Vector Math Library
C++ PPU Reference

Table of Contents

Introduction

Library Summary

Library Contents

Item	Description
Vectormath	The namespace containing the Vectormath library.
Vectormath::Aos	The namespace containing array-of-structures (AoS) classes.
Vectormath::Aos::Matrix3	A 3x3 matrix in array-of-structures format.
Vectormath::Aos::Matrix4	A 4x4 matrix in array-of-structures format.
Vectormath::Aos::Point3	A 3-D point in array-of-structures format.
Vectormath::Aos::Quat	A quaternion in array-of-structures format.
Vectormath::Aos::Transform3	A 3x4 transformation matrix in array-of-structures format.
Vectormath::Aos::Vector3	A 3-D vector in array-of-structures format.
Vectormath::Aos::Vector4	A 4-D vector in array-of-structures format.
Vectormath::Soa	The namespace containing structure-of-arrays (SoA) classes.
Vectormath::Soa::Matrix3	A set of four 3x3 matrices in structure-of-arrays format.
Vectormath::Soa::Matrix4	A set of four 4x4 matrices in structure-of-arrays format.
Vectormath::Soa::Point3	A set of four 3-D points in structure-of-arrays format.
Vectormath::Soa::Quat	A set of four quaternions in structure-of-arrays format.
Vectormath::Soa::Transform3	A set of four 3x4 transformation matrices in structure-of-arrays format.
Vectormath::Soa::Vector3	A set of four 3-D vectors in structure-of-arrays format.
Vectormath::Soa::Vector4	A set of four 4-D vectors in structure-of-arrays format.

Vectormath

Summary

Vectormath

The namespace containing the Vectormath library.

Definition

```
namespace Vectormath {}
```

Description

The namespace containing the Vectormath library.

Inner Classes, Structures, and Namespaces

Item	Description
Vectormath::Aos	The namespace containing array-of-structures (AoS) classes.
Vectormath::Soa	The namespace containing structure-of-arrays (SoA) classes.

Vectormath::Aos

Summary

Vectormath::Aos

The namespace containing array-of-structures (AoS) classes.

Definition

```
namespace Aos {}
```

Description

The namespace containing array-of-structures (AoS) classes.

Function Summary

Function	Description
absPerElem	Compute the absolute value of a 3-D vector per element.
absPerElem	Compute the absolute value of a 4-D vector per element.
absPerElem	Compute the absolute value of a 3-D point per element.
absPerElem	Compute the absolute value of a 3x3 matrix per element.
absPerElem	Compute the absolute value of a 4x4 matrix per element.
absPerElem	Compute the absolute value of a 3x4 transformation matrix per element.
affineInverse	Compute the inverse of a 4x4 matrix, which is expected to be an affine matrix.
appendScale	Append (post-multiply) a scale transformation to a 3x3 matrix.
appendScale	Append (post-multiply) a scale transformation to a 4x4 matrix.
appendScale	Append (post-multiply) a scale transformation to a 3x4 transformation matrix.
conj	Compute the conjugate of a quaternion.
copySignPerElem	Copy sign from one 3-D vector to another, per element.
copySignPerElem	Copy sign from one 4-D vector to another, per element.
copySignPerElem	Copy sign from one 3-D point to another, per element.
cross	Compute cross product of two 3-D vectors.
crossMatrix	Cross-product matrix of a 3-D vector.
crossMatrixMul	Create cross-product matrix and multiply.
determinant	Determinant of a 3x3 matrix.
determinant	Determinant of a 4x4 matrix.
dist	Compute the distance between two 3-D points.
distFromOrigin	Compute the distance of a 3-D point from the coordinate-system origin.
distSqr	Compute the square of the distance between two 3-D points.
distSqrFromOrigin	Compute the square of the distance of a 3-D point from the coordinate-system origin.
divPerElem	Divide two 3-D vectors per element.
divPerElem	Divide two 4-D vectors per element.
divPerElem	Divide two 3-D points per element.
dot	Compute the dot product of two 3-D vectors.
dot	Compute the dot product of two 4-D vectors.

Function	Description
dot	Compute the dot product of two quaternions.
inverse	Compute the inverse of a 3x3 matrix.
inverse	Compute the inverse of a 4x4 matrix.
inverse	Inverse of a 3x4 transformation matrix.
length	Compute the length of a 3-D vector.
length	Compute the length of a 4-D vector.
length	Compute the length of a quaternion.
lengthSqr	Compute the square of the length of a 3-D vector.
lengthSqr	Compute the square of the length of a 4-D vector.
lerp	Linear interpolation between two 3-D vectors.
lerp	Linear interpolation between two 3-D vectors (scalar data contained in vector data type).
lerp	Linear interpolation between two 4-D vectors.
lerp	Linear interpolation between two 4-D vectors (scalar data contained in vector data type).
lerp	Linear interpolation between two 3-D points.
lerp	Linear interpolation between two 3-D points (scalar data contained in vector data type).
lerp	Linear interpolation between two quaternions.
lerp	Linear interpolation between two quaternions (scalar data contained in vector data type).
loadXYZArray	Load four three-float 3-D vectors, stored in three quadwords.
loadXYZArray	Load four three-float 3-D points, stored in three quadwords.
maxElem	Maximum element of a 3-D vector.
maxElem	Maximum element of a 4-D vector.
maxElem	Maximum element of a 3-D point.
maxPerElem	Maximum of two 3-D vectors per element.
maxPerElem	Maximum of two 4-D vectors per element.
maxPerElem	Maximum of two 3-D points per element.
minElem	Minimum element of a 3-D vector.
minElem	Minimum element of a 4-D vector.
minElem	Minimum element of a 3-D point.
minPerElem	Minimum of two 3-D vectors per element.
minPerElem	Minimum of two 4-D vectors per element.
minPerElem	Minimum of two 3-D points per element.
mulPerElem	Multiply two 3-D vectors per element.
mulPerElem	Multiply two 4-D vectors per element.
mulPerElem	Multiply two 3-D points per element.
mulPerElem	Multiply two 3x3 matrices per element.
mulPerElem	Multiply two 4x4 matrices per element.
mulPerElem	Multiply two 3x4 transformation matrices per element.
norm	Compute the norm of a quaternion.
normalize	Normalize a 3-D vector.
normalize	Normalize a 4-D vector.
normalize	Normalize a quaternion.
operator *	Multiply a 3-D vector by a scalar.
operator *	Multiply a 3-D vector by a scalar (scalar data contained in vector data type).
operator *	Multiply a 4-D vector by a scalar.
operator *	Multiply a 4-D vector by a scalar (scalar data contained in vector data type).
operator *	Multiply a quaternion by a scalar.

Function	Description
operator *	Multiply a quaternion by a scalar (scalar data contained in vector data type).
operator *	Multiply a 3x3 matrix by a scalar.
operator *	Multiply a 3x3 matrix by a scalar (scalar data contained in vector data type).
operator *	Multiply a 4x4 matrix by a scalar.
operator *	Multiply a 4x4 matrix by a scalar (scalar data contained in vector data type).
orthoInverse	Compute the inverse of a 4x4 matrix, which is expected to be an affine matrix with an orthogonal upper-left 3x3 submatrix.
orthoInverse	Compute the inverse of a 3x4 transformation matrix, expected to have an orthogonal upper-left 3x3 submatrix.
outer	Outer product of two 3-D vectors.
outer	Outer product of two 4-D vectors.
prependScale	Prepend (pre-multiply) a scale transformation to a 3x3 matrix.
prependScale	Prepend (pre-multiply) a scale transformation to a 4x4 matrix.
prependScale	Prepend (pre-multiply) a scale transformation to a 3x4 transformation matrix.
print	Print a 3-D vector.
print	Print a 3-D vector and an associated string identifier.
print	Print a 4-D vector.
print	Print a 4-D vector and an associated string identifier.
print	Print a 3-D point.
print	Print a 3-D point and an associated string identifier.
print	Print a quaternion.
print	Print a quaternion and an associated string identifier.
print	Print a 3x3 matrix.
print	Print a 3x3 matrix and an associated string identifier.
print	Print a 4x4 matrix.
print	Print a 4x4 matrix and an associated string identifier.
print	Print a 3x4 transformation matrix.
print	Print a 3x4 transformation matrix and an associated string identifier.
projection	Scalar projection of a 3-D point on a unit-length 3-D vector.
recipPerElem	Compute the reciprocal of a 3-D vector per element.
recipPerElem	Compute the reciprocal of a 4-D vector per element.
recipPerElem	Compute the reciprocal of a 3-D point per element.
rotate	Use a unit-length quaternion to rotate a 3-D vector.
rowMul	Pre-multiply a row vector by a 3x3 matrix.
rsqrtPerElem	Compute the reciprocal square root of a 3-D vector per element.
rsqrtPerElem	Compute the reciprocal square root of a 4-D vector per element.
rsqrtPerElem	Compute the reciprocal square root of a 3-D point per element.
scale	Apply uniform scale to a 3-D point.
scale	Apply uniform scale to a 3-D point (scalar data contained in vector data type).
scale	Apply non-uniform scale to a 3-D point.
select	Conditionally select between two 3-D vectors.

Function	Description
select	Conditionally select between two 3-D vectors (scalar data contained in vector data type).
select	Conditionally select between two 4-D vectors.
select	Conditionally select between two 4-D vectors (scalar data contained in vector data type).
select	Conditionally select between two 3-D points.
select	Conditionally select between two 3-D points (scalar data contained in vector data type).
select	Conditionally select between two quaternions.
select	Conditionally select between two quaternions (scalar data contained in vector data type).
select	Conditionally select between two 3x3 matrices.
select	Conditionally select between two 3x3 matrices (scalar data contained in vector data type).
select	Conditionally select between two 4x4 matrices.
select	Conditionally select between two 4x4 matrices (scalar data contained in vector data type).
select	Conditionally select between two 3x4 transformation matrices.
select	Conditionally select between two 3x4 transformation matrices (scalar data contained in vector data type).
slerp	Spherical linear interpolation between two 3-D vectors.
slerp	Spherical linear interpolation between two 3-D vectors (scalar data contained in vector data type).
slerp	Spherical linear interpolation between two 4-D vectors.
slerp	Spherical linear interpolation between two 4-D vectors (scalar data contained in vector data type).
slerp	Spherical linear interpolation between two quaternions.
slerp	Spherical linear interpolation between two quaternions (scalar data contained in vector data type).
sqrtPerElem	Compute the square root of a 3-D vector per element.
sqrtPerElem	Compute the square root of a 4-D vector per element.
sqrtPerElem	Compute the square root of a 3-D point per element.
squad	Spherical quadrangle interpolation.
squad	Spherical quadrangle interpolation (scalar data contained in vector data type).
storeHalfFloats	Store eight 3-D vectors as half-floats.
storeHalfFloats	Store four 4-D vectors as half-floats.
storeHalfFloats	Store eight 3-D points as half-floats.
storeXYZ	Store x, y, and z elements of a 3-D vector in the first three words of a quadword. The value of the fourth word (the word with the highest address) remains unchanged.
storeXYZ	Store x, y, and z elements of a 3-D point in the first three words of a quadword. The value of the fourth word (the word with the highest address) remains unchanged.
storeXYZArray	Store four 3-D vectors in three quadwords.
storeXYZArray	Store four 3-D points in three quadwords.
sum	Compute the sum of all elements of a 3-D vector.
sum	Compute the sum of all elements of a 4-D vector.
sum	Compute the sum of all elements of a 3-D point.
transpose	Transpose of a 3x3 matrix.
transpose	Transpose of a 4x4 matrix.

Inner Classes, Structures, and Namespaces

Item	Description
Vectormath::Aos::Matrix3	A 3x3 matrix in array-of-structures format.
Vectormath::Aos::Matrix4	A 4x4 matrix in array-of-structures format.
Vectormath::Aos::Point3	A 3-D point in array-of-structures format.
Vectormath::Aos::Quat	A quaternion in array-of-structures format.
Vectormath::Aos::Transform3	A 3x4 transformation matrix in array-of-structures format.
Vectormath::Aos::Vector3	A 3-D vector in array-of-structures format.
Vectormath::Aos::Vector4	A 4-D vector in array-of-structures format.

3-D Vector Functions

absPerElem

Compute the absolute value of a 3-D vector per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector3 absPerElem(
                     Vector3 vec
              );
        }
}
```

Arguments

vec	3-D vector

Return Values

3-D vector in which each element is the absolute value of the corresponding element of vec

Description

Compute the absolute value of each element of a 3-D vector.

copySignPerElem

Copy sign from one 3-D vector to another, per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector3 copySignPerElem(
                     Vector3 vec0,
                     Vector3 vec1
              );
        }
}
```

Arguments

vec0	3-D vector
vec1	3-D vector

Return Values

3-D vector in which each element has the magnitude of the corresponding element of *vec0* and the sign of the corresponding element of *vec1*

Description

For each element, create a value composed of the magnitude of *vec0* and the sign of *vec1*.

cross

Compute cross product of two 3-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector3 cross(
                     Vector3 vec0,
                     Vector3 vec1
              );
        }
}
```

Arguments

vec0	3-D vector
vec1	3-D vector

Return Values

Cross product of the specified 3-D vectors

Description

Compute cross product of two 3-D vectors.

crossMatrix

Cross-product matrix of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Matrix3 crossMatrix(
                    Vector3 vec
              );
        }
}
```

Arguments

vec	3-D vector

Return Values

Cross-product matrix of *vec*

Description

Compute a matrix that, when multiplied by a 3-D vector, produces the same result as a cross product with that 3-D vector.

crossMatrixMul

Create cross-product matrix and multiply.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Matrix3 crossMatrixMul(
                     Vector3 vec,
                     const Matrix3 &mat
              );
        }
}
```

Arguments

vec	3-D vector
mat	3x3 matrix

Return Values

Product of cross-product matrix of *vec* and *mat*

Description

Multiply a cross-product matrix by another matrix.

Notes

Faster than separately creating a cross-product matrix and multiplying.

divPerElem

Divide two 3-D vectors per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector3 divPerElem(
                     Vector3 vec0,
                     Vector3 vec1
              );
        }
}
```

Arguments

vec0	3-D vector
vec1	3-D vector

Return Values

3-D vector in which each element is the quotient of the corresponding elements of the specified 3-D vectors

Description

Divide two 3-D vectors element by element.

Notes

Floating-point behavior matches standard library function divf4.

dot

Compute the dot product of two 3-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const floatInVec dot(
                     Vector3 vec0,
                     Vector3 vec1
              );
        }
}
```

Arguments

vec0	3-D vector
vec1	3-D vector

Return Values

Dot product of the specified 3-D vectors

Description

Compute the dot product of two 3-D vectors.

length

Compute the length of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const floatInVec length(
                     Vector3 vec
              );
        }
}
```

Arguments

vec	3-D vector

Return Values

Length of the specified 3-D vector

Description

Compute the length of a 3-D vector.

lengthSqr

Compute the square of the length of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const floatInVec lengthSqr(
                     Vector3 vec
              );
        }
}
```

Arguments

vec	3-D vector

Return Values

Square of the length of the specified 3-D vector

Description

Compute the square of the length of a 3-D vector.

lerp

Linear interpolation between two 3-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector3 lerp(
                     float t,
                     Vector3 vec0,
                     Vector3 vec1
              );
        }
}
```

Arguments

t	Interpolation parameter
vec0	3-D vector
vec1	3-D vector

Return Values

Interpolated 3-D vector

Description

Linearly interpolate between two 3-D vectors.

Notes

Does not clamp `t` between 0 and 1.

lerp

Linear interpolation between two 3-D vectors (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector3 lerp(
                     floatInVec t,
                     Vector3 vec0,
                     Vector3 vec1
              );
        }
}
```

Arguments

t	Interpolation parameter
vec0	3-D vector
vec1	3-D vector

Return Values

Interpolated 3-D vector

Description

Linearly interpolate between two 3-D vectors.

Notes

Does not clamp *t* between 0 and 1.

loadXYZArray

Load four three-float 3-D vectors, stored in three quadwords.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline void loadXYZArray(
                     Vector3 &vec0,
                     Vector3 &vec1,
                     Vector3 &vec2,
                     Vector3 &vec3,
                     const vec_float4 *threeQuads
              );
        }
}
```

Arguments

vec0	An output 3-D vector
vec1	An output 3-D vector
vec2	An output 3-D vector
vec3	An output 3-D vector
threeQuads	Array of 3 quadwords containing 12 floats

Return Values

None

Description

Load four three-float 3-D vectors, stored in three quadwords as {x0,y0,z0,x1,y1,z1,x2,y2,z2,x3,y3,z3}, into four 3-D vectors.

maxElem

Maximum element of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const floatInVec maxElem(
                     Vector3 vec
              );
        }
}
```

Arguments

vec	3-D vector

Return Values

Maximum value of all elements of *vec*

Description

Compute the maximum value of all elements of a 3-D vector.

maxPerElem

Maximum of two 3-D vectors per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector3 maxPerElem(
                    Vector3 vec0,
                    Vector3 vec1
              );
        }
}
```

Arguments

vec0	3-D vector
vec1	3-D vector

Return Values

3-D vector in which each element is the maximum of the corresponding elements of the specified 3-D vectors

Description

Create a 3-D vector in which each element is the maximum of the corresponding elements of the specified 3-D vectors.

minElem

Minimum element of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const floatInVec minElem(
                     Vector3 vec
              );
        }
}
```

Arguments

vec	3-D vector

Return Values

Minimum value of all elements of *vec*

Description

Compute the minimum value of all elements of a 3-D vector.

minPerElem

Minimum of two 3-D vectors per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector3 minPerElem(
                    Vector3 vec0,
                    Vector3 vec1
              );
        }
}
```

Arguments

vec0	3-D vector
vec1	3-D vector

Return Values

3-D vector in which each element is the minimum of the corresponding elements of the specified 3-D vectors

Description

Create a 3-D vector in which each element is the minimum of the corresponding elements of two specified 3-D vectors.

mulPerElem

Multiply two 3-D vectors per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector3 mulPerElem(
                     Vector3 vec0,
                     Vector3 vec1
              );
        }
}
```

Arguments

vec0	3-D vector
vec1	3-D vector

Return Values

3-D vector in which each element is the product of the corresponding elements of the specified 3-D vectors

Description

Multiply two 3-D vectors element by element.

normalize

Normalize a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector3 normalize(
                    Vector3 vec
              );
        }
}
```

Arguments

vec	3-D vector

Return Values

The specified 3-D vector scaled to unit length

Description

Compute a normalized 3-D vector.

Notes

The result is unpredictable when all elements of vec are at or near zero.

operator *

Multiply a 3-D vector by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector3 operator *(
                     float scalar,
                     Vector3 vec
              );
        }
}
```

Arguments

scalar	Scalar value
vec	3-D vector

Return Values

Scalar product of *vec* and *scalar*

Description

Multiply a 3-D vector by a scalar.

operator *

Multiply a 3-D vector by a scalar (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector3 operator *(
                     floatInVec scalar,
                     Vector3 vec
              );
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)
vec	3-D vector

Return Values

Scalar product of *vec* and *scalar*

Description

Multiply a 3-D vector by a scalar.

outer

Outer product of two 3-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Matrix3 outer(
                     Vector3 vec0,
                     Vector3 vec1
              );
        }
}
```

Arguments

vec0	3-D vector
vec1	3-D vector

Return Values

The 3x3 matrix product of a column-vector, *vec0*, and a row-vector, *vec1*

Description

Compute the outer product of two 3-D vectors.

print

Print a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline void print(
                     Vector3 vec
              );
        }
}
```

Arguments

vec	3-D vector

Return Values

None

Description

Print a 3-D vector. Prints the 3-D vector transposed, that is, as a row instead of a column.

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

print

Print a 3-D vector and an associated string identifier.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline void print(
                     Vector3 vec,
                     const char *name
              );
        }
}
```

Arguments

vec	3-D vector
name	String printed with the 3-D vector

Return Values

None

Description

Print a 3-D vector and an associated string identifier. Prints the 3-D vector transposed, that is, as a row instead of a column.

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

recipPerElem

Compute the reciprocal of a 3-D vector per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector3 recipPerElem(
                    Vector3 vec
              );
        }
}
```

Arguments

vec	3-D vector

Return Values

3-D vector in which each element is the reciprocal of the corresponding element of the specified 3-D vector

Description

Create a 3-D vector in which each element is the reciprocal of the corresponding element of the specified 3-D vector.

Notes

Floating-point behavior matches standard library function recipf4.

rowMul

Pre-multiply a row vector by a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector3 rowMul(
                     Vector3 vec,
                     const Matrix3 &mat
              );
        }
}
```

Arguments

vec	3-D vector
mat	3x3 matrix

Return Values

Product of a row-vector and a 3x3 matrix

Description

Transpose a 3-D vector into a row vector and pre-multiply by 3x3 matrix.

Notes

Slower than column post-multiply.

rsqrtPerElem

Compute the reciprocal square root of a 3-D vector per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector3 rsqrtPerElem(
                     Vector3 vec
              );
        }
}
```

Arguments

vec	3-D vector

Return Values

3-D vector in which each element is the reciprocal square root of the corresponding element of the specified 3-D vector

Description

Create a 3-D vector in which each element is the reciprocal square root of the corresponding element of the specified 3-D vector.

Notes

Floating-point behavior matches standard library function rsqrtf4.

select

Conditionally select between two 3-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
             inline const Vector3 select(
                    Vector3 vec0,
                    Vector3 vec1,
                    bool select1
             );
        }
}
```

Arguments

vec0	3-D vector
vec1	3-D vector
select1	False selects the vec0 argument, true selects the vec1 argument

Return Values

Equal to `vec0` if `select1` is false, or to `vec1` if `select1` is true

Description

Conditionally select one of the 3-D vector arguments.

Notes

This function uses a conditional select instruction to avoid a branch. However, the transfer of `select1` to a VMX register may use more processing time than a branch. Use the boolInVec version for better performance.

select

Conditionally select between two 3-D vectors (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector3 select(
                     Vector3 vec0,
                     Vector3 vec1,
                     boolInVec select1
              );
        }
}
```

Arguments

vec0	3-D vector
vec1	3-D vector
select1	False selects the vec0 argument, true selects the vec1 argument

Return Values

Equal to *vec0* if *select1* is false, or to *vec1* if *select1* is true

Description

Conditionally select one of the 3-D vector arguments.

Notes

This function uses a conditional select instruction to avoid a branch.

slerp

Spherical linear interpolation between two 3-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector3 slerp(
                     float t,
                     Vector3 unitVec0,
                     Vector3 unitVec1
              );
        }
}
```

Arguments

t	Interpolation parameter
unitVec0	3-D vector, expected to be unit-length
unitVec1	3-D vector, expected to be unit-length

Return Values

Interpolated 3-D vector

Description

Perform spherical linear interpolation between two 3-D vectors.

Notes

The result is unpredictable if the vectors point in opposite directions. Does not clamp `t` between 0 and 1.

slerp

Spherical linear interpolation between two 3-D vectors (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector3 slerp(
                     floatInVec t,
                     Vector3 unitVec0,
                     Vector3 unitVec1
              );
        }
}
```

Arguments

t	Interpolation parameter
unitVec0	3-D vector, expected to be unit-length
unitVec1	3-D vector, expected to be unit-length

Return Values

Interpolated 3-D vector

Description

Perform spherical linear interpolation between two 3-D vectors.

Notes

The result is unpredictable if the vectors point in opposite directions. Does not clamp `t` between 0 and 1.

sqrtPerElem

Compute the square root of a 3-D vector per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector3 sqrtPerElem(
                    Vector3 vec
              );
        }
}
```

Arguments

vec	3-D vector

Return Values

3-D vector in which each element is the square root of the corresponding element of the specified 3-D vector

Description

Create a 3-D vector in which each element is the square root of the corresponding element of the specified 3-D vector.

Notes

Floating-point behavior matches standard library function sqrtf4.

storeHalfFloats

Store eight 3-D vectors as half-floats.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline void storeHalfFloats(
                     Vector3 vec0,
                     Vector3 vec1,
                     Vector3 vec2,
                     Vector3 vec3,
                     Vector3 vec4,
                     Vector3 vec5,
                     Vector3 vec6,
                     Vector3 vec7,
                     vec_ushort8 *threeQuads
              );
        }
}
```

Arguments

vec0	3-D vector
vec1	3-D vector
vec2	3-D vector
vec3	3-D vector
vec4	3-D vector
vec5	3-D vector
vec6	3-D vector
vec7	3-D vector
threeQuads	An output array of 3 quadwords containing 24 half-floats

Return Values

None

Description

Store eight 3-D vectors in three quadwords of half-float values. The output is {x0,y0,z0,x1,y1,z1,x2,y2,z2,x3,y3,z3,x4,y4,z4,x5,y5,z5,x6,y6,z6,x7,y7,z7}.

storeXYZ

Store x, y, and z elements of a 3-D vector in the first three words of a quadword. The value of the fourth word (the word with the highest address) remains unchanged.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline void storeXYZ(
                     Vector3 vec,
                     vec_float4 *quad
              );
        }
}
```

Arguments

vec	3-D vector
quad	Pointer to a quadword in which x, y, and z will be stored

Return Values

None

Description

Store x, y, and z elements of a 3-D vector in the first three words of a quadword. The value of the fourth word (the word with the highest address) remains unchanged.

storeXYZArray

Store four 3-D vectors in three quadwords.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
             inline void storeXYZArray(
                    Vector3 vec0,
                    Vector3 vec1,
                    Vector3 vec2,
                    Vector3 vec3,
                    vec_float4 *threeQuads
             );
        }
}
```

Arguments

vec0	3-D vector
vec1	3-D vector
vec2	3-D vector
vec3	3-D vector
threeQuads	An output array of 3 quadwords containing 12 floats

Return Values

None

Description

Store four 3-D vectors in three quadwords as {x0,y0,z0,x1,y1,z1,x2,y2,z2,x3,y3,z3}.

sum

Compute the sum of all elements of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const floatInVec sum(
                     Vector3 vec
              );
        }
}
```

Arguments

vec	3-D vector

Return Values

Sum of all elements of *vec*

Description

Compute the sum of all elements of a 3-D vector.

4-D Vector Functions

absPerElem

Compute the absolute value of a 4-D vector per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector4 absPerElem(
                     Vector4 vec
              );
        }
}
```

Arguments

vec	4-D vector

Return Values

4-D vector in which each element is the absolute value of the corresponding element of vec

Description

Compute the absolute value of each element of a 4-D vector.

copySignPerElem

Copy sign from one 4-D vector to another, per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector4 copySignPerElem(
                    Vector4 vec0,
                    Vector4 vec1
              );
        }
}
```

Arguments

vec0	4-D vector
vec1	4-D vector

Return Values

4-D vector in which each element has the magnitude of the corresponding element of *vec0* and the sign of the corresponding element of *vec1*

Description

For each element, create a value composed of the magnitude of *vec0* and the sign of *vec1*.

divPerElem

Divide two 4-D vectors per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector4 divPerElem(
                     Vector4 vec0,
                     Vector4 vec1
              );
        }
}
```

Arguments

vec0	4-D vector
vec1	4-D vector

Return Values

4-D vector in which each element is the quotient of the corresponding elements of the specified 4-D vectors

Description

Divide two 4-D vectors element by element.

Notes

Floating-point behavior matches standard library function divf4.

dot

Compute the dot product of two 4-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const floatInVec dot(
                     Vector4 vec0,
                     Vector4 vec1
              );
        }
}
```

Arguments

vec0	4-D vector
vec1	4-D vector

Return Values

Dot product of the specified 4-D vectors

Description

Compute the dot product of two 4-D vectors.

length

Compute the length of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const floatInVec length(
                     Vector4 vec
              );
        }
}
```

Arguments

vec	4-D vector

Return Values

Length of the specified 4-D vector

Description

Compute the length of a 4-D vector.

lengthSqr

Compute the square of the length of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const floatInVec lengthSqr(
                     Vector4 vec
              );
        }
}
```

Arguments

vec	4-D vector

Return Values

Square of the length of the specified 4-D vector

Description

Compute the square of the length of a 4-D vector.

lerp

Linear interpolation between two 4-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector4 lerp(
                     float t,
                     Vector4 vec0,
                     Vector4 vec1
              );
        }
}
```

Arguments

t	Interpolation parameter
vec0	4-D vector
vec1	4-D vector

Return Values

Interpolated 4-D vector

Description

Linearly interpolate between two 4-D vectors.

Notes

Does not clamp *t* between 0 and 1.

lerp

Linear interpolation between two 4-D vectors (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector4 lerp(
                     floatInVec t,
                     Vector4 vec0,
                     Vector4 vec1
              );
        }
}
```

Arguments

t	Interpolation parameter
vec0	4-D vector
vec1	4-D vector

Return Values

Interpolated 4-D vector

Description

Linearly interpolate between two 4-D vectors.

Notes

Does not clamp *t* between 0 and 1.

maxElem

Maximum element of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
             inline const floatInVec maxElem(
                    Vector4 vec
             );
        }
}
```

Arguments

vec	4-D vector

Return Values

Maximum value of all elements of *vec*

Description

Compute the maximum value of all elements of a 4-D vector.

maxPerElem

Maximum of two 4-D vectors per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector4 maxPerElem(
                    Vector4 vec0,
                    Vector4 vec1
              );
        }
}
```

Arguments

vec0	4-D vector
vec1	4-D vector

Return Values

4-D vector in which each element is the maximum of the corresponding elements of the specified 4-D vectors

Description

Create a 4-D vector in which each element is the maximum of the corresponding elements of the specified 4-D vectors.

minElem

Minimum element of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const floatInVec minElem(
                     Vector4 vec
              );
        }
}
```

Arguments

vec	4-D vector

Return Values

Minimum value of all elements of *vec*

Description

Compute the minimum value of all elements of a 4-D vector.

minPerElem

Minimum of two 4-D vectors per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
             inline const Vector4 minPerElem(
                  Vector4 vec0,
                  Vector4 vec1
             );
        }
}
```

Arguments

vec0	4-D vector
vec1	4-D vector

Return Values

4-D vector in which each element is the minimum of the corresponding elements of the specified 4-D vectors

Description

Create a 4-D vector in which each element is the minimum of the corresponding elements of two specified 4-D vectors.

mulPerElem

Multiply two 4-D vectors per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector4 mulPerElem(
                    Vector4 vec0,
                    Vector4 vec1
              );
        }
}
```

Arguments

vec0	4-D vector
vec1	4-D vector

Return Values

4-D vector in which each element is the product of the corresponding elements of the specified 4-D vectors

Description

Multiply two 4-D vectors element by element.

normalize

Normalize a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector4 normalize(
                      Vector4 vec
              );
        }
}
```

Arguments

vec	4-D vector

Return Values

The specified 4-D vector scaled to unit length

Description

Compute a normalized 4-D vector.

Notes

The result is unpredictable when all elements of vec are at or near zero.

operator *

Multiply a 4-D vector by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector4 operator *(
                     float scalar,
                     Vector4 vec
              );
        }
}
```

Arguments

scalar	Scalar value
vec	4-D vector

Return Values

Scalar product of *vec* and *scalar*

Description

Multiply a 4-D vector by a scalar.

operator *

Multiply a 4-D vector by a scalar (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector4 operator *(
                     floatInVec scalar,
                     Vector4 vec
              );
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)
vec	4-D vector

Return Values

Scalar product of *vec* and *scalar*

Description

Multiply a 4-D vector by a scalar.

outer

Outer product of two 4-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Matrix4 outer(
                     Vector4 vec0,
                     Vector4 vec1
              );
        }
}
```

Arguments

vec0	4-D vector
vec1	4-D vector

Return Values

The 4x4 matrix product of a column-vector, *vec0*, and a row-vector, *vec1*

Description

Compute the outer product of two 4-D vectors.

print

Print a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline void print(
                     Vector4 vec
              );
        }
}
```

Arguments

vec	4-D vector

Return Values

None

Description

Print a 4-D vector. Prints the 4-D vector transposed, that is, as a row instead of a column.

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

print

Print a 4-D vector and an associated string identifier.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline void print(
                     Vector4 vec,
                     const char *name
              );
        }
}
```

Arguments

vec	4-D vector
name	String printed with the 4-D vector

Return Values

None

Description

Print a 4-D vector and an associated string identifier. Prints the 4-D vector transposed, that is, as a row instead of a column.

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

recipPerElem

Compute the reciprocal of a 4-D vector per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector4 recipPerElem(
                     Vector4 vec
              );
        }
}
```

Arguments

vec	4-D vector

Return Values

4-D vector in which each element is the reciprocal of the corresponding element of the specified 4-D vector

Description

Create a 4-D vector in which each element is the reciprocal of the corresponding element of the specified 4-D vector.

Notes

Floating-point behavior matches standard library function recipf4.

rsqrtPerElem

Compute the reciprocal square root of a 4-D vector per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector4 rsqrtPerElem(
                     Vector4 vec
              );
        }
}
```

Arguments

vec	4-D vector

Return Values

4-D vector in which each element is the reciprocal square root of the corresponding element of the specified 4-D vector

Description

Create a 4-D vector in which each element is the reciprocal square root of the corresponding element of the specified 4-D vector.

Notes

Floating-point behavior matches standard library function rsqrtf4.

select

Conditionally select between two 4-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
             inline const Vector4 select(
                    Vector4 vec0,
                    Vector4 vec1,
                    bool select1
             );
        }
}
```

Arguments

vec0	4-D vector
vec1	4-D vector
select1	False selects the vec0 argument, true selects the vec1 argument

Return Values

Equal to *vec0* if *select1* is false, or to *vec1* if *select1* is true

Description

Conditionally select one of the 4-D vector arguments.

Notes

This function uses a conditional select instruction to avoid a branch. However, the transfer of `select1` to a VMX register may use more processing time than a branch. Use the boolInVec version for better performance.

select

Conditionally select between two 4-D vectors (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
             inline const Vector4 select(
                    Vector4 vec0,
                    Vector4 vec1,
                    boolInVec select1
             );
        }
}
```

Arguments

vec0	4-D vector
vec1	4-D vector
select1	False selects the vec0 argument, true selects the vec1 argument

Return Values

Equal to *vec0* if *select1* is false, or to *vec1* if *select1* is true

Description

Conditionally select one of the 4-D vector arguments.

Notes

This function uses a conditional select instruction to avoid a branch.

slerp

Spherical linear interpolation between two 4-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector4 slerp(
                     float t,
                     Vector4 unitVec0,
                     Vector4 unitVec1
              );
        }
}
```

Arguments

t	Interpolation parameter
unitVec0	4-D vector, expected to be unit-length
unitVec1	4-D vector, expected to be unit-length

Return Values

Interpolated 4-D vector

Description

Perform spherical linear interpolation between two 4-D vectors.

Notes

The result is unpredictable if the vectors point in opposite directions. Does not clamp `t` between 0 and 1.

slerp

Spherical linear interpolation between two 4-D vectors (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector4 slerp(
                     floatInVec t,
                     Vector4 unitVec0,
                     Vector4 unitVec1
              );
        }
}
```

Arguments

t	Interpolation parameter
unitVec0	4-D vector, expected to be unit-length
unitVec1	4-D vector, expected to be unit-length

Return Values

Interpolated 4-D vector

Description

Perform spherical linear interpolation between two 4-D vectors.

Notes

The result is unpredictable if the vectors point in opposite directions. Does not clamp *t* between 0 and 1.

sqrtPerElem

Compute the square root of a 4-D vector per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector4 sqrtPerElem(
                     Vector4 vec
              );
        }
}
```

Arguments

vec	4-D vector

Return Values

4-D vector in which each element is the square root of the corresponding element of the specified 4-D vector

Description

Create a 4-D vector in which each element is the square root of the corresponding element of the specified 4-D vector.

Notes

Floating-point behavior matches standard library function sqrtf4.

storeHalfFloats

Store four 4-D vectors as half-floats.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
             inline void storeHalfFloats(
                    Vector4 vec0,
                    Vector4 vec1,
                    Vector4 vec2,
                    Vector4 vec3,
                    vec_ushort8 *twoQuads
             );
        }
}
```

Arguments

vec0	4-D vector
vec1	4-D vector
vec2	4-D vector
vec3	4-D vector
twoQuads	An output array of 2 quadwords containing 16 half-floats

Return Values

None

Description

Store four 4-D vectors in two quadwords of half-float values. The output is {x0,y0,z0,w0,x1,y1,z1,w1,x2,y2,z2,w2,x3,y3,z3,w3}.

sum

Compute the sum of all elements of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const floatInVec sum(
                     Vector4 vec
              );
        }
}
```

Arguments

vec	4-D vector

Return Values

Sum of all elements of *vec*

Description

Compute the sum of all elements of a 4-D vector.

3-D Point Functions

absPerElem

Compute the absolute value of a 3-D point per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Point3 absPerElem(
                     Point3 pnt
              );
        }
}
```

Arguments

pnt	3-D point

Return Values

3-D point in which each element is the absolute value of the corresponding element of pnt

Description

Compute the absolute value of each element of a 3-D point.

copySignPerElem

Copy sign from one 3-D point to another, per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Point3 copySignPerElem(
                     Point3 pnt0,
                     Point3 pnt1
              );
        }
}
```

Arguments

pnt0	3-D point
pnt1	3-D point

Return Values

3-D point in which each element has the magnitude of the corresponding element of *pnt0* and the sign of the corresponding element of *pnt1*

Description

For each element, create a value composed of the magnitude of *pnt0* and the sign of *pnt1*.

dist

Compute the distance between two 3-D points.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const floatInVec dist(
                     Point3 pnt0,
                     Point3 pnt1
              );
        }
}
```

Arguments

pnt0	3-D point
pnt1	3-D point

Return Values

Distance between two 3-D points

Description

Compute the distance between two 3-D points.

distFromOrigin

Compute the distance of a 3-D point from the coordinate-system origin.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const floatInVec distFromOrigin(
                     Point3 pnt
              );
        }
}
```

Arguments

pnt	3-D point

Return Values

Distance of a 3-D point from the origin

Description

Compute the distance of a 3-D point from the coordinate-system origin.

distSqr

Compute the square of the distance between two 3-D points.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const floatInVec distSqr(
                     Point3 pnt0,
                     Point3 pnt1
              );
        }
}
```

Arguments

pnt0	3-D point
pnt1	3-D point

Return Values

Square of the distance between two 3-D points

Description

Compute the square of the distance between two 3-D points.

distSqrFromOrigin

Compute the square of the distance of a 3-D point from the coordinate-system origin.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const floatInVec distSqrFromOrigin(
                     Point3 pnt
              );
        }
}
```

Arguments

pnt	3-D point

Return Values

Square of the distance of a 3-D point from the origin

Description

Compute the square of the distance of a 3-D point from the coordinate-system origin.

divPerElem

Divide two 3-D points per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Point3 divPerElem(
                    Point3 pnt0,
                    Point3 pnt1
              );
        }
}
```

Arguments

pnt0	3-D point
pnt1	3-D point

Return Values

3-D point in which each element is the quotient of the corresponding elements of the specified 3-D points

Description

Divide two 3-D points element by element.

