

[simd / simd_double3](#)

Type Alias

simd_double3

A vector of three 64-bit floating-point elements.

```
typealias simd_double3 = SIMD3<Double>
```

Topics

Functions to Create Three-Element Vectors From Other Vectors

```
func simd_make_double3(simd_double2) -> simd_double3
```

Returns a new vector from the specified two-element vector, and other elements set to zero.

```
func simd_make_double3(simd_double3) -> simd_double3
```

Returns a new vector from the specified vector.

```
func simd_make_double3(simd_double4) -> simd_double3
```

Returns a new vector by truncating the specified four-element vector.

```
func simd_make_double3(simd_double8) -> simd_double3
```

Returns a new vector by truncating the specified eight-element vector.

```
func simd_make_double3_undef(simd_double2) -> simd_double3
```

Returns a new vector from the specified two-element vector, and other elements undefined.

Functions to Create Three-Element Vectors From Scalar Values

```
func simd_make_double3(Double) -> simd_double3
```

Returns a new vector with the first element set to a scalar value, and other elements set to zero.

```
func simd_make_double3(Double, Double, Double) -> simd_double3
```

Returns a new vector from the specified scalar values.

```
func simd_make_double3_undef(Double) -> simd_double3
```

Returns a new vector with the first element set to a scalar value, and other elements undefined.

Functions to Create Three-Element Vectors From Combinations of Vectors and Scalar Values

```
func simd_make_double3(Double, simd_double2) -> simd_double3
```

Returns a new vector from a scalar value and a vector.

```
func simd_make_double3(simd_double2, Double) -> simd_double3
```

Returns a new vector from a vector and a scalar value.

Common Functions

```
func simd_abs(simd_double3) -> simd_double3
```

Returns the absolute value of each element in a vector.

```
func abs(SIMD3<Double>) -> SIMD3<Double>
```

Returns the absolute value of each element in a vector.

```
func simd_clamp(simd_double3, simd_double3, simd_double3) -> simd_double3
```

Returns each element in a vector clamped to a specified range.

```
func clamp(SIMD3<Double>, min: Double, max: Double) -> SIMD3<Double>
```

Returns each element in a vector clamped to a specified range.

```
func clamp(SIMD3<Double>, min: SIMD3<Double>, max: SIMD3<Double>) -> SIMD3<Double>
```

Returns each element in a vector clamped to a specified range.

```
func simd_equal(simd_double3, simd_double3) -> simd_bool
```

Returns true if every element in a vector is exactly equal to the corresponding element in a second vector, and otherwise returns false.

```
func simd_fract(simd_double3) -> simd_double3
```

Returns the fractional part of each element in a vector.

```
func fract(SIMD3<Double>) -> SIMD3<Double>
```

Returns the fractional part of each element in a vector.

```
func simd_sign(simd_double3) -> simd_double3
```

Returns the sign of each element in a vector.

```
func sign(SIMD3<Double>) -> SIMD3<Double>
```

Returns the sign of each element in a vector.

```
func simd_step(simd_double3, simd_double3) -> simd_double3
```

Returns zero for each element in a vector less than a specified edge, and otherwise returns one.

```
func step(SIMD3<Double>, edge: SIMD3<Double>) -> SIMD3<Double>
```

Returns zero for each element in a vector less than a specified edge, and otherwise returns one.

Reduce Functions

```
func simd_reduce_add(simd_double3) -> Double
```

Returns the sum of all elements in a vector.

```
func reduce_add(SIMD3<Double>) -> Double
```

Returns the sum of all elements in a vector.

```
func simd_reduce_max(simd_double3) -> Double
```

Returns the maximum value in a vector.

```
func reduce_max(SIMD3<Double>) -> Double
```

Returns the maximum value in a vector.

```
func simd_reduce_min(simd_double3) -> Double
```

Returns the minimum value in a vector.

```
func reduce_min(SIMD3<Double>) -> Double
```

Returns the minimum value in a vector.

Interpolation Functions

```
func simd_mix(simd_double3, simd_double3, simd_double3) -> simd_double3  
    Returns an element-wise linearly interpolated value between two vectors.
```

```
func mix(SIMD3<Double>, SIMD3<Double>, t: Double) -> SIMD3<Double>  
    Returns an element-wise linearly interpolated value between two vectors.
```

```
func mix(SIMD3<Double>, SIMD3<Double>, t: SIMD3<Double>) -> SIMD3<Double>  
    Returns an element-wise linearly interpolated value between two vectors.
```

```
func simd_smoothstep(simd_double3, simd_double3, simd_double3) -> simd_double3  
    Returns an element-wise smoothly interpolated value between two vectors.
```

```
func smoothstep(SIMD3<Double>, edge0: SIMD3<Double>, edge1: SIMD3<Double>) -> SIMD3<Double>
```

Returns an element-wise smoothly interpolated value between two vectors.

