SFML

* [Main Page](http://docs.google.com/index.htm)
* [Modules](http://docs.google.com/modules.htm)
* [Classes](http://docs.google.com/annotated.htm)
* [Files](http://docs.google.com/files.htm)
* [File List](http://docs.google.com/files.htm)
* [include](http://docs.google.com/dir_f3190241575fd2bd132a392ae6942f4a.htm)
* [SFML](http://docs.google.com/dir_692f376662c82a26cfe4cfa3aceebe24.htm)
* [System](http://docs.google.com/dir_60c5c649f8df3b69a45a020d59f81335.htm)

Vector3.inl

1

2 //

3 // SFML - Simple and Fast Multimedia Library

4 // Copyright (C) 2007-2012 Laurent Gomila (laurent.gom@gmail.com)

5 //

6 // This software is provided 'as-is', without any express or implied warranty.

7 // In no event will the authors be held liable for any damages arising from the use of this software.

8 //

9 // Permission is granted to anyone to use this software for any purpose,

10 // including commercial applications, and to alter it and redistribute it freely,

11 // subject to the following restrictions:

12 //

13 // 1. The origin of this software must not be misrepresented;

14 // you must not claim that you wrote the original software.

15 // If you use this software in a product, an acknowledgment

16 // in the product documentation would be appreciated but is not required.

17 //

18 // 2. Altered source versions must be plainly marked as such,

19 // and must not be misrepresented as being the original software.

20 //

21 // 3. This notice may not be removed or altered from any source distribution.

22 //

24

25

27 template <typename T>

28 inline Vector3<T>::Vector3() :

29 x(0),

30 y(0),

31 z(0)

32 {

33

34 }

35

36

38 template <typename T>

39 inline [Vector3<T>::Vector3](http://docs.google.com/classsf_1_1Vector3.htm#aee8be1985c6e45e381ad4071265636f9)(T X, T Y, T Z) :

40 x(X),

41 y(Y),

42 z(Z)

43 {

44

45 }

46

47

49 template <typename T>

50 template <typename U>

51 inline [Vector3<T>::Vector3](http://docs.google.com/classsf_1_1Vector3.htm#aee8be1985c6e45e381ad4071265636f9)(const Vector3<U>& vector) :

52 x(static\_cast<T>(vector.x)),

53 y(static\_cast<T>(vector.y)),

54 z(static\_cast<T>(vector.z))

55 {

56 }

57

58

60 template <typename T>

61 inline Vector3<T> operator -(const Vector3<T>& left)

62 {

63  return Vector3<T>(-left.x, -left.y, -left.z);

64 }

65

66

68 template <typename T>

69 inline Vector3<T>& operator +=(Vector3<T>& left, const Vector3<T>& right)

70 {

71  left.x += right.x;

72  left.y += right.y;

73  left.z += right.z;

74

75  return left;

76 }

77

78

80 template <typename T>

81 inline Vector3<T>& operator -=(Vector3<T>& left, const Vector3<T>& right)

82 {

83  left.x -= right.x;

84  left.y -= right.y;

85  left.z -= right.z;

86

87  return left;

88 }

89

90

92 template <typename T>

93 inline Vector3<T> operator +(const Vector3<T>& left, const Vector3<T>& right)

94 {

95  return Vector3<T>(left.x + right.x, left.y + right.y, left.z + right.z);

96 }

97

98

100 template <typename T>

101 inline Vector3<T> operator -(const Vector3<T>& left, const Vector3<T>& right)

102 {

103  return Vector3<T>(left.x - right.x, left.y - right.y, left.z - right.z);

104 }

105

106

108 template <typename T>

109 inline Vector3<T> operator \*(const Vector3<T>& left, T right)

110 {

111  return Vector3<T>(left.x \* right, left.y \* right, left.z \* right);

112 }

113

114

116 template <typename T>

117 inline Vector3<T> operator \*(T left, const Vector3<T>& right)

118 {

119  return Vector3<T>(right.x \* left, right.y \* left, right.z \* left);

120 }

121

122

124 template <typename T>

125 inline Vector3<T>& operator \*=(Vector3<T>& left, T right)

126 {

127  left.x \*= right;

128  left.y \*= right;

129  left.z \*= right;

130

131  return left;

132 }

133

134

136 template <typename T>

137 inline Vector3<T> operator /(const Vector3<T>& left, T right)

138 {

139  return Vector3<T>(left.x / right, left.y / right, left.z / right);

140 }

141

142

144 template <typename T>

145 inline Vector3<T>& operator /=(Vector3<T>& left, T right)

146 {

147  left.x /= right;

148  left.y /= right;

149  left.z /= right;

150

151  return left;

152 }

153

154

156 template <typename T>

157 inline bool operator ==(const Vector3<T>& left, const Vector3<T>& right)

158 {

159  return (left.x == right.x) && (left.y == right.y) && (left.z == right.z);

160 }

161

162

164 template <typename T>

165 inline bool operator !=(const Vector3<T>& left, const Vector3<T>& right)

166 {

167  return (left.x != right.x) || (left.y != right.y) || (left.z != right.z);

168 }

Copyright � Laurent Gomila  ::  Documentation generated by [doxygen](http://www.doxygen.org/)  ::