Notes

Floating-point behavior matches standard library function divf4.

lerp

Linear interpolation between two 3-D points.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Point3 lerp(
                     float t,
                     Point3 pnt0,
                     Point3 pnt1
              );
        }
}
```

Arguments

t	Interpolation parameter
pnt0	3-D point
pnt1	3-D point

Return Values

Interpolated 3-D point

Description

Linearly interpolate between two 3-D points.

Notes

Does not clamp `t` between 0 and 1.

lerp

Linear interpolation between two 3-D points (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Point3 lerp(
                     floatInVec t,
                     Point3 pnt0,
                     Point3 pnt1
              );
        }
}
```

Arguments

t	Interpolation parameter
pnt0	3-D point
pnt1	3-D point

Return Values

Interpolated 3-D point

Description

Linearly interpolate between two 3-D points.

Notes

Does not clamp *t* between 0 and 1.

loadXYZArray

Load four three-float 3-D points, stored in three quadwords.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline void loadXYZArray(
                     Point3 &pnt0,
                     Point3 &pnt1,
                     Point3 &pnt2,
                     Point3 &pnt3,
                     const vec_float4 *threeQuads
              );
        }
}
```

Arguments

pnt0	An output 3-D point
pnt1	An output 3-D point
pnt2	An output 3-D point
pnt3	An output 3-D point
threeQuads	Array of 3 quadwords containing 12 floats

Return Values

None

Description

Load four three-float 3-D points, stored in three quadwords as {x0,y0,z0,x1,y1,z1,x2,y2,z2,x3,y3,z3}, into four 3-D points.

maxElem

Maximum element of a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const floatInVec maxElem(
                     Point3 pnt
              );
        }
}
```

Arguments

pnt	3-D point

Return Values

Maximum value of all elements of *pnt*

Description

Compute the maximum value of all elements of a 3-D point.

maxPerElem

Maximum of two 3-D points per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Point3 maxPerElem(
                    Point3 pnt0,
                    Point3 pnt1
              );
        }
}
```

Arguments

pnt0	3-D point
pnt1	3-D point

Return Values

3-D point in which each element is the maximum of the corresponding elements of the specified 3-D points

Description

Create a 3-D point in which each element is the maximum of the corresponding elements of the specified 3-D points.

minElem

Minimum element of a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const floatInVec minElem(
                     Point3 pnt
              );
        }
}
```

Arguments

pnt	3-D point

Return Values

Minimum value of all elements of *pnt*

Description

Compute the minimum value of all elements of a 3-D point.

minPerElem

Minimum of two 3-D points per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Point3 minPerElem(
                     Point3 pnt0,
                     Point3 pnt1
              );
        }
}
```

Arguments

pnt0	3-D point
pnt1	3-D point

Return Values

3-D point in which each element is the minimum of the corresponding elements of the specified 3-D points

Description

Create a 3-D point in which each element is the minimum of the corresponding elements of two specified 3-D points.

mulPerElem

Multiply two 3-D points per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Point3 mulPerElem(
                      Point3 pnt0,
                      Point3 pnt1
              );
        }
}
```

Arguments

pnt0	3-D point
pnt1	3-D point

Return Values

3-D point in which each element is the product of the corresponding elements of the specified 3-D points

Description

Multiply two 3-D points element by element.

print

Print a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline void print(
                     Point3 pnt
              );
        }
}
```

Arguments

pnt	3-D point

Return Values

None

Description

Print a 3-D point. Prints the 3-D point transposed, that is, as a row instead of a column.

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

print

Print a 3-D point and an associated string identifier.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline void print(
                     Point3 pnt,
                     const char *name
              );
        }
}
```

Arguments

pnt	3-D point
name	String printed with the 3-D point

Return Values

None

Description

Print a 3-D point and an associated string identifier. Prints the 3-D point transposed, that is, as a row instead of a column.

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

projection

Scalar projection of a 3-D point on a unit-length 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const floatInVec projection(
                     Point3 pnt,
                     Vector3 unitVec
              );
        }
}
```

Arguments

pnt	3-D point
unitVec	3-D vector, expected to be unit-length

Return Values

Scalar projection of the 3-D point on the unit-length 3-D vector

Description

Scalar projection of a 3-D point on a unit-length 3-D vector (dot product).

recipPerElem

Compute the reciprocal of a 3-D point per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Point3 recipPerElem(
                     Point3 pnt
              );
        }
}
```

Arguments

pnt	3-D point

Return Values

3-D point in which each element is the reciprocal of the corresponding element of the specified 3-D point

Description

Create a 3-D point in which each element is the reciprocal of the corresponding element of the specified 3-D point.

Notes

Floating-point behavior matches standard library function recipf4.

rsqrtPerElem

Compute the reciprocal square root of a 3-D point per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Point3 rsqrtPerElem(
                     Point3 pnt
              );
        }
}
```

Arguments

pnt	3-D point

Return Values

3-D point in which each element is the reciprocal square root of the corresponding element of the specified 3-D point

Description

Create a 3-D point in which each element is the reciprocal square root of the corresponding element of the specified 3-D point.

Notes

Floating-point behavior matches standard library function rsqrtf4.

scale

Apply uniform scale to a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Point3 scale(
                      Point3 pnt,
                      float scaleVal
              );
        }
}
```

Arguments

pnt	3-D point
scaleVal	Scalar value

Return Values

3-D point in which every element is multiplied by the scalar value

Description

Apply uniform scale to a 3-D point.

scale

Apply uniform scale to a 3-D point (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Point3 scale(
                    Point3 pnt,
                    floatInVec scaleVal
              );
        }
}
```

Arguments

pnt	3-D point
scaleVal	Scalar value (stored in vector data type)

Return Values

3-D point in which every element is multiplied by the scalar value

Description

Apply uniform scale to a 3-D point.

scale

Apply non-uniform scale to a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Point3 scale(
                     Point3 pnt,
                     Vector3 scaleVec
              );
        }
}
```

Arguments

pnt	3-D point
scaleVec	3-D vector

Return Values

3-D point in which each element is the product of the corresponding elements of the specified 3-D point and 3-D vector

Description

Apply non-uniform scale to a 3-D point.

select

Conditionally select between two 3-D points.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
             inline const Point3 select(
                   Point3 pnt0,
                   Point3 pnt1,
                   bool select1
             );
        }
}
```

Arguments

pnt0	3-D point
pnt1	3-D point
select1	False selects the pnt0 argument, true selects the pnt1 argument

Return Values

Equal to `pnt0` if `select1` is false, or to `pnt1` if `select1` is true

Description

Conditionally select one of the 3-D point arguments.

Notes

This function uses a conditional select instruction to avoid a branch. However, the transfer of `select1` to a VMX register may use more processing time than a branch. Use the boolInVec version for better performance.

select

Conditionally select between two 3-D points (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Point3 select(
                     Point3 pnt0,
                     Point3 pnt1,
                     boolInVec select1
              );
        }
}
```

Arguments

pnt0	3-D point
pnt1	3-D point
select1	False selects the pnt0 argument, true selects the pnt1 argument

Return Values

Equal to `pnt0` if `select1` is false, or to `pnt1` if `select1` is true

Description

Conditionally select one of the 3-D point arguments.

Notes

This function uses a conditional select instruction to avoid a branch.

sqrtPerElem

Compute the square root of a 3-D point per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Point3 sqrtPerElem(
                     Point3 pnt
              );
        }
}
```

Arguments

pnt	3-D point

Return Values

3-D point in which each element is the square root of the corresponding element of the specified 3-D point

Description

Create a 3-D point in which each element is the square root of the corresponding element of the specified 3-D point.

Notes

Floating-point behavior matches standard library function sqrtf4.

storeHalfFloats

Store eight 3-D points as half-floats.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
             inline void storeHalfFloats(
                    Point3 pnt0,
                    Point3 pnt1,
                    Point3 pnt2,
                    Point3 pnt3,
                    Point3 pnt4,
                    Point3 pnt5,
                    Point3 pnt6,
                    Point3 pnt7,
                    vec_ushort8 *threeQuads
             );
        }
}
```

Arguments

pnt0	3-D point
pnt1	3-D point
pnt2	3-D point
pnt3	3-D point
pnt4	3-D point
pnt5	3-D point
pnt6	3-D point
pnt7	3-D point
threeQuads	An output array of 3 quadwords containing 24 half-floats

Return Values

None

Description

Store eight 3-D points in three quadwords of half-float values. The output is {x0,y0,z0,x1,y1,z1,x2,y2,z2,x3,y3,z3,x4,y4,z4,x5,y5,z5,x6,y6,z6,x7,y7,z7}.

storeXYZ

Store x, y, and z elements of a 3-D point in the first three words of a quadword. The value of the fourth word (the word with the highest address) remains unchanged.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline void storeXYZ(
                     Point3 pnt,
                     vec_float4 *quad
              );
        }
}
```

Arguments

pnt	3-D point
quad	Pointer to a quadword in which x, y, and z will be stored

Return Values

None

Description

Store x, y, and z elements of a 3-D point in the first three words of a quadword. The value of the fourth word (the word with the highest address) remains unchanged.

storeXYZArray

Store four 3-D points in three quadwords.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline void storeXYZArray(
                     Point3 pnt0,
                     Point3 pnt1,
                     Point3 pnt2,
                     Point3 pnt3,
                     vec_float4 *threeQuads
              );
        }
}
```

Arguments

pnt0	3-D point
pnt1	3-D point
pnt2	3-D point
pnt3	3-D point
threeQuads	An output array of 3 quadwords containing 12 floats

Return Values

None

Description

Store four 3-D points in three quadwords as {x0,y0,z0,x1,y1,z1,x2,y2,z2,x3,y3,z3}.

sum

Compute the sum of all elements of a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const floatInVec sum(
                     Point3 pnt
              );
        }
}
```

Arguments

pnt	3-D point

Return Values

Sum of all elements of *pnt*

Description

Compute the sum of all elements of a 3-D point.

Quaternion Functions

conj

Compute the conjugate of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Quat conj(
                     Quat quat
              );
        }
}
```

Arguments

quat	Quaternion

Return Values

Conjugate of the specified quaternion

Description

Compute the conjugate of a quaternion.

dot

Compute the dot product of two quaternions.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const floatInVec dot(
                     Quat quat0,
                     Quat quat1
              );
        }
}
```

Arguments

quat0	Quaternion
quat1	Quaternion

Return Values

Dot product of the specified quaternions

Description

Compute the dot product of two quaternions.

length

Compute the length of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const floatInVec length(
                     Quat quat
              );
        }
}
```

Arguments

quat	Quaternion

Return Values

Length of the specified quaternion

Description

Compute the length of a quaternion.

lerp

Linear interpolation between two quaternions.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Quat lerp(
                     float t,
                     Quat quat0,
                     Quat quat1
              );
        }
}
```

Arguments

t	Interpolation parameter
quat0	Quaternion
quat1	Quaternion

Return Values

Interpolated quaternion

Description

Linearly interpolate between two quaternions.

Notes

Does not clamp `t` between 0 and 1.

lerp

Linear interpolation between two quaternions (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Quat lerp(
                     floatInVec t,
                     Quat quat0,
                     Quat quat1
              );
        }
}
```

Arguments

t	Interpolation parameter
quat0	Quaternion
quat1	Quaternion

Return Values

Interpolated quaternion

Description

Linearly interpolate between two quaternions.

Notes

Does not clamp `t` between 0 and 1.

norm

Compute the norm of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const floatInVec norm(
                           Quat quat
              );
        }
}
```

Arguments

quat	Quaternion

Return Values

The norm of the specified quaternion

Description

Compute the norm, equal to the square of the length, of a quaternion.

normalize

Normalize a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Quat normalize(
                    Quat quat
              );
        }
}
```

Arguments

quat	Quaternion

Return Values

The specified quaternion scaled to unit length

Description

Compute a normalized quaternion.

Notes

The result is unpredictable when all elements of quat are at or near zero.

operator *

Multiply a quaternion by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Quat operator *(
                     float scalar,
                     Quat quat
              );
        }
}
```

Arguments

scalar	Scalar value
quat	Quaternion

Return Values

Scalar product of *quat* and *scalar*

Description

Multiply a quaternion by a scalar.

operator *

Multiply a quaternion by a scalar (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Quat operator *(
                     floatInVec scalar,
                     Quat quat
              );
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)
quat	Quaternion

Return Values

Scalar product of *quat* and *scalar*

Description

Multiply a quaternion by a scalar.

print

Print a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline void print(
                     Quat quat
              );
        }
}
```

Arguments

quat	Quaternion

Return Values

None

Description

Print a quaternion.

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

print

Print a quaternion and an associated string identifier.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline void print(
                     Quat quat,
                     const char *name
              );
        }
}
```

Arguments

quat	Quaternion
name	String printed with the quaternion

Return Values

None

Description

Print a quaternion and an associated string identifier.

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

rotate

Use a unit-length quaternion to rotate a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Vector3 rotate(
                     Quat unitQuat,
                     Vector3 vec
              );
        }
}
```

Arguments

unitQuat	Quaternion, expected to be unit-length
vec	3-D vector

Return Values

The rotated 3-D vector, equivalent to unitQuat*Quat(vec,0)*conj(unitQuat)

Description

Rotate a 3-D vector by applying a unit-length quaternion.

select

Conditionally select between two quaternions.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
             inline const Quat select(
                   Quat quat0,
                   Quat quat1,
                   bool select1
             );
        }
}
```

Arguments

quat0	Quaternion
quat1	Quaternion
select1	False selects the quat0 argument, true selects the quat1 argument

Return Values

Equal to *quat0* if *select1* is false, or to *quat1* if *select1* is true

Description

Conditionally select one of the quaternion arguments.

Notes

This function uses a conditional select instruction to avoid a branch. However, the transfer of *select1* to a VMX register may use more processing time than a branch. Use the boolInVec version for better performance.

select

Conditionally select between two quaternions (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Quat select(
                    Quat quat0,
                    Quat quat1,
                    boolInVec select1
              );
        }
}
```

Arguments

quat0	Quaternion
quat1	Quaternion
select1	False selects the quat0 argument, true selects the quat1 argument

Return Values

Equal to *quat0* if *select1* is false, or to *quat1* if *select1* is true

Description

Conditionally select one of the quaternion arguments.

Notes

This function uses a conditional select instruction to avoid a branch.

slerp

Spherical linear interpolation between two quaternions.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Quat slerp(
                     float t,
                     Quat unitQuat0,
                     Quat unitQuat1
              );
        }
}
```

Arguments

t	Interpolation parameter
unitQuat0	Quaternion, expected to be unit-length
unitQuat1	Quaternion, expected to be unit-length

Return Values

Interpolated quaternion

Description

Perform spherical linear interpolation between two quaternions.

Notes

Interpolates along the shortest path between orientations. Does not clamp *t* between 0 and 1.

slerp

Spherical linear interpolation between two quaternions (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Quat slerp(
                     floatInVec t,
                     Quat unitQuat0,
                     Quat unitQuat1
              );
        }
}
```

Arguments

t	Interpolation parameter
unitQuat0	Quaternion, expected to be unit-length
unitQuat1	Quaternion, expected to be unit-length

Return Values

Interpolated quaternion

Description

Perform spherical linear interpolation between two quaternions.

Notes

Interpolates along the shortest path between orientations. Does not clamp *t* between 0 and 1.

squad

Spherical quadrangle interpolation.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
             inline const Quat squad(
                    float t,
                    Quat unitQuat0,
                    Quat unitQuat1,
                    Quat unitQuat2,
                    Quat unitQuat3
             );
        }
}
```

Arguments

t	Interpolation parameter
unitQuat0	Quaternion, expected to be unit-length
unitQuat1	Quaternion, expected to be unit-length
unitQuat2	Quaternion, expected to be unit-length
unitQuat3	Quaternion, expected to be unit-length

Return Values

Interpolated quaternion

Description

Perform spherical quadrangle interpolation between four quaternions.

squad

Spherical quadrangle interpolation (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
             inline const Quat squad(
                    floatInVec t,
                    Quat unitQuat0,
                    Quat unitQuat1,
                    Quat unitQuat2,
                    Quat unitQuat3
             );
        }
}
```

Arguments

t	Interpolation parameter
unitQuat0	Quaternion, expected to be unit-length
unitQuat1	Quaternion, expected to be unit-length
unitQuat2	Quaternion, expected to be unit-length
unitQuat3	Quaternion, expected to be unit-length

Return Values

Interpolated quaternion

Description

Perform spherical quadrangle interpolation between four quaternions.

3x3 Matrix Functions

absPerElem

Compute the absolute value of a 3x3 matrix per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Matrix3 absPerElem(
                     const Matrix3 &mat
              );
        }
}
```

Arguments

mat	3x3 matrix

Return Values

3x3 matrix in which each element is the absolute value of the corresponding element of the specified 3x3 matrix

Description

Compute the absolute value of each element of a 3x3 matrix.

appendScale

Append (post-multiply) a scale transformation to a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Matrix3 appendScale(
                     const Matrix3 &mat,
                     Vector3 scaleVec
              );
        }
}
```

Arguments

mat	3x3 matrix
scaleVec	3-D vector

Return Values

The product of *mat* and a scale transformation created from *scaleVec*

Description

Post-multiply a 3x3 matrix by a scale transformation whose diagonal scale factors are contained in the 3-D vector.

Notes

Faster than creating and multiplying a scale transformation matrix.

determinant

Determinant of a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const floatInVec determinant(
                     const Matrix3 &mat
              );
        }
}
```

Arguments

mat	3x3 matrix

Return Values

The determinant of *mat*

Description

Compute the determinant of a 3x3 matrix.

inverse

Compute the inverse of a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Matrix3 inverse(
                     const Matrix3 &mat
              );
        }
}
```

Arguments

mat	3x3 matrix

Return Values

Inverse of *mat*

Description

Compute the inverse of a 3x3 matrix.

Notes

Result is unpredictable when the determinant of *mat* is equal to or near 0.

mulPerElem

Multiply two 3x3 matrices per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Matrix3 mulPerElem(
                     const Matrix3 &mat0,
                     const Matrix3 &mat1
              );
        }
}
```

Arguments

mat0	3x3 matrix
mat1	3x3 matrix

Return Values

3x3 matrix in which each element is the product of the corresponding elements of the specified 3x3 matrices

Description

Multiply two 3x3 matrices element by element.

operator *

Multiply a 3x3 matrix by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Matrix3 operator *(
                     float scalar,
                     const Matrix3 &mat
              );
        }
}
```

Arguments

scalar	Scalar value
mat	3x3 matrix

Return Values

Scalar product of *mat* and *scalar*

Description

Multiply a 3x3 matrix by a scalar.

operator *

Multiply a 3x3 matrix by a scalar (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Matrix3 operator *(
                     floatInVec scalar,
                     const Matrix3 &mat
              );
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)
mat	3x3 matrix

Return Values

Scalar product of *mat* and *scalar*

Description

Multiply a 3x3 matrix by a scalar.

prependScale

Prepend (pre-multiply) a scale transformation to a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Matrix3 prependScale(
                     Vector3 scaleVec,
                     const Matrix3 &mat
              );
        }
}
```

Arguments

scaleVec	3-D vector
mat	3x3 matrix

Return Values

The product of a scale transformation created from *scaleVec* and *mat*

Description

Pre-multiply a 3x3 matrix by a scale transformation whose diagonal scale factors are contained in the 3-D vector.

Notes

Faster than creating and multiplying a scale transformation matrix.

print

Print a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline void print(
                     const Matrix3 &mat
              );
        }
}
```

Arguments

mat	3x3 matrix

Return Values

None

Description

Print a 3x3 matrix. Unlike the printing of vectors, the 3x3 matrix is printed with the correct orientation (columns appear vertically).

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

print

Print a 3x3 matrix and an associated string identifier.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline void print(
                     const Matrix3 &mat,
                     const char *name
              );
        }
}
```

Arguments

mat	3x3 matrix
name	String printed with the 3x3 matrix

Return Values

None

Description

Print a 3x3 matrix and an associated string identifier. Unlike the printing of vectors, the 3x3 matrix is printed with the correct orientation (columns appear vertically).

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

select

Conditionally select between two 3x3 matrices.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
             inline const Matrix3 select(
                    const Matrix3 &mat0,
                    const Matrix3 &mat1,
                    bool select1
             );
        }
}
```

Arguments

mat0	3x3 matrix
mat1	3x3 matrix
select1	False selects the mat0 argument, true selects the mat1 argument

Return Values

Equal to *mat0* if *select1* is false, or to *mat1* if *select1* is true

Description

Conditionally select one of the 3x3 matrix arguments.

Notes

This function uses a conditional select instruction to avoid a branch. However, the transfer of *select1* to a VMX register may use more processing time than a branch. Use the boolInVec version for better performance.

select

Conditionally select between two 3x3 matrices (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Matrix3 select(
                     const Matrix3 &mat0,
                     const Matrix3 &mat1,
                     boolInVec select1
              );
        }
}
```

Arguments

mat0	3x3 matrix
mat1	3x3 matrix
select1	False selects the mat0 argument, true selects the mat1 argument

Return Values

Equal to *mat0* if *select1* is false, or to *mat1* if *select1* is true

Description

Conditionally select one of the 3x3 matrix arguments.

Notes

This function uses a conditional select instruction to avoid a branch.

transpose

Transpose of a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Matrix3 transpose(
                     const Matrix3 &mat
              );
        }
}
```

Arguments

mat	3x3 matrix

Return Values

mat transposed

Description

Compute the transpose of a 3x3 matrix.

4x4 Matrix Functions

absPerElem

Compute the absolute value of a 4x4 matrix per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Matrix4 absPerElem(
                     const Matrix4 &mat
              );
        }
}
```

Arguments

mat	4x4 matrix

Return Values

4x4 matrix in which each element is the absolute value of the corresponding element of the specified 4x4 matrix

Description

Compute the absolute value of each element of a 4x4 matrix.

affineInverse

Compute the inverse of a 4x4 matrix, which is expected to be an affine matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Matrix4 affineInverse(
                     const Matrix4 &mat
              );
        }
}
```

Arguments

mat	4x4 matrix

Return Values

Inverse of the specified 4x4 matrix

Description

Naming the upper-left 3x3 submatrix of the specified 4x4 matrix as M, and its translation component as v, compute a matrix whose upper-left 3x3 submatrix is inverse(M), whose translation vector is -inverse(M)*v, and whose bottom row is (0,0,0,1).

Notes

This can be used to achieve better performance than a general inverse when the specified 4x4 matrix meets the given restrictions. The result is unpredictable when the determinant of `mat` is equal to or near 0.

appendScale

Append (post-multiply) a scale transformation to a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Matrix4 appendScale(
                     const Matrix4 &mat,
                     Vector3 scaleVec
              );
        }
}
```

Arguments

mat	4x4 matrix
scaleVec	3-D vector

Return Values

The product of `mat` and a scale transformation created from `scaleVec`

Description

Post-multiply a 4x4 matrix by a scale transformation whose diagonal scale factors are contained in the 3-D vector.

Notes

Faster than creating and multiplying a scale transformation matrix.

determinant

Determinant of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const floatInVec determinant(
                     const Matrix4 &mat
              );
        }
}
```

Arguments

mat	4x4 matrix

Return Values

The determinant of `mat`

Description

Compute the determinant of a 4x4 matrix.

inverse

Compute the inverse of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Matrix4 inverse(
                     const Matrix4 &mat
              );
        }
}
```

Arguments

mat	4x4 matrix

Return Values

Inverse of *mat*

Description

Compute the inverse of a 4x4 matrix.

Notes

Result is unpredictable when the determinant of *mat* is equal to or near 0.

mulPerElem

Multiply two 4x4 matrices per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Matrix4 mulPerElem(
                     const Matrix4 &mat0,
                     const Matrix4 &mat1
              );
        }
}
```

Arguments

mat0	4x4 matrix
mat1	4x4 matrix

Return Values

4x4 matrix in which each element is the product of the corresponding elements of the specified 4x4 matrices

Description

Multiply two 4x4 matrices element by element.

operator *

Multiply a 4x4 matrix by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Matrix4 operator *(
                     float scalar,
                     const Matrix4 &mat
              );
        }
}
```

Arguments

scalar	Scalar value
mat	4x4 matrix

Return Values

Scalar product of *mat* and *scalar*

Description

Multiply a 4x4 matrix by a scalar.

operator *

Multiply a 4x4 matrix by a scalar (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Matrix4 operator *(
                     floatInVec scalar,
                     const Matrix4 &mat
              );
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)
mat	4x4 matrix

Return Values

Scalar product of *mat* and *scalar*

Description

Multiply a 4x4 matrix by a scalar.

orthoInverse

Compute the inverse of a 4x4 matrix, which is expected to be an affine matrix with an orthogonal upper-left 3x3 submatrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Matrix4 orthoInverse(
                     const Matrix4 &mat
              );
        }
}
```

Arguments

mat	4x4 matrix

Return Values

Inverse of the specified 4x4 matrix

Description

Naming the upper-left 3x3 submatrix of the specified 4x4 matrix as M, and its translation component as v, compute a matrix whose upper-left 3x3 submatrix is transpose(M), whose translation vector is -transpose(M)*v, and whose bottom row is (0,0,0,1).

Notes

This can be used to achieve better performance than a general inverse when the specified 4x4 matrix meets the given restrictions.

prependScale

Prepend (pre-multiply) a scale transformation to a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Matrix4 prependScale(
                     Vector3 scaleVec,
                     const Matrix4 &mat
              );
        }
}
```

Arguments

scaleVec	3-D vector
mat	4x4 matrix

Return Values

The product of a scale transformation created from *scaleVec* and *mat*

Description

Pre-multiply a 4x4 matrix by a scale transformation whose diagonal scale factors are contained in the 3-D vector.

Notes

Faster than creating and multiplying a scale transformation matrix.

print

Print a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline void print(
                     const Matrix4 &mat
              );
        }
}
```

Arguments

mat	4x4 matrix

Return Values

None

Description

Print a 4x4 matrix. Unlike the printing of vectors, the 4x4 matrix is printed with the correct orientation (columns appear vertically).

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

print

Print a 4x4 matrix and an associated string identifier.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline void print(
                     const Matrix4 &mat,
                     const char *name
              );
        }
}
```

Arguments

mat	4x4 matrix
name	String printed with the 4x4 matrix

Return Values

None

Description

Print a 4x4 matrix and an associated string identifier. Unlike the printing of vectors, the 4x4 matrix is printed with the correct orientation (columns appear vertically).

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

select

Conditionally select between two 4x4 matrices.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
             inline const Matrix4 select(
                    const Matrix4 &mat0,
                    const Matrix4 &mat1,
                    bool select1
             );
        }
}
```

Arguments

mat0	4x4 matrix
mat1	4x4 matrix
select1	False selects the mat0 argument, true selects the mat1 argument

Return Values

Equal to *mat0* if *select1* is false, or to *mat1* if *select1* is true

Description

Conditionally select one of the 4x4 matrix arguments.

Notes

This function uses a conditional select instruction to avoid a branch. However, the transfer of *select1* to a VMX register may use more processing time than a branch. Use the boolInVec version for better performance.

select

Conditionally select between two 4x4 matrices (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
             inline const Matrix4 select(
                    const Matrix4 &mat0,
                    const Matrix4 &mat1,
                    boolInVec select1
             );
        }
}
```

Arguments

mat0	4x4 matrix
mat1	4x4 matrix
select1	False selects the mat0 argument, true selects the mat1 argument

Return Values

Equal to *mat0* if *select1* is false, or to *mat1* if *select1* is true

Description

Conditionally select one of the 4x4 matrix arguments.

Notes

This function uses a conditional select instruction to avoid a branch.

transpose

Transpose of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Matrix4 transpose(
                     const Matrix4 &mat
              );
        }
}
```

Arguments

mat	4x4 matrix

Return Values

mat transposed

Description

Compute the transpose of a 4x4 matrix.

3x4 Transformation Matrix Functions

absPerElem

Compute the absolute value of a 3x4 transformation matrix per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Transform3 absPerElem(
                     const Transform3 &tfrm
              );
        }
}
```

Arguments

tfrm	3x4 transformation matrix

Return Values

3x4 transformation matrix in which each element is the absolute value of the corresponding element of the specified 3x4 transformation matrix

Description

Compute the absolute value of each element of a 3x4 transformation matrix.

appendScale

Append (post-multiply) a scale transformation to a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Transform3 appendScale(
                     const Transform3 &tfrm,
                     Vector3 scaleVec
              );
        }
}
```

Arguments

tfrm	3x4 transformation matrix
scaleVec	3-D vector

Return Values

The product of *tfrm* and a scale transformation created from *scaleVec*

Description

Post-multiply a 3x4 transformation matrix by a scale transformation whose diagonal scale factors are contained in the 3-D vector.

Notes

Faster than creating and multiplying a scale transformation matrix.

inverse

Inverse of a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Transform3 inverse(
                     const Transform3 &tfrm
              );
        }
}
```

Arguments

tfrm	3x4 transformation matrix

Return Values

Inverse of *tfrm*

Description

Compute the inverse of a 3x4 transformation matrix.

Notes

Result is unpredictable when the determinant of the left 3x3 submatrix is equal to or near 0.

mulPerElem

Multiply two 3x4 transformation matrices per element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Transform3 mulPerElem(
                     const Transform3 &tfrm0,
                     const Transform3 &tfrm1
              );
        }
}
```

Arguments

tfrm0	3x4 transformation matrix
tfrm1	3x4 transformation matrix

Return Values

3x4 transformation matrix in which each element is the product of the corresponding elements of the specified 3x4 transformation matrices

Description

Multiply two 3x4 transformation matrices element by element.

orthoInverse

Compute the inverse of a 3x4 transformation matrix, expected to have an orthogonal upper-left 3x3 submatrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Transform3 orthoInverse(
                     const Transform3 &tfrm
              );
        }
}
```

Arguments

tfrm	3x4 transformation matrix

Return Values

Inverse of the specified 3x4 transformation matrix

Description

Naming the upper-left 3x3 submatrix of the specified 3x4 transformation matrix as M, and its translation component as v, compute a matrix whose upper-left 3x3 submatrix is transpose(M), and whose translation vector is -transpose(M)*v.

Notes

This can be used to achieve better performance than a general inverse when the specified 3x4 transformation matrix meets the given restrictions.

prependScale

Prepend (pre-multiply) a scale transformation to a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Transform3 prependScale(
                     Vector3 scaleVec,
                     const Transform3 &tfrm
              );
        }
}
```

Arguments

scaleVec	3-D vector
tfrm	3x4 transformation matrix

Return Values

The product of a scale transformation created from *scaleVec* and *tfrm*

Description

Pre-multiply a 3x4 transformation matrix by a scale transformation whose diagonal scale factors are contained in the 3-D vector.

Notes

Faster than creating and multiplying a scale transformation matrix.

print

Print a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline void print(
                     const Transform3 &tfrm
              );
        }
}
```

Arguments

tfrm	3x4 transformation matrix

Return Values

None

Description

Print a 3x4 transformation matrix. Unlike the printing of vectors, the 3x4 transformation matrix is printed with the correct orientation (columns appear vertically).

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

print

Print a 3x4 transformation matrix and an associated string identifier.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline void print(
                     const Transform3 &tfrm,
                     const char *name
              );
        }
}
```

Arguments

tfrm	3x4 transformation matrix
name	String printed with the 3x4 transformation matrix

Return Values

None

Description

Print a 3x4 transformation matrix and an associated string identifier. Unlike the printing of vectors, the 3x4 transformation matrix is printed with the correct orientation (columns appear vertically).

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

select

Conditionally select between two 3x4 transformation matrices.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              inline const Transform3 select(
                     const Transform3 &tfrm0,
                     const Transform3 &tfrm1,
                     bool select1
              );
        }
}
```

Arguments

tfrm0	3x4 transformation matrix
tfrm1	3x4 transformation matrix
select1	False selects the tfrm0 argument, true selects the tfrm1 argument

Return Values

Equal to *tfrm0* if *select1* is false, or to *tfrm1* if *select1* is true

Description

Conditionally select one of the 3x4 transformation matrix arguments.

Notes

This function uses a conditional select instruction to avoid a branch. However, the transfer of *select1* to a VMX register may use more processing time than a branch. Use the boolInVec version for better performance.

select

Conditionally select between two 3x4 transformation matrices (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
                inline const Transform3 select(
                        const Transform3 &tfrm0,
                        const Transform3 &tfrm1,
                        boolInVec select1
                );
        }
}
```

Arguments

tfrm0	3x4 transformation matrix
tfrm1	3x4 transformation matrix
select1	False selects the tfrm0 argument, true selects the tfrm1 argument

Return Values

Equal to `tfrm0` if `select1` is false, or to `tfrm1` if `select1` is true

Description

Conditionally select one of the 3x4 transformation matrix arguments.

Notes

This function uses a conditional select instruction to avoid a branch.

Vectormath::Aos::Matrix3

Summary

Vectormath::Aos::Matrix3

A 3x3 matrix in array-of-structures format.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
class Matrix3;
```

Description

A class representing a 3x3 matrix stored in array-of-structures (AoS) format.

Methods Summary

Methods	Description
getCol	Get the column of a 3x3 matrix referred to by the specified index.
getCol0	Get column 0 of a 3x3 matrix.
getCol1	Get column 1 of a 3x3 matrix.
getCol2	Get column 2 of a 3x3 matrix.
getElem	Get the element of a 3x3 matrix referred to by column and row indices.
getRow	Get the row of a 3x3 matrix referred to by the specified index.
identity	Construct an identity 3x3 matrix.
Matrix3	Default constructor; does no initialization.
Matrix3	Copy a 3x3 matrix.
Matrix3	Construct a 3x3 matrix containing the specified columns.
Matrix3	Construct a 3x3 rotation matrix from a unit-length quaternion.
Matrix3	Set all elements of a 3x3 matrix to the same scalar value.
Matrix3	Set all elements of a 3x3 matrix to the same scalar value (scalar data contained in vector data type).
operator *	Multiply a 3x3 matrix by a scalar.
operator *	Multiply a 3x3 matrix by a scalar (scalar data contained in vector data type).
operator *	Multiply a 3x3 matrix by a 3-D vector.
operator *	Multiply two 3x3 matrices.
operator *=	Perform compound assignment and multiplication by a scalar.
operator *=	Perform compound assignment and multiplication by a scalar (scalar data contained in vector data type).
operator *=	Perform compound assignment and multiplication by a 3x3 matrix.
operator+	Add two 3x3 matrices.
operator+=	Perform compound assignment and addition with a 3x3 matrix.
operator-	Subtract a 3x3 matrix from another 3x3 matrix.
operator-	Negate all elements of a 3x3 matrix.

Methods	Description
operator-=	Perform compound assignment and subtraction by a 3x3 matrix.
operator=	Assign one 3x3 matrix to another.
operator[]	Subscripting operator to set or get a column.
operator[]	Subscripting operator to get a column.
rotation	Construct a 3x3 matrix to rotate around a unit-length 3-D vector.
rotation	Construct a 3x3 matrix to rotate around a unit-length 3-D vector (scalar data contained in vector data type).
rotation	Construct a rotation matrix from a unit-length quaternion.
rotationX	Construct a 3x3 matrix to rotate around the x axis.
rotationX	Construct a 3x3 matrix to rotate around the x axis (scalar data contained in vector data type).
rotationY	Construct a 3x3 matrix to rotate around the y axis.
rotationY	Construct a 3x3 matrix to rotate around the y axis (scalar data contained in vector data type).
rotationZ	Construct a 3x3 matrix to rotate around the z axis.
rotationZ	Construct a 3x3 matrix to rotate around the z axis (scalar data contained in vector data type).
rotationZYX	Construct a 3x3 matrix to rotate around the x, y, and z axes.
scale	Construct a 3x3 matrix to perform scaling.
setCol	Set the column of a 3x3 matrix referred to by the specified index.
setCol0	Set column 0 of a 3x3 matrix.
setCol1	Set column 1 of a 3x3 matrix.
setCol2	Set column 2 of a 3x3 matrix.
setElem	Set the element of a 3x3 matrix referred to by column and row indices.
setElem	Set the element of a 3x3 matrix referred to by column and row indices (scalar data contained in vector data type).
setRow	Set the row of a 3x3 matrix referred to by the specified index.

Constructors and Destructors

Matrix3

Default constructor; does no initialization.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline Matrix3();
              }
        }
}
```

Arguments

None

Return Values

None

Description

Default constructor; does no initialization.

Matrix3

Copy a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline Matrix3(
                            const Matrix3 &mat
                     );
              }
        }
}
```

Arguments

mat	3x3 matrix

Return Values

None

Description

Construct a copy of a 3x3 matrix.

Matrix3

Construct a 3x3 matrix containing the specified columns.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline Matrix3(
                            Vector3 col0,
                            Vector3 col1,
                            Vector3 col2
                     );
              }
        }
}
```

Arguments

col0	3-D vector
col1	3-D vector
col2	3-D vector

Return Values

None

Description

Construct a 3x3 matrix containing the specified columns.

