magnitude.

Returns

Normal 2D vector.

5.1.3.5 normalise()

```
void math::Vector2::normalise ( )  [inline]
```

Normalises the vector. Vec.normal(), divides x and y components by magnitude.

5.1.3.6 operator∗()

```
Vector2 math::Vector2::operator* (
            double value )  [inline]
```

Overloads ∗ operator. Vec∗Vec, multiplies the x and y components of the 2D vectors.

Parameters

in	*value*	The double value used in the operation

Returns

Result of the multiplication.

5.1.3.7 operator∗=()

```
void math::Vector2::operator*= (
            double value )  [inline]
```

Overloads ∗= operator. Vec∗=Vec, multiplies the x and y components of the 2D vectors.

Parameters

in	*value*	The double value used in the operation

5.1.3.8 operator+()

```
Vector2 math::Vector2::operator+ (
            const Vector2 & other )  [inline]
```

Overloads + operator. Vec+Vec, sum of the x and y components of the 2D vector.

Parameters

other	The other vector used in the operation

Returns

Result of the addition.

5.1.3.9 operator+=()

```
void math::Vector2::operator+= (
            const Vector2 & other )  [inline]
```

Overloads += operator. Vec+=Vec, sum of the x and y components.

Parameters

other	The other vector used in the operation

5.1.3.10 operator-()

```
Vector2 math::Vector2::operator- (
            const Vector2 & other )  [inline]
```

Overloads - operator. Vec-Vec, subtraction of x and y components.

Parameters

other	The other vector used in the operation

Returns

Result of the subtraction.

5.1.3.11 operator-=()

```
void math::Vector2::operator-= (
            const Vector2 & other )  [inline]
```

Overloads -= operator. Vec-=Vec, subtraction of x and y components.

Parameters

other	The other vector used in the operation

5.1.3.12 operator/()

```
Vector2 math::Vector2::operator/ (
            double value )  [inline]
```

Overloads / operator. Vec/Vec, divide left vector by right vector.

Parameters

in	*value*	The double value used in the operation

Returns

Result of the division.

5.1.3.13 operator/=()

```
void math::Vector2::operator/= (
            double value )  [inline]
```

Overloads /= operator. Vec/=Vec, divide left vector by right vector.

Parameters

in	*value*	The double value used in the operation

5.1.3.14 operator=()

```
void math::Vector2::operator= (
            const Vector2 & other )  [inline]
```

Overloads = operator. Vec=Vec, assign x and y components to left hand side 2D vector.

Parameters

in	*other*	The other vector used in the operation

5.1.3.15 operator==()

```
bool math::Vector2::operator== (
            const Vector2 & other )  [inline]
```

Overloads == operator. Vec==Vec, compares if x and y components are the same.

Parameters

other	The other vector used in the operation

Returns

True if vectors are the same, False otherwise.

5.1.4 Member Data Documentation

5.1.4.1 x

```
double math::Vector2::x
```

5.1.4.2 y

```
double math::Vector2::y
```

The documentation for this class was generated from the following file:

- src/Vector2.hpp

5.2 math::Vector3 Class Reference

Vector3 structure. Operations available: Vec+Vec, Vec+=Vec, Vec-Vec, Vec-=Vec, Vec=Vec, Vec==Vec, Vec∗d, Vec∗=d, Vec/d, Vec/=d, Vec.doc(Vec), Vec.cross(Vec), Vec.normal(), Vec.normalise(), Vec.magnitude(), Vec.↩ magnitudeSquare()

```
#include <Vector3.hpp>
```

Public Member Functions

- Vector3 ()
- Vector3 (double x, double y, double z)

  Constructor. Uses the x, y and z provided to build the 3D vector.
- ∼Vector3 ()

  Destroys the object.
- Vector3 operator+ (const Vector3 &other)

  Overloads + operator. Vec+Vec, sum of the x, y and z components of the 3D vector.
- void operator+= (const Vector3 &other)

  Overloads += operator. Vec+=Vec, sum of the x, y and z components.
- Vector3 operator- (const Vector3 &other)

  Overloads - operator. Vec-Vec, subtraction of x, y and z components.
- void operator-= (const Vector3 &other)

  Overloads -= operator. Vec-=Vec, subtraction of x, y and z components.
- void operator= (const Vector3 &other)

  Overloads equal operator. Vec=Vec, assign x, y and z components to left hand side 3D vector.
- bool operator== (const Vector3 &other)

  Overloads == operator. Vec==Vec, compares if x, y and z components are the same.
- Vector3 operator∗ (double other)