Extrema Functions

```
func simd_min(simd_double3, simd_double3) -> simd_double3  
    Returns the minimum value of each element in a vector.
```

```
func min(SIMD3<Double>, Double) -> SIMD3<Double>  
    Returns the minimum value of each element in a vector.
```

```
func min(SIMD3<Double>, SIMD3<Double>) -> SIMD3<Double>  
    Returns the minimum value of each element in a vector.
```

```
func fmin(SIMD3<Double>, SIMD3<Double>) -> SIMD3<Double>  
    Returns the minimum value of each element in a vector.
```

```
func simd_max(simd_double3, simd_double3) -> simd_double3  
    Returns the maximum value of each element in a vector.
```

```
func max(SIMD3<Double>, Double) -> SIMD3<Double>  
    Returns the maximum value of each element in a vector.
```

```
func max(SIMD3<Double>, SIMD3<Double>) -> SIMD3<Double>  
    Returns the maximum value of each element in a vector.
```

```
func fmax(SIMD3<Double>, SIMD3<Double>) -> SIMD3<Double>
```

Returns the maximum value of each element in a vector.

Reciprocal and Reciprocal Square Root Functions

`func simd_recip(simd_double3) -> simd_double3`

Returns the reciprocal of each element in a vector.

`func recip(SIMD3<Double>) -> SIMD3<Double>`

Returns the reciprocal of each element in a vector.

`func simd_rsqrt(simd_double3) -> simd_double3`

Returns the reciprocal square root of each element in a vector.

`func rsqrt(SIMD3<Double>) -> SIMD3<Double>`

Returns the reciprocal square root of each element in a vector.

`func simd_precise_recip(simd_double3) -> simd_double3`

Returns the precise reciprocal of each element in a vector.

`func simd_precise_rsqrt(simd_double3) -> simd_double3`

Returns the precise reciprocal square root of each element in a vector.

`func simd_fast_recip(simd_double3) -> simd_double3`

Returns the fast reciprocal of each element in a vector.

`func simd_fast_rsqrt(simd_double3) -> simd_double3`

Returns the fast reciprocal square root of each element in a vector.

Exponential and Logarithmic Functions

`func exp(simd_double3) -> simd_double3`

Returns e raised to the power of each element in a vector.

`func exp2(simd_double3) -> simd_double3`

Returns 2 raised to the power of each element in a vector.

`func exp10(simd_double3) -> simd_double3`

Returns 10 raised to the power of each element in a vector.

`func expm1(simd_double3) -> simd_double3`

Returns $e^x - 1$ for each element in a vector.

```
func log(simd_double3) -> simd_double3
```

Returns the natural logarithm of each element in a vector.

```
func log2(simd_double3) -> simd_double3
```

Returns the base 2 logarithm of each element in a vector.

```
func log10(simd_double3) -> simd_double3
```

Returns the base 10 logarithm of each element in a vector.

```
func log1p(simd_double3) -> simd_double3
```

Returns $\log(1+x)$ of each element in a vector.

Geometry Functions

```
func cross(SIMD3<Double>, SIMD3<Double>) -> SIMD3<Double>
```

Returns the cross product of two vectors.

```
func dot(SIMD3<Double>, SIMD3<Double>) -> Double
```

Returns the dot product of two vectors.

```
func normalize(SIMD3<Double>) -> SIMD3<Double>
```

Returns a vector pointing in the same direction of the supplied vector with a length of 1.

```
func project(SIMD3<Double>, SIMD3<Double>) -> SIMD3<Double>
```

Returns the first vector projected onto the second vector.

```
func reflect(SIMD3<Double>, n: SIMD3<Double>) -> SIMD3<Double>
```

Returns the reflection direction of an incident vector and a unit normal vector.

```
func refract(SIMD3<Double>, n: SIMD3<Double>, eta: Double) -> SIMD3<Double>
```

Returns the refraction direction of an incident vector, a unit normal vector, and an index of refraction eta.

Vector Norm Functions

```
func norm_one(SIMD3<Double>) -> Double
```

Returns the sum of the absolute values of a vector.

```
func norm_inf(SIMD3<Double>) -> Double
```

Returns the maximum absolute value of a vector.

Length and Distance Functions

```
func length(SIMD3<Double>) -> Double
```

Returns the length of a vector.

```
func length_squared(SIMD3<Double>) -> Double
```

Returns the square of the length of a vector.

```
func distance(SIMD3<Double>, SIMD3<Double>) -> Double
```

Returns the distance between two vectors.

```
func distance_squared(SIMD3<Double>, SIMD3<Double>) -> Double
```

Hyperbolic Functions

```
func acosh(simd_double3) -> simd_double3
```

Returns the inverse hyperbolic cosine of each element in a vector.

```
func asinh(simd_double3) -> simd_double3
```

Returns the inverse hyperbolic sine of each element in a vector.

```
func atanh(simd_double3) -> simd_double3
```

Returns the inverse hyperbolic tangent of each element in a vector.

```
func cosh(simd_double3) -> simd_double3
```

Returns the hyperbolic cosine of each element in a vector.

```
func sinh(simd_double3) -> simd_double3
```

Returns the hyperbolic sine of each element in a vector.

```
func tanh(simd_double3) -> simd_double3
```

Returns the hyperbolic tangent of each element in a vector.