Matrix3

Construct a 3x3 rotation matrix from a unit-length quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     explicit inline Matrix3(
                            Quat unitQuat
                     );
              }
        }
}
```

Arguments

unitQuat	Quaternion, expected to be unit-length

Return Values

None

Description

Construct a 3x3 matrix that applies the same rotation as the specified unit-length quaternion.

Matrix3

Set all elements of a 3x3 matrix to the same scalar value.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     explicit inline Matrix3(
                            float scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

None

Description

Construct a 3x3 matrix with all elements set to the scalar value argument.

Matrix3

Set all elements of a 3x3 matrix to the same scalar value (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     explicit inline Matrix3(
                            floatInVec scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)

Return Values

None

Description

Construct a 3x3 matrix with all elements set to the scalar value argument.

Operator Methods

operator *

Multiply a 3x3 matrix by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline const Matrix3 operator *(
                            float scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

Product of the specified 3x3 matrix and scalar

Description

Multiply a 3x3 matrix by a scalar.

operator *

Multiply a 3x3 matrix by a scalar (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline const Matrix3 operator *(
                            floatInVec scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)

Return Values

Product of the specified 3x3 matrix and scalar

Description

Multiply a 3x3 matrix by a scalar.

operator *

Multiply a 3x3 matrix by a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline const Vector3 operator *(
                            Vector3 vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

Product of the specified 3x3 matrix and 3-D vector

Description

Multiply a 3x3 matrix by a 3-D vector.

operator *

Multiply two 3x3 matrices.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline const Matrix3 operator *(
                            const Matrix3 &mat
                     );
              }
        }
}
```

Arguments

mat	3x3 matrix

Return Values

Product of the specified 3x3 matrices

Description

Multiply two 3x3 matrices.

operator *=

Perform compound assignment and multiplication by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline Matrix3 &operator *=(
                            float scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

A reference to the resulting 3x3 matrix

Description

Perform compound assignment and multiplication by a scalar.

operator *=

Perform compound assignment and multiplication by a scalar (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline Matrix3 &operator *=(
                            floatInVec scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)

Return Values

A reference to the resulting 3x3 matrix

Description

Perform compound assignment and multiplication by a scalar.

operator *=

Perform compound assignment and multiplication by a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline Matrix3 &operator *=(
                            const Matrix3 &mat
                     );
              }
        }
}
```

Arguments

mat	3x3 matrix

Return Values

A reference to the resulting 3x3 matrix

Description

Perform compound assignment and multiplication by a 3x3 matrix.

operator+

Add two 3x3 matrices.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline const Matrix3 operator+(
                            const Matrix3 &mat
                     );
              }
        }
}
```

Arguments

mat	3x3 matrix

Return Values

Sum of the specified 3x3 matrices

Description

Add two 3x3 matrices.

operator+=

Perform compound assignment and addition with a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline Matrix3 &operator+=(
                            const Matrix3 &mat
                     );
              }
        }
}
```

Arguments

mat	3x3 matrix

Return Values

A reference to the resulting 3x3 matrix

Description

Perform compound assignment and addition with a 3x3 matrix.

operator-

Subtract a 3x3 matrix from another 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline const Matrix3 operator-(
                            const Matrix3 &mat
                     );
              }
        }
}
```

Arguments

mat	3x3 matrix

Return Values

Difference of the specified 3x3 matrices

Description

Subtract a 3x3 matrix from another 3x3 matrix.

operator-

Negate all elements of a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline const Matrix3 operator-();
              }
        }
}
```

Arguments

None

Return Values

3x3 matrix containing negated elements of the specified 3x3 matrix

Description

Negate all elements of a 3x3 matrix.

operator-=

Perform compound assignment and subtraction by a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline Matrix3 &operator-=(
                            const Matrix3 &mat
                     );
              }
        }
}
```

Arguments

mat	3x3 matrix

Return Values

A reference to the resulting 3x3 matrix

Description

Perform compound assignment and subtraction by a 3x3 matrix.

operator=

Assign one 3x3 matrix to another.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline Matrix3 &operator=(
                            const Matrix3 &mat
                     );
              }
        }
}
```

Arguments

mat	3x3 matrix

Return Values

A reference to the resulting 3x3 matrix

Description

Assign one 3x3 matrix to another.

operator[]

Subscripting operator to set or get a column.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline Vector3 &operator[](
                            int col
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-2

Return Values

A reference to indexed column

Description

Subscripting operator invoked when applied to non-const Matrix3.

operator[]

Subscripting operator to get a column.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline const Vector3 operator[](
                            int col
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-2

Return Values

Indexed column

Description

Subscripting operator invoked when applied to const Matrix3.

Public Static Methods

identity

Construct an identity 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     static inline const Matrix3 identity();
              }
        }
}
```

Arguments

None

Return Values

The constructed 3x3 matrix

Description

Construct an identity 3x3 matrix in which non-diagonal elements are zero and diagonal elements are 1.

rotation

Construct a 3x3 matrix to rotate around a unit-length 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     static inline const Matrix3 rotation(
                            float radians,
                            Vector3 unitVec
                     );
              }
        }
}
```

Arguments

radians	Scalar value
unitVec	3-D vector, expected to be unit-length

Return Values

The constructed 3x3 matrix

Description

Construct a 3x3 matrix to rotate around a unit-length 3-D vector by the specified radians angle.

rotation

Construct a 3x3 matrix to rotate around a unit-length 3-D vector (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     static inline const Matrix3 rotation(
                            floatInVec radians,
                            Vector3 unitVec
                     );
              }
        }
}
```

Arguments

radians	Scalar value (stored in vector data type)
unitVec	3-D vector, expected to be unit-length

Return Values

The constructed 3x3 matrix

Description

Construct a 3x3 matrix to rotate around a unit-length 3-D vector by the specified radians angle.

rotation

Construct a rotation matrix from a unit-length quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     static inline const Matrix3 rotation(
                            Quat unitQuat
                     );
              }
        }
}
```

Arguments

unitQuat	Quaternion, expected to be unit-length

Return Values

The constructed 3x3 matrix

Description

Construct a 3x3 matrix that applies the same rotation as the specified unit-length quaternion.

rotationX

Construct a 3x3 matrix to rotate around the x axis.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     static inline const Matrix3 rotationX(
                            float radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value

Return Values

The constructed 3x3 matrix

Description

Construct a 3x3 matrix to rotate around the x axis by the specified radians angle.

rotationX

Construct a 3x3 matrix to rotate around the x axis (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     static inline const Matrix3 rotationX(
                            floatInVec radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value (stored in vector data type)

Return Values

The constructed 3x3 matrix

Description

Construct a 3x3 matrix to rotate around the x axis by the specified radians angle.

rotationY

Construct a 3x3 matrix to rotate around the y axis.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     static inline const Matrix3 rotationY(
                            float radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value

Return Values

The constructed 3x3 matrix

Description

Construct a 3x3 matrix to rotate around the y axis by the specified radians angle.

rotationY

Construct a 3x3 matrix to rotate around the y axis (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     static inline const Matrix3 rotationY(
                            floatInVec radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value (stored in vector data type)

Return Values

The constructed 3x3 matrix

Description

Construct a 3x3 matrix to rotate around the y axis by the specified radians angle.

rotationZ

Construct a 3x3 matrix to rotate around the z axis.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     static inline const Matrix3 rotationZ(
                            float radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value

Return Values

The constructed 3x3 matrix

Description

Construct a 3x3 matrix to rotate around the z axis by the specified radians angle.

rotationZ

Construct a 3x3 matrix to rotate around the z axis (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     static inline const Matrix3 rotationZ(
                            floatInVec radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value (stored in vector data type)

Return Values

The constructed 3x3 matrix

Description

Construct a 3x3 matrix to rotate around the z axis by the specified radians angle.

rotationZYX

Construct a 3x3 matrix to rotate around the x, y, and z axes.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     static inline const Matrix3 rotationZYX(
                            Vector3 radiansXYZ
                     );
              }
        }
}
```

Arguments

radiansXYZ	3-D vector

Return Values

The constructed 3x3 matrix

Description

Construct a 3x3 matrix to rotate around the x, y, and z axes by the radians angles contained in a 3-D vector. Equivalent to `rotationZ(radiansXYZ.getZ()) * rotationY(radiansXYZ.getY()) * rotationX(radiansXYZ.getX())`.

scale

Construct a 3x3 matrix to perform scaling.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     static inline const Matrix3 scale(
                            Vector3 scaleVec
                     );
              }
        }
}
```

Arguments

scaleVec	3-D vector

Return Values

The constructed 3x3 matrix

Description

Construct a 3x3 matrix to perform scaling, in which the non-diagonal elements are zero and the diagonal elements are set to the elements of *scaleVec*.

Public Instance Methods

getCol

Get the column of a 3x3 matrix referred to by the specified index.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline const Vector3 getCol(
                            int col
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-2

Return Values

The column referred to by the specified index

Description

Get the column of a 3x3 matrix referred to by the specified index.

getCol0

Get column 0 of a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline const Vector3 getCol0();
              }
        }
}
```

Arguments

None

Return Values

Column 0

Description

Get column 0 of a 3x3 matrix.

getCol1

Get column 1 of a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline const Vector3 getCol1();
              }
        }
}
```

Arguments

None

Return Values

Column 1

Description

Get column 1 of a 3x3 matrix.

getCol2

Get column 2 of a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline const Vector3 getCol2();
              }
        }
}
```

Arguments

None

Return Values

Column 2

Description

Get column 2 of a 3x3 matrix.

getElem

Get the element of a 3x3 matrix referred to by column and row indices.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline const floatInVec getElem(
                            int col,
                            int row
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-2
row	Index, expected in the range 0-2

Return Values

Element selected by *col* and *row*

Description

Get the element of a 3x3 matrix referred to by column and row indices.

getRow

Get the row of a 3x3 matrix referred to by the specified index.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline const Vector3 getRow(
                            int row
                     );
              }
        }
}
```

Arguments

row	Index, expected in the range 0-2

Return Values

The row referred to by the specified index

Description

Get the row of a 3x3 matrix referred to by the specified index.

setCol

Set the column of a 3x3 matrix referred to by the specified index.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline Matrix3 &setCol(
                            int col,
                            Vector3 vec
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-2
vec	3-D vector

Return Values

A reference to the resulting 3x3 matrix

Description

Set the column of a 3x3 matrix referred to by the specified index.

setCol0

Set column 0 of a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline Matrix3 &setCol0(
                            Vector3 col0
                     );
              }
        }
}
```

Arguments

col0	3-D vector

Return Values

A reference to the resulting 3x3 matrix

Description

Set column 0 of a 3x3 matrix.

setCol1

Set column 1 of a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline Matrix3 &setCol1(
                            Vector3 col1
                     );
              }
        }
}
```

Arguments

col1	3-D vector

Return Values

A reference to the resulting 3x3 matrix

Description

Set column 1 of a 3x3 matrix.

setCol2

Set column 2 of a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline Matrix3 &setCol2(
                            Vector3 col2
                     );
              }
        }
}
```

Arguments

col2	3-D vector

Return Values

A reference to the resulting 3x3 matrix

Description

Set column 2 of a 3x3 matrix.

setElem

Set the element of a 3x3 matrix referred to by column and row indices.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline Matrix3 &setElem(
                            int col,
                            int row,
                            float val
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-2
row	Index, expected in the range 0-2
val	Scalar value

Return Values

A reference to the resulting 3x3 matrix

Description

Set the element of a 3x3 matrix referred to by column and row indices.

setElem

Set the element of a 3x3 matrix referred to by column and row indices (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline Matrix3 &setElem(
                            int col,
                            int row,
                            floatInVec val
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-2
row	Index, expected in the range 0-2
val	Scalar value (stored in vector data type)

Return Values

A reference to the resulting 3x3 matrix

Description

Set the element of a 3x3 matrix referred to by column and row indices.

setRow

Set the row of a 3x3 matrix referred to by the specified index.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix3 {
                     inline Matrix3 &setRow(
                            int row,
                            Vector3 vec
                     );
              }
        }
}
```

Arguments

row	Index, expected in the range 0-2
vec	3-D vector

Return Values

A reference to the resulting 3x3 matrix

Description

Set the row of a 3x3 matrix referred to by the specified index.

Vectormath::Aos::Matrix4

Summary

Vectormath::Aos::Matrix4

A 4x4 matrix in array-of-structures format.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
class Matrix4;
```

Description

A class representing a 4x4 matrix stored in array-of-structures (AoS) format.

Methods Summary

Methods	Description
frustum	Construct a perspective projection matrix based on frustum.
getCol	Get the column of a 4x4 matrix referred to by the specified index.
getCol0	Get column 0 of a 4x4 matrix.
getCol1	Get column 1 of a 4x4 matrix.
getCol2	Get column 2 of a 4x4 matrix.
getCol3	Get column 3 of a 4x4 matrix.
getElem	Get the element of a 4x4 matrix referred to by column and row indices.
getRow	Get the row of a 4x4 matrix referred to by the specified index.
getTranslation	Get the translation component of a 4x4 matrix.
getUpper3x3	Get the upper-left 3x3 submatrix of a 4x4 matrix.
identity	Construct an identity 4x4 matrix.
lookAt	Construct viewing matrix based on eye position, position looked at, and up direction.
Matrix4	Default constructor; does no initialization.
Matrix4	Copy a 4x4 matrix.
Matrix4	Construct a 4x4 matrix containing the specified columns.
Matrix4	Construct a 4x4 matrix from a 3x4 transformation matrix.
Matrix4	Construct a 4x4 matrix from a 3x3 matrix and a 3-D vector.
Matrix4	Construct a 4x4 matrix from a unit-length quaternion and a 3-D vector.
Matrix4	Set all elements of a 4x4 matrix to the same scalar value.
Matrix4	Set all elements of a 4x4 matrix to the same scalar value (scalar data contained in vector data type).
operator *	Multiply a 4x4 matrix by a scalar.
operator *	Multiply a 4x4 matrix by a scalar (scalar data contained in vector data type).
operator *	Multiply a 4x4 matrix by a 4-D vector.
operator *	Multiply a 4x4 matrix by a 3-D vector.
operator *	Multiply a 4x4 matrix by a 3-D point.
operator *	Multiply two 4x4 matrices.
operator *	Multiply a 4x4 matrix by a 3x4 transformation matrix.

Methods	Description
operator *=	Perform compound assignment and multiplication by a scalar.
operator *=	Perform compound assignment and multiplication by a scalar (scalar data contained in vector data type).
operator *=	Perform compound assignment and multiplication by a 4x4 matrix.
operator *=	Perform compound assignment and multiplication by a 3x4 transformation matrix.
operator+	Add two 4x4 matrices.
operator+=	Perform compound assignment and addition with a 4x4 matrix.
operator-	Subtract a 4x4 matrix from another 4x4 matrix.
operator-	Negate all elements of a 4x4 matrix.
operator-=	Perform compound assignment and subtraction by a 4x4 matrix.
operator=	Assign one 4x4 matrix to another.
operator[]	Subscripting operator to set or get a column.
operator[]	Subscripting operator to get a column.
orthographic	Construct an orthographic projection matrix.
perspective	Construct a perspective projection matrix.
rotation	Construct a 4x4 matrix to rotate around a unit-length 3-D vector.
rotation	Construct a 4x4 matrix to rotate around a unit-length 3-D vector (scalar data contained in vector data type).
rotation	Construct a rotation matrix from a unit-length quaternion.
rotationX	Construct a 4x4 matrix to rotate around the x axis.
rotationX	Construct a 4x4 matrix to rotate around the x axis (scalar data contained in vector data type).
rotationY	Construct a 4x4 matrix to rotate around the y axis.
rotationY	Construct a 4x4 matrix to rotate around the y axis (scalar data contained in vector data type).
rotationZ	Construct a 4x4 matrix to rotate around the z axis.
rotationZ	Construct a 4x4 matrix to rotate around the z axis (scalar data contained in vector data type).
rotationZYX	Construct a 4x4 matrix to rotate around the x, y, and z axes.
scale	Construct a 4x4 matrix to perform scaling.
setCol	Set the column of a 4x4 matrix referred to by the specified index.
setCol0	Set column 0 of a 4x4 matrix.
setCol1	Set column 1 of a 4x4 matrix.
setCol2	Set column 2 of a 4x4 matrix.
setCol3	Set column 3 of a 4x4 matrix.
setElem	Set the element of a 4x4 matrix referred to by column and row indices.
setElem	Set the element of a 4x4 matrix referred to by column and row indices (scalar data contained in vector data type).
setRow	Set the row of a 4x4 matrix referred to by the specified index.
setTranslation	Set translation component.
setUpper3x3	Set the upper-left 3x3 submatrix.
translation	Construct a 4x4 matrix to perform translation.

Constructors and Destructors

Matrix4

Default constructor; does no initialization.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline Matrix4();
              }
        }
}
```

Arguments

None

Return Values

None

Description

Default constructor; does no initialization.

Matrix4

Copy a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline Matrix4(
                            const Matrix4 &mat
                     );
              }
        }
}
```

Arguments

mat	4x4 matrix

Return Values

None

Description

Construct a copy of a 4x4 matrix.

Matrix4

Construct a 4x4 matrix containing the specified columns.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline Matrix4(
                            Vector4 col0,
                            Vector4 col1,
                            Vector4 col2,
                            Vector4 col3
                     );
              }
        }
}
```

Arguments

col0	4-D vector
col1	4-D vector
col2	4-D vector
col3	4-D vector

Return Values

None

Description

Construct a 4x4 matrix containing the specified columns.

Matrix4

Construct a 4x4 matrix from a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     explicit inline Matrix4(
                            const Transform3 &mat
                     );
              }
        }
}
```

Arguments

mat	3x4 transformation matrix

Return Values

None

Description

Construct a 4x4 matrix whose upper 3x4 elements are equal to the 3x4 transformation matrix argument and whose bottom row is equal to (0,0,0,1).

Matrix4

Construct a 4x4 matrix from a 3x3 matrix and a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline Matrix4(
                            const Matrix3 &mat,
                            Vector3 translateVec
                     );
              }
        }
}
```

Arguments

mat	3x3 matrix
translateVec	3-D vector

Return Values

None

Description

Construct a 4x4 matrix whose upper 3x3 elements are equal to the 3x3 matrix argument, whose translation component is equal to the 3-D vector argument, and whose bottom row is (0,0,0,1).

Matrix4

Construct a 4x4 matrix from a unit-length quaternion and a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline Matrix4(
                            Quat unitQuat,
                            Vector3 translateVec
                     );
              }
        }
}
```

Arguments

unitQuat	Quaternion, expected to be unit-length
translateVec	3-D vector

Return Values

None

Description

Construct a 4x4 matrix whose upper-left 3x3 submatrix is a rotation matrix converted from the unit-length quaternion argument, whose translation component is equal to the 3-D vector argument, and whose bottom row is (0,0,0,1).

Matrix4

Set all elements of a 4x4 matrix to the same scalar value.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     explicit inline Matrix4(
                            float scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

None

Description

Construct a 4x4 matrix with all elements set to the scalar value argument.

Matrix4

Set all elements of a 4x4 matrix to the same scalar value (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     explicit inline Matrix4(
                            floatInVec scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)

Return Values

None

Description

Construct a 4x4 matrix with all elements set to the scalar value argument.

Operator Methods

operator *

Multiply a 4x4 matrix by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline const Matrix4 operator *(
                            float scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

Product of the specified 4x4 matrix and scalar

Description

Multiply a 4x4 matrix by a scalar.

operator *

Multiply a 4x4 matrix by a scalar (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline const Matrix4 operator *(
                            floatInVec scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)

Return Values

Product of the specified 4x4 matrix and scalar

Description

Multiply a 4x4 matrix by a scalar.

operator *

Multiply a 4x4 matrix by a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline const Vector4 operator *(
                            Vector4 vec
                     );
              }
        }
}
```

Arguments

vec	4-D vector

Return Values

Product of the specified 4x4 matrix and 4-D vector

Description

Multiply a 4x4 matrix by a 4-D vector.

operator *

Multiply a 4x4 matrix by a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline const Vector4 operator *(
                           Vector3 vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

Product of the specified 4x4 matrix and 3-D vector

Description

Multiply a 4x4 matrix by a 3-D vector treated as if it were a 4-D vector with the w element equal to 0.

operator *

Multiply a 4x4 matrix by a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline const Vector4 operator *(
                           Point3 pnt
                     );
              }
        }
}
```

Arguments

pnt	3-D point

Return Values

Product of the specified 4x4 matrix and 3-D point

Description

Multiply a 4x4 matrix by a 3-D point treated as if it were a 4-D vector with the w element equal to 1.

operator *

Multiply two 4x4 matrices.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline const Matrix4 operator *(
                            const Matrix4 &mat
                     );
              }
        }
}
```

Arguments

mat	4x4 matrix

Return Values

Product of the specified 4x4 matrices

Description

Multiply two 4x4 matrices.

operator *

Multiply a 4x4 matrix by a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline const Matrix4 operator *(
                            const Transform3 &tfrm
                     );
              }
        }
}
```

Arguments

tfrm	3x4 transformation matrix

Return Values

Product of the specified 4x4 matrix and 3x4 transformation matrix

Description

Multiply a 4x4 matrix by a 3x4 transformation matrix treated as if it were a 4x4 matrix with the bottom row equal to (0,0,0,1).

operator *=

Perform compound assignment and multiplication by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline Matrix4 &operator *=(
                            float scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

A reference to the resulting 4x4 matrix

Description

Perform compound assignment and multiplication by a scalar.

operator *=

Perform compound assignment and multiplication by a scalar (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
                class Matrix4 {
                        inline Matrix4 &operator *=(
                                floatInVec scalar
                        );
                }
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)

Return Values

A reference to the resulting 4x4 matrix

Description

Perform compound assignment and multiplication by a scalar.

operator *=

Perform compound assignment and multiplication by a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline Matrix4 &operator *=(
                            const Matrix4 &mat
                     );
              }
        }
}
```

Arguments

mat	4x4 matrix

Return Values

A reference to the resulting 4x4 matrix

Description

Perform compound assignment and multiplication by a 4x4 matrix.

operator *=

Perform compound assignment and multiplication by a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline Matrix4 &operator *=(
                            const Transform3 &tfrm
                     );
              }
        }
}
```

Arguments

tfrm	3x4 transformation matrix

Return Values

A reference to the resulting 4x4 matrix

Description

Perform compound assignment and multiplication by a 3x4 transformation matrix.

operator+

Add two 4x4 matrices.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline const Matrix4 operator+(
                            const Matrix4 &mat
                     );
              }
        }
}
```

Arguments

mat	4x4 matrix

Return Values

Sum of the specified 4x4 matrices

Description

Add two 4x4 matrices.

operator+=

Perform compound assignment and addition with a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline Matrix4 &operator+=(
                            const Matrix4 &mat
                     );
              }
        }
}
```

Arguments

mat	4x4 matrix

Return Values

A reference to the resulting 4x4 matrix

Description

Perform compound assignment and addition with a 4x4 matrix.

operator-

Subtract a 4x4 matrix from another 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline const Matrix4 operator-(
                            const Matrix4 &mat
                     );
              }
        }
}
```

Arguments

mat	4x4 matrix

Return Values

Difference of the specified 4x4 matrices

Description

Subtract a 4x4 matrix from another 4x4 matrix.

operator-

Negate all elements of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline const Matrix4 operator-();
              }
        }
}
```

Arguments

None

Return Values

4x4 matrix containing negated elements of the specified 4x4 matrix

Description

Negate all elements of a 4x4 matrix.

operator-=

Perform compound assignment and subtraction by a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline Matrix4 &operator-=(
                            const Matrix4 &mat
                     );
              }
        }
}
```

Arguments

mat	4x4 matrix

Return Values

A reference to the resulting 4x4 matrix

Description

Perform compound assignment and subtraction by a 4x4 matrix.

operator=

Assign one 4x4 matrix to another.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline Matrix4 &operator=(
                            const Matrix4 &mat
                     );
              }
        }
}
```

Arguments

mat	4x4 matrix

Return Values

A reference to the resulting 4x4 matrix

Description

Assign one 4x4 matrix to another.

operator[]

Subscripting operator to set or get a column.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline Vector4 &operator[](
                            int col
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-3

Return Values

A reference to indexed column

Description

Subscripting operator invoked when applied to non-const Matrix4.

operator[]

Subscripting operator to get a column.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline const Vector4 operator[](
                            int col
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-3

Return Values

Indexed column

Description

Subscripting operator invoked when applied to const Matrix4.

Public Static Methods

frustum

Construct a perspective projection matrix based on frustum.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     static inline const Matrix4 frustum(
                            float left,
                            float right,
                            float bottom,
                            float top,
                            float zNear,
                            float zFar
                     );
              }
        }
}
```

Arguments

left	Scalar value
right	Scalar value
bottom	Scalar value
top	Scalar value
zNear	Scalar value
zFar	Scalar value

Return Values

The constructed 4x4 matrix

Description

Construct a perspective projection matrix based on frustum, equal to:

```
2*zNear/(right-left)   0          (right+left)/(right-left)      0
            0      2*zNear/(top-bottom)  (top+bottom)/(top-bottom)      0
            0              0        -(zFar+zNear)/(zFar-zNear)
-2*zFar*zNear/(zFar-zNear)
            0              0                  -1                0        .
```

identity

Construct an identity 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     static inline const Matrix4 identity();
              }
        }
}
```

Arguments

None

Return Values

The constructed 4x4 matrix

Description

Construct an identity 4x4 matrix in which non-diagonal elements are zero and diagonal elements are 1.

lookAt

Construct viewing matrix based on eye position, position looked at, and up direction.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     static inline const Matrix4 lookAt(
                            Point3 eyePos,
                            Point3 lookAtPos,
                            Vector3 upVec
                     );
              }
        }
}
```

Arguments

eyePos	3-D point
lookAtPos	3-D point
upVec	3-D vector

Return Values

The constructed 4x4 matrix

Description

Construct the inverse of a coordinate frame that is centered at the eye position, with z axis directed away from lookAtPos, and y axis oriented to best match the up direction.

orthographic

Construct an orthographic projection matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     static inline const Matrix4 orthographic(
                            float left,
                            float right,
                            float bottom,
                            float top,
                            float zNear,
                            float zFar
                     );
              }
        }
}
```

Arguments

left	Scalar value
right	Scalar value
bottom	Scalar value
top	Scalar value
zNear	Scalar value
zFar	Scalar value

Return Values

The constructed 4x4 matrix

Description

Construct an orthographic projection matrix, equal to

```
2/(right-left)       0           0        -(right+left)/(right-left)
    0         2/(top-bottom)     0        -(top+bottom)/(top-bottom)
    0              0      -2/(zFar-zNear) -(zFar+zNear)/(zFar-zNear)
    0              0           0                   1        .
```

perspective

Construct a perspective projection matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
             class Matrix4 {
                    static inline const Matrix4 perspective(
                           float fovyRadians,
                           float aspect,
                           float zNear,
                           float zFar
                    );
             }
        }
}
```

Arguments

fovyRadians	Scalar value
aspect	Scalar value
zNear	Scalar value
zFar	Scalar value

Return Values

The constructed 4x4 matrix

Description

Construct a perspective projection matrix, equal to:

```
cot(fovyRadians/2)/aspect   0                0                0
           0           cot(fovyRadians/2)      0                0
           0                  0   (zFar+zNear)/(zNear-zFar)
2*zFar*zNear/(zNear-zFar)
           0                  0            -1                0       .
```

rotation

Construct a 4x4 matrix to rotate around a unit-length 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     static inline const Matrix4 rotation(
                            float radians,
                            Vector3 unitVec
                     );
              }
        }
}
```

Arguments

radians	Scalar value
unitVec	3-D vector, expected to be unit-length

Return Values

The constructed 4x4 matrix

Description

Construct a 4x4 matrix to rotate around a unit-length 3-D vector by the specified radians angle.

rotation

Construct a 4x4 matrix to rotate around a unit-length 3-D vector (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     static inline const Matrix4 rotation(
                            floatInVec radians,
                            Vector3 unitVec
                     );
              }
        }
}
```

Arguments

radians	Scalar value (stored in vector data type)
unitVec	3-D vector, expected to be unit-length

Return Values

The constructed 4x4 matrix

Description

Construct a 4x4 matrix to rotate around a unit-length 3-D vector by the specified radians angle.

rotation

Construct a rotation matrix from a unit-length quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     static inline const Matrix4 rotation(
                            Quat unitQuat
                     );
              }
        }
}
```

Arguments

unitQuat	Quaternion, expected to be unit-length

Return Values

The constructed 4x4 matrix

Description

Construct a 4x4 matrix that applies the same rotation as the specified unit-length quaternion.

rotationX

Construct a 4x4 matrix to rotate around the x axis.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     static inline const Matrix4 rotationX(
                            float radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value

Return Values

The constructed 4x4 matrix

Description

Construct a 4x4 matrix to rotate around the x axis by the specified radians angle.

rotationX

Construct a 4x4 matrix to rotate around the x axis (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     static inline const Matrix4 rotationX(
                            floatInVec radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value (stored in vector data type)

Return Values

The constructed 4x4 matrix

Description

Construct a 4x4 matrix to rotate around the x axis by the specified radians angle.

rotationY

Construct a 4x4 matrix to rotate around the y axis.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     static inline const Matrix4 rotationY(
                            float radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value

Return Values

The constructed 4x4 matrix

Description

Construct a 4x4 matrix to rotate around the y axis by the specified radians angle.

rotationY

Construct a 4x4 matrix to rotate around the y axis (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     static inline const Matrix4 rotationY(
                            floatInVec radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value (stored in vector data type)

Return Values

The constructed 4x4 matrix

Description

Construct a 4x4 matrix to rotate around the y axis by the specified radians angle.

rotationZ

Construct a 4x4 matrix to rotate around the z axis.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     static inline const Matrix4 rotationZ(
                            float radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value

Return Values

The constructed 4x4 matrix

Description

Construct a 4x4 matrix to rotate around the z axis by the specified radians angle.

rotationZ

Construct a 4x4 matrix to rotate around the z axis (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     static inline const Matrix4 rotationZ(
                            floatInVec radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value (stored in vector data type)

Return Values

The constructed 4x4 matrix

Description

Construct a 4x4 matrix to rotate around the z axis by the specified radians angle.

rotationZYX

Construct a 4x4 matrix to rotate around the x, y, and z axes.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     static inline const Matrix4 rotationZYX(
                            Vector3 radiansXYZ
                     );
              }
        }
}
```

Arguments

radiansXYZ	3-D vector

Return Values

The constructed 4x4 matrix

Description

Construct a 4x4 matrix to rotate around the x, y, and z axes by the radians angles contained in a 3-D vector. Equivalent to *rotationZ(radiansXYZ.getZ()) * rotationY(radiansXYZ.getY()) * rotationX(radiansXYZ.getX())*.

scale

Construct a 4x4 matrix to perform scaling.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     static inline const Matrix4 scale(
                            Vector3 scaleVec
                     );
              }
        }
}
```

Arguments

scaleVec	3-D vector

Return Values

The constructed 4x4 matrix

Description

Construct a 4x4 matrix to perform scaling, in which the non-diagonal elements are zero and the diagonal elements are set to the elements of *scaleVec*.

translation

Construct a 4x4 matrix to perform translation.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     static inline const Matrix4 translation(
                            Vector3 translateVec
                     );
              }
        }
}
```

Arguments

translateVec	3-D vector

Return Values

The constructed 4x4 matrix

Description

Construct a 4x4 matrix to perform translation, which is an identity matrix except for the translation component, with coordinates equal to those in *translateVec*.

Public Instance Methods

getCol

Get the column of a 4x4 matrix referred to by the specified index.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline const Vector4 getCol(
                            int col
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-3

Return Values

The column referred to by the specified index

Description

Get the column of a 4x4 matrix referred to by the specified index.

getCol0

Get column 0 of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline const Vector4 getCol0();
              }
        }
}
```

Arguments

None

Return Values

Column 0

Description

Get column 0 of a 4x4 matrix.

getCol1

Get column 1 of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline const Vector4 getCol1();
              }
        }
}
```

Arguments

None

Return Values

Column 1

Description

Get column 1 of a 4x4 matrix.

getCol2

Get column 2 of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline const Vector4 getCol2();
              }
        }
}
```

Arguments

None

Return Values

Column 2

Description

Get column 2 of a 4x4 matrix.

getCol3

Get column 3 of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline const Vector4 getCol3();
              }
        }
}
```

Arguments

None

Return Values

Column 3

Description

Get column 3 of a 4x4 matrix.

getElem

Get the element of a 4x4 matrix referred to by column and row indices.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
             class Matrix4 {
                   inline const floatInVec getElem(
                         int col,
                         int row
                   );
             }
        }
}
```

Arguments

col	Index, expected in the range 0-3
row	Index, expected in the range 0-3

Return Values

Element selected by *col* and *row*

Description

Get the element of a 4x4 matrix referred to by column and row indices.

getRow

Get the row of a 4x4 matrix referred to by the specified index.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
                class Matrix4 {
                        inline const Vector4 getRow(
                                int row
                        );
                }
        }
}
```

Arguments

row	Index, expected in the range 0-3

Return Values

The row referred to by the specified index

Description

Get the row of a 4x4 matrix referred to by the specified index.

getTranslation

Get the translation component of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline const Vector3 getTranslation();
              }
        }
}
```

Arguments

None

Return Values

Translation component

Description

Get the translation component of a 4x4 matrix.

getUpper3x3

Get the upper-left 3x3 submatrix of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
             class Matrix4 {
                    inline const Matrix3 getUpper3x3();
             }
        }
}
```

Arguments

None

Return Values

Upper-left 3x3 submatrix

Description

Get the upper-left 3x3 submatrix of a 4x4 matrix.

setCol

Set the column of a 4x4 matrix referred to by the specified index.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline Matrix4 &setCol(
                            int col,
                            Vector4 vec
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-3
vec	4-D vector

Return Values

A reference to the resulting 4x4 matrix

Description

Set the column of a 4x4 matrix referred to by the specified index.

setCol0

Set column 0 of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline Matrix4 &setCol0(
                            Vector4 col0
                     );
              }
        }
}
```

Arguments

col0	4-D vector

Return Values

A reference to the resulting 4x4 matrix

Description

Set column 0 of a 4x4 matrix.

setCol1

Set column 1 of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline Matrix4 &setCol1(
                            Vector4 col1
                     );
              }
        }
}
```

Arguments

col1	4-D vector

Return Values

A reference to the resulting 4x4 matrix

Description

Set column 1 of a 4x4 matrix.

setCol2

Set column 2 of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline Matrix4 &setCol2(
                            Vector4 col2
                     );
              }
        }
}
```

Arguments

col2	4-D vector

Return Values

A reference to the resulting 4x4 matrix

Description

Set column 2 of a 4x4 matrix.

setCol3

Set column 3 of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline Matrix4 &setCol3(
                            Vector4 col3
                     );
              }
        }
}
```

Arguments

col3	4-D vector

Return Values

A reference to the resulting 4x4 matrix

Description

Set column 3 of a 4x4 matrix.

setElem

Set the element of a 4x4 matrix referred to by column and row indices.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline Matrix4 &setElem(
                            int col,
                            int row,
                            float val
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-3
row	Index, expected in the range 0-3
val	Scalar value

Return Values

A reference to the resulting 4x4 matrix

Description

Set the element of a 4x4 matrix referred to by column and row indices.

setElem

Set the element of a 4x4 matrix referred to by column and row indices (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline Matrix4 &setElem(
                            int col,
                            int row,
                            floatInVec val
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-3
row	Index, expected in the range 0-3
val	Scalar value (stored in vector data type)

Return Values

A reference to the resulting 4x4 matrix

Description

Set the element of a 4x4 matrix referred to by column and row indices.

setRow

Set the row of a 4x4 matrix referred to by the specified index.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline Matrix4 &setRow(
                            int row,
                            Vector4 vec
                     );
              }
        }
}
```

Arguments

row	Index, expected in the range 0-3
vec	4-D vector

Return Values

A reference to the resulting 4x4 matrix

Description

Set the row of a 4x4 matrix referred to by the specified index.

setTranslation

Set translation component.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline Matrix4 &setTranslation(
                            Vector3 translateVec
                     );
              }
        }
}
```

Arguments

translateVec	3-D vector

Return Values

A reference to the resulting 4x4 matrix

Description

Set the translation component of a 4x4 matrix equal to the specified 3-D vector.

Notes

This function does not change the bottom row elements.

setUpper3x3

Set the upper-left 3x3 submatrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Matrix4 {
                     inline Matrix4 &setUpper3x3(
                            const Matrix3 &mat3
                     );
              }
        }
}
```

Arguments

mat3	3x3 matrix

Return Values

A reference to the resulting 4x4 matrix

Description

Set the upper-left 3x3 submatrix elements of a 4x4 matrix equal to the specified 3x3 matrix.

Notes

This function does not change the bottom row elements.

Vectormath::Aos::Point3

Summary

Vectormath::Aos::Point3

A 3-D point in array-of-structures format.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
class Point3;
```

Description

A class representing a 3-D point stored in array-of-structures (AoS) format.

Methods Summary

Methods	Description
get128	Get vector float data from a 3-D point.
getElem	Get an x, y, or z element of a 3-D point by index.
getX	Get the x element of a 3-D point.
getY	Get the y element of a 3-D point.
getZ	Get the z element of a 3-D point.
operator+	Add a 3-D point to a 3-D vector.
operator+=	Perform compound assignment and addition with a 3-D vector.
operator-	Subtract a 3-D point from another 3-D point.
operator-	Subtract a 3-D vector from a 3-D point.
operator-=	Perform compound assignment and subtraction by a 3-D vector.
operator=	Assign one 3-D point to another.
operator[]	Subscripting operator to set or get an element.
operator[]	Subscripting operator to get an element.
Point3	Default constructor; does no initialization.
Point3	Construct a 3-D point from x, y, and z elements.
Point3	Construct a 3-D point from x, y, and z elements (scalar data contained in vector data type).
Point3	Copy elements from a 3-D vector into a 3-D point.
Point3	Set all elements of a 3-D point to the same scalar value.
Point3	Set all elements of a 3-D point to the same scalar value (scalar data contained in vector data type).
Point3	Set vector float data in a 3-D point.
setElem	Set an x, y, or z element of a 3-D point by index.
setElem	Set an x, y, or z element of a 3-D point by index (scalar data contained in vector data type).
setX	Set the x element of a 3-D point.
setX	Set the x element of a 3-D point (scalar data contained in vector data type).
setY	Set the y element of a 3-D point.
setY	Set the y element of a 3-D point (scalar data contained in vector data type).
setZ	Set the z element of a 3-D point.

Methods	Description
setZ	Set the z element of a 3-D point (scalar data contained in vector data type).

Constructors and Destructors

Point3

Default constructor; does no initialization.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     inline Point3();
              }
        }
}
```

Arguments

None

Return Values

None

Description

Default constructor; does no initialization.

Point3

Construct a 3-D point from x, y, and z elements.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     inline Point3(
                            float x,
                            float y,
                            float z
                     );
              }
        }
}
```

Arguments

x	Scalar value
y	Scalar value
z	Scalar value

Return Values

None

Description

Construct a 3-D point containing the specified x, y, and z elements.

Point3

Construct a 3-D point from x, y, and z elements (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     inline Point3(
                            floatInVec x,
                            floatInVec y,
                            floatInVec z
                     );
              }
        }
}
```

Arguments

x	Scalar value (stored in vector data type)
y	Scalar value (stored in vector data type)
z	Scalar value (stored in vector data type)

Return Values

None

Description

Construct a 3-D point containing the specified x, y, and z elements.

Point3

Copy elements from a 3-D vector into a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     explicit inline Point3(
                            Vector3 vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

None

Description

Construct a 3-D point containing the x, y, and z elements of the specified 3-D vector.

Point3

Set all elements of a 3-D point to the same scalar value.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     explicit inline Point3(
                            float scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

None

Description

Construct a 3-D point with all elements set to the scalar value argument.

Point3

Set all elements of a 3-D point to the same scalar value (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     explicit inline Point3(
                            floatInVec scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)

Return Values

None

Description

Construct a 3-D point with all elements set to the scalar value argument.