  Overloads multiplication with double. Vec∗Vec, multiplies the x, y and z components of the 3D vectors.
- void operator∗= (double other)

  Overloads ∗= operator. Vec∗=Vec, multiplies the x, y and z components of the 3D vectors.
- Vector3 operator/ (double other)

  Overloads division with double. Vec/Vec, divide left vector by right vector.
- void operator/= (double other)

  Overloads /= operator. Vec/=Vec, divide left vector by right vector.
- double dot (const Vector3 &other)

  Dot product. Vec.dot(Vec), result of the dot product operation. V1 is first vector and V2 second vector, DotProduct↩ Result = V1.x ∗ V2.x + V1.y ∗ V2.y + V1.z ∗ V2.z.
- Vector3 cross (const Vector3 &other)

  Cross product. V1 is the first vector and V2 is the second vector, CrossProduct = [V1.y∗V2.z - V1.z∗V2.y, V1.z∗V2.x - V1.x∗V2.z, V1.x∗V2.y-V1.y∗V2.x].

- Vector3 normal ()

  Computes the vector mormal. Vec.normal(), divides x, y and z components by magnitude.
- void normalise ()

  Normalises the vector. Vec.normal(), divides x, y and z components by magnitude.
- double magnitude ()

  Computes the vector magnitude. Magnitude = sqrt(x∗x + y∗y + z∗z).
- double magnitudeSquared ()

  Computes the magnitude squared. MagnitudeSquared = x∗x + y∗y + z∗z.

Public Attributes

- double x
- double y
- double z

5.2.1 Detailed Description

Vector3 structure. Operations available: Vec+Vec, Vec+=Vec, Vec-Vec, Vec-=Vec, Vec=Vec, Vec==Vec, Vec∗d, Vec∗=d, Vec/d, Vec/=d, Vec.doc(Vec), Vec.cross(Vec), Vec.normal(), Vec.normalise(), Vec.magnitude(), Vec.↩ magnitudeSquare()

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Vector3() [1/2]

```
math::Vector3::Vector3 ( )  [inline]
```

Default constructor. Empty construcor sets x=0, y=0 and z=0.

5.2.2.2 Vector3() [2/2]

```
math::Vector3::Vector3 (
            double x,
            double y,
            double z )  [inline]
```

Constructor. Uses the x, y and z provided to build the 3D vector.

Parameters

in	*x*	x value
in	*y*	y value
in	*z*	z value

5.2.2.3 ~Vector3()

```
math::Vector3::~Vector3 ( )  [inline]
```

Destroys the object.

5.2.3 Member Function Documentation

5.2.3.1 cross()

```
Vector3 math::Vector3::cross (
            const Vector3 & other )  [inline]
```

Cross product. V1 is the first vector and V2 is the second vector, CrossProduct = [V1.y∗V2.z - V1.z∗V2.y, V1.z∗V2.x - V1.x∗V2.z, V1.x∗V2.y-V1.y∗V2.x].

Parameters

in	*other*	The other Vector3 used in the operation

Returns

Result of the cross product.

5.2.3.2 dot()

```
double math::Vector3::dot (
            const Vector3 & other )  [inline]
```

Dot product. Vec.dot(Vec), result of the dot product operation. V1 is first vector and V2 second vector, DotProduct↩ Result = V1.x ∗ V2.x + V1.y ∗ V2.y + V1.z ∗ V2.z.

Parameters

other	The other vector

Returns

Result of the dot product.

5.2.3.3 magnitude()

```
double math::Vector3::magnitude ( )  [inline]
```

Computes the vector magnitude. Magnitude = sqrt(x∗x + y∗y + z∗z).

Returns

Magnitude computed.

5.2.3.4 magnitudeSquared()

```
double math::Vector3::magnitudeSquared ( )  [inline]
```

Computes the magnitude squared. MagnitudeSquared = x∗x + y∗y + z∗z.

Returns

Magnitude squared.

5.2.3.5 normal()

```
Vector3 math::Vector3::normal ( )  [inline]
```

Computes the vector mormal. Vec.normal(), divides x, y and z components by magnitude.

Returns

Normal 3D vector.

5.2.3.6 normalise()

```
void math::Vector3::normalise ( )  [inline]
```

Normalises the vector. Vec.normal(), divides x, y and z components by magnitude.

5.2.3.7 operator∗()

```
Vector3 math::Vector3::operator* (
            double other )  [inline]
```

Overloads multiplication with double. Vec∗Vec, multiplies the x, y and z components of the 3D vectors.

Parameters

in	*other*	Double value used in the operation

Returns

Multiplication result.