Logic Functions

```
func simd_select(simd_double3, simd_double3, simd_long3) -> simd_double3
```

Returns a vector that contains elements from either the first or second parameter, based on the corresponding element's high-order bit in the mask.

```
func simd_bitselect(simd_double3, simd_double3, simd_long3) -> simd_double3
```

Returns a vector that contains elements from either the first or second parameter, based on the corresponding element in the third parameter.

Math Functions

```
func cbrt(simd_double3) -> simd_double3
```

Returns the cube root of each element in a vector.

```
func ceil(SIMD3<Double>) -> SIMD3<Double>
```

Returns the ceiling of each element in a vector.

```
func erf(simd_double3) -> simd_double3
```

Returns the error function for each element in a vector.

```
func erfc(simd_double3) -> simd_double3
```

Returns the complementary error function for each element in a vector.

```
func floor(SIMD3<Double>) -> SIMD3<Double>
```

Returns the floor of each element in a vector.

```
func fma(simd_double3, simd_double3, simd_double3) -> simd_double3
```

Returns the multiply-add result for corresponding elements in three vectors.

```
func fmod(simd_double3, simd_double3) -> simd_double3
```

Returns the modulus after dividing each element in a vector by the corresponding element in a second vector.

```
func hypot(simd_double3, simd_double3) -> simd_double3
```

Returns the hypotenuse of a right-angled triangle with the sides that are adjacent to the right angle that two vectors define.

```
func lgamma(simd_double3) -> simd_double3
```

Returns the natural logarithm of the absolute value of the gamma function of each element in a vector.

```
func nextafter(simd_double3, simd_double3) -> simd_double3
```

Returns the next representable value of each element in a vector in the direction of the corresponding element in a second vector.

```
func pow(simd_double3, simd_double3) -> simd_double3
```

Returns each element in a vector raised to the power of the corresponding element in a second vector.

```
func remainder(simd_double3, simd_double3) -> simd_double3
```

Returns the remainder after dividing each element in an array by the corresponding element in a second array of double-precision values.

```
func round(simd_double3) -> simd_double3
```

Returns each element in a vector rounded to the nearest integer.

```
func simd_muladd(simd_double3, simd_double3, simd_double3) -> simd_double3
```

Returns the multiply-add result for corresponding elements in three vectors.

```
func tgamma(simd_double3) -> simd_double3
```

Returns the gamma function for each element in a vector.

```
func trunc(SIMD3<Double>) -> SIMD3<Double>
```

Returns each element in a vector rounded toward zero to the nearest integer.

Trigonometric Functions

```
func acos(simd_double3) -> simd_double3
```

Returns the arccosine of each element in a vector.

```
func asin(simd_double3) -> simd_double3
```

Returns the arcsine of each element in a vector.

```
func atan(simd_double3) -> simd_double3
```

Returns the arctangent of each element in a vector.

```
func atan2(simd_double3, simd_double3) -> simd_double3
```

Returns the arctangent of each pair of corresponding elements in two vectors.

```
func cos(simd_double3) -> simd_double3
```

Returns the cosine of each element in a vector.

```
func cospi(simd_double3) -> simd_double3
```

Returns the cosine of each element in a vector multiplied by pi.

```
func sin(simd_double3) -> simd_double3
```

Returns the sine of each element in a vector.

```
func sinpi(simd_double3) -> simd_double3
```

Returns the sine of each element in a vector multiplied by pi.

```
func sincos(simd_double3) -> (sin: simd_double3, cos: simd_double3)
```

Returns the sine and cosine of each element in a vector.

```
func sincospi(simd_double3) -> (sin: simd_double3, cos: simd_double3)
```

Returns the sine and cosine of each element in a vector multiplied by pi.

```
func tan(simd_double3) -> simd_double3
```

Returns the tangent of each element in a vector.

```
func tanpi(simd_double3) -> simd_double3
```

Returns the tangent of each element in a vector multiplied by pi.

Classification Functions

```
func isfinite(simd_double3) -> simd_long3
```

Returns true for each element that is finite in a vector.

```
func isnan(simd_double3) -> simd_long3
```

Returns true for each element that is infinite in a vector.

```
func isnan(simd_double3) -> simd_long3
```

Returns true for each element that is not a number (NaN) in a vector.

```
func isnormal(simd_double3) -> simd_long3
```

Returns true for each element that is normal in a vector.

Alternative Type Alias

```
typealias vector_double3
```

```
typealias double3      Deprecated
```

See Also

[Vector data types](#)

```
typealias simd_double1
```

A vector of one 64-bit floating-point element.

```
typealias simd_double2
```

A vector of two 64-bit floating-point elements.

```
typealias simd_double4
```

A vector of four 64-bit floating-point elements.

```
typealias simd_double8
```

A vector of eight 64-bit floating-point elements.