Point3

Set vector float data in a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     explicit inline Point3(
                            vec_float4 vf4
                     );
              }
        }
}
```

Arguments

vf4	Scalar value

Return Values

None

Description

Construct a 3-D point whose internal vector float data is set to the vector float argument.

Operator Methods

operator+

Add a 3-D point to a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     inline const Point3 operator+(
                            Vector3 vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

Sum of the specified 3-D point and 3-D vector

Description

Add a 3-D point to a 3-D vector.

operator+=

Perform compound assignment and addition with a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     inline Point3 &operator+=(
                            Vector3 vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

A reference to the resulting 3-D point

Description

Perform compound assignment and addition with a 3-D vector.

operator-

Subtract a 3-D point from another 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     inline const Vector3 operator-(
                            Point3 pnt
                     );
              }
        }
}
```

Arguments

pnt	3-D point

Return Values

Difference of the specified 3-D points

Description

Subtract a 3-D point from another 3-D point.

operator-

Subtract a 3-D vector from a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     inline const Point3 operator-(
                            Vector3 vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

Difference of the specified 3-D point and 3-D vector

Description

Subtract a 3-D vector from a 3-D point.

operator-=

Perform compound assignment and subtraction by a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     inline Point3 &operator-=(
                            Vector3 vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

A reference to the resulting 3-D point

Description

Perform compound assignment and subtraction by a 3-D vector.

operator=

Assign one 3-D point to another.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     inline Point3 &operator=(
                            Point3 pnt
                     );
              }
        }
}
```

Arguments

pnt	3-D point

Return Values

A reference to the resulting 3-D point

Description

Assign one 3-D point to another.

operator[]

Subscripting operator to set or get an element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     inline VecIdx operator[](
                            int idx
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-2

Return Values

VecIdx which holds a reference to the selected element

Description

Subscripting operator invoked when applied to non-const Point3.

operator[]

Subscripting operator to get an element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     inline const floatInVec operator[](
                            int idx
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-2

Return Values

Indexed element

Description

Subscripting operator invoked when applied to const Point3.

Public Instance Methods

get128

Get vector float data from a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     inline vec_float4 get128();
              }
        }
}
```

Arguments

None

Return Values

Internal vector float data

Description

Get internal vector float data from a 3-D point.

getElem

Get an x, y, or z element of a 3-D point by index.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     inline const floatInVec getElem(
                            int idx
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-2

Return Values

Element selected by the specified index

Description

Get an x, y, or z element of a 3-D point by specifying an index of 0, 1, or 2, respectively.

getX

Get the x element of a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     inline const floatInVec getX();
              }
        }
}
```

Arguments

None

Return Values

x element of a 3-D point

Description

Get the x element of a 3-D point.

getY

Get the y element of a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     inline const floatInVec getY();
              }
        }
}
```

Arguments

None

Return Values

y element of a 3-D point

Description

Get the y element of a 3-D point.

getZ

Get the z element of a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     inline const floatInVec getZ();
              }
        }
}
```

Arguments

None

Return Values

z element of a 3-D point

Description

Get the z element of a 3-D point.

setElem

Set an x, y, or z element of a 3-D point by index.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     inline Point3 &setElem(
                            int idx,
                            float value
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-2
value	Scalar value

Return Values

A reference to the resulting 3-D point

Description

Set an x, y, or z element of a 3-D point by specifying an index of 0, 1, or 2, respectively.

setElem

Set an x, y, or z element of a 3-D point by index (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     inline Point3 &setElem(
                            int idx,
                            floatInVec value
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-2
value	Scalar value (stored in vector data type)

Return Values

A reference to the resulting 3-D point

Description

Set an x, y, or z element of a 3-D point by specifying an index of 0, 1, or 2, respectively.

setX

Set the x element of a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     inline Point3 &setX(
                            float x
                     );
              }
        }
}
```

Arguments

x	Scalar value

Return Values

A reference to the resulting 3-D point

Description

Set the x element of a 3-D point to the specified scalar value.

setX

Set the x element of a 3-D point (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     inline Point3 &setX(
                            floatInVec x
                     );
              }
        }
}
```

Arguments

x	Scalar value (stored in vector data type)

Return Values

A reference to the resulting 3-D point

Description

Set the x element of a 3-D point to the specified scalar value.

setY

Set the y element of a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     inline Point3 &setY(
                            float y
                     );
              }
        }
}
```

Arguments

y	Scalar value

Return Values

A reference to the resulting 3-D point

Description

Set the y element of a 3-D point to the specified scalar value.

setY

Set the y element of a 3-D point (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     inline Point3 &setY(
                            floatInVec y
                     );
              }
        }
}
```

Arguments

y	Scalar value (stored in vector data type)

Return Values

A reference to the resulting 3-D point

Description

Set the y element of a 3-D point to the specified scalar value.

setZ

Set the z element of a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     inline Point3 &setZ(
                            float z
                     );
              }
        }
}
```

Arguments

z	Scalar value

Return Values

A reference to the resulting 3-D point

Description

Set the z element of a 3-D point to the specified scalar value.

setZ

Set the z element of a 3-D point (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Point3 {
                     inline Point3 &setZ(
                            floatInVec z
                     );
              }
        }
}
```

Arguments

z	Scalar value (stored in vector data type)

Return Values

A reference to the resulting 3-D point

Description

Set the z element of a 3-D point to the specified scalar value.

Vectormath::Aos::Quat

Summary

Vectormath::Aos::Quat

A quaternion in array-of-structures format.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
class Quat;
```

Description

A class representing a quaternion stored in array-of-structures (AoS) format.

Methods Summary

Methods	Description
get128	Get vector float data from a quaternion.
getElem	Get an x, y, z, or w element of a quaternion by index.
getW	Get the w element of a quaternion.
getX	Get the x element of a quaternion.
getXYZ	Get the x, y, and z elements of a quaternion.
getY	Get the y element of a quaternion.
getZ	Get the z element of a quaternion.
identity	Construct an identity quaternion.
operator *	Multiply two quaternions.
operator *	Multiply a quaternion by a scalar.
operator *	Multiply a quaternion by a scalar (scalar data contained in vector data type).
operator *=	Perform compound assignment and multiplication by a quaternion.
operator *=	Perform compound assignment and multiplication by a scalar.
operator *=	Perform compound assignment and multiplication by a scalar (scalar data contained in vector data type).
operator+	Add two quaternions.
operator+=	Perform compound assignment and addition with a quaternion.
operator-	Subtract a quaternion from another quaternion.
operator-	Negate all elements of a quaternion.
operator-=	Perform compound assignment and subtraction by a quaternion.
operator/	Divide a quaternion by a scalar.
operator/	Divide a quaternion by a scalar (scalar data contained in vector data type).
operator/=	Perform compound assignment and division by a scalar.
operator/=	Perform compound assignment and division by a scalar (scalar data contained in vector data type).
operator=	Assign one quaternion to another.
operator[]	Subscripting operator to set or get an element.
operator[]	Subscripting operator to get an element.
Quat	Default constructor; does no initialization.

Methods	Description
Quat	Construct a quaternion from x, y, z, and w elements.
Quat	Construct a quaternion from x, y, z, and w elements (scalar data contained in vector data type).
Quat	Construct a quaternion from a 3-D vector and a scalar.
Quat	Construct a quaternion from a 3-D vector and a scalar (scalar data contained in vector data type).
Quat	Copy elements from a 4-D vector into a quaternion.
Quat	Convert a rotation matrix to a unit-length quaternion.
Quat	Set all elements of a quaternion to the same scalar value.
Quat	Set all elements of a quaternion to the same scalar value (scalar data contained in vector data type).
Quat	Set vector float data in a quaternion.
rotation	Construct a quaternion to rotate between two unit-length 3-D vectors.
rotation	Construct a quaternion to rotate around a unit-length 3-D vector.
rotation	Construct a quaternion to rotate around a unit-length 3-D vector (scalar data contained in vector data type).
rotationX	Construct a quaternion to rotate around the x axis.
rotationX	Construct a quaternion to rotate around the x axis (scalar data contained in vector data type).
rotationY	Construct a quaternion to rotate around the y axis.
rotationY	Construct a quaternion to rotate around the y axis (scalar data contained in vector data type).
rotationZ	Construct a quaternion to rotate around the z axis.
rotationZ	Construct a quaternion to rotate around the z axis (scalar data contained in vector data type).
setElem	Set an x, y, z, or w element of a quaternion by index.
setElem	Set an x, y, z, or w element of a quaternion by index (scalar data contained in vector data type).
setW	Set the w element of a quaternion.
setW	Set the w element of a quaternion (scalar data contained in vector data type).
setX	Set the x element of a quaternion.
setX	Set the x element of a quaternion (scalar data contained in vector data type).
setXYZ	Set the x, y, and z elements of a quaternion.
setY	Set the y element of a quaternion.
setY	Set the y element of a quaternion (scalar data contained in vector data type).
setZ	Set the z element of a quaternion.
setZ	Set the z element of a quaternion (scalar data contained in vector data type).

Constructors and Destructors

Quat

Default constructor; does no initialization.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline Quat();
              }
        }
}
```

Arguments

None

Return Values

None

Description

Default constructor; does no initialization.

Quat

Construct a quaternion from x, y, z, and w elements.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline Quat(
                            float x,
                            float y,
                            float z,
                            float w
                     );
              }
        }
}
```

Arguments

x	Scalar value
y	Scalar value
z	Scalar value
w	Scalar value

Return Values

None

Description

Construct a quaternion containing the specified x, y, z, and w elements.

Quat

Construct a quaternion from x, y, z, and w elements (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline Quat(
                            floatInVec x,
                            floatInVec y,
                            floatInVec z,
                            floatInVec w
                     );
              }
        }
}
```

Arguments

x	Scalar value (stored in vector data type)
y	Scalar value (stored in vector data type)
z	Scalar value (stored in vector data type)
w	Scalar value (stored in vector data type)

Return Values

None

Description

Construct a quaternion containing the specified x, y, z, and w elements.

Quat

Construct a quaternion from a 3-D vector and a scalar.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline Quat(
                            Vector3 xyz,
                            float w
                     );
              }
        }
}
```

Arguments

xyz	3-D vector
w	Scalar value

Return Values

None

Description

Construct a quaternion with the x, y, and z elements of the specified 3-D vector and with the w element set to the specified scalar.

Quat

Construct a quaternion from a 3-D vector and a scalar (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline Quat(
                            Vector3 xyz,
                            floatInVec w
                     );
              }
        }
}
```

Arguments

xyz	3-D vector
w	Scalar value (stored in vector data type)

Return Values

None

Description

Construct a quaternion with the x, y, and z elements of the specified 3-D vector and with the w element set to the specified scalar.

Quat

Copy elements from a 4-D vector into a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     explicit inline Quat(
                            Vector4 vec
                     );
              }
        }
}
```

Arguments

vec	4-D vector

Return Values

None

Description

Construct a quaternion containing the x, y, z, and w elements of the specified 4-D vector.

Quat

Convert a rotation matrix to a unit-length quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     explicit inline Quat(
                            const Matrix3 &rotMat
                     );
              }
        }
}
```

Arguments

rotMat	3x3 matrix, expected to be a rotation matrix

Return Values

None

Description

Construct a unit-length quaternion representing the same transformation as a rotation matrix.

Quat

Set all elements of a quaternion to the same scalar value.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     explicit inline Quat(
                            float scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

None

Description

Construct a quaternion with all elements set to the scalar value argument.

Quat

Set all elements of a quaternion to the same scalar value (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     explicit inline Quat(
                            floatInVec scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)

Return Values

None

Description

Construct a quaternion with all elements set to the scalar value argument.

Quat

Set vector float data in a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     explicit inline Quat(
                            vec_float4 vf4
                     );
              }
        }
}
```

Arguments

vf4	Scalar value

Return Values

None

Description

Construct a quaternion whose internal vector float data is set to the vector float argument.

Operator Methods

operator *

Multiply two quaternions.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline const Quat operator *(
                            Quat quat
                     );
              }
        }
}
```

Arguments

quat	Quaternion

Return Values

Product of the specified quaternions

Description

Multiply two quaternions.

operator *

Multiply a quaternion by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline const Quat operator *(
                            float scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

Product of the specified quaternion and scalar

Description

Multiply a quaternion by a scalar.

operator *

Multiply a quaternion by a scalar (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline const Quat operator *(
                            floatInVec scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)

Return Values

Product of the specified quaternion and scalar

Description

Multiply a quaternion by a scalar.

operator *=

Perform compound assignment and multiplication by a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline Quat &operator *=(
                            Quat quat
                     );
              }
        }
}
```

Arguments

quat	Quaternion

Return Values

A reference to the resulting quaternion

Description

Perform compound assignment and multiplication by a quaternion.

operator *=

Perform compound assignment and multiplication by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline Quat &operator *=(
                            float scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

A reference to the resulting quaternion

Description

Perform compound assignment and multiplication by a scalar.

operator *=

Perform compound assignment and multiplication by a scalar (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline Quat &operator *=(
                            floatInVec scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)

Return Values

A reference to the resulting quaternion

Description

Perform compound assignment and multiplication by a scalar.

operator+

Add two quaternions.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline const Quat operator+(
                            Quat quat
                     );
              }
        }
}
```

Arguments

quat	Quaternion

Return Values

Sum of the specified quaternions

Description

Add two quaternions.

operator+=

Perform compound assignment and addition with a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline Quat &operator+=(
                            Quat quat
                     );
              }
        }
}
```

Arguments

quat	Quaternion

Return Values

A reference to the resulting quaternion

Description

Perform compound assignment and addition with a quaternion.

operator-

Subtract a quaternion from another quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline const Quat operator-(
                            Quat quat
                     );
              }
        }
}
```

Arguments

quat	Quaternion

Return Values

Difference of the specified quaternions

Description

Subtract a quaternion from another quaternion.

operator-

Negate all elements of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline const Quat operator-();
              }
        }
}
```

Arguments

None

Return Values

Quaternion containing negated elements of the specified quaternion

Description

Negate all elements of a quaternion.

operator-=

Perform compound assignment and subtraction by a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline Quat &operator-=(
                            Quat quat
                     );
              }
        }
}
```

Arguments

quat	Quaternion

Return Values

A reference to the resulting quaternion

Description

Perform compound assignment and subtraction by a quaternion.

operator/

Divide a quaternion by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline const Quat operator/(
                            float scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

Quotient of the specified quaternion and scalar

Description

Divide a quaternion by a scalar.

operator/

Divide a quaternion by a scalar (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline const Quat operator/(
                            floatInVec scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)

Return Values

Quotient of the specified quaternion and scalar

Description

Divide a quaternion by a scalar.

operator/=

Perform compound assignment and division by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline Quat &operator/=(
                            float scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

A reference to the resulting quaternion

Description

Perform compound assignment and division by a scalar.

operator/=

Perform compound assignment and division by a scalar (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline Quat &operator/=(
                            floatInVec scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)

Return Values

A reference to the resulting quaternion

Description

Perform compound assignment and division by a scalar.

operator=

Assign one quaternion to another.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline Quat &operator=(
                            Quat quat
                     );
              }
        }
}
```

Arguments

quat	Quaternion

Return Values

A reference to the resulting quaternion

Description

Assign one quaternion to another.

operator[]

Subscripting operator to set or get an element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline VecIdx operator[](
                            int idx
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-3

Return Values

VecIdx which holds a reference to the selected element

Description

Subscripting operator invoked when applied to non-const Quat.

operator[]

Subscripting operator to get an element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline const floatInVec operator[](
                            int idx
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-3

Return Values

Indexed element

Description

Subscripting operator invoked when applied to const Quat.

Public Static Methods

identity

Construct an identity quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     static inline const Quat identity();
              }
        }
}
```

Arguments

None

Return Values

The constructed quaternion

Description

Construct an identity quaternion equal to (0,0,0,1).

rotation

Construct a quaternion to rotate between two unit-length 3-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     static inline const Quat rotation(
                            Vector3 unitVec0,
                            Vector3 unitVec1
                     );
              }
        }
}
```

Arguments

unitVec0	3-D vector, expected to be unit-length
unitVec1	3-D vector, expected to be unit-length

Return Values

The constructed quaternion

Description

Construct a quaternion to rotate between two unit-length 3-D vectors.

Notes

The result is unpredictable if `unitVec0` and `unitVec1` point in opposite directions.

rotation

Construct a quaternion to rotate around a unit-length 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     static inline const Quat rotation(
                            float radians,
                            Vector3 unitVec
                     );
              }
        }
}
```

Arguments

radians	Scalar value
unitVec	3-D vector, expected to be unit-length

Return Values

The constructed quaternion

Description

Construct a quaternion to rotate around a unit-length 3-D vector by the specified radians angle.

rotation

Construct a quaternion to rotate around a unit-length 3-D vector (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     static inline const Quat rotation(
                            floatInVec radians,
                            Vector3 unitVec
                     );
              }
        }
}
```

Arguments

radians	Scalar value (stored in vector data type)
unitVec	3-D vector, expected to be unit-length

Return Values

The constructed quaternion

Description

Construct a quaternion to rotate around a unit-length 3-D vector by the specified radians angle.

rotationX

Construct a quaternion to rotate around the x axis.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     static inline const Quat rotationX(
                            float radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value

Return Values

The constructed quaternion

Description

Construct a quaternion to rotate around the x axis by the specified radians angle.

rotationX

Construct a quaternion to rotate around the x axis (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     static inline const Quat rotationX(
                            floatInVec radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value (stored in vector data type)

Return Values

The constructed quaternion

Description

Construct a quaternion to rotate around the x axis by the specified radians angle.

rotationY

Construct a quaternion to rotate around the y axis.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     static inline const Quat rotationY(
                            float radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value

Return Values

The constructed quaternion

Description

Construct a quaternion to rotate around the y axis by the specified radians angle.

rotationY

Construct a quaternion to rotate around the y axis (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     static inline const Quat rotationY(
                            floatInVec radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value (stored in vector data type)

Return Values

The constructed quaternion

Description

Construct a quaternion to rotate around the y axis by the specified radians angle.

rotationZ

Construct a quaternion to rotate around the z axis.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     static inline const Quat rotationZ(
                            float radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value

Return Values

The constructed quaternion

Description

Construct a quaternion to rotate around the z axis by the specified radians angle.

rotationZ

Construct a quaternion to rotate around the z axis (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     static inline const Quat rotationZ(
                            floatInVec radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value (stored in vector data type)

Return Values

The constructed quaternion

Description

Construct a quaternion to rotate around the z axis by the specified radians angle.

Public Instance Methods

get128

Get vector float data from a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline vec_float4 get128();
              }
        }
}
```

Arguments

None

Return Values

Internal vector float data

Description

Get internal vector float data from a quaternion.

getElem

Get an x, y, z, or w element of a quaternion by index.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline const floatInVec getElem(
                            int idx
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-3

Return Values

Element selected by the specified index

Description

Get an x, y, z, or w element of a quaternion by specifying an index of 0, 1, 2, or 3, respectively.

getW

Get the w element of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline const floatInVec getW();
              }
        }
}
```

Arguments

None

Return Values

w element of a quaternion

Description

Get the w element of a quaternion.

getX

Get the x element of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline const floatInVec getX();
              }
        }
}
```

Arguments

None

Return Values

x element of a quaternion

Description

Get the x element of a quaternion.

getXYZ

Get the x, y, and z elements of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline const Vector3 getXYZ();
              }
        }
}
```

Arguments

None

Return Values

3-D vector containing x, y, and z elements

Description

Extract a quaternion's x, y, and z elements into a 3-D vector.

getY

Get the y element of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline const floatInVec getY();
              }
        }
}
```

Arguments

None

Return Values

y element of a quaternion

Description

Get the y element of a quaternion.

getZ

Get the z element of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline const floatInVec getZ();
              }
        }
}
```

Arguments

None

Return Values

z element of a quaternion

Description

Get the z element of a quaternion.

setElem

Set an x, y, z, or w element of a quaternion by index.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline Quat &setElem(
                            int idx,
                            float value
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-3
value	Scalar value

Return Values

A reference to the resulting quaternion

Description

Set an x, y, z, or w element of a quaternion by specifying an index of 0, 1, 2, or 3, respectively.

setElem

Set an x, y, z, or w element of a quaternion by index (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline Quat &setElem(
                            int idx,
                            floatInVec value
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-3
value	Scalar value (stored in vector data type)

Return Values

A reference to the resulting quaternion

Description

Set an x, y, z, or w element of a quaternion by specifying an index of 0, 1, 2, or 3, respectively.

setW

Set the w element of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline Quat &setW(
                            float w
                     );
              }
        }
}
```

Arguments

w	Scalar value

Return Values

A reference to the resulting quaternion

Description

Set the w element of a quaternion to the specified scalar value.

setW

Set the w element of a quaternion (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline Quat &setW(
                            floatInVec w
                     );
              }
        }
}
```

Arguments

w	Scalar value (stored in vector data type)

Return Values

A reference to the resulting quaternion

Description

Set the w element of a quaternion to the specified scalar value.

setX

Set the x element of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline Quat &setX(
                            float x
                     );
              }
        }
}
```

Arguments

x	Scalar value

Return Values

A reference to the resulting quaternion

Description

Set the x element of a quaternion to the specified scalar value.

setX

Set the x element of a quaternion (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline Quat &setX(
                            floatInVec x
                     );
              }
        }
}
```

Arguments

x	Scalar value (stored in vector data type)

Return Values

A reference to the resulting quaternion

Description

Set the x element of a quaternion to the specified scalar value.

setXYZ

Set the x, y, and z elements of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline Quat &setXYZ(
                            Vector3 vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

A reference to the resulting quaternion

Description

Set the x, y, and z elements to those of the specified 3-D vector.

Notes

This function does not change the w element.

setY

Set the y element of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline Quat &setY(
                            float y
                     );
              }
        }
}
```

Arguments

y	Scalar value

Return Values

A reference to the resulting quaternion

Description

Set the y element of a quaternion to the specified scalar value.

setY

Set the y element of a quaternion (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline Quat &setY(
                            floatInVec y
                     );
              }
        }
}
```

Arguments

y	Scalar value (stored in vector data type)

Return Values

A reference to the resulting quaternion

Description

Set the y element of a quaternion to the specified scalar value.

setZ

Set the z element of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline Quat &setZ(
                            float z
                     );
              }
        }
}
```

Arguments

z	Scalar value

Return Values

A reference to the resulting quaternion

Description

Set the z element of a quaternion to the specified scalar value.

setZ

Set the z element of a quaternion (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Quat {
                     inline Quat &setZ(
                            floatInVec z
                     );
              }
        }
}
```

Arguments

z	Scalar value (stored in vector data type)

Return Values

A reference to the resulting quaternion

Description

Set the z element of a quaternion to the specified scalar value.

Vectormath::Aos::Transform3

Summary

Vectormath::Aos::Transform3

A 3x4 transformation matrix in array-of-structures format.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
class Transform3;
```

Description

A class representing a 3x4 transformation matrix stored in array-of-structures (AoS) format.

Methods Summary

Methods	Description
getCol	Get the column of a 3x4 transformation matrix referred to by the specified index.
getCol0	Get column 0 of a 3x4 transformation matrix.
getCol1	Get column 1 of a 3x4 transformation matrix.
getCol2	Get column 2 of a 3x4 transformation matrix.
getCol3	Get column 3 of a 3x4 transformation matrix.
getElem	Get the element of a 3x4 transformation matrix referred to by column and row indices.
getRow	Get the row of a 3x4 transformation matrix referred to by the specified index.
getTranslation	Get the translation component of a 3x4 transformation matrix.
getUpper3x3	Get the upper-left 3x3 submatrix of a 3x4 transformation matrix.
identity	Construct an identity 3x4 transformation matrix.
operator *	Multiply a 3x4 transformation matrix by a 3-D vector.
operator *	Multiply a 3x4 transformation matrix by a 3-D point.
operator *	Multiply two 3x4 transformation matrices.
operator *=	Perform compound assignment and multiplication by a 3x4 transformation matrix.
operator=	Assign one 3x4 transformation matrix to another.
operator[]	Subscripting operator to set or get a column.
operator[]	Subscripting operator to get a column.
rotation	Construct a 3x4 transformation matrix to rotate around a unit-length 3-D vector.
rotation	Construct a 3x4 transformation matrix to rotate around a unit-length 3-D vector (scalar data contained in vector data type).
rotation	Construct a rotation matrix from a unit-length quaternion.
rotationX	Construct a 3x4 transformation matrix to rotate around the x axis.
rotationX	Construct a 3x4 transformation matrix to rotate around the x axis (scalar data contained in vector data type).
rotationY	Construct a 3x4 transformation matrix to rotate around the y axis.

Methods	Description
rotationY	Construct a 3x4 transformation matrix to rotate around the y axis (scalar data contained in vector data type).
rotationZ	Construct a 3x4 transformation matrix to rotate around the z axis.
rotationZ	Construct a 3x4 transformation matrix to rotate around the z axis (scalar data contained in vector data type).
rotationZYX	Construct a 3x4 transformation matrix to rotate around the x, y, and z axes.
scale	Construct a 3x4 transformation matrix to perform scaling.
setCol	Set the column of a 3x4 transformation matrix referred to by the specified index.
setCol0	Set column 0 of a 3x4 transformation matrix.
setCol1	Set column 1 of a 3x4 transformation matrix.
setCol2	Set column 2 of a 3x4 transformation matrix.
setCol3	Set column 3 of a 3x4 transformation matrix.
setElem	Set the element of a 3x4 transformation matrix referred to by column and row indices.
setElem	Set the element of a 3x4 transformation matrix referred to by column and row indices (scalar data contained in vector data type).
setRow	Set the row of a 3x4 transformation matrix referred to by the specified index.
setTranslation	Set translation component.
setUpper3x3	Set the upper-left 3x3 submatrix.
Transform3	Default constructor; does no initialization.
Transform3	Copy a 3x4 transformation matrix.
Transform3	Construct a 3x4 transformation matrix containing the specified columns.
Transform3	Construct a 3x4 transformation matrix from a 3x3 matrix and a 3-D vector.
Transform3	Construct a 3x4 transformation matrix from a unit-length quaternion and a 3-D vector.
Transform3	Set all elements of a 3x4 transformation matrix to the same scalar value.
Transform3	Set all elements of a 3x4 transformation matrix to the same scalar value (scalar data contained in vector data type).
translation	Construct a 3x4 transformation matrix to perform translation.

Constructors and Destructors

Transform3

Default constructor; does no initialization.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline Transform3();
              }
        }
}
```

Arguments

None

Return Values

None

Description

Default constructor; does no initialization.

Transform3

Copy a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline Transform3(
                            const Transform3 &tfrm
                     );
              }
        }
}
```

Arguments

tfrm	3x4 transformation matrix

Return Values

None

Description

Construct a copy of a 3x4 transformation matrix.

Transform3

Construct a 3x4 transformation matrix containing the specified columns.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline Transform3(
                            Vector3 col0,
                            Vector3 col1,
                            Vector3 col2,
                            Vector3 col3
                     );
              }
        }
}
```

Arguments

col0	3-D vector
col1	3-D vector
col2	3-D vector
col3	3-D vector

Return Values

None

Description

Construct a 3x4 transformation matrix containing the specified columns.

Transform3

Construct a 3x4 transformation matrix from a 3x3 matrix and a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
                class Transform3 {
                        inline Transform3(
                                const Matrix3 &tfrm,
                                Vector3 translateVec
                        );
                }
        }
}
```

Arguments

tfrm	3x3 matrix
translateVec	3-D vector

Return Values

None

Description

Construct a 3x4 transformation matrix whose upper 3x3 elements are equal to the 3x3 matrix argument and whose translation component is equal to the 3-D vector argument.

Transform3

Construct a 3x4 transformation matrix from a unit-length quaternion and a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline Transform3(
                            Quat unitQuat,
                            Vector3 translateVec
                     );
              }
        }
}
```

Arguments

unitQuat	Quaternion, expected to be unit-length
translateVec	3-D vector

Return Values

None

Description

Construct a 3x4 transformation matrix whose upper-left 3x3 submatrix is a rotation matrix converted from the unit-length quaternion argument and whose translation component is equal to the 3-D vector argument.

Transform3

Set all elements of a 3x4 transformation matrix to the same scalar value.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     explicit inline Transform3(
                            float scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

None

Description

Construct a 3x4 transformation matrix with all elements set to the scalar value argument.

Transform3

Set all elements of a 3x4 transformation matrix to the same scalar value (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     explicit inline Transform3(
                            floatInVec scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)

Return Values

None

Description

Construct a 3x4 transformation matrix with all elements set to the scalar value argument.

Operator Methods

operator *

Multiply a 3x4 transformation matrix by a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline const Vector3 operator *(
                            Vector3 vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

Product of the specified 3x4 transformation matrix and 3-D vector

Description

Applies the 3x3 upper-left submatrix (but not the translation component) of a 3x4 transformation matrix to a 3-D vector.

operator *

Multiply a 3x4 transformation matrix by a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline const Point3 operator *(
                            Point3 pnt
                     );
              }
        }
}
```

Arguments

pnt	3-D point

Return Values

Product of the specified 3x4 transformation matrix and 3-D point

Description

Applies the 3x3 upper-left submatrix and the translation component of a 3x4 transformation matrix to a 3-D point.

operator *

Multiply two 3x4 transformation matrices.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline const Transform3 operator *(
                            const Transform3 &tfrm
                     );
              }
        }
}
```

Arguments

tfrm	3x4 transformation matrix

Return Values

Product of the specified 3x4 transformation matrices

Description

Multiply two 3x4 transformation matrices.

operator *=

Perform compound assignment and multiplication by a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline Transform3 &operator *=(
                            const Transform3 &tfrm
                     );
              }
        }
}
```

Arguments

tfrm	3x4 transformation matrix

Return Values

A reference to the resulting 3x4 transformation matrix

Description

Perform compound assignment and multiplication by a 3x4 transformation matrix.

operator=

Assign one 3x4 transformation matrix to another.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline Transform3 &operator=(
                            const Transform3 &tfrm
                     );
              }
        }
}
```

Arguments

tfrm	3x4 transformation matrix

Return Values

A reference to the resulting 3x4 transformation matrix

Description

Assign one 3x4 transformation matrix to another.

operator[]

Subscripting operator to set or get a column.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline Vector3 &operator[](
                            int col
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-3

Return Values

A reference to indexed column

Description

Subscripting operator invoked when applied to non-const Transform3.

operator[]

Subscripting operator to get a column.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline const Vector3 operator[](
                            int col
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-3

Return Values

Indexed column

Description

Subscripting operator invoked when applied to const Transform3.

Public Static Methods

identity

Construct an identity 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     static inline const Transform3 identity();
              }
        }
}
```

Arguments

None

Return Values

The constructed 3x4 transformation matrix

Description

Construct an identity 3x4 transformation matrix in which non-diagonal elements are zero and diagonal elements are 1.

rotation

Construct a 3x4 transformation matrix to rotate around a unit-length 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     static inline const Transform3 rotation(
                            float radians,
                            Vector3 unitVec
                     );
              }
        }
}
```

Arguments

radians	Scalar value
unitVec	3-D vector, expected to be unit-length

Return Values

The constructed 3x4 transformation matrix

Description

Construct a 3x4 transformation matrix to rotate around a unit-length 3-D vector by the specified radians angle.

rotation

Construct a 3x4 transformation matrix to rotate around a unit-length 3-D vector (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     static inline const Transform3 rotation(
                            floatInVec radians,
                            Vector3 unitVec
                     );
              }
        }
}
```

Arguments

radians	Scalar value (stored in vector data type)
unitVec	3-D vector, expected to be unit-length

Return Values

The constructed 3x4 transformation matrix

Description

Construct a 3x4 transformation matrix to rotate around a unit-length 3-D vector by the specified radians angle.

rotation

Construct a rotation matrix from a unit-length quaternion.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     static inline const Transform3 rotation(
                            Quat unitQuat
                     );
              }
        }
}
```

Arguments

unitQuat	Quaternion, expected to be unit-length

Return Values

The constructed 3x4 transformation matrix

Description

Construct a 3x4 transformation matrix that applies the same rotation as the specified unit-length quaternion.

rotationX

Construct a 3x4 transformation matrix to rotate around the x axis.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     static inline const Transform3 rotationX(
                            float radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value

Return Values

The constructed 3x4 transformation matrix

Description

Construct a 3x4 transformation matrix to rotate around the x axis by the specified radians angle.

rotationX

Construct a 3x4 transformation matrix to rotate around the x axis (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     static inline const Transform3 rotationX(
                            floatInVec radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value (stored in vector data type)

Return Values

The constructed 3x4 transformation matrix

Description

Construct a 3x4 transformation matrix to rotate around the x axis by the specified radians angle.

rotationY

Construct a 3x4 transformation matrix to rotate around the y axis.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     static inline const Transform3 rotationY(
                            float radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value

Return Values

The constructed 3x4 transformation matrix

Description

Construct a 3x4 transformation matrix to rotate around the y axis by the specified radians angle.

rotationY

Construct a 3x4 transformation matrix to rotate around the y axis (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     static inline const Transform3 rotationY(
                            floatInVec radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value (stored in vector data type)

Return Values

The constructed 3x4 transformation matrix

Description

Construct a 3x4 transformation matrix to rotate around the y axis by the specified radians angle.

rotationZ

Construct a 3x4 transformation matrix to rotate around the z axis.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     static inline const Transform3 rotationZ(
                            float radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value

Return Values

The constructed 3x4 transformation matrix

Description

Construct a 3x4 transformation matrix to rotate around the z axis by the specified radians angle.

rotationZ

Construct a 3x4 transformation matrix to rotate around the z axis (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     static inline const Transform3 rotationZ(
                            floatInVec radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value (stored in vector data type)

Return Values

The constructed 3x4 transformation matrix

Description

Construct a 3x4 transformation matrix to rotate around the z axis by the specified radians angle.

rotationZYX

Construct a 3x4 transformation matrix to rotate around the x, y, and z axes.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     static inline const Transform3 rotationZYX(
                            Vector3 radiansXYZ
                     );
              }
        }
}
```

Arguments

radiansXYZ	3-D vector

Return Values

The constructed 3x4 transformation matrix

Description

Construct a 3x4 transformation matrix to rotate around the x, y, and z axes by the radians angles contained in a 3-D vector. Equivalent to *rotationZ(radiansXYZ.getZ()) * rotationY(radiansXYZ.getY()) * rotationX(radiansXYZ.getX())*.

scale

Construct a 3x4 transformation matrix to perform scaling.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
             class Transform3 {
                    static inline const Transform3 scale(
                           Vector3 scaleVec
                    );
             }
        }
}
```

Arguments

scaleVec	3-D vector

Return Values

The constructed 3x4 transformation matrix

Description

Construct a 3x4 transformation matrix to perform scaling, in which the non-diagonal elements are zero and the diagonal elements are set to the elements of *scaleVec*.

translation

Construct a 3x4 transformation matrix to perform translation.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     static inline const Transform3 translation(
                            Vector3 translateVec
                     );
              }
        }
}
```

Arguments

translateVec	3-D vector

Return Values

The constructed 3x4 transformation matrix

Description

Construct a 3x4 transformation matrix to perform translation, which is an identity matrix except for the translation component, with coordinates equal to those in *translateVec*.

Public Instance Methods

getCol

Get the column of a 3x4 transformation matrix referred to by the specified index.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline const Vector3 getCol(
                            int col
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-3

Return Values

The column referred to by the specified index

Description

Get the column of a 3x4 transformation matrix referred to by the specified index.

getCol0

Get column 0 of a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline const Vector3 getCol0();
              }
        }
}
```

Arguments

None

Return Values

Column 0

Description

Get column 0 of a 3x4 transformation matrix.

getCol1

Get column 1 of a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline const Vector3 getCol1();
              }
        }
}
```

Arguments

None

Return Values

Column 1

Description

Get column 1 of a 3x4 transformation matrix.

getCol2

Get column 2 of a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline const Vector3 getCol2();
              }
        }
}
```

Arguments

None

Return Values

Column 2

Description

Get column 2 of a 3x4 transformation matrix.

getCol3

Get column 3 of a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline const Vector3 getCol3();
              }
        }
}
```

Arguments

None

Return Values

Column 3

Description

Get column 3 of a 3x4 transformation matrix.

getElem

Get the element of a 3x4 transformation matrix referred to by column and row indices.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline const floatInVec getElem(
                            int col,
                            int row
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-3
row	Index, expected in the range 0-2

Return Values

Element selected by *col* and *row*

Description

Get the element of a 3x4 transformation matrix referred to by column and row indices.

getRow

Get the row of a 3x4 transformation matrix referred to by the specified index.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline const Vector4 getRow(
                            int row
                     );
              }
        }
}
```

Arguments

row	Index, expected in the range 0-2

Return Values

The row referred to by the specified index

Description

Get the row of a 3x4 transformation matrix referred to by the specified index.

getTranslation

Get the translation component of a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline const Vector3 getTranslation();
              }
        }
}
```

Arguments

None

Return Values

Translation component

Description

Get the translation component of a 3x4 transformation matrix.

getUpper3x3

Get the upper-left 3x3 submatrix of a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline const Matrix3 getUpper3x3();
              }
        }
}
```

Arguments

None

Return Values

Upper-left 3x3 submatrix

Description

Get the upper-left 3x3 submatrix of a 3x4 transformation matrix.

setCol

Set the column of a 3x4 transformation matrix referred to by the specified index.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline Transform3 &setCol(
                            int col,
                            Vector3 vec
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-3
vec	3-D vector

Return Values

A reference to the resulting 3x4 transformation matrix

Description

Set the column of a 3x4 transformation matrix referred to by the specified index.

setCol0

Set column 0 of a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline Transform3 &setCol0(
                            Vector3 col0
                     );
              }
        }
}
```

Arguments

col0	3-D vector

Return Values

A reference to the resulting 3x4 transformation matrix

Description

Set column 0 of a 3x4 transformation matrix.

setCol1

Set column 1 of a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline Transform3 &setCol1(
                            Vector3 col1
                     );
              }
        }
}
```

Arguments

col1	3-D vector

Return Values

A reference to the resulting 3x4 transformation matrix

Description

Set column 1 of a 3x4 transformation matrix.

setCol2

Set column 2 of a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline Transform3 &setCol2(
                            Vector3 col2
                     );
              }
        }
}
```

Arguments

col2	3-D vector

Return Values

A reference to the resulting 3x4 transformation matrix

Description

Set column 2 of a 3x4 transformation matrix.

setCol3

Set column 3 of a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline Transform3 &setCol3(
                            Vector3 col3
                     );
              }
        }
}
```

Arguments

col3	3-D vector

Return Values

A reference to the resulting 3x4 transformation matrix

Description

Set column 3 of a 3x4 transformation matrix.

setElem

Set the element of a 3x4 transformation matrix referred to by column and row indices.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline Transform3 &setElem(
                            int col,
                            int row,
                            float val
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-3
row	Index, expected in the range 0-2
val	Scalar value

Return Values

A reference to the resulting 3x4 transformation matrix

Description

Set the element of a 3x4 transformation matrix referred to by column and row indices.

setElem

Set the element of a 3x4 transformation matrix referred to by column and row indices (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline Transform3 &setElem(
                            int col,
                            int row,
                            floatInVec val
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-3
row	Index, expected in the range 0-2
val	Scalar value (stored in vector data type)

Return Values

A reference to the resulting 3x4 transformation matrix

Description

Set the element of a 3x4 transformation matrix referred to by column and row indices.

setRow

Set the row of a 3x4 transformation matrix referred to by the specified index.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline Transform3 &setRow(
                            int row,
                            Vector4 vec
                     );
              }
        }
}
```

Arguments

row	Index, expected in the range 0-2
vec	4-D vector

Return Values

A reference to the resulting 3x4 transformation matrix

Description

Set the row of a 3x4 transformation matrix referred to by the specified index.

setTranslation

Set translation component.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline Transform3 &setTranslation(
                            Vector3 translateVec
                     );
              }
        }
}
```

Arguments

translateVec	3-D vector

Return Values

A reference to the resulting 3x4 transformation matrix

Description

Set the translation component of a 3x4 transformation matrix equal to the specified 3-D vector.

setUpper3x3

Set the upper-left 3x3 submatrix.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Transform3 {
                     inline Transform3 &setUpper3x3(
                            const Matrix3 &mat3
                     );
              }
        }
}
```

Arguments

mat3	3x3 matrix

Return Values

A reference to the resulting 3x4 transformation matrix

Description

Set the upper-left 3x3 submatrix elements of a 3x4 transformation matrix equal to the specified 3x3 matrix.

Vectormath::Aos::Vector3

Summary

Vectormath::Aos::Vector3

A 3-D vector in array-of-structures format.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
class Vector3;
```

Description

A class representing a 3-D vector stored in array-of-structures (AoS) format.