5.2.3.8 operator∗=()

```
void math::Vector3::operator*= (
            double other )  [inline]
```

Overloads ∗= operator. Vec∗=Vec, multiplies the x, y and z components of the 3D vectors.

Parameters

in	*other*	Double value used in the operation

5.2.3.9 operator+()

```
Vector3 math::Vector3::operator+ (
            const Vector3 & other )  [inline]
```

Overloads + operator. Vec+Vec, sum of the x, y and z components of the 3D vector.

Parameters

in	*other*	The other Vector3 used in the operation

Returns

Sum of the two vectors.

5.2.3.10 operator+=()

```
void math::Vector3::operator+= (
            const Vector3 & other )  [inline]
```

Overloads += operator. Vec+=Vec, sum of the x, y and z components.

Parameters

in	*other*	The other Vector3 used in the operation

5.2.3.11 operator-()

```
Vector3 math::Vector3::operator- (
            const Vector3 & other )  [inline]
```

Overloads - operator. Vec-Vec, subtraction of x, y and z components.

Parameters

in	*other*	The other Vector3 used in the operation

Returns

Substraction of the two vectors.

5.2.3.12 operator-=()

```
void math::Vector3::operator-= (
            const Vector3 & other )  [inline]
```

Overloads -= operator. Vec-=Vec, subtraction of x, y and z components.

Parameters

in	*other*	The other Vector3 used in the operation

5.2.3.13 operator/()

```
Vector3 math::Vector3::operator/ (
            double other )  [inline]
```

Overloads division with double. Vec/Vec, divide left vector by right vector.

Parameters

in	*other*	Double value used in the operation

Returns

Division result.

5.2.3.14 operator/=()

```
void math::Vector3::operator/= (
            double other )  [inline]
```

Overloads /= operator. Vec/=Vec, divide left vector by right vector.

Parameters

in	*other*	Double value used in the operation

5.2.3.15 operator=()

```
void math::Vector3::operator= (
            const Vector3 & other )  [inline]
```

Overloads equal operator. Vec=Vec, assign x, y and z components to left hand side 3D vector.

Parameters

in	*other*	The other Vector3 used in the operation

5.2.3.16 operator==()

```
bool math::Vector3::operator== (
            const Vector3 & other )  [inline]
```

Overloads == operator. Vec==Vec, compares if x, y and z components are the same.

Parameters

other	The other vector

Returns

True if vectors are the same, False otherwise.

5.2.4 Member Data Documentation

5.2.4.1 x

```
double math::Vector3::x
```

5.2.4.2 y

```
double math::Vector3::y
```

5.2.4.3 z

```
double math::Vector3::z
```

The documentation for this class was generated from the following file:

- src/Vector3.hpp

Chapter 6

File Documentation

6.1 src/MathLibrary.hpp File Reference

```
#include "Vector2.hpp"
#include "Vector3.hpp"
```

6.2 src/Vector2.hpp File Reference

```
#include <cmath>
```

Classes

- class math::Vector2

  2D Vector class containing 2D vector operations. Operations available: Vec+Vec, Vec+=Vec, Vec-Vec, Vec-=Vec, Vec=Vec, Vec==Vec, Vec∗d, Vec∗=d, Vec/d, Vec/=d, Vec.doc(Vec), Vec.normal(), Vec.normalise(), Vec.magnitude(), Vec.magnitudeSquare()

Namespaces

- math

6.3 src/Vector3.hpp File Reference

```
#include <cmath>
```

Classes

- class math::Vector3

  Vector3 structure. Operations available: Vec+Vec, Vec+=Vec, Vec-Vec, Vec-=Vec, Vec=Vec, Vec==Vec, Vec∗d, Vec∗=d, Vec/d, Vec/=d, Vec.doc(Vec), Vec.cross(Vec), Vec.normal(), Vec.normalise(), Vec.magnitude(), Vec.magnitudeSquare()

Namespaces

- math

6.4 test/main.cpp File Reference

```
#include "gtest/gtest.h"
```

Functions

- int main (int argc, char ∗argv[ ])

6.4.1 Function Documentation

6.4.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

6.5 test/src/Vector2Tests.cpp File Reference

```
#include "gtest/gtest.h"
#include "MathLibrary.hpp"
```

