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.hpp

1

2 //

3 // SFML - Simple and Fast Multimedia Library

4 // Copyright (C) 2007-2013 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 #ifndef SFML\_VECTOR3\_HPP

26 #define SFML\_VECTOR3\_HPP

27

28

29 namespace sf

30 {

36 template <typename T>

[37](http://docs.google.com/classsf_1_1Vector3.htm) class [Vector3](http://docs.google.com/classsf_1_1Vector3.htm)

38 {

39 public :

40

47  [Vector3](http://docs.google.com/classsf_1_1Vector3.htm#aee8be1985c6e45e381ad4071265636f9)();

48

57  [Vector3](http://docs.google.com/classsf_1_1Vector3.htm#aee8be1985c6e45e381ad4071265636f9)(T X, T Y, T Z);

58

70  template <typename U>

71  explicit [Vector3](http://docs.google.com/classsf_1_1Vector3.htm#aee8be1985c6e45e381ad4071265636f9)(const [Vector3<U>](http://docs.google.com/classsf_1_1Vector3.htm)& vector);

72

74  // Member data

[76](http://docs.google.com/classsf_1_1Vector3.htm#a3cb0c769390bc37c346bb1a69e510d16)  T [x](http://docs.google.com/classsf_1_1Vector3.htm#a3cb0c769390bc37c346bb1a69e510d16);

[77](http://docs.google.com/classsf_1_1Vector3.htm#a6590d50ccb862c5efc5512e974e9b794)  T [y](http://docs.google.com/classsf_1_1Vector3.htm#a6590d50ccb862c5efc5512e974e9b794);

[78](http://docs.google.com/classsf_1_1Vector3.htm#a2f36ab4b552c028e3a9734c1ad4df7d1)  T [z](http://docs.google.com/classsf_1_1Vector3.htm#a2f36ab4b552c028e3a9734c1ad4df7d1);

79 };

80

90 template <typename T>

91 [Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm) operator -(const [Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm)& left);

92

106 template <typename T>

107 [Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm)& operator +=([Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm)& left, const [Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm)& right);

108

122 template <typename T>

123 [Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm)& operator -=([Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm)& left, const [Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm)& right);

124

135 template <typename T>

136 [Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm) operator +(const [Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm)& left, const [Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm)& right);

137

148 template <typename T>

149 [Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm) operator -(const [Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm)& left, const [Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm)& right);

150

161 template <typename T>

162 [Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm) operator \*(const [Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm)& left, T right);

163

174 template <typename T>

175 [Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm) operator \*(T left, const [Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm)& right);

176

190 template <typename T>

191 [Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm)& operator \*=([Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm)& left, T right);

192

203 template <typename T>

204 [Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm) operator /(const [Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm)& left, T right);

205

219 template <typename T>

220 [Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm)& operator /=([Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm)& left, T right);

221

234 template <typename T>

235 bool operator ==(const [Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm)& left, const [Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm)& right);

236

249 template <typename T>

250 bool operator !=(const [Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm)& left, const [Vector3<T>](http://docs.google.com/classsf_1_1Vector3.htm)& right);

251

252 #include <SFML/System/Vector3.inl>

253

254 // Define the most common types

255 typedef [Vector3<int>](http://docs.google.com/classsf_1_1Vector3.htm) [Vector3i](http://docs.google.com/classsf_1_1Vector3.htm);

256 typedef [Vector3<float>](http://docs.google.com/classsf_1_1Vector3.htm) [Vector3f](http://docs.google.com/classsf_1_1Vector3.htm);

257

258 } // namespace sf

259

260

261 #endif // SFML\_VECTOR3\_HPP

262

263

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