Methods Summary

Methods	Description
get128	Get vector float data from a 3-D vector.
getElem	Get an x, y, or z element of a 3-D vector by index.
getX	Get the x element of a 3-D vector.
getY	Get the y element of a 3-D vector.
getZ	Get the z element of a 3-D vector.
operator *	Multiply a 3-D vector by a scalar.
operator *	Multiply a 3-D vector by a scalar (scalar data contained in vector data type).
operator *=	Perform compound assignment and multiplication by a scalar.
operator *=	Perform compound assignment and multiplication by a scalar (scalar data contained in vector data type).
operator+	Add two 3-D vectors.
operator+	Add a 3-D vector to a 3-D point.
operator+=	Perform compound assignment and addition with a 3-D vector.
operator-	Subtract a 3-D vector from another 3-D vector.
operator-	Negate all elements of a 3-D vector.
operator-=	Perform compound assignment and subtraction by a 3-D vector.
operator/	Divide a 3-D vector by a scalar.
operator/	Divide a 3-D vector by a scalar (scalar data contained in vector data type).
operator/=	Perform compound assignment and division by a scalar.
operator/=	Perform compound assignment and division by a scalar (scalar data contained in vector data type).
operator=	Assign one 3-D vector to another.
operator[]	Subscripting operator to set or get an element.
operator[]	Subscripting operator to get an element.
setElem	Set an x, y, or z element of a 3-D vector by index.
setElem	Set an x, y, or z element of a 3-D vector by index (scalar data contained in vector data type).
setX	Set the x element of a 3-D vector.
setX	Set the x element of a 3-D vector (scalar data contained in vector data type).

Methods	Description
setY	Set the y element of a 3-D vector.
setY	Set the y element of a 3-D vector (scalar data contained in vector data type).
setZ	Set the z element of a 3-D vector.
setZ	Set the z element of a 3-D vector (scalar data contained in vector data type).
Vector3	Default constructor; does no initialization.
Vector3	Construct a 3-D vector from x, y, and z elements.
Vector3	Construct a 3-D vector from x, y, and z elements (scalar data contained in vector data type).
Vector3	Copy elements from a 3-D point into a 3-D vector.
Vector3	Set all elements of a 3-D vector to the same scalar value.
Vector3	Set all elements of a 3-D vector to the same scalar value (scalar data contained in vector data type).
Vector3	Set vector float data in a 3-D vector.
xAxis	Construct x axis.
yAxis	Construct y axis.
zAxis	Construct z axis.

Constructors and Destructors

Vector3

Default constructor; does no initialization.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline Vector3();
              }
        }
}
```

Arguments

None

Return Values

None

Description

Default constructor; does no initialization.

Vector3

Construct a 3-D vector from x, y, and z elements.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline Vector3(
                            float x,
                            float y,
                            float z
                     );
              }
        }
}
```

Arguments

x	Scalar value
y	Scalar value
z	Scalar value

Return Values

None

Description

Construct a 3-D vector containing the specified x, y, and z elements.

Vector3

Construct a 3-D vector from x, y, and z elements (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline Vector3(
                            floatInVec x,
                            floatInVec y,
                            floatInVec z
                     );
              }
        }
}
```

Arguments

x	Scalar value (stored in vector data type)
y	Scalar value (stored in vector data type)
z	Scalar value (stored in vector data type)

Return Values

None

Description

Construct a 3-D vector containing the specified x, y, and z elements.

Vector3

Copy elements from a 3-D point into a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     explicit inline Vector3(
                            Point3 pnt
                     );
              }
        }
}
```

Arguments

pnt	3-D point

Return Values

None

Description

Construct a 3-D vector containing the x, y, and z elements of the specified 3-D point.

Vector3

Set all elements of a 3-D vector to the same scalar value.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     explicit inline Vector3(
                            float scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

None

Description

Construct a 3-D vector with all elements set to the scalar value argument.

Vector3

Set all elements of a 3-D vector to the same scalar value (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     explicit inline Vector3(
                            floatInVec scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)

Return Values

None

Description

Construct a 3-D vector with all elements set to the scalar value argument.

Vector3

Set vector float data in a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     explicit inline Vector3(
                            vec_float4 vf4
                     );
              }
        }
}
```

Arguments

vf4	Scalar value

Return Values

None

Description

Construct a 3-D vector whose internal vector float data is set to the vector float argument.

Operator Methods

operator *

Multiply a 3-D vector by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline const Vector3 operator *(
                            float scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

Product of the specified 3-D vector and scalar

Description

Multiply a 3-D vector by a scalar.

operator *

Multiply a 3-D vector by a scalar (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline const Vector3 operator *(
                            floatInVec scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)

Return Values

Product of the specified 3-D vector and scalar

Description

Multiply a 3-D vector by a scalar.

operator *=

Perform compound assignment and multiplication by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline Vector3 &operator *=(
                            float scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

A reference to the resulting 3-D vector

Description

Perform compound assignment and multiplication by a scalar.

operator *=

Perform compound assignment and multiplication by a scalar (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
             class Vector3 {
                    inline Vector3 &operator *=(
                          floatInVec scalar
                    );
             }
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)

Return Values

A reference to the resulting 3-D vector

Description

Perform compound assignment and multiplication by a scalar.

operator+

Add two 3-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline const Vector3 operator+(
                            Vector3 vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

Sum of the specified 3-D vectors

Description

Add two 3-D vectors.

operator+

Add a 3-D vector to a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline const Point3 operator+(
                           Point3 pnt
                     );
              }
        }
}
```

Arguments

pnt	3-D point

Return Values

Sum of the specified 3-D vector and 3-D point

Description

Add a 3-D vector to a 3-D point.

operator+=

Perform compound assignment and addition with a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline Vector3 &operator+=(
                            Vector3 vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

A reference to the resulting 3-D vector

Description

Perform compound assignment and addition with a 3-D vector.

operator-

Subtract a 3-D vector from another 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline const Vector3 operator-(
                            Vector3 vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

Difference of the specified 3-D vectors

Description

Subtract a 3-D vector from another 3-D vector.

operator-

Negate all elements of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline const Vector3 operator-();
              }
        }
}
```

Arguments

None

Return Values

3-D vector containing negated elements of the specified 3-D vector

Description

Negate all elements of a 3-D vector.

operator-=

Perform compound assignment and subtraction by a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline Vector3 &operator-=(
                            Vector3 vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

A reference to the resulting 3-D vector

Description

Perform compound assignment and subtraction by a 3-D vector.

operator/

Divide a 3-D vector by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline const Vector3 operator/(
                            float scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

Quotient of the specified 3-D vector and scalar

Description

Divide a 3-D vector by a scalar.

operator/

Divide a 3-D vector by a scalar (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline const Vector3 operator/(
                            floatInVec scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)

Return Values

Quotient of the specified 3-D vector and scalar

Description

Divide a 3-D vector by a scalar.

operator/=

Perform compound assignment and division by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline Vector3 &operator/=(
                            float scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

A reference to the resulting 3-D vector

Description

Perform compound assignment and division by a scalar.

operator/=

Perform compound assignment and division by a scalar (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline Vector3 &operator/=(
                            floatInVec scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)

Return Values

A reference to the resulting 3-D vector

Description

Perform compound assignment and division by a scalar.

operator=

Assign one 3-D vector to another.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline Vector3 &operator=(
                            Vector3 vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

A reference to the resulting 3-D vector

Description

Assign one 3-D vector to another.

operator[]

Subscripting operator to set or get an element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline VecIdx operator[](
                            int idx
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-2

Return Values

VecIdx which holds a reference to the selected element

Description

Subscripting operator invoked when applied to non-const Vector3.

operator[]

Subscripting operator to get an element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline const floatInVec operator[](
                            int idx
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-2

Return Values

Indexed element

Description

Subscripting operator invoked when applied to const Vector3.

Public Static Methods

xAxis

Construct x axis.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     static inline const Vector3 xAxis();
              }
        }
}
```

Arguments

None

Return Values

The constructed 3-D vector

Description

Construct a 3-D vector equal to (1,0,0).

yAxis

Construct y axis.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     static inline const Vector3 yAxis();
              }
        }
}
```

Arguments

None

Return Values

The constructed 3-D vector

Description

Construct a 3-D vector equal to (0,1,0).

zAxis

Construct z axis.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     static inline const Vector3 zAxis();
              }
        }
}
```

Arguments

None

Return Values

The constructed 3-D vector

Description

Construct a 3-D vector equal to (0,0,1).

Public Instance Methods

get128

Get vector float data from a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline vec_float4 get128();
              }
        }
}
```

Arguments

None

Return Values

Internal vector float data

Description

Get internal vector float data from a 3-D vector.

getElem

Get an x, y, or z element of a 3-D vector by index.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline const floatInVec getElem(
                            int idx
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-2

Return Values

Element selected by the specified index

Description

Get an x, y, or z element of a 3-D vector by specifying an index of 0, 1, or 2, respectively.

getX

Get the x element of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline const floatInVec getX();
              }
        }
}
```

Arguments

None

Return Values

x element of a 3-D vector

Description

Get the x element of a 3-D vector.

getY

Get the y element of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline const floatInVec getY();
              }
        }
}
```

Arguments

None

Return Values

y element of a 3-D vector

Description

Get the y element of a 3-D vector.

getZ

Get the z element of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline const floatInVec getZ();
              }
        }
}
```

Arguments

None

Return Values

z element of a 3-D vector

Description

Get the z element of a 3-D vector.

setElem

Set an x, y, or z element of a 3-D vector by index.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline Vector3 &setElem(
                            int idx,
                            float value
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-2
value	Scalar value

Return Values

A reference to the resulting 3-D vector

Description

Set an x, y, or z element of a 3-D vector by specifying an index of 0, 1, or 2, respectively.

setElem

Set an x, y, or z element of a 3-D vector by index (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline Vector3 &setElem(
                            int idx,
                            floatInVec value
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-2
value	Scalar value (stored in vector data type)

Return Values

A reference to the resulting 3-D vector

Description

Set an x, y, or z element of a 3-D vector by specifying an index of 0, 1, or 2, respectively.

setX

Set the x element of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline Vector3 &setX(
                            float x
                     );
              }
        }
}
```

Arguments

x	Scalar value

Return Values

A reference to the resulting 3-D vector

Description

Set the x element of a 3-D vector to the specified scalar value.

setX

Set the x element of a 3-D vector (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline Vector3 &setX(
                            floatInVec x
                     );
              }
        }
}
```

Arguments

x	Scalar value (stored in vector data type)

Return Values

A reference to the resulting 3-D vector

Description

Set the x element of a 3-D vector to the specified scalar value.

setY

Set the y element of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline Vector3 &setY(
                            float y
                     );
              }
        }
}
```

Arguments

y	Scalar value

Return Values

A reference to the resulting 3-D vector

Description

Set the y element of a 3-D vector to the specified scalar value.

setY

Set the y element of a 3-D vector (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline Vector3 &setY(
                            floatInVec y
                     );
              }
        }
}
```

Arguments

y	Scalar value (stored in vector data type)

Return Values

A reference to the resulting 3-D vector

Description

Set the y element of a 3-D vector to the specified scalar value.

setZ

Set the z element of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline Vector3 &setZ(
                            float z
                     );
              }
        }
}
```

Arguments

z	Scalar value

Return Values

A reference to the resulting 3-D vector

Description

Set the z element of a 3-D vector to the specified scalar value.

setZ

Set the z element of a 3-D vector (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector3 {
                     inline Vector3 &setZ(
                            floatInVec z
                     );
              }
        }
}
```

Arguments

z	Scalar value (stored in vector data type)

Return Values

A reference to the resulting 3-D vector

Description

Set the z element of a 3-D vector to the specified scalar value.

Vectormath::Aos::Vector4

Summary

Vectormath::Aos::Vector4

A 4-D vector in array-of-structures format.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
class Vector4;
```

Description

A class representing a 4-D vector stored in array-of-structures (AoS) format.

Methods Summary

Methods	Description
get128	Get vector float data from a 4-D vector.
getElem	Get an x, y, z, or w element of a 4-D vector by index.
getW	Get the w element of a 4-D vector.
getX	Get the x element of a 4-D vector.
getXYZ	Get the x, y, and z elements of a 4-D vector.
getY	Get the y element of a 4-D vector.
getZ	Get the z element of a 4-D vector.
operator *	Multiply a 4-D vector by a scalar.
operator *	Multiply a 4-D vector by a scalar (scalar data contained in vector data type).
operator *=	Perform compound assignment and multiplication by a scalar.
operator *=	Perform compound assignment and multiplication by a scalar (scalar data contained in vector data type).
operator+	Add two 4-D vectors.
operator+=	Perform compound assignment and addition with a 4-D vector.
operator-	Subtract a 4-D vector from another 4-D vector.
operator-	Negate all elements of a 4-D vector.
operator-=	Perform compound assignment and subtraction by a 4-D vector.
operator/	Divide a 4-D vector by a scalar.
operator/	Divide a 4-D vector by a scalar (scalar data contained in vector data type).
operator/=	Perform compound assignment and division by a scalar.
operator/=	Perform compound assignment and division by a scalar (scalar data contained in vector data type).
operator=	Assign one 4-D vector to another.
operator[]	Subscripting operator to set or get an element.
operator[]	Subscripting operator to get an element.
setElem	Set an x, y, z, or w element of a 4-D vector by index.
setElem	Set an x, y, z, or w element of a 4-D vector by index (scalar data contained in vector data type).
setW	Set the w element of a 4-D vector.

Methods	Description
setW	Set the w element of a 4-D vector (scalar data contained in vector data type).
setX	Set the x element of a 4-D vector.
setX	Set the x element of a 4-D vector (scalar data contained in vector data type).
setXYZ	Set the x, y, and z elements of a 4-D vector.
setY	Set the y element of a 4-D vector.
setY	Set the y element of a 4-D vector (scalar data contained in vector data type).
setZ	Set the z element of a 4-D vector.
setZ	Set the z element of a 4-D vector (scalar data contained in vector data type).
Vector4	Default constructor; does no initialization.
Vector4	Construct a 4-D vector from x, y, z, and w elements.
Vector4	Construct a 4-D vector from x, y, z, and w elements (scalar data contained in vector data type).
Vector4	Construct a 4-D vector from a 3-D vector and a scalar.
Vector4	Construct a 4-D vector from a 3-D vector and a scalar (scalar data contained in vector data type).
Vector4	Copy x, y, and z from a 3-D vector into a 4-D vector, and set w to 0.
Vector4	Copy x, y, and z from a 3-D point into a 4-D vector, and set w to 1.
Vector4	Copy elements from a quaternion into a 4-D vector.
Vector4	Set all elements of a 4-D vector to the same scalar value.
Vector4	Set all elements of a 4-D vector to the same scalar value (scalar data contained in vector data type).
Vector4	Set vector float data in a 4-D vector.
wAxis	Construct w axis.
xAxis	Construct x axis.
yAxis	Construct y axis.
zAxis	Construct z axis.

Constructors and Destructors

Vector4

Default constructor; does no initialization.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline Vector4();
              }
        }
}
```

Arguments

None

Return Values

None

Description

Default constructor; does no initialization.

Vector4

Construct a 4-D vector from x, y, z, and w elements.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline Vector4(
                            float x,
                            float y,
                            float z,
                            float w
                     );
              }
        }
}
```

Arguments

x	Scalar value
y	Scalar value
z	Scalar value
w	Scalar value

Return Values

None

Description

Construct a 4-D vector containing the specified x, y, z, and w elements.

Vector4

Construct a 4-D vector from x, y, z, and w elements (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline Vector4(
                            floatInVec x,
                            floatInVec y,
                            floatInVec z,
                            floatInVec w
                     );
              }
        }
}
```

Arguments

x	Scalar value (stored in vector data type)
y	Scalar value (stored in vector data type)
z	Scalar value (stored in vector data type)
w	Scalar value (stored in vector data type)

Return Values

None

Description

Construct a 4-D vector containing the specified x, y, z, and w elements.

Vector4

Construct a 4-D vector from a 3-D vector and a scalar.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline Vector4(
                            Vector3 xyz,
                            float w
                     );
              }
        }
}
```

Arguments

xyz	3-D vector
w	Scalar value

Return Values

None

Description

Construct a 4-D vector with the x, y, and z elements of the specified 3-D vector and with the w element set to the specified scalar.

Vector4

Construct a 4-D vector from a 3-D vector and a scalar (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline Vector4(
                            Vector3 xyz,
                            floatInVec w
                     );
              }
        }
}
```

Arguments

xyz	3-D vector
w	Scalar value (stored in vector data type)

Return Values

None

Description

Construct a 4-D vector with the x, y, and z elements of the specified 3-D vector and with the w element set to the specified scalar.

Vector4

Copy x, y, and z from a 3-D vector into a 4-D vector, and set w to 0.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     explicit inline Vector4(
                            Vector3 vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

None

Description

Construct a 4-D vector with the x, y, and z elements of the specified 3-D vector and with the w element set to 0.

Vector4

Copy x, y, and z from a 3-D point into a 4-D vector, and set w to 1.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     explicit inline Vector4(
                            Point3 pnt
                     );
              }
        }
}
```

Arguments

pnt	3-D point

Return Values

None

Description

Construct a 4-D vector with the x, y, and z elements of the specified 3-D point and with the w element set to 1.

Vector4

Copy elements from a quaternion into a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     explicit inline Vector4(
                            Quat quat
                     );
              }
        }
}
```

Arguments

quat	Quaternion

Return Values

None

Description

Construct a 4-D vector containing the x, y, z, and w elements of the specified quaternion.

Vector4

Set all elements of a 4-D vector to the same scalar value.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     explicit inline Vector4(
                            float scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

None

Description

Construct a 4-D vector with all elements set to the scalar value argument.

Vector4

Set all elements of a 4-D vector to the same scalar value (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     explicit inline Vector4(
                            floatInVec scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)

Return Values

None

Description

Construct a 4-D vector with all elements set to the scalar value argument.

Vector4

Set vector float data in a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
             class Vector4 {
                    explicit inline Vector4(
                           vec_float4 vf4
                    );
             }
        }
}
```

Arguments

vf4	Scalar value

Return Values

None

Description

Construct a 4-D vector whose internal vector float data is set to the vector float argument.

Operator Methods

operator *

Multiply a 4-D vector by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline const Vector4 operator *(
                            float scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

Product of the specified 4-D vector and scalar

Description

Multiply a 4-D vector by a scalar.

operator *

Multiply a 4-D vector by a scalar (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline const Vector4 operator *(
                            floatInVec scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)

Return Values

Product of the specified 4-D vector and scalar

Description

Multiply a 4-D vector by a scalar.

operator *=

Perform compound assignment and multiplication by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline Vector4 &operator *=(
                            float scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

A reference to the resulting 4-D vector

Description

Perform compound assignment and multiplication by a scalar.

operator *=

Perform compound assignment and multiplication by a scalar (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline Vector4 &operator *=(
                            floatInVec scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)

Return Values

A reference to the resulting 4-D vector

Description

Perform compound assignment and multiplication by a scalar.

operator+

Add two 4-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline const Vector4 operator+(
                            Vector4 vec
                     );
              }
        }
}
```

Arguments

vec	4-D vector

Return Values

Sum of the specified 4-D vectors

Description

Add two 4-D vectors.

operator+=

Perform compound assignment and addition with a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline Vector4 &operator+=(
                            Vector4 vec
                     );
              }
        }
}
```

Arguments

vec	4-D vector

Return Values

A reference to the resulting 4-D vector

Description

Perform compound assignment and addition with a 4-D vector.

operator-

Subtract a 4-D vector from another 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline const Vector4 operator-(
                            Vector4 vec
                     );
              }
        }
}
```

Arguments

vec	4-D vector

Return Values

Difference of the specified 4-D vectors

Description

Subtract a 4-D vector from another 4-D vector.

operator-

Negate all elements of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline const Vector4 operator-();
              }
        }
}
```

Arguments

None

Return Values

4-D vector containing negated elements of the specified 4-D vector

Description

Negate all elements of a 4-D vector.

operator-=

Perform compound assignment and subtraction by a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline Vector4 &operator-=(
                            Vector4 vec
                     );
              }
        }
}
```

Arguments

vec	4-D vector

Return Values

A reference to the resulting 4-D vector

Description

Perform compound assignment and subtraction by a 4-D vector.

operator/

Divide a 4-D vector by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline const Vector4 operator/(
                            float scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

Quotient of the specified 4-D vector and scalar

Description

Divide a 4-D vector by a scalar.

operator/

Divide a 4-D vector by a scalar (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline const Vector4 operator/(
                            floatInVec scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)

Return Values

Quotient of the specified 4-D vector and scalar

Description

Divide a 4-D vector by a scalar.

operator/=

Perform compound assignment and division by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline Vector4 &operator/=(
                            float scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

A reference to the resulting 4-D vector

Description

Perform compound assignment and division by a scalar.

operator/=

Perform compound assignment and division by a scalar (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline Vector4 &operator/=(
                            floatInVec scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value (stored in vector data type)

Return Values

A reference to the resulting 4-D vector

Description

Perform compound assignment and division by a scalar.

operator=

Assign one 4-D vector to another.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline Vector4 &operator=(
                            Vector4 vec
                     );
              }
        }
}
```

Arguments

vec	4-D vector

Return Values

A reference to the resulting 4-D vector

Description

Assign one 4-D vector to another.

operator[]

Subscripting operator to set or get an element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline VecIdx operator[](
                            int idx
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-3

Return Values

VecIdx which holds a reference to the selected element

Description

Subscripting operator invoked when applied to non-const Vector4.

operator[]

Subscripting operator to get an element.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline const floatInVec operator[](
                            int idx
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-3

Return Values

Indexed element

Description

Subscripting operator invoked when applied to const Vector4.

Public Static Methods

wAxis

Construct w axis.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     static inline const Vector4 wAxis();
              }
        }
}
```

Arguments

None

Return Values

The constructed 4-D vector

Description

Construct a 4-D vector equal to (0,0,0,1).

xAxis

Construct x axis.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     static inline const Vector4 xAxis();
              }
        }
}
```

Arguments

None

Return Values

The constructed 4-D vector

Description

Construct a 4-D vector equal to (1,0,0,0).

yAxis

Construct y axis.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     static inline const Vector4 yAxis();
              }
        }
}
```

Arguments

None

Return Values

The constructed 4-D vector

Description

Construct a 4-D vector equal to (0,1,0,0).

zAxis

Construct z axis.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     static inline const Vector4 zAxis();
              }
        }
}
```

Arguments

None

Return Values

The constructed 4-D vector

Description

Construct a 4-D vector equal to (0,0,1,0).

Public Instance Methods

get128

Get vector float data from a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline vec_float4 get128();
              }
        }
}
```

Arguments

None

Return Values

Internal vector float data

Description

Get internal vector float data from a 4-D vector.

getElem

Get an x, y, z, or w element of a 4-D vector by index.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline const floatInVec getElem(
                            int idx
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-3

Return Values

Element selected by the specified index

Description

Get an x, y, z, or w element of a 4-D vector by specifying an index of 0, 1, 2, or 3, respectively.

getW

Get the w element of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline const floatInVec getW();
              }
        }
}
```

Arguments

None

Return Values

w element of a 4-D vector

Description

Get the w element of a 4-D vector.

getX

Get the x element of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline const floatInVec getX();
              }
        }
}
```

Arguments

None

Return Values

x element of a 4-D vector

Description

Get the x element of a 4-D vector.

getXYZ

Get the x, y, and z elements of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline const Vector3 getXYZ();
              }
        }
}
```

Arguments

None

Return Values

3-D vector containing x, y, and z elements

Description

Extract a 4-D vector's x, y, and z elements into a 3-D vector.

getY

Get the y element of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline const floatInVec getY();
              }
        }
}
```

Arguments

None

Return Values

y element of a 4-D vector

Description

Get the y element of a 4-D vector.

getZ

Get the z element of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline const floatInVec getZ();
              }
        }
}
```

Arguments

None

Return Values

z element of a 4-D vector

Description

Get the z element of a 4-D vector.

setElem

Set an x, y, z, or w element of a 4-D vector by index.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline Vector4 &setElem(
                            int idx,
                            float value
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-3
value	Scalar value

Return Values

A reference to the resulting 4-D vector

Description

Set an x, y, z, or w element of a 4-D vector by specifying an index of 0, 1, 2, or 3, respectively.

setElem

Set an x, y, z, or w element of a 4-D vector by index (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline Vector4 &setElem(
                            int idx,
                            floatInVec value
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-3
value	Scalar value (stored in vector data type)

Return Values

A reference to the resulting 4-D vector

Description

Set an x, y, z, or w element of a 4-D vector by specifying an index of 0, 1, 2, or 3, respectively.

setW

Set the w element of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline Vector4 &setW(
                            float w
                     );
              }
        }
}
```

Arguments

w	Scalar value

Return Values

A reference to the resulting 4-D vector

Description

Set the w element of a 4-D vector to the specified scalar value.

setW

Set the w element of a 4-D vector (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline Vector4 &setW(
                            floatInVec w
                     );
              }
        }
}
```

Arguments

w	Scalar value (stored in vector data type)

Return Values

A reference to the resulting 4-D vector

Description

Set the w element of a 4-D vector to the specified scalar value.

setX

Set the x element of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline Vector4 &setX(
                            float x
                     );
              }
        }
}
```

Arguments

x	Scalar value

Return Values

A reference to the resulting 4-D vector

Description

Set the x element of a 4-D vector to the specified scalar value.

setX

Set the x element of a 4-D vector (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline Vector4 &setX(
                            floatInVec x
                     );
              }
        }
}
```

Arguments

x	Scalar value (stored in vector data type)

Return Values

A reference to the resulting 4-D vector

Description

Set the x element of a 4-D vector to the specified scalar value.

setXYZ

Set the x, y, and z elements of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline Vector4 &setXYZ(
                            Vector3 vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

A reference to the resulting 4-D vector

Description

Set the x, y, and z elements to those of the specified 3-D vector.

Notes

This function does not change the w element.

setY

Set the y element of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline Vector4 &setY(
                            float y
                     );
              }
        }
}
```

Arguments

y	Scalar value

Return Values

A reference to the resulting 4-D vector

Description

Set the y element of a 4-D vector to the specified scalar value.

setY

Set the y element of a 4-D vector (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
             class Vector4 {
                    inline Vector4 &setY(
                           floatInVec y
                    );
             }
        }
}
```

Arguments

y	Scalar value (stored in vector data type)

Return Values

A reference to the resulting 4-D vector

Description

Set the y element of a 4-D vector to the specified scalar value.

setZ

Set the z element of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline Vector4 &setZ(
                            float z
                     );
              }
        }
}
```

Arguments

z	Scalar value

Return Values

A reference to the resulting 4-D vector

Description

Set the z element of a 4-D vector to the specified scalar value.

setZ

Set the z element of a 4-D vector (scalar data contained in vector data type).

Definition

```
#include <vectormath/cpp/vectormath_aos.h>
namespace Vectormath {
        namespace Aos {
              class Vector4 {
                     inline Vector4 &setZ(
                            floatInVec z
                     );
              }
        }
}
```

Arguments

z	Scalar value (stored in vector data type)

Return Values

A reference to the resulting 4-D vector

Description

Set the z element of a 4-D vector to the specified scalar value.

Vectormath::Soa

Summary

Vectormath::Soa

The namespace containing structure-of-arrays (SoA) classes.

Definition

```
namespace Soa {}
```

Description

The namespace containing structure-of-arrays (SoA) classes.

Function Summary

Function	Description
absPerElem	Compute the absolute value of a 3-D vector per element.
absPerElem	Compute the absolute value of a 4-D vector per element.
absPerElem	Compute the absolute value of a 3-D point per element.
absPerElem	Compute the absolute value of a 3x3 matrix per element.
absPerElem	Compute the absolute value of a 4x4 matrix per element.
absPerElem	Compute the absolute value of a 3x4 transformation matrix per element.
affineInverse	Compute the inverse of a 4x4 matrix, which is expected to be an affine matrix.
appendScale	Append (post-multiply) a scale transformation to a 3x3 matrix.
appendScale	Append (post-multiply) a scale transformation to a 4x4 matrix.
appendScale	Append (post-multiply) a scale transformation to a 3x4 transformation matrix.
conj	Compute the conjugate of a quaternion.
copySignPerElem	Copy sign from one 3-D vector to another, per element.
copySignPerElem	Copy sign from one 4-D vector to another, per element.
copySignPerElem	Copy sign from one 3-D point to another, per element.
cross	Compute cross product of two 3-D vectors.
crossMatrix	Cross-product matrix of a 3-D vector.
crossMatrixMul	Create cross-product matrix and multiply.
determinant	Determinant of a 3x3 matrix.
determinant	Determinant of a 4x4 matrix.
dist	Compute the distance between two 3-D points.
distFromOrigin	Compute the distance of a 3-D point from the coordinate-system origin.
distSqr	Compute the square of the distance between two 3-D points.
distSqrFromOrigin	Compute the square of the distance of a 3-D point from the coordinate-system origin.
divPerElem	Divide two 3-D vectors per element.
divPerElem	Divide two 4-D vectors per element.
divPerElem	Divide two 3-D points per element.
dot	Compute the dot product of two 3-D vectors.
dot	Compute the dot product of two 4-D vectors.

Function	Description
dot	Compute the dot product of two quaternions.
inverse	Compute the inverse of a 3x3 matrix.
inverse	Compute the inverse of a 4x4 matrix.
inverse	Inverse of a 3x4 transformation matrix.
length	Compute the length of a 3-D vector.
length	Compute the length of a 4-D vector.
length	Compute the length of a quaternion.
lengthSqr	Compute the square of the length of a 3-D vector.
lengthSqr	Compute the square of the length of a 4-D vector.
lerp	Linear interpolation between two 3-D vectors.
lerp	Linear interpolation between two 4-D vectors.
lerp	Linear interpolation between two 3-D points.
lerp	Linear interpolation between two quaternions.
loadXYZArray	Load four three-float 3-D vectors, stored in three quadwords.
loadXYZArray	Load four three-float 3-D points, stored in three quadwords.
maxElem	Maximum element of a 3-D vector.
maxElem	Maximum element of a 4-D vector.
maxElem	Maximum element of a 3-D point.
maxPerElem	Maximum of two 3-D vectors per element.
maxPerElem	Maximum of two 4-D vectors per element.
maxPerElem	Maximum of two 3-D points per element.
minElem	Minimum element of a 3-D vector.
minElem	Minimum element of a 4-D vector.
minElem	Minimum element of a 3-D point.
minPerElem	Minimum of two 3-D vectors per element.
minPerElem	Minimum of two 4-D vectors per element.
minPerElem	Minimum of two 3-D points per element.
mulPerElem	Multiply two 3-D vectors per element.
mulPerElem	Multiply two 4-D vectors per element.
mulPerElem	Multiply two 3-D points per element.
mulPerElem	Multiply two 3x3 matrices per element.
mulPerElem	Multiply two 4x4 matrices per element.
mulPerElem	Multiply two 3x4 transformation matrices per element.
norm	Compute the norm of a quaternion.
normalize	Normalize a 3-D vector.
normalize	Normalize a 4-D vector.
normalize	Normalize a quaternion.
operator *	Multiply a 3-D vector by a scalar.
operator *	Multiply a 4-D vector by a scalar.
operator *	Multiply a quaternion by a scalar.
operator *	Multiply a 3x3 matrix by a scalar.
operator *	Multiply a 4x4 matrix by a scalar.
orthoInverse	Compute the inverse of a 4x4 matrix, which is expected to be an affine matrix with an orthogonal upper-left 3x3 submatrix.
orthoInverse	Compute the inverse of a 3x4 transformation matrix, expected to have an orthogonal upper-left 3x3 submatrix.
outer	Outer product of two 3-D vectors.
outer	Outer product of two 4-D vectors.
prependScale	Prepend (pre-multiply) a scale transformation to a 3x3 matrix.

Function	Description
prependScale	Prepend (pre-multiply) a scale transformation to a 4x4 matrix.
prependScale	Prepend (pre-multiply) a scale transformation to a 3x4 transformation matrix.
print	Print a 3-D vector.
print	Print a 3-D vector and an associated string identifier.
print	Print a 4-D vector.
print	Print a 4-D vector and an associated string identifier.
print	Print a 3-D point.
print	Print a 3-D point and an associated string identifier.
print	Print a quaternion.
print	Print a quaternion and an associated string identifier.
print	Print a 3x3 matrix.
print	Print a 3x3 matrix and an associated string identifier.
print	Print a 4x4 matrix.
print	Print a 4x4 matrix and an associated string identifier.
print	Print a 3x4 transformation matrix.
print	Print a 3x4 transformation matrix and an associated string identifier.
projection	Scalar projection of a 3-D point on a unit-length 3-D vector.
recipPerElem	Compute the reciprocal of a 3-D vector per element.
recipPerElem	Compute the reciprocal of a 4-D vector per element.
recipPerElem	Compute the reciprocal of a 3-D point per element.
rotate	Use a unit-length quaternion to rotate a 3-D vector.
rowMul	Pre-multiply a row vector by a 3x3 matrix.
rsqrtPerElem	Compute the reciprocal square root of a 3-D vector per element.
rsqrtPerElem	Compute the reciprocal square root of a 4-D vector per element.
rsqrtPerElem	Compute the reciprocal square root of a 3-D point per element.
scale	Apply uniform scale to a 3-D point.
scale	Apply non-uniform scale to a 3-D point.
select	Conditionally select between two 3-D vectors.
select	Conditionally select between two 4-D vectors.
select	Conditionally select between two 3-D points.
select	Conditionally select between two quaternions.
select	Conditionally select between two 3x3 matrices.
select	Conditionally select between two 4x4 matrices.
select	Conditionally select between two 3x4 transformation matrices.
slerp	Spherical linear interpolation between two 3-D vectors.
slerp	Spherical linear interpolation between two 4-D vectors.
slerp	Spherical linear interpolation between two quaternions.
sqrtPerElem	Compute the square root of a 3-D vector per element.
sqrtPerElem	Compute the square root of a 4-D vector per element.
sqrtPerElem	Compute the square root of a 3-D point per element.
squad	Spherical quadrangle interpolation.
storeHalfFloats	Store eight slots of two SoA 3-D vectors as half-floats.
storeHalfFloats	Store four slots of an SoA 4-D vector as half-floats.
storeHalfFloats	Store eight slots of two SoA 3-D points as half-floats.
storeXYZArray	Store four slots of an SoA 3-D vector in three quadwords.
storeXYZArray	Store four slots of an SoA 3-D point in three quadwords.

Function	Description
sum	Compute the sum of all elements of a 3-D vector.
sum	Compute the sum of all elements of a 4-D vector.
sum	Compute the sum of all elements of a 3-D point.
transpose	Transpose of a 3x3 matrix.
transpose	Transpose of a 4x4 matrix.

Inner Classes, Structures, and Namespaces

Item	Description
Vectormath::Soa::Matrix3	A set of four 3x3 matrices in structure-of-arrays format.
Vectormath::Soa::Matrix4	A set of four 4x4 matrices in structure-of-arrays format.
Vectormath::Soa::Point3	A set of four 3-D points in structure-of-arrays format.
Vectormath::Soa::Quat	A set of four quaternions in structure-of-arrays format.
Vectormath::Soa::Transform3	A set of four 3x4 transformation matrices in structure-of-arrays format.
Vectormath::Soa::Vector3	A set of four 3-D vectors in structure-of-arrays format.
Vectormath::Soa::Vector4	A set of four 4-D vectors in structure-of-arrays format.

3-D Vector Functions

absPerElem

Compute the absolute value of a 3-D vector per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector3 absPerElem(
                     const Vector3 &vec
              );
        }
}
```

Arguments

vec	3-D vector

Return Values

3-D vector in which each element is the absolute value of the corresponding element of vec

Description

Compute the absolute value of each element of a 3-D vector.

copySignPerElem

Copy sign from one 3-D vector to another, per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector3 copySignPerElem(
                     const Vector3 &vec0,
                     const Vector3 &vec1
              );
        }
}
```

Arguments

vec0	3-D vector
vec1	3-D vector

Return Values

3-D vector in which each element has the magnitude of the corresponding element of *vec0* and the sign of the corresponding element of *vec1*

Description

For each element, create a value composed of the magnitude of *vec0* and the sign of *vec1*.

cross

Compute cross product of two 3-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector3 cross(
                     const Vector3 &vec0,
                     const Vector3 &vec1
              );
        }
}
```

Arguments

vec0	3-D vector
vec1	3-D vector

Return Values

Cross product of the specified 3-D vectors

Description

Compute cross product of two 3-D vectors.

crossMatrix

Cross-product matrix of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Matrix3 crossMatrix(
                     const Vector3 &vec
              );
        }
}
```

Arguments

vec	3-D vector

Return Values

Cross-product matrix of *vec*

Description

Compute a matrix that, when multiplied by a 3-D vector, produces the same result as a cross product with that 3-D vector.

crossMatrixMul

Create cross-product matrix and multiply.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Matrix3 crossMatrixMul(
                     const Vector3 &vec,
                     const Matrix3 &mat
              );
        }
}
```

Arguments

vec	3-D vector
mat	3x3 matrix

Return Values

Product of cross-product matrix of *vec* and *mat*

Description

Multiply a cross-product matrix by another matrix.

Notes

Faster than separately creating a cross-product matrix and multiplying.

divPerElem

Divide two 3-D vectors per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector3 divPerElem(
                     const Vector3 &vec0,
                     const Vector3 &vec1
              );
        }
}
```

Arguments

vec0	3-D vector
vec1	3-D vector

Return Values

3-D vector in which each element is the quotient of the corresponding elements of the specified 3-D vectors

Description

Divide two 3-D vectors element by element.

Notes

Floating-point behavior matches standard library function divf4.

dot

Compute the dot product of two 3-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline vec_float4 dot(
                     const Vector3 &vec0,
                     const Vector3 &vec1
              );
        }
}
```

Arguments

vec0	3-D vector
vec1	3-D vector

Return Values

Dot product of the specified 3-D vectors

Description

Compute the dot product of two 3-D vectors.

length

Compute the length of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline vec_float4 length(
                     const Vector3 &vec
              );
        }
}
```

Arguments

vec	3-D vector

Return Values

Length of the specified 3-D vector

Description

Compute the length of a 3-D vector.

lengthSqr

Compute the square of the length of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline vec_float4 lengthSqr(
                     const Vector3 &vec
              );
        }
}
```

Arguments

vec	3-D vector

Return Values

Square of the length of the specified 3-D vector

Description

Compute the square of the length of a 3-D vector.

lerp

Linear interpolation between two 3-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector3 lerp(
                     vec_float4 t,
                     const Vector3 &vec0,
                     const Vector3 &vec1
              );
        }
}
```

Arguments

t	Interpolation parameter
vec0	3-D vector
vec1	3-D vector

Return Values

Interpolated 3-D vector

Description

Linearly interpolate between two 3-D vectors.