Functions

- TEST (Vector2, DefaultConstructor)
- TEST (Vector2, Constructor)
- TEST (Vector2, Vector2_Plus_Vector2)
- TEST (Vector2, Vector2_PlusEqual_Vector2)
- TEST (Vector2, Vector2_Minus_Vector2)
- TEST (Vector2, Vector2_MinusEqual_Vector2)
- TEST (Vector2, Vector2_Assign_Vector2)
- TEST (Vector2, Vector2_Equal_Vector2)
- TEST (Vector2, Vector2_Times_Value)
- TEST (Vector2, Vector2_TimesEqual_Value)
- TEST (Vector2, Vector2_DividedBy_Value)
- TEST (Vector2, Vector2_DividedByEqual_Value)
- TEST (Vector2, DotProduct)
- TEST (Vector2, Normal)
- TEST (Vector2, Normalise)
- TEST (Vector2, Magnitude)
- TEST (Vector2, MagnitudeSquared)

6.5.1 Function Documentation

6.5.1.1 TEST() [1/17]

```
TEST (
            Vector2 ,
            DefaultConstructor  )
```

6.5.1.2 TEST() [2/17]

```
TEST (
            Vector2 ,
            Constructor  )
```

6.5.1.3 TEST() [3/17]

```
TEST (
            Vector2 ,
            Vector2_Plus_Vector2  )
```

6.5.1.4 TEST() [4/17]

```
TEST (
            Vector2 ,
            Vector2_PlusEqual_Vector2  )
```

6.5.1.5 TEST() [5/17]

```
TEST (
            Vector2 ,
            Vector2_Minus_Vector2  )
```

6.5.1.6 TEST() [6/17]

```
TEST (
            Vector2 ,
            Vector2_MinusEqual_Vector2  )
```

6.5.1.7 TEST() [7/17]

```
TEST (
            Vector2 ,
            Vector2_Assign_Vector2  )
```

6.5.1.8 TEST() [8/17]

```
TEST (
            Vector2 ,
            Vector2_Equal_Vector2  )
```

6.5.1.9 TEST() [9/17]

```
TEST (
            Vector2 ,
            Vector2_Times_Value  )
```

6.5.1.10 TEST() [10/17]

```
TEST (
            Vector2 ,
            Vector2_TimesEqual_Value  )
```

6.5.1.11 TEST() [11/17]

```
TEST (
            Vector2 ,
            Vector2_DividedBy_Value  )
```

6.5.1.12 TEST() [12/17]

```
TEST (
            Vector2 ,
            Vector2_DividedByEqual_Value  )
```

6.5.1.13 TEST() [13/17]

```
TEST (
            Vector2 ,
            DotProduct  )
```

6.5.1.14 TEST() [14/17]

```
TEST (
            Vector2 ,
            Normal  )
```

6.5.1.15 TEST() [15/17]

```
TEST (
            Vector2 ,
            Normalise  )
```

6.5.1.16 TEST() [16/17]

```
TEST (
            Vector2 ,
            Magnitude  )
```

6.5.1.17 TEST() [17/17]

```
TEST (
            Vector2 ,
            MagnitudeSquared  )
```

6.6 test/src/Vector3Tests.cpp File Reference

```
#include "gtest/gtest.h"
#include "MathLibrary.hpp"
```

Functions

- TEST (Vector3, DefaultConstructor)
- TEST (Vector3, Constructor)
- TEST (Vector3, Vector3_Plus_Vector3)
- TEST (Vector3, Vector3_PlusEqual_Vector3)
- TEST (Vector3, Vector3_Minus_Vector3)
- TEST (Vector3, Vector3_MinusEqual_Vector3)
- TEST (Vector3, Vector3_Assign_Vector3)
- TEST (Vector3, Vector3_Equal_Vector3)
- TEST (Vector3, Vector3_Times_Value)
- TEST (Vector3, Vector3_TimesEqual_Value)
- TEST (Vector3, Vector3_DividedBy_Value)
- TEST (Vector3, Vector3_DividedByEqual_Value)
- TEST (Vector3, DotProduct)
- TEST (Vector3, CrossProduct)
- TEST (Vector3, Normal)
- TEST (Vector3, Normalise)
- TEST (Vector3, Magnitude)
- TEST (Vector3, MagnitudeSquared)