Notes

Does not clamp `t` between 0 and 1.

loadXYZArray

Load four three-float 3-D vectors, stored in three quadwords.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline void loadXYZArray(
                     Vector3 &vec,
                     const vec_float4 *threeQuads
              );
        }
}
```

Arguments

vec	An output 3-D vector
threeQuads	Array of 3 quadwords containing 12 floats

Return Values

None

Description

Load four three-float 3-D vectors, stored in three quadwords as {x0,y0,z0,x1,y1,z1,x2,y2,z2,x3,y3,z3}, into four slots of an SoA 3-D vector.

maxElem

Maximum element of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline vec_float4 maxElem(
                     const Vector3 &vec
              );
        }
}
```

Arguments

vec	3-D vector

Return Values

Maximum value of all elements of *vec*

Description

Compute the maximum value of all elements of a 3-D vector.

maxPerElem

Maximum of two 3-D vectors per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector3 maxPerElem(
                     const Vector3 &vec0,
                     const Vector3 &vec1
              );
        }
}
```

Arguments

vec0	3-D vector
vec1	3-D vector

Return Values

3-D vector in which each element is the maximum of the corresponding elements of the specified 3-D vectors

Description

Create a 3-D vector in which each element is the maximum of the corresponding elements of the specified 3-D vectors.

minElem

Minimum element of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline vec_float4 minElem(
                     const Vector3 &vec
              );
        }
}
```

Arguments

vec	3-D vector

Return Values

Minimum value of all elements of *vec*

Description

Compute the minimum value of all elements of a 3-D vector.

minPerElem

Minimum of two 3-D vectors per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector3 minPerElem(
                     const Vector3 &vec0,
                     const Vector3 &vec1
              );
        }
}
```

Arguments

vec0	3-D vector
vec1	3-D vector

Return Values

3-D vector in which each element is the minimum of the corresponding elements of the specified 3-D vectors

Description

Create a 3-D vector in which each element is the minimum of the corresponding elements of two specified 3-D vectors.

mulPerElem

Multiply two 3-D vectors per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector3 mulPerElem(
                     const Vector3 &vec0,
                     const Vector3 &vec1
              );
        }
}
```

Arguments

vec0	3-D vector
vec1	3-D vector

Return Values

3-D vector in which each element is the product of the corresponding elements of the specified 3-D vectors

Description

Multiply two 3-D vectors element by element.

normalize

Normalize a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector3 normalize(
                     const Vector3 &vec
              );
        }
}
```

Arguments

vec	3-D vector

Return Values

The specified 3-D vector scaled to unit length

Description

Compute a normalized 3-D vector.

Notes

The result is unpredictable when all elements of vec are at or near zero.

operator *

Multiply a 3-D vector by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector3 operator *(
                     vec_float4 scalar,
                     const Vector3 &vec
              );
        }
}
```

Arguments

scalar	Scalar value
vec	3-D vector

Return Values

Scalar product of *vec* and *scalar*

Description

Multiply a 3-D vector by a scalar.

outer

Outer product of two 3-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Matrix3 outer(
                     const Vector3 &vec0,
                     const Vector3 &vec1
              );
        }
}
```

Arguments

vec0	3-D vector
vec1	3-D vector

Return Values

The 3x3 matrix product of a column-vector, *vec0*, and a row-vector, *vec1*

Description

Compute the outer product of two 3-D vectors.

print

Print a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline void print(
                     const Vector3 &vec
              );
        }
}
```

Arguments

vec	3-D vector

Return Values

None

Description

Print a 3-D vector. Prints the 3-D vector transposed, that is, as a row instead of a column.

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

print

Print a 3-D vector and an associated string identifier.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline void print(
                     const Vector3 &vec,
                     const char *name
              );
        }
}
```

Arguments

vec	3-D vector
name	String printed with the 3-D vector

Return Values

None

Description

Print a 3-D vector and an associated string identifier. Prints the 3-D vector transposed, that is, as a row instead of a column.

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

recipPerElem

Compute the reciprocal of a 3-D vector per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector3 recipPerElem(
                     const Vector3 &vec
              );
        }
}
```

Arguments

vec	3-D vector

Return Values

3-D vector in which each element is the reciprocal of the corresponding element of the specified 3-D vector

Description

Create a 3-D vector in which each element is the reciprocal of the corresponding element of the specified 3-D vector.

Notes

Floating-point behavior matches standard library function recipf4.

rowMul

Pre-multiply a row vector by a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector3 rowMul(
                     const Vector3 &vec,
                     const Matrix3 &mat
              );
        }
}
```

Arguments

vec	3-D vector
mat	3x3 matrix

Return Values

Product of a row-vector and a 3x3 matrix

Description

Transpose a 3-D vector into a row vector and pre-multiply by 3x3 matrix.

rsqrtPerElem

Compute the reciprocal square root of a 3-D vector per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector3 rsqrtPerElem(
                     const Vector3 &vec
              );
        }
}
```

Arguments

vec	3-D vector

Return Values

3-D vector in which each element is the reciprocal square root of the corresponding element of the specified 3-D vector

Description

Create a 3-D vector in which each element is the reciprocal square root of the corresponding element of the specified 3-D vector.

Notes

Floating-point behavior matches standard library function rsqrtf4.

select

Conditionally select between two 3-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector3 select(
                     const Vector3 &vec0,
                     const Vector3 &vec1,
                     vec_uint4 select1
              );
        }
}
```

Arguments

vec0	3-D vector
vec1	3-D vector
select1	For each of the four word slots, this mask selects either the 3-D vector in the corresponding slot of *vec0* or the 3-D vector in the corresponding slot of *vec1*. A 0 bit selects from *vec0* whereas a 1 bit selects from *vec1*. Identical bits should be set for each word of the mask.

Return Values

Each slot of the result is equal to the 3-D vector at the corresponding slot of *vec0* or *vec1*, depending on the value of *select1* at the corresponding slot. A value of 0 selects the slot of *vec0*, and a value of 0xFFFFFFFF selects the slot of *vec1*

Description

Conditionally select one of the 3-D vectors at each of the corresponding slots of *vec0* or *vec1*.

Notes

This function uses a conditional select instruction to avoid a branch.

slerp

Spherical linear interpolation between two 3-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector3 slerp(
                     vec_float4 t,
                     const Vector3 &unitVec0,
                     const Vector3 &unitVec1
              );
        }
}
```

Arguments

t	Interpolation parameter
unitVec0	3-D vector, expected to be unit-length
unitVec1	3-D vector, expected to be unit-length

Return Values

Interpolated 3-D vector

Description

Perform spherical linear interpolation between two 3-D vectors.

Notes

The result is unpredictable if the vectors point in opposite directions. Does not clamp *t* between 0 and 1.

sqrtPerElem

Compute the square root of a 3-D vector per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector3 sqrtPerElem(
                     const Vector3 &vec
              );
        }
}
```

Arguments

vec	3-D vector

Return Values

3-D vector in which each element is the square root of the corresponding element of the specified 3-D vector

Description

Create a 3-D vector in which each element is the square root of the corresponding element of the specified 3-D vector.

Notes

Floating-point behavior matches standard library function sqrtf4.

storeHalfFloats

Store eight slots of two SoA 3-D vectors as half-floats.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline void storeHalfFloats(
                     const Vector3 &vec0,
                     const Vector3 &vec1,
                     vec_ushort8 *threeQuads
              );
        }
}
```

Arguments

vec0	3-D vector
vec1	3-D vector
threeQuads	An output array of 3 quadwords containing 24 half-floats

Return Values

None

Description

Store eight slots of two SoA 3-D vectors in three quadwords of half-float values. Numbering slots of *vec0* as 0..3 and slots of *vec1* as 4..7, the output is {x0,y0,z0,x1,y1,z1,x2,y2,z2,x3,y3,z3,x4,y4,z4,x5,y5,z5,x6,y6,z6,x7,y7,z7}.

storeXYZArray

Store four slots of an SoA 3-D vector in three quadwords.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline void storeXYZArray(
                     const Vector3 &vec,
                     vec_float4 *threeQuads
              );
        }
}
```

Arguments

vec	3-D vector
threeQuads	An output array of 3 quadwords containing 12 floats

Return Values

None

Description

Store four slots of an SoA 3-D vector in three quadwords as {x0,y0,z0,x1,y1,z1,x2,y2,z2,x3,y3,z3}.

sum

Compute the sum of all elements of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline vec_float4 sum(
                     const Vector3 &vec
              );
        }
}
```

Arguments

vec	3-D vector

Return Values

Sum of all elements of *vec*

Description

Compute the sum of all elements of a 3-D vector.

4-D Vector Functions

absPerElem

Compute the absolute value of a 4-D vector per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector4 absPerElem(
                     const Vector4 &vec
              );
        }
}
```

Arguments

vec	4-D vector

Return Values

4-D vector in which each element is the absolute value of the corresponding element of vec

Description

Compute the absolute value of each element of a 4-D vector.

copySignPerElem

Copy sign from one 4-D vector to another, per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
            inline const Vector4 copySignPerElem(
                   const Vector4 &vec0,
                   const Vector4 &vec1
            );
        }
}
```

Arguments

vec0	4-D vector
vec1	4-D vector

Return Values

4-D vector in which each element has the magnitude of the corresponding element of *vec0* and the sign of the corresponding element of *vec1*

Description

For each element, create a value composed of the magnitude of *vec0* and the sign of *vec1*.

divPerElem

Divide two 4-D vectors per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector4 divPerElem(
                     const Vector4 &vec0,
                     const Vector4 &vec1
              );
        }
}
```

Arguments

vec0	4-D vector
vec1	4-D vector

Return Values

4-D vector in which each element is the quotient of the corresponding elements of the specified 4-D vectors

Description

Divide two 4-D vectors element by element.

Notes

Floating-point behavior matches standard library function divf4.

dot

Compute the dot product of two 4-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline vec_float4 dot(
                     const Vector4 &vec0,
                     const Vector4 &vec1
              );
        }
}
```

Arguments

vec0	4-D vector
vec1	4-D vector

Return Values

Dot product of the specified 4-D vectors

Description

Compute the dot product of two 4-D vectors.

length

Compute the length of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline vec_float4 length(
                     const Vector4 &vec
              );
        }
}
```

Arguments

vec	4-D vector

Return Values

Length of the specified 4-D vector

Description

Compute the length of a 4-D vector.

lengthSqr

Compute the square of the length of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline vec_float4 lengthSqr(
                     const Vector4 &vec
              );
        }
}
```

Arguments

vec	4-D vector

Return Values

Square of the length of the specified 4-D vector

Description

Compute the square of the length of a 4-D vector.

lerp

Linear interpolation between two 4-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector4 lerp(
                     vec_float4 t,
                     const Vector4 &vec0,
                     const Vector4 &vec1
              );
        }
}
```

Arguments

t	Interpolation parameter
vec0	4-D vector
vec1	4-D vector

Return Values

Interpolated 4-D vector

Description

Linearly interpolate between two 4-D vectors.

Notes

Does not clamp *t* between 0 and 1.

maxElem

Maximum element of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline vec_float4 maxElem(
                     const Vector4 &vec
              );
        }
}
```

Arguments

vec	4-D vector

Return Values

Maximum value of all elements of *vec*

Description

Compute the maximum value of all elements of a 4-D vector.

maxPerElem

Maximum of two 4-D vectors per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector4 maxPerElem(
                     const Vector4 &vec0,
                     const Vector4 &vec1
              );
        }
}
```

Arguments

vec0	4-D vector
vec1	4-D vector

Return Values

4-D vector in which each element is the maximum of the corresponding elements of the specified 4-D vectors

Description

Create a 4-D vector in which each element is the maximum of the corresponding elements of the specified 4-D vectors.

minElem

Minimum element of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline vec_float4 minElem(
                     const Vector4 &vec
              );
        }
}
```

Arguments

vec	4-D vector

Return Values

Minimum value of all elements of *vec*

Description

Compute the minimum value of all elements of a 4-D vector.

minPerElem

Minimum of two 4-D vectors per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector4 minPerElem(
                     const Vector4 &vec0,
                     const Vector4 &vec1
              );
        }
}
```

Arguments

vec0	4-D vector
vec1	4-D vector

Return Values

4-D vector in which each element is the minimum of the corresponding elements of the specified 4-D vectors

Description

Create a 4-D vector in which each element is the minimum of the corresponding elements of two specified 4-D vectors.

mulPerElem

Multiply two 4-D vectors per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector4 mulPerElem(
                     const Vector4 &vec0,
                     const Vector4 &vec1
              );
        }
}
```

Arguments

vec0	4-D vector
vec1	4-D vector

Return Values

4-D vector in which each element is the product of the corresponding elements of the specified 4-D vectors

Description

Multiply two 4-D vectors element by element.

normalize

Normalize a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector4 normalize(
                     const Vector4 &vec
              );
        }
}
```

Arguments

vec	4-D vector

Return Values

The specified 4-D vector scaled to unit length

Description

Compute a normalized 4-D vector.

Notes

The result is unpredictable when all elements of vec are at or near zero.

operator *

Multiply a 4-D vector by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector4 operator *(
                     vec_float4 scalar,
                     const Vector4 &vec
              );
        }
}
```

Arguments

scalar	Scalar value
vec	4-D vector

Return Values

Scalar product of *vec* and *scalar*

Description

Multiply a 4-D vector by a scalar.

outer

Outer product of two 4-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Matrix4 outer(
                     const Vector4 &vec0,
                     const Vector4 &vec1
              );
        }
}
```

Arguments

vec0	4-D vector
vec1	4-D vector

Return Values

The 4x4 matrix product of a column-vector, *vec0*, and a row-vector, *vec1*

Description

Compute the outer product of two 4-D vectors.

print

Print a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline void print(
                     const Vector4 &vec
              );
        }
}
```

Arguments

vec	4-D vector

Return Values

None

Description

Print a 4-D vector. Prints the 4-D vector transposed, that is, as a row instead of a column.

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

print

Print a 4-D vector and an associated string identifier.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline void print(
                     const Vector4 &vec,
                     const char *name
              );
        }
}
```

Arguments

vec	4-D vector
name	String printed with the 4-D vector

Return Values

None

Description

Print a 4-D vector and an associated string identifier. Prints the 4-D vector transposed, that is, as a row instead of a column.

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

recipPerElem

Compute the reciprocal of a 4-D vector per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector4 recipPerElem(
                     const Vector4 &vec
              );
        }
}
```

Arguments

vec	4-D vector

Return Values

4-D vector in which each element is the reciprocal of the corresponding element of the specified 4-D vector

Description

Create a 4-D vector in which each element is the reciprocal of the corresponding element of the specified 4-D vector.

Notes

Floating-point behavior matches standard library function recipf4.

rsqrtPerElem

Compute the reciprocal square root of a 4-D vector per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector4 rsqrtPerElem(
                     const Vector4 &vec
              );
        }
}
```

Arguments

vec	4-D vector

Return Values

4-D vector in which each element is the reciprocal square root of the corresponding element of the specified 4-D vector

Description

Create a 4-D vector in which each element is the reciprocal square root of the corresponding element of the specified 4-D vector.

Notes

Floating-point behavior matches standard library function rsqrtf4.

select

Conditionally select between two 4-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector4 select(
                     const Vector4 &vec0,
                     const Vector4 &vec1,
                     vec_uint4 select1
              );
        }
}
```

Arguments

vec0	4-D vector
vec1	4-D vector
select1	For each of the four word slots, this mask selects either the 4-D vector in the corresponding slot of `vec0` or the 4-D vector in the corresponding slot of `vec1`. A 0 bit selects from `vec0` whereas a 1 bit selects from `vec1`. Identical bits should be set for each word of the mask.

Return Values

Each slot of the result is equal to the 4-D vector at the corresponding slot of `vec0` or `vec1`, depending on the value of `select1` at the corresponding slot. A value of 0 selects the slot of `vec0`, and a value of 0xFFFFFFFF selects the slot of `vec1`

Description

Conditionally select one of the 4-D vectors at each of the corresponding slots of `vec0` or `vec1`.

Notes

This function uses a conditional select instruction to avoid a branch.

slerp

Spherical linear interpolation between two 4-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector4 slerp(
                     vec_float4 t,
                     const Vector4 &unitVec0,
                     const Vector4 &unitVec1
              );
        }
}
```

Arguments

t	Interpolation parameter
unitVec0	4-D vector, expected to be unit-length
unitVec1	4-D vector, expected to be unit-length

Return Values

Interpolated 4-D vector

Description

Perform spherical linear interpolation between two 4-D vectors.

Notes

The result is unpredictable if the vectors point in opposite directions. Does not clamp *t* between 0 and 1.

sqrtPerElem

Compute the square root of a 4-D vector per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector4 sqrtPerElem(
                     const Vector4 &vec
              );
        }
}
```

Arguments

vec	4-D vector

Return Values

4-D vector in which each element is the square root of the corresponding element of the specified 4-D vector

Description

Create a 4-D vector in which each element is the square root of the corresponding element of the specified 4-D vector.

Notes

Floating-point behavior matches standard library function sqrtf4.

storeHalfFloats

Store four slots of an SoA 4-D vector as half-floats.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
             inline void storeHalfFloats(
                    const Vector4 &vec,
                    vec_ushort8 *twoQuads
             );
        }
}
```

Arguments

vec	4-D vector
twoQuads	An output array of 2 quadwords containing 16 half-floats

Return Values

None

Description

Store four slots of an SoA 4-D vector in two quadwords of half-float values. Numbering slots of *vec* as 0..3, the output is {x0,y0,z0,w0,x1,y1,z1,w1,x2,y2,z2,w2,x3,y3,z3,w3}.

sum

Compute the sum of all elements of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline vec_float4 sum(
                     const Vector4 &vec
              );
        }
}
```

Arguments

vec	4-D vector

Return Values

Sum of all elements of *vec*

Description

Compute the sum of all elements of a 4-D vector.

3-D Point Functions

absPerElem

Compute the absolute value of a 3-D point per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Point3 absPerElem(
                     const Point3 &pnt
              );
        }
}
```

Arguments

pnt	3-D point

Return Values

3-D point in which each element is the absolute value of the corresponding element of pnt

Description

Compute the absolute value of each element of a 3-D point.

copySignPerElem

Copy sign from one 3-D point to another, per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Point3 copySignPerElem(
                     const Point3 &pnt0,
                     const Point3 &pnt1
              );
        }
}
```

Arguments

pnt0	3-D point
pnt1	3-D point

Return Values

3-D point in which each element has the magnitude of the corresponding element of *pnt0* and the sign of the corresponding element of *pnt1*

Description

For each element, create a value composed of the magnitude of *pnt0* and the sign of *pnt1*.

dist

Compute the distance between two 3-D points.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline vec_float4 dist(
                     const Point3 &pnt0,
                     const Point3 &pnt1
              );
        }
}
```

Arguments

pnt0	3-D point
pnt1	3-D point

Return Values

Distance between two 3-D points

Description

Compute the distance between two 3-D points.

distFromOrigin

Compute the distance of a 3-D point from the coordinate-system origin.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline vec_float4 distFromOrigin(
                     const Point3 &pnt
              );
        }
}
```

Arguments

pnt	3-D point

Return Values

Distance of a 3-D point from the origin

Description

Compute the distance of a 3-D point from the coordinate-system origin.

distSqr

Compute the square of the distance between two 3-D points.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline vec_float4 distSqr(
                     const Point3 &pnt0,
                     const Point3 &pnt1
              );
        }
}
```

Arguments

pnt0	3-D point
pnt1	3-D point

Return Values

Square of the distance between two 3-D points

Description

Compute the square of the distance between two 3-D points.

distSqrFromOrigin

Compute the square of the distance of a 3-D point from the coordinate-system origin.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline vec_float4 distSqrFromOrigin(
                     const Point3 &pnt
              );
        }
}
```

Arguments

pnt	3-D point

Return Values

Square of the distance of a 3-D point from the origin

Description

Compute the square of the distance of a 3-D point from the coordinate-system origin.

divPerElem

Divide two 3-D points per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Point3 divPerElem(
                     const Point3 &pnt0,
                     const Point3 &pnt1
              );
        }
}
```

Arguments

pnt0	3-D point
pnt1	3-D point

Return Values

3-D point in which each element is the quotient of the corresponding elements of the specified 3-D points

Description

Divide two 3-D points element by element.

Notes

Floating-point behavior matches standard library function divf4.

lerp

Linear interpolation between two 3-D points.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Point3 lerp(
                     vec_float4 t,
                     const Point3 &pnt0,
                     const Point3 &pnt1
              );
        }
}
```

Arguments

t	Interpolation parameter
pnt0	3-D point
pnt1	3-D point

Return Values

Interpolated 3-D point

Description

Linearly interpolate between two 3-D points.

Notes

Does not clamp *t* between 0 and 1.

loadXYZArray

Load four three-float 3-D points, stored in three quadwords.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline void loadXYZArray(
                     Point3 &pnt,
                     const vec_float4 *threeQuads
              );
        }
}
```

Arguments

pnt	An output 3-D point
threeQuads	Array of 3 quadwords containing 12 floats

Return Values

None

Description

Load four three-float 3-D points, stored in three quadwords as {x0,y0,z0,x1,y1,z1,x2,y2,z2,x3,y3,z3}, into four slots of an SoA 3-D point.

maxElem

Maximum element of a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline vec_float4 maxElem(
                     const Point3 &pnt
              );
        }
}
```

Arguments

pnt	3-D point

Return Values

Maximum value of all elements of *pnt*

Description

Compute the maximum value of all elements of a 3-D point.

maxPerElem

Maximum of two 3-D points per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Point3 maxPerElem(
                     const Point3 &pnt0,
                     const Point3 &pnt1
              );
        }
}
```

Arguments

pnt0	3-D point
pnt1	3-D point

Return Values

3-D point in which each element is the maximum of the corresponding elements of the specified 3-D points

Description

Create a 3-D point in which each element is the maximum of the corresponding elements of the specified 3-D points.

minElem

Minimum element of a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline vec_float4 minElem(
                     const Point3 &pnt
              );
        }
}
```

Arguments

pnt	3-D point

Return Values

Minimum value of all elements of *pnt*

Description

Compute the minimum value of all elements of a 3-D point.

minPerElem

Minimum of two 3-D points per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Point3 minPerElem(
                     const Point3 &pnt0,
                     const Point3 &pnt1
              );
        }
}
```

Arguments

pnt0	3-D point
pnt1	3-D point

Return Values

3-D point in which each element is the minimum of the corresponding elements of the specified 3-D points

Description

Create a 3-D point in which each element is the minimum of the corresponding elements of two specified 3-D points.

mulPerElem

Multiply two 3-D points per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Point3 mulPerElem(
                     const Point3 &pnt0,
                     const Point3 &pnt1
              );
        }
}
```

Arguments

pnt0	3-D point
pnt1	3-D point

Return Values

3-D point in which each element is the product of the corresponding elements of the specified 3-D points

Description

Multiply two 3-D points element by element.

print

Print a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline void print(
                     const Point3 &pnt
              );
        }
}
```

Arguments

pnt	3-D point

Return Values

None

Description

Print a 3-D point. Prints the 3-D point transposed, that is, as a row instead of a column.

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

print

Print a 3-D point and an associated string identifier.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline void print(
                     const Point3 &pnt,
                     const char *name
              );
        }
}
```

Arguments

pnt	3-D point
name	String printed with the 3-D point

Return Values

None

Description

Print a 3-D point and an associated string identifier. Prints the 3-D point transposed, that is, as a row instead of a column.

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

projection

Scalar projection of a 3-D point on a unit-length 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline vec_float4 projection(
                     const Point3 &pnt,
                     const Vector3 &unitVec
              );
        }
}
```

Arguments

pnt	3-D point
unitVec	3-D vector, expected to be unit-length

Return Values

Scalar projection of the 3-D point on the unit-length 3-D vector

Description

Scalar projection of a 3-D point on a unit-length 3-D vector (dot product).

recipPerElem

Compute the reciprocal of a 3-D point per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Point3 recipPerElem(
                     const Point3 &pnt
              );
        }
}
```

Arguments

pnt	3-D point

Return Values

3-D point in which each element is the reciprocal of the corresponding element of the specified 3-D point

Description

Create a 3-D point in which each element is the reciprocal of the corresponding element of the specified 3-D point.

Notes

Floating-point behavior matches standard library function recipf4.

rsqrtPerElem

Compute the reciprocal square root of a 3-D point per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Point3 rsqrtPerElem(
                     const Point3 &pnt
              );
        }
}
```

Arguments

pnt	3-D point

Return Values

3-D point in which each element is the reciprocal square root of the corresponding element of the specified 3-D point

Description

Create a 3-D point in which each element is the reciprocal square root of the corresponding element of the specified 3-D point.

Notes

Floating-point behavior matches standard library function rsqrtf4.

scale

Apply uniform scale to a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Point3 scale(
                     const Point3 &pnt,
                     vec_float4 scaleVal
              );
        }
}
```

Arguments

pnt	3-D point
scaleVal	Scalar value

Return Values

3-D point in which every element is multiplied by the scalar value

Description

Apply uniform scale to a 3-D point.

scale

Apply non-uniform scale to a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Point3 scale(
                     const Point3 &pnt,
                     const Vector3 &scaleVec
              );
        }
}
```

Arguments

pnt	3-D point
scaleVec	3-D vector

Return Values

3-D point in which each element is the product of the corresponding elements of the specified 3-D point and 3-D vector

Description

Apply non-uniform scale to a 3-D point.

select

Conditionally select between two 3-D points.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Point3 select(
                     const Point3 &pnt0,
                     const Point3 &pnt1,
                     vec_uint4 select1
              );
        }
}
```

Arguments

pnt0	3-D point
pnt1	3-D point
select1	For each of the four word slots, this mask selects either the 3-D point in the corresponding slot of `pnt0` or the 3-D point in the corresponding slot of `pnt1`. A 0 bit selects from `pnt0` whereas a 1 bit selects from `pnt1`. Identical bits should be set for each word of the mask.

Return Values

Each slot of the result is equal to the 3-D point at the corresponding slot of `pnt0` or `pnt1`, depending on the value of `select1` at the corresponding slot. A value of 0 selects the slot of `pnt0`, and a value of 0xFFFFFFFF selects the slot of `pnt1`

Description

Conditionally select one of the 3-D points at each of the corresponding slots of `pnt0` or `pnt1`.

Notes

This function uses a conditional select instruction to avoid a branch.

sqrtPerElem

Compute the square root of a 3-D point per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Point3 sqrtPerElem(
                     const Point3 &pnt
              );
        }
}
```

Arguments

pnt	3-D point

Return Values

3-D point in which each element is the square root of the corresponding element of the specified 3-D point

Description

Create a 3-D point in which each element is the square root of the corresponding element of the specified 3-D point.

Notes

Floating-point behavior matches standard library function sqrtf4.

storeHalfFloats

Store eight slots of two SoA 3-D points as half-floats.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline void storeHalfFloats(
                     const Point3 &pnt0,
                     const Point3 &pnt1,
                     vec_ushort8 *threeQuads
              );
        }
}
```

Arguments

pnt0	3-D point
pnt1	3-D point
threeQuads	An output array of 3 quadwords containing 24 half-floats

Return Values

None

Description

Store eight slots of two SoA 3-D points in three quadwords of half-float values. Numbering slots of *pnt0* as 0..3 and slots of *pnt1* as 4..7, the output is {x0,y0,z0,x1,y1,z1,x2,y2,z2,x3,y3,z3,x4,y4,z4,x5,y5,z5,x6,y6,z6,x7,y7,z7}.

storeXYZArray

Store four slots of an SoA 3-D point in three quadwords.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline void storeXYZArray(
                     const Point3 &pnt,
                     vec_float4 *threeQuads
              );
        }
}
```

Arguments

pnt	3-D point
threeQuads	An output array of 3 quadwords containing 12 floats

Return Values

None

Description

Store four slots of an SoA 3-D point in three quadwords as {x0,y0,z0,x1,y1,z1,x2,y2,z2,x3,y3,z3}.

sum

Compute the sum of all elements of a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline vec_float4 sum(
                     const Point3 &pnt
              );
        }
}
```

Arguments

pnt	3-D point

Return Values

Sum of all elements of *pnt*

Description

Compute the sum of all elements of a 3-D point.

Quaternion Functions

conj

Compute the conjugate of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Quat conj(
                     const Quat &quat
              );
        }
}
```

Arguments

quat	Quaternion

Return Values

Conjugate of the specified quaternion

Description

Compute the conjugate of a quaternion.

dot

Compute the dot product of two quaternions.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline vec_float4 dot(
                     const Quat &quat0,
                     const Quat &quat1
              );
        }
}
```

Arguments

quat0	Quaternion
quat1	Quaternion

Return Values

Dot product of the specified quaternions

Description

Compute the dot product of two quaternions.

length

Compute the length of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline vec_float4 length(
                     const Quat &quat
              );
        }
}
```

Arguments

quat	Quaternion

Return Values

Length of the specified quaternion

Description

Compute the length of a quaternion.

lerp

Linear interpolation between two quaternions.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Quat lerp(
                     vec_float4 t,
                     const Quat &quat0,
                     const Quat &quat1
              );
        }
}
```

Arguments

t	Interpolation parameter
quat0	Quaternion
quat1	Quaternion

Return Values

Interpolated quaternion

Description

Linearly interpolate between two quaternions.

Notes

Does not clamp `t` between 0 and 1.

norm

Compute the norm of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline vec_float4 norm(
                     const Quat &quat
              );
        }
}
```

Arguments

quat	Quaternion

Return Values

The norm of the specified quaternion

Description

Compute the norm, equal to the square of the length, of a quaternion.

normalize

Normalize a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Quat normalize(
                     const Quat &quat
              );
        }
}
```

Arguments

quat	Quaternion

Return Values

The specified quaternion scaled to unit length

Description

Compute a normalized quaternion.

Notes

The result is unpredictable when all elements of quat are at or near zero.

operator *

Multiply a quaternion by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Quat operator *(
                     vec_float4 scalar,
                     const Quat &quat
              );
        }
}
```

Arguments

scalar	Scalar value
quat	Quaternion

Return Values

Scalar product of *quat* and *scalar*

Description

Multiply a quaternion by a scalar.

print

Print a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline void print(
                     const Quat &quat
              );
        }
}
```

Arguments

quat	Quaternion

Return Values

None

Description

Print a quaternion.

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

print

Print a quaternion and an associated string identifier.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline void print(
                     const Quat &quat,
                     const char *name
              );
        }
}
```

Arguments

quat	Quaternion
name	String printed with the quaternion

Return Values

None

Description

Print a quaternion and an associated string identifier.

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

rotate

Use a unit-length quaternion to rotate a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Vector3 rotate(
                     const Quat &unitQuat,
                     const Vector3 &vec
              );
        }
}
```

Arguments

unitQuat	Quaternion, expected to be unit-length
vec	3-D vector

Return Values

The rotated 3-D vector, equivalent to unitQuat*Quat(vec,0)*conj(unitQuat)

Description

Rotate a 3-D vector by applying a unit-length quaternion.

select

Conditionally select between two quaternions.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Quat select(
                     const Quat &quat0,
                     const Quat &quat1,
                     vec_uint4 select1
              );
        }
}
```

Arguments

quat0	Quaternion
quat1	Quaternion
select1	For each of the four word slots, this mask selects either the quaternion in the corresponding slot of *quat0* or the quaternion in the corresponding slot of *quat1*. A 0 bit selects from *quat0* whereas a 1 bit selects from *quat1*. Identical bits should be set for each word of the mask.

Return Values

Each slot of the result is equal to the quaternion at the corresponding slot of *quat0* or *quat1*, depending on the value of *select1* at the corresponding slot. A value of 0 selects the slot of *quat0*, and a value of 0xFFFFFFFF selects the slot of *quat1*

Description

Conditionally select one of the quaternions at each of the corresponding slots of *quat0* or *quat1*.

Notes

This function uses a conditional select instruction to avoid a branch.

slerp

Spherical linear interpolation between two quaternions.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Quat slerp(
                     vec_float4 t,
                     const Quat &unitQuat0,
                     const Quat &unitQuat1
              );
        }
}
```

Arguments

t	Interpolation parameter
unitQuat0	Quaternion, expected to be unit-length
unitQuat1	Quaternion, expected to be unit-length

Return Values

Interpolated quaternion

Description

Perform spherical linear interpolation between two quaternions.

Notes

Interpolates along the shortest path between orientations. Does not clamp `t` between 0 and 1.

squad

Spherical quadrangle interpolation.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
             inline const Quat squad(
                    vec_float4 t,
                    const Quat &unitQuat0,
                    const Quat &unitQuat1,
                    const Quat &unitQuat2,
                    const Quat &unitQuat3
             );
        }
}
```

Arguments

t	Interpolation parameter
unitQuat0	Quaternion, expected to be unit-length
unitQuat1	Quaternion, expected to be unit-length
unitQuat2	Quaternion, expected to be unit-length
unitQuat3	Quaternion, expected to be unit-length

Return Values

Interpolated quaternion

Description

Perform spherical quadrangle interpolation between four quaternions.

3x3 Matrix Functions

absPerElem

Compute the absolute value of a 3x3 matrix per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Matrix3 absPerElem(
                     const Matrix3 &mat
              );
        }
}
```

Arguments

mat	3x3 matrix

Return Values

3x3 matrix in which each element is the absolute value of the corresponding element of the specified 3x3 matrix

Description

Compute the absolute value of each element of a 3x3 matrix.

appendScale

Append (post-multiply) a scale transformation to a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Matrix3 appendScale(
                     const Matrix3 &mat,
                     const Vector3 &scaleVec
              );
        }
}
```

Arguments

mat	3x3 matrix
scaleVec	3-D vector

Return Values

The product of *mat* and a scale transformation created from *scaleVec*

Description

Post-multiply a 3x3 matrix by a scale transformation whose diagonal scale factors are contained in the 3-D vector.

Notes

Faster than creating and multiplying a scale transformation matrix.

determinant

Determinant of a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline vec_float4 determinant(
                     const Matrix3 &mat
              );
        }
}
```

Arguments

mat	3x3 matrix

Return Values

The determinant of *mat*

Description

Compute the determinant of a 3x3 matrix.

inverse

Compute the inverse of a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Matrix3 inverse(
                     const Matrix3 &mat
              );
        }
}
```

Arguments

mat	3x3 matrix

Return Values

Inverse of *mat*

Description

Compute the inverse of a 3x3 matrix.

Notes

Result is unpredictable when the determinant of *mat* is equal to or near 0.

mulPerElem

Multiply two 3x3 matrices per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Matrix3 mulPerElem(
                     const Matrix3 &mat0,
                     const Matrix3 &mat1
              );
        }
}
```

Arguments

mat0	3x3 matrix
mat1	3x3 matrix

Return Values

3x3 matrix in which each element is the product of the corresponding elements of the specified 3x3 matrices

Description

Multiply two 3x3 matrices element by element.

operator *

Multiply a 3x3 matrix by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Matrix3 operator *(
                     vec_float4 scalar,
                     const Matrix3 &mat
              );
        }
}
```

Arguments

scalar	Scalar value
mat	3x3 matrix

Return Values

Scalar product of *mat* and *scalar*

Description

Multiply a 3x3 matrix by a scalar.

prependScale

Prepend (pre-multiply) a scale transformation to a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Matrix3 prependScale(
                     const Vector3 &scaleVec,
                     const Matrix3 &mat
              );
        }
}
```

Arguments

scaleVec	3-D vector
mat	3x3 matrix

Return Values

The product of a scale transformation created from *scaleVec* and *mat*

Description

Pre-multiply a 3x3 matrix by a scale transformation whose diagonal scale factors are contained in the 3-D vector.

Notes

Faster than creating and multiplying a scale transformation matrix.

print

Print a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline void print(
                     const Matrix3 &mat
              );
        }
}
```

Arguments

mat	3x3 matrix

Return Values

None

Description

Print a 3x3 matrix. Unlike the printing of vectors, the 3x3 matrix is printed with the correct orientation (columns appear vertically).

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

print

Print a 3x3 matrix and an associated string identifier.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline void print(
                     const Matrix3 &mat,
                     const char *name
              );
        }
}
```

Arguments

mat	3x3 matrix
name	String printed with the 3x3 matrix

Return Values

None

Description

Print a 3x3 matrix and an associated string identifier. Unlike the printing of vectors, the 3x3 matrix is printed with the correct orientation (columns appear vertically).

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

select

Conditionally select between two 3x3 matrices.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Matrix3 select(
                     const Matrix3 &mat0,
                     const Matrix3 &mat1,
                     vec_uint4 select1
              );
        }
}
```

Arguments

mat0	3x3 matrix
mat1	3x3 matrix
select1	For each of the four word slots, this mask selects either the 3x3 matrix in the corresponding slot of `mat0` or the 3x3 matrix in the corresponding slot of `mat1`. A 0 bit selects from `mat0` whereas a 1 bit selects from `mat1`. Identical bits should be set for each word of the mask.

Return Values

Each slot of the result is equal to the 3x3 matrix at the corresponding slot of `mat0` or `mat1`, depending on the value of `select1` at the corresponding slot. A value of 0 selects the slot of `mat0` and a value of 0xFFFFFFFF selects the slot of `mat1`

Description

Conditionally select one of the 3x3 matrices at each of the corresponding slots of `mat0` or `mat1`.

Notes

This function uses a conditional select instruction to avoid a branch.

transpose

Transpose of a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Matrix3 transpose(
                     const Matrix3 &mat
              );
        }
}
```

Arguments

mat	3x3 matrix

Return Values

mat transposed

Description

Compute the transpose of a 3x3 matrix.

4x4 Matrix Functions

absPerElem

Compute the absolute value of a 4x4 matrix per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Matrix4 absPerElem(
                     const Matrix4 &mat
              );
        }
}
```

Arguments

mat	4x4 matrix

Return Values

4x4 matrix in which each element is the absolute value of the corresponding element of the specified 4x4 matrix

Description

Compute the absolute value of each element of a 4x4 matrix.

affineInverse

Compute the inverse of a 4x4 matrix, which is expected to be an affine matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Matrix4 affineInverse(
                     const Matrix4 &mat
              );
        }
}
```

Arguments

mat	4x4 matrix

Return Values

Inverse of the specified 4x4 matrix

Description

Naming the upper-left 3x3 submatrix of the specified 4x4 matrix as M, and its translation component as v, compute a matrix whose upper-left 3x3 submatrix is inverse(M), whose translation vector is -inverse(M)*v, and whose bottom row is (0,0,0,1).

Notes

This can be used to achieve better performance than a general inverse when the specified 4x4 matrix meets the given restrictions. The result is unpredictable when the determinant of *mat* is equal to or near 0.

appendScale

Append (post-multiply) a scale transformation to a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Matrix4 appendScale(
                     const Matrix4 &mat,
                     const Vector3 &scaleVec
              );
        }
}
```

Arguments

mat	4x4 matrix
scaleVec	3-D vector

Return Values

The product of *mat* and a scale transformation created from *scaleVec*

Description

Post-multiply a 4x4 matrix by a scale transformation whose diagonal scale factors are contained in the 3-D vector.

Notes

Faster than creating and multiplying a scale transformation matrix.

determinant

Determinant of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline vec_float4 determinant(
                     const Matrix4 &mat
              );
        }
}
```

Arguments

mat	4x4 matrix

Return Values

The determinant of *mat*

Description

Compute the determinant of a 4x4 matrix.

inverse

Compute the inverse of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Matrix4 inverse(
                     const Matrix4 &mat
              );
        }
}
```

Arguments

mat	4x4 matrix

Return Values

Inverse of *mat*

Description

Compute the inverse of a 4x4 matrix.

Notes

Result is unpredictable when the determinant of *mat* is equal to or near 0.

mulPerElem

Multiply two 4x4 matrices per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Matrix4 mulPerElem(
                     const Matrix4 &mat0,
                     const Matrix4 &mat1
              );
        }
}
```

Arguments

mat0	4x4 matrix
mat1	4x4 matrix

Return Values

4x4 matrix in which each element is the product of the corresponding elements of the specified 4x4 matrices

Description

Multiply two 4x4 matrices element by element.

operator *

Multiply a 4x4 matrix by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Matrix4 operator *(
                     vec_float4 scalar,
                     const Matrix4 &mat
              );
        }
}
```

Arguments

scalar	Scalar value
mat	4x4 matrix

Return Values

Scalar product of *mat* and *scalar*

Description

Multiply a 4x4 matrix by a scalar.

orthoInverse

Compute the inverse of a 4x4 matrix, which is expected to be an affine matrix with an orthogonal upper-left 3x3 submatrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Matrix4 orthoInverse(
                     const Matrix4 &mat
              );
        }
}
```

Arguments

mat	4x4 matrix

Return Values

Inverse of the specified 4x4 matrix

Description

Naming the upper-left 3x3 submatrix of the specified 4x4 matrix as M, and its translation component as v, compute a matrix whose upper-left 3x3 submatrix is transpose(M), whose translation vector is -transpose(M)*v, and whose bottom row is (0,0,0,1).

Notes

This can be used to achieve better performance than a general inverse when the specified 4x4 matrix meets the given restrictions.

prependScale

Prepend (pre-multiply) a scale transformation to a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Matrix4 prependScale(
                     const Vector3 &scaleVec,
                     const Matrix4 &mat
              );
        }
}
```

Arguments

scaleVec	3-D vector
mat	4x4 matrix

Return Values

The product of a scale transformation created from *scaleVec* and *mat*

Description

Pre-multiply a 4x4 matrix by a scale transformation whose diagonal scale factors are contained in the 3-D vector.

Notes

Faster than creating and multiplying a scale transformation matrix.

print

Print a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline void print(
                     const Matrix4 &mat
              );
        }
}
```

Arguments

mat	4x4 matrix

Return Values

None

Description

Print a 4x4 matrix. Unlike the printing of vectors, the 4x4 matrix is printed with the correct orientation (columns appear vertically).

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

print

Print a 4x4 matrix and an associated string identifier.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline void print(
                     const Matrix4 &mat,
                     const char *name
              );
        }
}
```

Arguments

mat	4x4 matrix
name	String printed with the 4x4 matrix

Return Values

None

Description

Print a 4x4 matrix and an associated string identifier. Unlike the printing of vectors, the 4x4 matrix is printed with the correct orientation (columns appear vertically).

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

select

Conditionally select between two 4x4 matrices.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Matrix4 select(
                     const Matrix4 &mat0,
                     const Matrix4 &mat1,
                     vec_uint4 select1
              );
        }
}
```

Arguments

mat0	4x4 matrix
mat1	4x4 matrix
select1	For each of the four word slots, this mask selects either the 4x4 matrix in the corresponding slot of `mat0` or the 4x4 matrix in the corresponding slot of `mat1`. A 0 bit selects from `mat0` whereas a 1 bit selects from `mat1`. Identical bits should be set for each word of the mask.

Return Values

Each slot of the result is equal to the 4x4 matrix at the corresponding slot of *mat0* or *mat1*, depending on the value of *select1* at the corresponding slot. A value of 0 selects the slot of *mat0* and a value of 0xFFFFFFFF selects the slot of *mat1*

Description

Conditionally select one of the 4x4 matrices at each of the corresponding slots of *mat0* or *mat1*.

Notes

This function uses a conditional select instruction to avoid a branch.

transpose

Transpose of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Matrix4 transpose(
                     const Matrix4 &mat
              );
        }
}
```

Arguments

mat	4x4 matrix

Return Values

mat transposed

Description

Compute the transpose of a 4x4 matrix.

3x4 Transformation Matrix Functions

absPerElem

Compute the absolute value of a 3x4 transformation matrix per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Transform3 absPerElem(
                     const Transform3 &tfrm
              );
        }
}
```

Arguments

tfrm	3x4 transformation matrix

Return Values

3x4 transformation matrix in which each element is the absolute value of the corresponding element of the specified 3x4 transformation matrix

Description

Compute the absolute value of each element of a 3x4 transformation matrix.

appendScale

Append (post-multiply) a scale transformation to a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Transform3 appendScale(
                     const Transform3 &tfrm,
                     const Vector3 &scaleVec
              );
        }
}
```

Arguments

tfrm	3x4 transformation matrix
scaleVec	3-D vector

Return Values

The product of `tfrm` and a scale transformation created from `scaleVec`

Description

Post-multiply a 3x4 transformation matrix by a scale transformation whose diagonal scale factors are contained in the 3-D vector.

Notes

Faster than creating and multiplying a scale transformation matrix.

inverse

Inverse of a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Transform3 inverse(
                     const Transform3 &tfrm
              );
        }
}
```

Arguments

tfrm	3x4 transformation matrix

Return Values

Inverse of *tfrm*

Description

Compute the inverse of a 3x4 transformation matrix.

Notes

Result is unpredictable when the determinant of the left 3x3 submatrix is equal to or near 0.

mulPerElem

Multiply two 3x4 transformation matrices per element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
             inline const Transform3 mulPerElem(
                    const Transform3 &tfrm0,
                    const Transform3 &tfrm1
             );
        }
}
```

Arguments

tfrm0	3x4 transformation matrix
tfrm1	3x4 transformation matrix

Return Values

3x4 transformation matrix in which each element is the product of the corresponding elements of the specified 3x4 transformation matrices

Description

Multiply two 3x4 transformation matrices element by element.

orthoInverse

Compute the inverse of a 3x4 transformation matrix, expected to have an orthogonal upper-left 3x3 submatrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Transform3 orthoInverse(
                     const Transform3 &tfrm
              );
        }
}
```

Arguments

tfrm	3x4 transformation matrix

Return Values

Inverse of the specified 3x4 transformation matrix

Description

Naming the upper-left 3x3 submatrix of the specified 3x4 transformation matrix as M, and its translation component as v, compute a matrix whose upper-left 3x3 submatrix is transpose(M), and whose translation vector is -transpose(M)*v.

Notes

This can be used to achieve better performance than a general inverse when the specified 3x4 transformation matrix meets the given restrictions.

prependScale

Prepend (pre-multiply) a scale transformation to a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Transform3 prependScale(
                     const Vector3 &scaleVec,
                     const Transform3 &tfrm
              );
        }
}
```

Arguments

scaleVec	3-D vector
tfrm	3x4 transformation matrix

Return Values

The product of a scale transformation created from *scaleVec* and *tfrm*

Description

Pre-multiply a 3x4 transformation matrix by a scale transformation whose diagonal scale factors are contained in the 3-D vector.

Notes

Faster than creating and multiplying a scale transformation matrix.

print

Print a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline void print(
                     const Transform3 &tfrm
              );
        }
}
```

Arguments

tfrm	3x4 transformation matrix

Return Values

None

Description

Print a 3x4 transformation matrix. Unlike the printing of vectors, the 3x4 transformation matrix is printed with the correct orientation (columns appear vertically).

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

print

Print a 3x4 transformation matrix and an associated string identifier.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline void print(
                     const Transform3 &tfrm,
                     const char *name
              );
        }
}
```

Arguments

tfrm	3x4 transformation matrix
name	String printed with the 3x4 transformation matrix

Return Values

None

Description

Print a 3x4 transformation matrix and an associated string identifier. Unlike the printing of vectors, the 3x4 transformation matrix is printed with the correct orientation (columns appear vertically).

Notes

Function is only defined when _VECTORMATH_DEBUG is defined.

select

Conditionally select between two 3x4 transformation matrices.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              inline const Transform3 select(
                     const Transform3 &tfrm0,
                     const Transform3 &tfrm1,
                     vec_uint4 select1
              );
        }
}
```

Arguments

tfrm0	3x4 transformation matrix
tfrm1	3x4 transformation matrix
select1	For each of the four word slots, this mask selects either the 3x4 transformation matrix in the corresponding slot of *tfrm0* or the 3x4 transformation matrix in the corresponding slot of *tfrm1*. A 0 bit selects from *tfrm0* whereas a 1 bit selects from *tfrm1*. Identical bits should be set for each word of the mask.

Return Values

Each slot of the result is equal to the 3x4 transformation matrix at the corresponding slot of *tfrm0* or *tfrm1*, depending on the value of *select1* at the corresponding slot. A value of 0 selects the slot of *tfrm0* and a value of 0xFFFFFFFF selects the slot of *tfrm1*

Description

Conditionally select one of the 3x4 transformation matrices at each of the corresponding slots of *tfrm0* or *tfrm1*.

Notes

This function uses a conditional select instruction to avoid a branch.

Vectormath::Soa::Matrix3

Summary

Vectormath::Soa::Matrix3

A set of four 3x3 matrices in structure-of-arrays format.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
class Matrix3;
```

Description

A class representing a set of four 3x3 matrices stored in structure-of-arrays (SoA) format.

Methods Summary

Methods	Description
get4Aos	Extract four AoS 3x3 matrices.
getCol	Get the column of a 3x3 matrix referred to by the specified index.
getCol0	Get column 0 of a 3x3 matrix.
getCol1	Get column 1 of a 3x3 matrix.
getCol2	Get column 2 of a 3x3 matrix.
getElem	Get the element of a 3x3 matrix referred to by column and row indices.
getRow	Get the row of a 3x3 matrix referred to by the specified index.
identity	Construct an identity 3x3 matrix.
Matrix3	Default constructor; does no initialization.
Matrix3	Copy a 3x3 matrix.
Matrix3	Construct a 3x3 matrix containing the specified columns.
Matrix3	Construct a 3x3 rotation matrix from a unit-length quaternion.
Matrix3	Set all elements of a 3x3 matrix to the same scalar value.
Matrix3	Replicate an AoS 3x3 matrix.
Matrix3	Insert four AoS 3x3 matrices.
operator *	Multiply a 3x3 matrix by a scalar.
operator *	Multiply a 3x3 matrix by a 3-D vector.
operator *	Multiply two 3x3 matrices.
operator *=	Perform compound assignment and multiplication by a scalar.
operator *=	Perform compound assignment and multiplication by a 3x3 matrix.
operator+	Add two 3x3 matrices.
operator+=	Perform compound assignment and addition with a 3x3 matrix.
operator-	Subtract a 3x3 matrix from another 3x3 matrix.
operator-	Negate all elements of a 3x3 matrix.
operator-=	Perform compound assignment and subtraction by a 3x3 matrix.
operator=	Assign one 3x3 matrix to another.
operator[]	Subscripting operator to set or get a column.

Methods	Description
operator[]	Subscripting operator to get a column.
rotation	Construct a 3x3 matrix to rotate around a unit-length 3-D vector.
rotation	Construct a rotation matrix from a unit-length quaternion.
rotationX	Construct a 3x3 matrix to rotate around the x axis.
rotationY	Construct a 3x3 matrix to rotate around the y axis.
rotationZ	Construct a 3x3 matrix to rotate around the z axis.
rotationZYX	Construct a 3x3 matrix to rotate around the x, y, and z axes.
scale	Construct a 3x3 matrix to perform scaling.
setCol	Set the column of a 3x3 matrix referred to by the specified index.
setCol0	Set column 0 of a 3x3 matrix.
setCol1	Set column 1 of a 3x3 matrix.
setCol2	Set column 2 of a 3x3 matrix.
setElem	Set the element of a 3x3 matrix referred to by column and row indices.
setRow	Set the row of a 3x3 matrix referred to by the specified index.

Constructors and Destructors

Matrix3

Default constructor; does no initialization.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline Matrix3();
              }
        }
}
```

Arguments

None

Return Values

None

Description

Default constructor; does no initialization.

Matrix3

Copy a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline Matrix3(
                            const Matrix3 &mat
                     );
              }
        }
}
```

Arguments

mat	3x3 matrix

Return Values

None

Description

Construct a copy of a 3x3 matrix.

Matrix3

Construct a 3x3 matrix containing the specified columns.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline Matrix3(
                            const Vector3 &col0,
                            const Vector3 &col1,
                            const Vector3 &col2
                     );
              }
        }
}
```

Arguments

col0	3-D vector
col1	3-D vector
col2	3-D vector

Return Values

None

Description

Construct a 3x3 matrix containing the specified columns.

Matrix3

Construct a 3x3 rotation matrix from a unit-length quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     explicit inline Matrix3(
                            const Quat &unitQuat
                     );
              }
        }
}
```

Arguments

unitQuat	Quaternion, expected to be unit-length

Return Values

None

Description

Construct a 3x3 matrix that applies the same rotation as the specified unit-length quaternion.

Matrix3

Set all elements of a 3x3 matrix to the same scalar value.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     explicit inline Matrix3(
                            vec_float4 scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

None

Description

Construct a 3x3 matrix with all elements set to the scalar value argument.

Matrix3

Replicate an AoS 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline Matrix3(
                            const Aos::Matrix3 &mat
                     );
              }
        }
}
```

Arguments

mat	AoS 3x3 matrix

Return Values

None

Description

Replicate an AoS 3x3 matrix in all four slots of an SoA 3x3 matrix.

Matrix3

Insert four AoS 3x3 matrices.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline Matrix3(
                            const Aos::Matrix3 &mat0,
                            const Aos::Matrix3 &mat1,
                            const Aos::Matrix3 &mat2,
                            const Aos::Matrix3 &mat3
                     );
              }
        }
}
```

Arguments

mat0	AoS 3x3 matrix
mat1	AoS 3x3 matrix
mat2	AoS 3x3 matrix
mat3	AoS 3x3 matrix

Return Values

None

Description

Insert four AoS 3x3 matrices into four slots of an SoA 3x3 matrix (transpose the data format).

Operator Methods

operator *

Multiply a 3x3 matrix by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline const Matrix3 operator *(
                            vec_float4 scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

Product of the specified 3x3 matrix and scalar

Description

Multiply a 3x3 matrix by a scalar.

operator *

Multiply a 3x3 matrix by a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline const Vector3 operator *(
                            const Vector3 &vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

Product of the specified 3x3 matrix and 3-D vector

Description

Multiply a 3x3 matrix by a 3-D vector.

operator *

Multiply two 3x3 matrices.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline const Matrix3 operator *(
                            const Matrix3 &mat
                     );
              }
        }
}
```

Arguments

mat	3x3 matrix

Return Values

Product of the specified 3x3 matrices

Description

Multiply two 3x3 matrices.

operator *=

Perform compound assignment and multiplication by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline Matrix3 &operator *=(
                            vec_float4 scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

A reference to the resulting 3x3 matrix

Description

Perform compound assignment and multiplication by a scalar.

operator *=

Perform compound assignment and multiplication by a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline Matrix3 &operator *=(
                            const Matrix3 &mat
                     );
              }
        }
}
```

Arguments

mat	3x3 matrix

Return Values

A reference to the resulting 3x3 matrix

Description

Perform compound assignment and multiplication by a 3x3 matrix.

operator+

Add two 3x3 matrices.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline const Matrix3 operator+(
                            const Matrix3 &mat
                     );
              }
        }
}
```

Arguments

mat	3x3 matrix

Return Values

Sum of the specified 3x3 matrices

Description

Add two 3x3 matrices.

operator+=

Perform compound assignment and addition with a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
                class Matrix3 {
                        inline Matrix3 &operator+=(
                                const Matrix3 &mat
                        );
                }
        }
}
```

Arguments

mat	3x3 matrix

Return Values

A reference to the resulting 3x3 matrix

Description

Perform compound assignment and addition with a 3x3 matrix.

operator-

Subtract a 3x3 matrix from another 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline const Matrix3 operator-(
                            const Matrix3 &mat
                     );
              }
        }
}
```

Arguments

mat	3x3 matrix

Return Values

Difference of the specified 3x3 matrices

Description

Subtract a 3x3 matrix from another 3x3 matrix.

operator-

Negate all elements of a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline const Matrix3 operator-();
              }
        }
}
```

Arguments

None

Return Values

3x3 matrix containing negated elements of the specified 3x3 matrix

Description

Negate all elements of a 3x3 matrix.

operator-=

Perform compound assignment and subtraction by a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline Matrix3 &operator-=(
                            const Matrix3 &mat
                     );
              }
        }
}
```

Arguments

mat	3x3 matrix

Return Values

A reference to the resulting 3x3 matrix

Description

Perform compound assignment and subtraction by a 3x3 matrix.

operator=

Assign one 3x3 matrix to another.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline Matrix3 &operator=(
                            const Matrix3 &mat
                     );
              }
        }
}
```

Arguments

mat	3x3 matrix

Return Values

A reference to the resulting 3x3 matrix

Description

Assign one 3x3 matrix to another.

operator[]

Subscripting operator to set or get a column.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline Vector3 &operator[](
                            int col
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-2

Return Values

A reference to indexed column

Description

Subscripting operator invoked when applied to non-const Matrix3.

operator[]

Subscripting operator to get a column.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline const Vector3 operator[](
                            int col
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-2

Return Values

Indexed column

Description

Subscripting operator invoked when applied to const Matrix3.

Public Static Methods

identity

Construct an identity 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     static inline const Matrix3 identity();
              }
        }
}
```

Arguments

None

Return Values

The constructed 3x3 matrix

Description

Construct an identity 3x3 matrix in which non-diagonal elements are zero and diagonal elements are 1.

rotation

Construct a 3x3 matrix to rotate around a unit-length 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     static inline const Matrix3 rotation(
                            vec_float4 radians,
                            const Vector3 &unitVec
                     );
              }
        }
}
```

Arguments

radians	Scalar value
unitVec	3-D vector, expected to be unit-length

Return Values

The constructed 3x3 matrix

Description

Construct a 3x3 matrix to rotate around a unit-length 3-D vector by the specified radians angle.

rotation

Construct a rotation matrix from a unit-length quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     static inline const Matrix3 rotation(
                            const Quat &unitQuat
                     );
              }
        }
}
```

Arguments

unitQuat	Quaternion, expected to be unit-length

Return Values

The constructed 3x3 matrix

Description

Construct a 3x3 matrix that applies the same rotation as the specified unit-length quaternion.

rotationX

Construct a 3x3 matrix to rotate around the x axis.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     static inline const Matrix3 rotationX(
                            vec_float4 radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value

Return Values

The constructed 3x3 matrix

Description

Construct a 3x3 matrix to rotate around the x axis by the specified radians angle.

rotationY

Construct a 3x3 matrix to rotate around the y axis.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     static inline const Matrix3 rotationY(
                            vec_float4 radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value

Return Values

The constructed 3x3 matrix

Description

Construct a 3x3 matrix to rotate around the y axis by the specified radians angle.

rotationZ

Construct a 3x3 matrix to rotate around the z axis.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     static inline const Matrix3 rotationZ(
                            vec_float4 radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value

Return Values

The constructed 3x3 matrix

Description

Construct a 3x3 matrix to rotate around the z axis by the specified radians angle.

rotationZYX

Construct a 3x3 matrix to rotate around the x, y, and z axes.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     static inline const Matrix3 rotationZYX(
                            const Vector3 &radiansXYZ
                     );
              }
        }
}
```

Arguments

radiansXYZ	3-D vector

Return Values

The constructed 3x3 matrix

Description

Construct a 3x3 matrix to rotate around the x, y, and z axes by the radians angles contained in a 3-D vector. Equivalent to `rotationZ(radiansXYZ.getZ()) * rotationY(radiansXYZ.getY()) * rotationX(radiansXYZ.getX())`.

scale

Construct a 3x3 matrix to perform scaling.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     static inline const Matrix3 scale(
                            const Vector3 &scaleVec
                     );
              }
        }
}
```

Arguments

scaleVec	3-D vector

Return Values

The constructed 3x3 matrix

Description

Construct a 3x3 matrix to perform scaling, in which the non-diagonal elements are zero and the diagonal elements are set to the elements of *scaleVec*.

Public Instance Methods

get4Aos

Extract four AoS 3x3 matrices.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline void get4Aos(
                            Aos::Matrix3 &result0,
                            Aos::Matrix3 &result1,
                            Aos::Matrix3 &result2,
                            Aos::Matrix3 &result3
                     );
              }
        }
}
```

Arguments

result0	An output AoS 3x3 matrix
result1	An output AoS 3x3 matrix
result2	An output AoS 3x3 matrix
result3	An output AoS 3x3 matrix

Return Values

None

Description

Extract four AoS 3x3 matrices from four slots of an SoA 3x3 matrix (transpose the data format).

getCol

Get the column of a 3x3 matrix referred to by the specified index.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline const Vector3 getCol(
                            int col
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-2

Return Values

The column referred to by the specified index

Description

Get the column of a 3x3 matrix referred to by the specified index.

getCol0

Get column 0 of a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline const Vector3 getCol0();
              }
        }
}
```

Arguments

None

Return Values

Column 0

Description

Get column 0 of a 3x3 matrix.

getCol1

Get column 1 of a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline const Vector3 getCol1();
              }
        }
}
```

Arguments

None

Return Values

Column 1

Description

Get column 1 of a 3x3 matrix.

getCol2

Get column 2 of a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline const Vector3 getCol2();
              }
        }
}
```

Arguments

None

Return Values

Column 2

Description

Get column 2 of a 3x3 matrix.

getElem

Get the element of a 3x3 matrix referred to by column and row indices.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline vec_float4 getElem(
                            int col,
                            int row
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-2
row	Index, expected in the range 0-2

Return Values

Element selected by *col* and *row*

Description

Get the element of a 3x3 matrix referred to by column and row indices.

getRow

Get the row of a 3x3 matrix referred to by the specified index.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline const Vector3 getRow(
                            int row
                     );
              }
        }
}
```

Arguments

row	Index, expected in the range 0-2

Return Values

The row referred to by the specified index

Description

Get the row of a 3x3 matrix referred to by the specified index.

setCol

Set the column of a 3x3 matrix referred to by the specified index.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline Matrix3 &setCol(
                            int col,
                            const Vector3 &vec
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-2
vec	3-D vector

Return Values

A reference to the resulting 3x3 matrix

Description

Set the column of a 3x3 matrix referred to by the specified index.

setCol0

Set column 0 of a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline Matrix3 &setCol0(
                            const Vector3 &col0
                     );
              }
        }
}
```

Arguments

col0	3-D vector

Return Values

A reference to the resulting 3x3 matrix

Description

Set column 0 of a 3x3 matrix.

setCol1

Set column 1 of a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
             class Matrix3 {
                    inline Matrix3 &setCol1(
                           const Vector3 &col1
                    );
             }
        }
}
```

Arguments

col1	3-D vector

Return Values

A reference to the resulting 3x3 matrix

Description

Set column 1 of a 3x3 matrix.

setCol2

Set column 2 of a 3x3 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline Matrix3 &setCol2(
                            const Vector3 &col2
                     );
              }
        }
}
```

Arguments

col2	3-D vector

Return Values

A reference to the resulting 3x3 matrix

Description

Set column 2 of a 3x3 matrix.

setElem

Set the element of a 3x3 matrix referred to by column and row indices.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline Matrix3 &setElem(
                            int col,
                            int row,
                            vec_float4 val
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-2
row	Index, expected in the range 0-2
val	Scalar value

Return Values

A reference to the resulting 3x3 matrix

Description

Set the element of a 3x3 matrix referred to by column and row indices.

setRow

Set the row of a 3x3 matrix referred to by the specified index.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix3 {
                     inline Matrix3 &setRow(
                            int row,
                            const Vector3 &vec
                     );
              }
        }
}
```

Arguments

row	Index, expected in the range 0-2
vec	3-D vector

Return Values

A reference to the resulting 3x3 matrix

Description

Set the row of a 3x3 matrix referred to by the specified index.

Vectormath::Soa::Matrix4

Summary

Vectormath::Soa::Matrix4

A set of four 4x4 matrices in structure-of-arrays format.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
class Matrix4;
```

Description

A class representing a set of four 4x4 matrices stored in structure-of-arrays (SoA) format.

Methods Summary

Methods	Description
frustum	Construct a perspective projection matrix based on frustum.
get4Aos	Extract four AoS 4x4 matrices.
getCol	Get the column of a 4x4 matrix referred to by the specified index.
getCol0	Get column 0 of a 4x4 matrix.
getCol1	Get column 1 of a 4x4 matrix.
getCol2	Get column 2 of a 4x4 matrix.
getCol3	Get column 3 of a 4x4 matrix.
getElem	Get the element of a 4x4 matrix referred to by column and row indices.
getRow	Get the row of a 4x4 matrix referred to by the specified index.
getTranslation	Get the translation component of a 4x4 matrix.
getUpper3x3	Get the upper-left 3x3 submatrix of a 4x4 matrix.
identity	Construct an identity 4x4 matrix.
lookAt	Construct viewing matrix based on eye position, position looked at, and up direction.
Matrix4	Default constructor; does no initialization.
Matrix4	Copy a 4x4 matrix.
Matrix4	Construct a 4x4 matrix containing the specified columns.
Matrix4	Construct a 4x4 matrix from a 3x4 transformation matrix.
Matrix4	Construct a 4x4 matrix from a 3x3 matrix and a 3-D vector.
Matrix4	Construct a 4x4 matrix from a unit-length quaternion and a 3-D vector.
Matrix4	Set all elements of a 4x4 matrix to the same scalar value.
Matrix4	Replicate an AoS 4x4 matrix.
Matrix4	Insert four AoS 4x4 matrices.
operator *	Multiply a 4x4 matrix by a scalar.
operator *	Multiply a 4x4 matrix by a 4-D vector.
operator *	Multiply a 4x4 matrix by a 3-D vector.
operator *	Multiply a 4x4 matrix by a 3-D point.
operator *	Multiply two 4x4 matrices.
operator *	Multiply a 4x4 matrix by a 3x4 transformation matrix.
operator *=	Perform compound assignment and multiplication by a scalar.

Methods	Description
operator *=	Perform compound assignment and multiplication by a 4x4 matrix.
operator *=	Perform compound assignment and multiplication by a 3x4 transformation matrix.
operator+	Add two 4x4 matrices.
operator+=	Perform compound assignment and addition with a 4x4 matrix.
operator-	Subtract a 4x4 matrix from another 4x4 matrix.
operator-	Negate all elements of a 4x4 matrix.
operator-=	Perform compound assignment and subtraction by a 4x4 matrix.
operator=	Assign one 4x4 matrix to another.
operator[]	Subscripting operator to set or get a column.
operator[]	Subscripting operator to get a column.
orthographic	Construct an orthographic projection matrix.
perspective	Construct a perspective projection matrix.
rotation	Construct a 4x4 matrix to rotate around a unit-length 3-D vector.
rotation	Construct a rotation matrix from a unit-length quaternion.
rotationX	Construct a 4x4 matrix to rotate around the x axis.
rotationY	Construct a 4x4 matrix to rotate around the y axis.
rotationZ	Construct a 4x4 matrix to rotate around the z axis.
rotationZYX	Construct a 4x4 matrix to rotate around the x, y, and z axes.
scale	Construct a 4x4 matrix to perform scaling.
setCol	Set the column of a 4x4 matrix referred to by the specified index.
setCol0	Set column 0 of a 4x4 matrix.
setCol1	Set column 1 of a 4x4 matrix.
setCol2	Set column 2 of a 4x4 matrix.
setCol3	Set column 3 of a 4x4 matrix.
setElem	Set the element of a 4x4 matrix referred to by column and row indices.
setRow	Set the row of a 4x4 matrix referred to by the specified index.
setTranslation	Set translation component.
setUpper3x3	Set the upper-left 3x3 submatrix.
translation	Construct a 4x4 matrix to perform translation.

Constructors and Destructors

Matrix4

Default constructor; does no initialization.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline Matrix4();
              }
        }
}
```

Arguments

None

Return Values

None

Description

Default constructor; does no initialization.

Matrix4

Copy a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline Matrix4(
                            const Matrix4 &mat
                     );
              }
        }
}
```

Arguments

mat	4x4 matrix

Return Values

None

Description

Construct a copy of a 4x4 matrix.

Matrix4

Construct a 4x4 matrix containing the specified columns.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline Matrix4(
                            const Vector4 &col0,
                            const Vector4 &col1,
                            const Vector4 &col2,
                            const Vector4 &col3
                     );
              }
        }
}
```

Arguments

col0	4-D vector
col1	4-D vector
col2	4-D vector
col3	4-D vector

Return Values

None

Description

Construct a 4x4 matrix containing the specified columns.

Matrix4

Construct a 4x4 matrix from a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     explicit inline Matrix4(
                            const Transform3 &mat
                     );
              }
        }
}
```

Arguments

mat	3x4 transformation matrix

Return Values

None

Description

Construct a 4x4 matrix whose upper 3x4 elements are equal to the 3x4 transformation matrix argument and whose bottom row is equal to (0,0,0,1).

Matrix4

Construct a 4x4 matrix from a 3x3 matrix and a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline Matrix4(
                            const Matrix3 &mat,
                            const Vector3 &translateVec
                     );
              }
        }
}
```

Arguments

mat	3x3 matrix
translateVec	3-D vector

Return Values

None

Description

Construct a 4x4 matrix whose upper 3x3 elements are equal to the 3x3 matrix argument, whose translation component is equal to the 3-D vector argument, and whose bottom row is (0,0,0,1).

Matrix4

Construct a 4x4 matrix from a unit-length quaternion and a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline Matrix4(
                            const Quat &unitQuat,
                            const Vector3 &translateVec
                     );
              }
        }
}
```

Arguments

unitQuat	Quaternion, expected to be unit-length
translateVec	3-D vector

Return Values

None

Description

Construct a 4x4 matrix whose upper-left 3x3 submatrix is a rotation matrix converted from the unit-length quaternion argument, whose translation component is equal to the 3-D vector argument, and whose bottom row is (0,0,0,1).

Matrix4

Set all elements of a 4x4 matrix to the same scalar value.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     explicit inline Matrix4(
                            vec_float4 scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

None

Description

Construct a 4x4 matrix with all elements set to the scalar value argument.

Matrix4

Replicate an AoS 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline Matrix4(
                            const Aos::Matrix4 &mat
                     );
              }
        }
}
```

Arguments

mat	AoS 4x4 matrix

Return Values

None

Description

Replicate an AoS 4x4 matrix in all four slots of an SoA 4x4 matrix.

Matrix4

Insert four AoS 4x4 matrices.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline Matrix4(
                            const Aos::Matrix4 &mat0,
                            const Aos::Matrix4 &mat1,
                            const Aos::Matrix4 &mat2,
                            const Aos::Matrix4 &mat3
                     );
              }
        }
}
```

Arguments

mat0	AoS 4x4 matrix
mat1	AoS 4x4 matrix
mat2	AoS 4x4 matrix
mat3	AoS 4x4 matrix

Return Values

None

Description

Insert four AoS 4x4 matrices into four slots of an SoA 4x4 matrix (transpose the data format).

Operator Methods

operator *

Multiply a 4x4 matrix by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline const Matrix4 operator *(
                            vec_float4 scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

Product of the specified 4x4 matrix and scalar

Description

Multiply a 4x4 matrix by a scalar.

operator *

Multiply a 4x4 matrix by a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline const Vector4 operator *(
                            const Vector4 &vec
                     );
              }
        }
}
```

Arguments

vec	4-D vector

Return Values

Product of the specified 4x4 matrix and 4-D vector

Description

Multiply a 4x4 matrix by a 4-D vector.

operator *

Multiply a 4x4 matrix by a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline const Vector4 operator *(
                            const Vector3 &vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

Product of the specified 4x4 matrix and 3-D vector

Description

Multiply a 4x4 matrix by a 3-D vector treated as if it were a 4-D vector with the w element equal to 0.

operator *

Multiply a 4x4 matrix by a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline const Vector4 operator *(
                            const Point3 &pnt
                     );
              }
        }
}
```

Arguments

pnt	3-D point

Return Values

Product of the specified 4x4 matrix and 3-D point

Description

Multiply a 4x4 matrix by a 3-D point treated as if it were a 4-D vector with the w element equal to 1.

operator *

Multiply two 4x4 matrices.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline const Matrix4 operator *(
                            const Matrix4 &mat
                     );
              }
        }
}
```

Arguments

mat	4x4 matrix

Return Values

Product of the specified 4x4 matrices

Description

Multiply two 4x4 matrices.

operator *

Multiply a 4x4 matrix by a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline const Matrix4 operator *(
                            const Transform3 &tfrm
                     );
              }
        }
}
```

Arguments

tfrm	3x4 transformation matrix

Return Values

Product of the specified 4x4 matrix and 3x4 transformation matrix

Description

Multiply a 4x4 matrix by a 3x4 transformation matrix treated as if it were a 4x4 matrix with the bottom row equal to (0,0,0,1).

operator *=

Perform compound assignment and multiplication by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline Matrix4 &operator *=(
                            vec_float4 scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

A reference to the resulting 4x4 matrix

Description

Perform compound assignment and multiplication by a scalar.

operator *=

Perform compound assignment and multiplication by a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline Matrix4 &operator *=(
                            const Matrix4 &mat
                     );
              }
        }
}
```

Arguments

mat	4x4 matrix

Return Values

A reference to the resulting 4x4 matrix

Description

Perform compound assignment and multiplication by a 4x4 matrix.

operator *=

Perform compound assignment and multiplication by a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline Matrix4 &operator *=(
                            const Transform3 &tfrm
                     );
              }
        }
}
```

Arguments

tfrm	3x4 transformation matrix

Return Values

A reference to the resulting 4x4 matrix

Description

Perform compound assignment and multiplication by a 3x4 transformation matrix.

operator+

Add two 4x4 matrices.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline const Matrix4 operator+(
                            const Matrix4 &mat
                     );
              }
        }
}
```

Arguments

mat	4x4 matrix

Return Values

Sum of the specified 4x4 matrices

Description

Add two 4x4 matrices.

operator+=

Perform compound assignment and addition with a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline Matrix4 &operator+=(
                            const Matrix4 &mat
                     );
              }
        }
}
```

Arguments

mat	4x4 matrix

Return Values

A reference to the resulting 4x4 matrix

Description

Perform compound assignment and addition with a 4x4 matrix.

operator-

Subtract a 4x4 matrix from another 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline const Matrix4 operator-(
                            const Matrix4 &mat
                     );
              }
        }
}
```

Arguments

mat	4x4 matrix

Return Values

Difference of the specified 4x4 matrices

Description

Subtract a 4x4 matrix from another 4x4 matrix.

operator-

Negate all elements of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline const Matrix4 operator-();
              }
        }
}
```

Arguments

None

Return Values

4x4 matrix containing negated elements of the specified 4x4 matrix

Description

Negate all elements of a 4x4 matrix.

operator-=

Perform compound assignment and subtraction by a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline Matrix4 &operator-=(
                            const Matrix4 &mat
                     );
              }
        }
}
```

Arguments

mat	4x4 matrix

Return Values

A reference to the resulting 4x4 matrix

Description

Perform compound assignment and subtraction by a 4x4 matrix.

operator=

Assign one 4x4 matrix to another.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline Matrix4 &operator=(
                            const Matrix4 &mat
                     );
              }
        }
}
```

Arguments

mat	4x4 matrix

Return Values

A reference to the resulting 4x4 matrix

Description

Assign one 4x4 matrix to another.

operator[]

Subscripting operator to set or get a column.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline Vector4 &operator[](
                            int col
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-3

Return Values

A reference to indexed column

Description

Subscripting operator invoked when applied to non-const Matrix4.

operator[]

Subscripting operator to get a column.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline const Vector4 operator[](
                            int col
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-3

Return Values

Indexed column

Description

Subscripting operator invoked when applied to const Matrix4.