6.6.1 Function Documentation

6.6.1.1 TEST() [1/18]

```
TEST (
            Vector3 ,
            DefaultConstructor  )
```

6.6.1.2 TEST() [2/18]

```
TEST (
            Vector3 ,
            Constructor  )
```

6.6.1.3 TEST() [3/18]

```
TEST (
            Vector3 ,
            Vector3_Plus_Vector3  )
```

6.6.1.4 TEST() [4/18]

```
TEST (
            Vector3 ,
            Vector3_PlusEqual_Vector3  )
```

6.6.1.5 TEST() [5/18]

```
TEST (
            Vector3 ,
            Vector3_Minus_Vector3  )
```

6.6.1.6 TEST() [6/18]

```
TEST (
            Vector3 ,
            Vector3_MinusEqual_Vector3  )
```

6.6.1.7 TEST() [7/18]

```
TEST (
            Vector3 ,
            Vector3_Assign_Vector3  )
```

6.6.1.8 TEST() [8/18]

```
TEST (
            Vector3 ,
            Vector3_Equal_Vector3  )
```

6.6.1.9 TEST() [9/18]

```
TEST (
            Vector3 ,
            Vector3_Times_Value  )
```

6.6.1.10 TEST() [10/18]

```
TEST (
            Vector3 ,
            Vector3_TimesEqual_Value  )
```

6.6.1.11 TEST() [11/18]

```
TEST (
            Vector3 ,
            Vector3_DividedBy_Value  )
```

6.6.1.12 TEST() [12/18]

```
TEST (
            Vector3 ,
            Vector3_DividedByEqual_Value  )
```

6.6.1.13 TEST() [13/18]

```
TEST (
            Vector3 ,
            DotProduct  )
```

6.6.1.14 TEST() [14/18]

```
TEST (
            Vector3 ,
            CrossProduct  )
```

6.6.1.15 TEST() [15/18]

```
TEST (
            Vector3 ,
            Normal  )
```

6.6.1.16 TEST() [16/18]

```
TEST (
            Vector3 ,
            Normalise  )
```

6.6.1.17 TEST() [17/18]

```
TEST (
            Vector3 ,
            Magnitude  )
```

6.6.1.18 TEST() [18/18]

```
TEST (
            Vector3 ,
            MagnitudeSquared  )
```

Index

~Vector2
    math::Vector2, 11
~Vector3
    math::Vector3, 18

cross
    math::Vector3, 18

dot
    math::Vector2, 11
    math::Vector3, 18

magnitude
    math::Vector2, 11
    math::Vector3, 18
magnitudeSquared
    math::Vector2, 11
    math::Vector3, 19
main
    main.cpp, 26
main.cpp
    main, 26
math, 7
math::Vector2, 9
    ~Vector2, 11
    dot, 11
    magnitude, 11
    magnitudeSquared, 11
    normal, 12
    normalise, 12
    operator∗, 12
    operator∗=, 12
    operator+, 13
    operator+=, 13
    operator-, 13
    operator-=, 14
    operator/, 14
    operator/=, 14
    operator=, 15
    operator==, 15
    Vector2, 10
    x, 15
    y, 15
math::Vector3, 16
    ~Vector3, 18
    cross, 18
    dot, 18
    magnitude, 18
    magnitudeSquared, 19
    normal, 19
    normalise, 19
    operator∗, 19
    operator∗=, 20
    operator+, 20
    operator+=, 20
    operator-, 21
    operator-=, 21
    operator/, 21
    operator/=, 22
    operator=, 22
    operator==, 22
    Vector3, 17
    x, 23
    y, 23
    z, 23

normal
    math::Vector2, 12
    math::Vector3, 19
normalise
    math::Vector2, 12
    math::Vector3, 19

operator∗
    math::Vector2, 12
    math::Vector3, 19
operator∗=
    math::Vector2, 12
    math::Vector3, 20
operator+
    math::Vector2, 13
    math::Vector3, 20
operator+=
    math::Vector2, 13
    math::Vector3, 20
operator-
    math::Vector2, 13
    math::Vector3, 21
operator-=
    math::Vector2, 14
    math::Vector3, 21
operator/
    math::Vector2, 14
    math::Vector3, 21
operator/=
    math::Vector2, 14
    math::Vector3, 22
operator=
    math::Vector2, 15
    math::Vector3, 22

operator==
    math::Vector2, 15
    math::Vector3, 22

src/MathLibrary.hpp, 25
src/Vector2.hpp, 25
src/Vector3.hpp, 25

TEST
    Vector2Tests.cpp, 27–30
    Vector3Tests.cpp, 30–33
test/main.cpp, 26
test/src/Vector2Tests.cpp, 26
test/src/Vector3Tests.cpp, 30

Vector2
    math::Vector2, 10
Vector2Tests.cpp
    TEST, 27–30
Vector3
    math::Vector3, 17
Vector3Tests.cpp
    TEST, 30–33

x
    math::Vector2, 15
    math::Vector3, 23

y
    math::Vector2, 15
    math::Vector3, 23

z
    math::Vector3, 23