Public Static Methods

frustum

Construct a perspective projection matrix based on frustum.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     static inline const Matrix4 frustum(
                            vec_float4 left,
                            vec_float4 right,
                            vec_float4 bottom,
                            vec_float4 top,
                            vec_float4 zNear,
                            vec_float4 zFar
                     );
              }
        }
}
```

Arguments

left	Scalar value
right	Scalar value
bottom	Scalar value
top	Scalar value
zNear	Scalar value
zFar	Scalar value

Return Values

The constructed 4x4 matrix

Description

Construct a perspective projection matrix based on frustum, equal to:

```
2*zNear/(right-left)   0         (right+left)/(right-left)      0
            0      2*zNear/(top-bottom)  (top+bottom)/(top-bottom)      0
            0              0        -(zFar+zNear)/(zFar-zNear)
-2*zFar*zNear/(zFar-zNear)
            0              0                  -1                0       .
```

identity

Construct an identity 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     static inline const Matrix4 identity();
              }
        }
}
```

Arguments

None

Return Values

The constructed 4x4 matrix

Description

Construct an identity 4x4 matrix in which non-diagonal elements are zero and diagonal elements are 1.

lookAt

Construct viewing matrix based on eye position, position looked at, and up direction.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     static inline const Matrix4 lookAt(
                            const Point3 &eyePos,
                            const Point3 &lookAtPos,
                            const Vector3 &upVec
                     );
              }
        }
}
```

Arguments

eyePos	3-D point
lookAtPos	3-D point
upVec	3-D vector

Return Values

The constructed 4x4 matrix

Description

Construct the inverse of a coordinate frame that is centered at the eye position, with z axis directed away from lookAtPos, and y axis oriented to best match the up direction.

orthographic

Construct an orthographic projection matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     static inline const Matrix4 orthographic(
                            vec_float4 left,
                            vec_float4 right,
                            vec_float4 bottom,
                            vec_float4 top,
                            vec_float4 zNear,
                            vec_float4 zFar
                     );
              }
        }
}
```

Arguments

left	Scalar value
right	Scalar value
bottom	Scalar value
top	Scalar value
zNear	Scalar value
zFar	Scalar value

Return Values

The constructed 4x4 matrix

Description

Construct an orthographic projection matrix, equal to

```
        2/(right-left)       0            0        -(right+left)/(right-left)
            0        2/(top-bottom)       0        -(top+bottom)/(top-bottom)
            0              0       -2/(zFar-zNear) -(zFar+zNear)/(zFar-zNear)
            0              0            0                 1        .
```

perspective

Construct a perspective projection matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     static inline const Matrix4 perspective(
                            vec_float4 fovyRadians,
                            vec_float4 aspect,
                            vec_float4 zNear,
                            vec_float4 zFar
                     );
              }
        }
}
```

Arguments

fovyRadians	Scalar value
aspect	Scalar value
zNear	Scalar value
zFar	Scalar value

Return Values

The constructed 4x4 matrix

Description

Construct a perspective projection matrix, equal to:

```
cot(fovyRadians/2)/aspect   0                0                 0
            0            cot(fovyRadians/2)       0                 0
            0                  0   (zFar+zNear)/(zNear-zFar)
2*zFar*zNear/(zNear-zFar)
            0                  0            -1                 0        .
```

rotation

Construct a 4x4 matrix to rotate around a unit-length 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     static inline const Matrix4 rotation(
                            vec_float4 radians,
                            const Vector3 &unitVec
                     );
              }
        }
}
```

Arguments

radians	Scalar value
unitVec	3-D vector, expected to be unit-length

Return Values

The constructed 4x4 matrix

Description

Construct a 4x4 matrix to rotate around a unit-length 3-D vector by the specified radians angle.

rotation

Construct a rotation matrix from a unit-length quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     static inline const Matrix4 rotation(
                            const Quat &unitQuat
                     );
              }
        }
}
```

Arguments

unitQuat	Quaternion, expected to be unit-length

Return Values

The constructed 4x4 matrix

Description

Construct a 4x4 matrix that applies the same rotation as the specified unit-length quaternion.

rotationX

Construct a 4x4 matrix to rotate around the x axis.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     static inline const Matrix4 rotationX(
                            vec_float4 radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value

Return Values

The constructed 4x4 matrix

Description

Construct a 4x4 matrix to rotate around the x axis by the specified radians angle.

rotationY

Construct a 4x4 matrix to rotate around the y axis.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     static inline const Matrix4 rotationY(
                            vec_float4 radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value

Return Values

The constructed 4x4 matrix

Description

Construct a 4x4 matrix to rotate around the y axis by the specified radians angle.

rotationZ

Construct a 4x4 matrix to rotate around the z axis.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     static inline const Matrix4 rotationZ(
                            vec_float4 radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value

Return Values

The constructed 4x4 matrix

Description

Construct a 4x4 matrix to rotate around the z axis by the specified radians angle.

rotationZYX

Construct a 4x4 matrix to rotate around the x, y, and z axes.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     static inline const Matrix4 rotationZYX(
                            const Vector3 &radiansXYZ
                     );
              }
        }
}
```

Arguments

radiansXYZ	3-D vector

Return Values

The constructed 4x4 matrix

Description

Construct a 4x4 matrix to rotate around the x, y, and z axes by the radians angles contained in a 3-D vector. Equivalent to `rotationZ(radiansXYZ.getZ()) * rotationY(radiansXYZ.getY()) * rotationX(radiansXYZ.getX())`.

scale

Construct a 4x4 matrix to perform scaling.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
                class Matrix4 {
                        static inline const Matrix4 scale(
                                const Vector3 &scaleVec
                        );
                }
        }
}
```

Arguments

scaleVec	3-D vector

Return Values

The constructed 4x4 matrix

Description

Construct a 4x4 matrix to perform scaling, in which the non-diagonal elements are zero and the diagonal elements are set to the elements of *scaleVec*.

translation

Construct a 4x4 matrix to perform translation.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     static inline const Matrix4 translation(
                            const Vector3 &translateVec
                     );
              }
        }
}
```

Arguments

translateVec	3-D vector

Return Values

The constructed 4x4 matrix

Description

Construct a 4x4 matrix to perform translation, which is an identity matrix except for the translation component, with coordinates equal to those in *translateVec*.

Public Instance Methods

get4Aos

Extract four AoS 4x4 matrices.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline void get4Aos(
                            Aos::Matrix4 &result0,
                            Aos::Matrix4 &result1,
                            Aos::Matrix4 &result2,
                            Aos::Matrix4 &result3
                     );
              }
        }
}
```

Arguments

result0	An output AoS 4x4 matrix
result1	An output AoS 4x4 matrix
result2	An output AoS 4x4 matrix
result3	An output AoS 4x4 matrix

Return Values

None

Description

Extract four AoS 4x4 matrices from four slots of an SoA 4x4 matrix (transpose the data format).

getCol

Get the column of a 4x4 matrix referred to by the specified index.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline const Vector4 getCol(
                            int col
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-3

Return Values

The column referred to by the specified index

Description

Get the column of a 4x4 matrix referred to by the specified index.

getCol0

Get column 0 of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline const Vector4 getCol0();
              }
        }
}
```

Arguments

None

Return Values

Column 0

Description

Get column 0 of a 4x4 matrix.

getCol1

Get column 1 of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline const Vector4 getCol1();
              }
        }
}
```

Arguments

None

Return Values

Column 1

Description

Get column 1 of a 4x4 matrix.

getCol2

Get column 2 of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline const Vector4 getCol2();
              }
        }
}
```

Arguments

None

Return Values

Column 2

Description

Get column 2 of a 4x4 matrix.

getCol3

Get column 3 of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline const Vector4 getCol3();
              }
        }
}
```

Arguments

None

Return Values

Column 3

Description

Get column 3 of a 4x4 matrix.

getElem

Get the element of a 4x4 matrix referred to by column and row indices.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline vec_float4 getElem(
                            int col,
                            int row
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-3
row	Index, expected in the range 0-3

Return Values

Element selected by *col* and *row*

Description

Get the element of a 4x4 matrix referred to by column and row indices.

getRow

Get the row of a 4x4 matrix referred to by the specified index.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline const Vector4 getRow(
                            int row
                     );
              }
        }
}
```

Arguments

row	Index, expected in the range 0-3

Return Values

The row referred to by the specified index

Description

Get the row of a 4x4 matrix referred to by the specified index.

getTranslation

Get the translation component of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline const Vector3 getTranslation();
              }
        }
}
```

Arguments

None

Return Values

Translation component

Description

Get the translation component of a 4x4 matrix.

getUpper3x3

Get the upper-left 3x3 submatrix of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline const Matrix3 getUpper3x3();
              }
        }
}
```

Arguments

None

Return Values

Upper-left 3x3 submatrix

Description

Get the upper-left 3x3 submatrix of a 4x4 matrix.

setCol

Set the column of a 4x4 matrix referred to by the specified index.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline Matrix4 &setCol(
                            int col,
                            const Vector4 &vec
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-3
vec	4-D vector

Return Values

A reference to the resulting 4x4 matrix

Description

Set the column of a 4x4 matrix referred to by the specified index.

setCol0

Set column 0 of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline Matrix4 &setCol0(
                            const Vector4 &col0
                     );
              }
        }
}
```

Arguments

col0	4-D vector

Return Values

A reference to the resulting 4x4 matrix

Description

Set column 0 of a 4x4 matrix.

setCol1

Set column 1 of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline Matrix4 &setCol1(
                            const Vector4 &col1
                     );
              }
        }
}
```

Arguments

col1	4-D vector

Return Values

A reference to the resulting 4x4 matrix

Description

Set column 1 of a 4x4 matrix.

setCol2

Set column 2 of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline Matrix4 &setCol2(
                            const Vector4 &col2
                     );
              }
        }
}
```

Arguments

col2	4-D vector

Return Values

A reference to the resulting 4x4 matrix

Description

Set column 2 of a 4x4 matrix.

setCol3

Set column 3 of a 4x4 matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline Matrix4 &setCol3(
                            const Vector4 &col3
                     );
              }
        }
}
```

Arguments

col3	4-D vector

Return Values

A reference to the resulting 4x4 matrix

Description

Set column 3 of a 4x4 matrix.

setElem

Set the element of a 4x4 matrix referred to by column and row indices.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline Matrix4 &setElem(
                            int col,
                            int row,
                            vec_float4 val
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-3
row	Index, expected in the range 0-3
val	Scalar value

Return Values

A reference to the resulting 4x4 matrix

Description

Set the element of a 4x4 matrix referred to by column and row indices.

setRow

Set the row of a 4x4 matrix referred to by the specified index.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline Matrix4 &setRow(
                            int row,
                            const Vector4 &vec
                     );
              }
        }
}
```

Arguments

row	Index, expected in the range 0-3
vec	4-D vector

Return Values

A reference to the resulting 4x4 matrix

Description

Set the row of a 4x4 matrix referred to by the specified index.

setTranslation

Set translation component.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline Matrix4 &setTranslation(
                            const Vector3 &translateVec
                     );
              }
        }
}
```

Arguments

translateVec	3-D vector

Return Values

A reference to the resulting 4x4 matrix

Description

Set the translation component of a 4x4 matrix equal to the specified 3-D vector.

Notes

This function does not change the bottom row elements.

setUpper3x3

Set the upper-left 3x3 submatrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Matrix4 {
                     inline Matrix4 &setUpper3x3(
                            const Matrix3 &mat3
                     );
              }
        }
}
```

Arguments

mat3	3x3 matrix

Return Values

A reference to the resulting 4x4 matrix

Description

Set the upper-left 3x3 submatrix elements of a 4x4 matrix equal to the specified 3x3 matrix.

Notes

This function does not change the bottom row elements.

Vectormath::Soa::Point3

Summary

Vectormath::Soa::Point3

A set of four 3-D points in structure-of-arrays format.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
class Point3;
```

Description

A class representing a set of four 3-D points stored in structure-of-arrays (SoA) format.

Methods Summary

Methods	Description
get4Aos	Extract four AoS 3-D points.
getElem	Get an x, y, or z element of a 3-D point by index.
getX	Get the x element of a 3-D point.
getY	Get the y element of a 3-D point.
getZ	Get the z element of a 3-D point.
operator+	Add a 3-D point to a 3-D vector.
operator+=	Perform compound assignment and addition with a 3-D vector.
operator-	Subtract a 3-D point from another 3-D point.
operator-	Subtract a 3-D vector from a 3-D point.
operator-=	Perform compound assignment and subtraction by a 3-D vector.
operator=	Assign one 3-D point to another.
operator[]	Subscripting operator to set or get an element.
operator[]	Subscripting operator to get an element.
Point3	Default constructor; does no initialization.
Point3	Copy a 3-D point.
Point3	Construct a 3-D point from x, y, and z elements.
Point3	Copy elements from a 3-D vector into a 3-D point.
Point3	Set all elements of a 3-D point to the same scalar value.
Point3	Replicate an AoS 3-D point.
Point3	Insert four AoS 3-D points.
setElem	Set an x, y, or z element of a 3-D point by index.
setX	Set the x element of a 3-D point.
setY	Set the y element of a 3-D point.
setZ	Set the z element of a 3-D point.

Constructors and Destructors

Point3

Default constructor; does no initialization.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Point3 {
                     inline Point3();
              }
        }
}
```

Arguments

None

Return Values

None

Description

Default constructor; does no initialization.

Point3

Copy a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Point3 {
                     inline Point3(
                            const Point3 &pnt
                     );
              }
        }
}
```

Arguments

pnt	3-D point

Return Values

None

Description

Construct a copy of a 3-D point.

Point3

Construct a 3-D point from x, y, and z elements.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Point3 {
                     inline Point3(
                            vec_float4 x,
                            vec_float4 y,
                            vec_float4 z
                     );
              }
        }
}
```

Arguments

x	Scalar value
y	Scalar value
z	Scalar value

Return Values

None

Description

Construct a 3-D point containing the specified x, y, and z elements.

Point3

Copy elements from a 3-D vector into a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Point3 {
                     explicit inline Point3(
                            const Vector3 &vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

None

Description

Construct a 3-D point containing the x, y, and z elements of the specified 3-D vector.

Point3

Set all elements of a 3-D point to the same scalar value.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Point3 {
                     explicit inline Point3(
                            vec_float4 scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

None

Description

Construct a 3-D point with all elements set to the scalar value argument.

Point3

Replicate an AoS 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Point3 {
                     inline Point3(
                            Aos::Point3 pnt
                     );
              }
        }
}
```

Arguments

pnt	AoS 3-D point

Return Values

None

Description

Replicate an AoS 3-D point in all four slots of an SoA 3-D point.

Point3

Insert four AoS 3-D points.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Point3 {
                     inline Point3(
                            Aos::Point3 pnt0,
                            Aos::Point3 pnt1,
                            Aos::Point3 pnt2,
                            Aos::Point3 pnt3
                     );
              }
        }
}
```

Arguments

pnt0	AoS 3-D point
pnt1	AoS 3-D point
pnt2	AoS 3-D point
pnt3	AoS 3-D point

Return Values

None

Description

Insert four AoS 3-D points into four slots of an SoA 3-D point (transpose the data format).

Operator Methods

operator+

Add a 3-D point to a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Point3 {
                     inline const Point3 operator+(
                            const Vector3 &vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

Sum of the specified 3-D point and 3-D vector

Description

Add a 3-D point to a 3-D vector.

operator+=

Perform compound assignment and addition with a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Point3 {
                     inline Point3 &operator+=(
                            const Vector3 &vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

A reference to the resulting 3-D point

Description

Perform compound assignment and addition with a 3-D vector.

operator-

Subtract a 3-D point from another 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Point3 {
                     inline const Vector3 operator-(
                            const Point3 &pnt
                     );
              }
        }
}
```

Arguments

pnt	3-D point

Return Values

Difference of the specified 3-D points

Description

Subtract a 3-D point from another 3-D point.

operator-

Subtract a 3-D vector from a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Point3 {
                     inline const Point3 operator-(
                            const Vector3 &vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

Difference of the specified 3-D point and 3-D vector

Description

Subtract a 3-D vector from a 3-D point.

operator-=

Perform compound assignment and subtraction by a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Point3 {
                     inline Point3 &operator-=(
                            const Vector3 &vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

A reference to the resulting 3-D point

Description

Perform compound assignment and subtraction by a 3-D vector.

operator=

Assign one 3-D point to another.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Point3 {
                     inline Point3 &operator=(
                            const Point3 &pnt
                     );
              }
        }
}
```

Arguments

pnt	3-D point

Return Values

A reference to the resulting 3-D point

Description

Assign one 3-D point to another.

operator[]

Subscripting operator to set or get an element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Point3 {
                     inline vec_float4 &operator[](
                            int idx
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-2

Return Values

A reference to indexed element

Description

Subscripting operator invoked when applied to non-const Point3.

operator[]

Subscripting operator to get an element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Point3 {
                     inline vec_float4 operator[](
                            int idx
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-2

Return Values

Indexed element

Description

Subscripting operator invoked when applied to const Point3.

Public Instance Methods

get4Aos

Extract four AoS 3-D points.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Point3 {
                     inline void get4Aos(
                            Aos::Point3 &result0,
                            Aos::Point3 &result1,
                            Aos::Point3 &result2,
                            Aos::Point3 &result3
                     );
              }
        }
}
```

Arguments

result0	An output AoS 3-D point
result1	An output AoS 3-D point
result2	An output AoS 3-D point
result3	An output AoS 3-D point

Return Values

None

Description

Extract four AoS 3-D points from four slots of an SoA 3-D point (transpose the data format).

getElem

Get an x, y, or z element of a 3-D point by index.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Point3 {
                     inline vec_float4 getElem(
                            int idx
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-2

Return Values

Element selected by the specified index

Description

Get an x, y, or z element of a 3-D point by specifying an index of 0, 1, or 2, respectively.

getX

Get the x element of a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Point3 {
                     inline vec_float4 getX();
              }
        }
}
```

Arguments

None

Return Values

x element of a 3-D point

Description

Get the x element of a 3-D point.

getY

Get the y element of a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Point3 {
                     inline vec_float4 getY();
              }
        }
}
```

Arguments

None

Return Values

y element of a 3-D point

Description

Get the y element of a 3-D point.

getZ

Get the z element of a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Point3 {
                     inline vec_float4 getZ();
              }
        }
}
```

Arguments

None

Return Values

z element of a 3-D point

Description

Get the z element of a 3-D point.

setElem

Set an x, y, or z element of a 3-D point by index.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Point3 {
                     inline Point3 &setElem(
                            int idx,
                            vec_float4 value
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-2
value	Scalar value

Return Values

A reference to the resulting 3-D point

Description

Set an x, y, or z element of a 3-D point by specifying an index of 0, 1, or 2, respectively.

setX

Set the x element of a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Point3 {
                     inline Point3 &setX(
                            vec_float4 x
                     );
              }
        }
}
```

Arguments

x	Scalar value

Return Values

A reference to the resulting 3-D point

Description

Set the x element of a 3-D point to the specified scalar value.

setY

Set the y element of a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Point3 {
                     inline Point3 &setY(
                            vec_float4 y
                     );
              }
        }
}
```

Arguments

y	Scalar value

Return Values

A reference to the resulting 3-D point

Description

Set the y element of a 3-D point to the specified scalar value.

setZ

Set the z element of a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Point3 {
                     inline Point3 &setZ(
                            vec_float4 z
                     );
              }
        }
}
```

Arguments

z	Scalar value

Return Values

A reference to the resulting 3-D point

Description

Set the z element of a 3-D point to the specified scalar value.

Vectormath::Soa::Quat

Summary

Vectormath::Soa::Quat

A set of four quaternions in structure-of-arrays format.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
class Quat;
```

Description

A class representing a set of four quaternions stored in structure-of-arrays (SoA) format.

Methods Summary

Methods	Description
get4Aos	Extract four AoS quaternions.
getElem	Get an x, y, z, or w element of a quaternion by index.
getW	Get the w element of a quaternion.
getX	Get the x element of a quaternion.
getXYZ	Get the x, y, and z elements of a quaternion.
getY	Get the y element of a quaternion.
getZ	Get the z element of a quaternion.
identity	Construct an identity quaternion.
operator *	Multiply two quaternions.
operator *	Multiply a quaternion by a scalar.
operator *=	Perform compound assignment and multiplication by a quaternion.
operator *=	Perform compound assignment and multiplication by a scalar.
operator+	Add two quaternions.
operator+=	Perform compound assignment and addition with a quaternion.
operator-	Subtract a quaternion from another quaternion.
operator-	Negate all elements of a quaternion.
operator-=	Perform compound assignment and subtraction by a quaternion.
operator/	Divide a quaternion by a scalar.
operator/=	Perform compound assignment and division by a scalar.
operator=	Assign one quaternion to another.
operator[]	Subscripting operator to set or get an element.
operator[]	Subscripting operator to get an element.
Quat	Default constructor; does no initialization.
Quat	Copy a quaternion.
Quat	Construct a quaternion from x, y, z, and w elements.
Quat	Construct a quaternion from a 3-D vector and a scalar.
Quat	Copy elements from a 4-D vector into a quaternion.
Quat	Convert a rotation matrix to a unit-length quaternion.
Quat	Set all elements of a quaternion to the same scalar value.
Quat	Replicate an AoS quaternion.

Methods	Description
Quat	Insert four AoS quaternions.
rotation	Construct a quaternion to rotate between two unit-length 3-D vectors.
rotation	Construct a quaternion to rotate around a unit-length 3-D vector.
rotationX	Construct a quaternion to rotate around the x axis.
rotationY	Construct a quaternion to rotate around the y axis.
rotationZ	Construct a quaternion to rotate around the z axis.
setElem	Set an x, y, z, or w element of a quaternion by index.
setW	Set the w element of a quaternion.
setX	Set the x element of a quaternion.
setXYZ	Set the x, y, and z elements of a quaternion.
setY	Set the y element of a quaternion.
setZ	Set the z element of a quaternion.

Constructors and Destructors

Quat

Default constructor; does no initialization.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline Quat();
              }
        }
}
```

Arguments

None

Return Values

None

Description

Default constructor; does no initialization.

Quat

Copy a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline Quat(
                            const Quat &quat
                     );
              }
        }
}
```

Arguments

quat	Quaternion

Return Values

None

Description

Construct a copy of a quaternion.

Quat

Construct a quaternion from x, y, z, and w elements.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline Quat(
                            vec_float4 x,
                            vec_float4 y,
                            vec_float4 z,
                            vec_float4 w
                     );
              }
        }
}
```

Arguments

x	Scalar value
y	Scalar value
z	Scalar value
w	Scalar value

Return Values

None

Description

Construct a quaternion containing the specified x, y, z, and w elements.

Quat

Construct a quaternion from a 3-D vector and a scalar.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline Quat(
                            const Vector3 &xyz,
                            vec_float4 w
                     );
              }
        }
}
```

Arguments

xyz	3-D vector
w	Scalar value

Return Values

None

Description

Construct a quaternion with the x, y, and z elements of the specified 3-D vector and with the w element set to the specified scalar.

Quat

Copy elements from a 4-D vector into a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     explicit inline Quat(
                            const Vector4 &vec
                     );
              }
        }
}
```

Arguments

vec	4-D vector

Return Values

None

Description

Construct a quaternion containing the x, y, z, and w elements of the specified 4-D vector.

Quat

Convert a rotation matrix to a unit-length quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     explicit inline Quat(
                            const Matrix3 &rotMat
                     );
              }
        }
}
```

Arguments

rotMat	3x3 matrix, expected to be a rotation matrix

Return Values

None

Description

Construct a unit-length quaternion representing the same transformation as a rotation matrix.

Quat

Set all elements of a quaternion to the same scalar value.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     explicit inline Quat(
                            vec_float4 scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

None

Description

Construct a quaternion with all elements set to the scalar value argument.

Quat

Replicate an AoS quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline Quat(
                            Aos::Quat quat
                     );
              }
        }
}
```

Arguments

quat	AoS quaternion

Return Values

None

Description

Replicate an AoS quaternion in all four slots of an SoA quaternion.

Quat

Insert four AoS quaternions.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline Quat(
                            Aos::Quat quat0,
                            Aos::Quat quat1,
                            Aos::Quat quat2,
                            Aos::Quat quat3
                     );
              }
        }
}
```

Arguments

quat0	AoS quaternion
quat1	AoS quaternion
quat2	AoS quaternion
quat3	AoS quaternion

Return Values

None

Description

Insert four AoS quaternions into four slots of an SoA quaternion (transpose the data format).

Operator Methods

operator *

Multiply two quaternions.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline const Quat operator *(
                            const Quat &quat
                     );
              }
        }
}
```

Arguments

quat	Quaternion

Return Values

Product of the specified quaternions

Description

Multiply two quaternions.

operator *

Multiply a quaternion by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline const Quat operator *(
                            vec_float4 scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

Product of the specified quaternion and scalar

Description

Multiply a quaternion by a scalar.

operator *=

Perform compound assignment and multiplication by a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline Quat &operator *=(
                            const Quat &quat
                     );
              }
        }
}
```

Arguments

quat	Quaternion

Return Values

A reference to the resulting quaternion

Description

Perform compound assignment and multiplication by a quaternion.

operator *=

Perform compound assignment and multiplication by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline Quat &operator *=(
                            vec_float4 scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

A reference to the resulting quaternion

Description

Perform compound assignment and multiplication by a scalar.

operator+

Add two quaternions.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline const Quat operator+(
                            const Quat &quat
                     );
              }
        }
}
```

Arguments

quat	Quaternion

Return Values

Sum of the specified quaternions

Description

Add two quaternions.

operator+=

Perform compound assignment and addition with a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline Quat &operator+=(
                            const Quat &quat
                     );
              }
        }
}
```

Arguments

quat	Quaternion

Return Values

A reference to the resulting quaternion

Description

Perform compound assignment and addition with a quaternion.

operator-

Subtract a quaternion from another quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline const Quat operator-(
                            const Quat &quat
                     );
              }
        }
}
```

Arguments

quat	Quaternion

Return Values

Difference of the specified quaternions

Description

Subtract a quaternion from another quaternion.

operator-

Negate all elements of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline const Quat operator-();
              }
        }
}
```

Arguments

None

Return Values

Quaternion containing negated elements of the specified quaternion

Description

Negate all elements of a quaternion.

operator-=

Perform compound assignment and subtraction by a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline Quat &operator-=(
                            const Quat &quat
                     );
              }
        }
}
```

Arguments

quat	Quaternion

Return Values

A reference to the resulting quaternion

Description

Perform compound assignment and subtraction by a quaternion.

operator/

Divide a quaternion by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline const Quat operator/(
                            vec_float4 scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

Quotient of the specified quaternion and scalar

Description

Divide a quaternion by a scalar.

operator/=

Perform compound assignment and division by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline Quat &operator/=(
                            vec_float4 scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

A reference to the resulting quaternion

Description

Perform compound assignment and division by a scalar.

operator=

Assign one quaternion to another.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline Quat &operator=(
                            const Quat &quat
                     );
              }
        }
}
```

Arguments

quat	Quaternion

Return Values

A reference to the resulting quaternion

Description

Assign one quaternion to another.

operator[]

Subscripting operator to set or get an element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline vec_float4 &operator[](
                            int idx
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-3

Return Values

A reference to indexed element

Description

Subscripting operator invoked when applied to non-const Quat.

operator[]

Subscripting operator to get an element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline vec_float4 operator[](
                            int idx
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-3

Return Values

Indexed element

Description

Subscripting operator invoked when applied to const Quat.

Public Static Methods

identity

Construct an identity quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     static inline const Quat identity();
              }
        }
}
```

Arguments

None

Return Values

The constructed quaternion

Description

Construct an identity quaternion equal to (0,0,0,1).

rotation

Construct a quaternion to rotate between two unit-length 3-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     static inline const Quat rotation(
                            const Vector3 &unitVec0,
                            const Vector3 &unitVec1
                     );
              }
        }
}
```

Arguments

unitVec0	3-D vector, expected to be unit-length
unitVec1	3-D vector, expected to be unit-length

Return Values

The constructed quaternion

Description

Construct a quaternion to rotate between two unit-length 3-D vectors.

Notes

The result is unpredictable if *unitVec0* and *unitVec1* point in opposite directions.

rotation

Construct a quaternion to rotate around a unit-length 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     static inline const Quat rotation(
                            vec_float4 radians,
                            const Vector3 &unitVec
                     );
              }
        }
}
```

Arguments

radians	Scalar value
unitVec	3-D vector, expected to be unit-length

Return Values

The constructed quaternion

Description

Construct a quaternion to rotate around a unit-length 3-D vector by the specified radians angle.

rotationX

Construct a quaternion to rotate around the x axis.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     static inline const Quat rotationX(
                            vec_float4 radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value

Return Values

The constructed quaternion

Description

Construct a quaternion to rotate around the x axis by the specified radians angle.

rotationY

Construct a quaternion to rotate around the y axis.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     static inline const Quat rotationY(
                            vec_float4 radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value

Return Values

The constructed quaternion

Description

Construct a quaternion to rotate around the y axis by the specified radians angle.

rotationZ

Construct a quaternion to rotate around the z axis.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     static inline const Quat rotationZ(
                            vec_float4 radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value

Return Values

The constructed quaternion

Description

Construct a quaternion to rotate around the z axis by the specified radians angle.

Public Instance Methods

get4Aos

Extract four AoS quaternions.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline void get4Aos(
                            Aos::Quat &result0,
                            Aos::Quat &result1,
                            Aos::Quat &result2,
                            Aos::Quat &result3
                     );
              }
        }
}
```

Arguments

result0	An output AoS quaternion
result1	An output AoS quaternion
result2	An output AoS quaternion
result3	An output AoS quaternion

Return Values

None

Description

Extract four AoS quaternions from four slots of an SoA quaternion (transpose the data format).

getElem

Get an x, y, z, or w element of a quaternion by index.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline vec_float4 getElem(
                            int idx
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-3

Return Values

Element selected by the specified index

Description

Get an x, y, z, or w element of a quaternion by specifying an index of 0, 1, 2, or 3, respectively.

getW

Get the w element of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline vec_float4 getW();
              }
        }
}
```

Arguments

None

Return Values

w element of a quaternion

Description

Get the w element of a quaternion.

getX

Get the x element of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline vec_float4 getX();
              }
        }
}
```

Arguments

None

Return Values

x element of a quaternion

Description

Get the x element of a quaternion.

getXYZ

Get the x, y, and z elements of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline const Vector3 getXYZ();
              }
        }
}
```

Arguments

None

Return Values

3-D vector containing x, y, and z elements

Description

Extract a quaternion's x, y, and z elements into a 3-D vector.

getY

Get the y element of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline vec_float4 getY();
              }
        }
}
```

Arguments

None

Return Values

y element of a quaternion

Description

Get the y element of a quaternion.

getZ

Get the z element of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline vec_float4 getZ();
              }
        }
}
```

Arguments

None

Return Values

z element of a quaternion

Description

Get the z element of a quaternion.

setElem

Set an x, y, z, or w element of a quaternion by index.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline Quat &setElem(
                            int idx,
                            vec_float4 value
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-3
value	Scalar value

Return Values

A reference to the resulting quaternion

Description

Set an x, y, z, or w element of a quaternion by specifying an index of 0, 1, 2, or 3, respectively.

setW

Set the w element of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline Quat &setW(
                            vec_float4 w
                     );
              }
        }
}
```

Arguments

w	Scalar value

Return Values

A reference to the resulting quaternion

Description

Set the w element of a quaternion to the specified scalar value.

setX

Set the x element of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline Quat &setX(
                            vec_float4 x
                     );
              }
        }
}
```

Arguments

x	Scalar value

Return Values

A reference to the resulting quaternion

Description

Set the x element of a quaternion to the specified scalar value.

setXYZ

Set the x, y, and z elements of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline Quat &setXYZ(
                            const Vector3 &vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

A reference to the resulting quaternion

Description

Set the x, y, and z elements to those of the specified 3-D vector.

Notes

This function does not change the w element.

setY

Set the y element of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline Quat &setY(
                            vec_float4 y
                     );
              }
        }
}
```

Arguments

y	Scalar value

Return Values

A reference to the resulting quaternion

Description

Set the y element of a quaternion to the specified scalar value.

setZ

Set the z element of a quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Quat {
                     inline Quat &setZ(
                            vec_float4 z
                     );
              }
        }
}
```

Arguments

z	Scalar value

Return Values

A reference to the resulting quaternion

Description

Set the z element of a quaternion to the specified scalar value.

Vectormath::Soa::Transform3

Summary

Vectormath::Soa::Transform3

A set of four 3x4 transformation matrices in structure-of-arrays format.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
class Transform3;
```

Description

A class representing a set of four 3x4 transformation matrices stored in structure-of-arrays (SoA) format.

Methods Summary

Methods	Description
get4Aos	Extract four AoS 3x4 transformation matrices.
getCol	Get the column of a 3x4 transformation matrix referred to by the specified index.
getCol0	Get column 0 of a 3x4 transformation matrix.
getCol1	Get column 1 of a 3x4 transformation matrix.
getCol2	Get column 2 of a 3x4 transformation matrix.
getCol3	Get column 3 of a 3x4 transformation matrix.
getElem	Get the element of a 3x4 transformation matrix referred to by column and row indices.
getRow	Get the row of a 3x4 transformation matrix referred to by the specified index.
getTranslation	Get the translation component of a 3x4 transformation matrix.
getUpper3x3	Get the upper-left 3x3 submatrix of a 3x4 transformation matrix.
identity	Construct an identity 3x4 transformation matrix.
operator *	Multiply a 3x4 transformation matrix by a 3-D vector.
operator *	Multiply a 3x4 transformation matrix by a 3-D point.
operator *	Multiply two 3x4 transformation matrices.
operator *=	Perform compound assignment and multiplication by a 3x4 transformation matrix.
operator=	Assign one 3x4 transformation matrix to another.
operator[]	Subscripting operator to set or get a column.
operator[]	Subscripting operator to get a column.
rotation	Construct a 3x4 transformation matrix to rotate around a unit-length 3-D vector.
rotation	Construct a rotation matrix from a unit-length quaternion.
rotationX	Construct a 3x4 transformation matrix to rotate around the x axis.
rotationY	Construct a 3x4 transformation matrix to rotate around the y axis.
rotationZ	Construct a 3x4 transformation matrix to rotate around the z axis.

Methods	Description
rotationZYX	Construct a 3x4 transformation matrix to rotate around the x, y, and z axes.
scale	Construct a 3x4 transformation matrix to perform scaling.
setCol	Set the column of a 3x4 transformation matrix referred to by the specified index.
setCol0	Set column 0 of a 3x4 transformation matrix.
setCol1	Set column 1 of a 3x4 transformation matrix.
setCol2	Set column 2 of a 3x4 transformation matrix.
setCol3	Set column 3 of a 3x4 transformation matrix.
setElem	Set the element of a 3x4 transformation matrix referred to by column and row indices.
setRow	Set the row of a 3x4 transformation matrix referred to by the specified index.
setTranslation	Set translation component.
setUpper3x3	Set the upper-left 3x3 submatrix.
Transform3	Default constructor; does no initialization.
Transform3	Copy a 3x4 transformation matrix.
Transform3	Construct a 3x4 transformation matrix containing the specified columns.
Transform3	Construct a 3x4 transformation matrix from a 3x3 matrix and a 3-D vector.
Transform3	Construct a 3x4 transformation matrix from a unit-length quaternion and a 3-D vector.
Transform3	Set all elements of a 3x4 transformation matrix to the same scalar value.
Transform3	Replicate an AoS 3x4 transformation matrix.
Transform3	Insert four AoS 3x4 transformation matrices.
translation	Construct a 3x4 transformation matrix to perform translation.

Constructors and Destructors

Transform3

Default constructor; does no initialization.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline Transform3();
              }
        }
}
```

Arguments

None

Return Values

None

Description

Default constructor; does no initialization.

Transform3

Copy a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline Transform3(
                            const Transform3 &tfrm
                     );
              }
        }
}
```

Arguments

tfrm	3x4 transformation matrix

Return Values

None

Description

Construct a copy of a 3x4 transformation matrix.

Transform3

Construct a 3x4 transformation matrix containing the specified columns.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
             class Transform3 {
                    inline Transform3(
                           const Vector3 &col0,
                           const Vector3 &col1,
                           const Vector3 &col2,
                           const Vector3 &col3
                    );
             }
        }
}
```

Arguments

col0	3-D vector
col1	3-D vector
col2	3-D vector
col3	3-D vector

Return Values

None

Description

Construct a 3x4 transformation matrix containing the specified columns.

Transform3

Construct a 3x4 transformation matrix from a 3x3 matrix and a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline Transform3(
                            const Matrix3 &tfrm,
                            const Vector3 &translateVec
                     );
              }
        }
}
```

Arguments

tfrm	3x3 matrix
translateVec	3-D vector

Return Values

None

Description

Construct a 3x4 transformation matrix whose upper 3x3 elements are equal to the 3x3 matrix argument and whose translation component is equal to the 3-D vector argument.

Transform3

Construct a 3x4 transformation matrix from a unit-length quaternion and a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline Transform3(
                            const Quat &unitQuat,
                            const Vector3 &translateVec
                     );
              }
        }
}
```

Arguments

unitQuat	Quaternion, expected to be unit-length
translateVec	3-D vector

Return Values

None

Description

Construct a 3x4 transformation matrix whose upper-left 3x3 submatrix is a rotation matrix converted from the unit-length quaternion argument and whose translation component is equal to the 3-D vector argument.

Transform3

Set all elements of a 3x4 transformation matrix to the same scalar value.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     explicit inline Transform3(
                            vec_float4 scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

None

Description

Construct a 3x4 transformation matrix with all elements set to the scalar value argument.

Transform3

Replicate an AoS 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline Transform3(
                            const Aos::Transform3 &tfrm
                     );
              }
        }
}
```

Arguments

tfrm	AoS 3x4 transformation matrix

Return Values

None

Description

Replicate an AoS 3x4 transformation matrix in all four slots of an SoA 3x4 transformation matrix.

Transform3

Insert four AoS 3x4 transformation matrices.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline Transform3(
                            const Aos::Transform3 &tfrm0,
                            const Aos::Transform3 &tfrm1,
                            const Aos::Transform3 &tfrm2,
                            const Aos::Transform3 &tfrm3
                     );
              }
        }
}
```

Arguments

tfrm0	AoS 3x4 transformation matrix
tfrm1	AoS 3x4 transformation matrix
tfrm2	AoS 3x4 transformation matrix
tfrm3	AoS 3x4 transformation matrix

Return Values

None

Description

Insert four AoS 3x4 transformation matrices into four slots of an SoA 3x4 transformation matrix (transpose the data format).

Operator Methods

operator *

Multiply a 3x4 transformation matrix by a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline const Vector3 operator *(
                            const Vector3 &vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

Product of the specified 3x4 transformation matrix and 3-D vector

Description

Applies the 3x3 upper-left submatrix (but not the translation component) of a 3x4 transformation matrix to a 3-D vector.

operator *

Multiply a 3x4 transformation matrix by a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline const Point3 operator *(
                            const Point3 &pnt
                     );
              }
        }
}
```

Arguments

pnt	3-D point

Return Values

Product of the specified 3x4 transformation matrix and 3-D point

Description

Applies the 3x3 upper-left submatrix and the translation component of a 3x4 transformation matrix to a 3-D point.

operator *

Multiply two 3x4 transformation matrices.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline const Transform3 operator *(
                            const Transform3 &tfrm
                     );
              }
        }
}
```

Arguments

tfrm	3x4 transformation matrix

Return Values

Product of the specified 3x4 transformation matrices

Description

Multiply two 3x4 transformation matrices.

operator *=

Perform compound assignment and multiplication by a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline Transform3 &operator *=(
                            const Transform3 &tfrm
                     );
              }
        }
}
```

Arguments

tfrm	3x4 transformation matrix

Return Values

A reference to the resulting 3x4 transformation matrix

Description

Perform compound assignment and multiplication by a 3x4 transformation matrix.

operator=

Assign one 3x4 transformation matrix to another.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline Transform3 &operator=(
                            const Transform3 &tfrm
                     );
              }
        }
}
```

Arguments

tfrm	3x4 transformation matrix

Return Values

A reference to the resulting 3x4 transformation matrix

Description

Assign one 3x4 transformation matrix to another.

operator[]

Subscripting operator to set or get a column.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline Vector3 &operator[](
                            int col
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-3

Return Values

A reference to indexed column

Description

Subscripting operator invoked when applied to non-const Transform3.

operator[]

Subscripting operator to get a column.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline const Vector3 operator[](
                            int col
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-3

Return Values

Indexed column

Description

Subscripting operator invoked when applied to const Transform3.

Public Static Methods

identity

Construct an identity 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     static inline const Transform3 identity();
              }
        }
}
```

Arguments

None

Return Values

The constructed 3x4 transformation matrix

Description

Construct an identity 3x4 transformation matrix in which non-diagonal elements are zero and diagonal elements are 1.

rotation

Construct a 3x4 transformation matrix to rotate around a unit-length 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     static inline const Transform3 rotation(
                            vec_float4 radians,
                            const Vector3 &unitVec
                     );
              }
        }
}
```

Arguments

radians	Scalar value
unitVec	3-D vector, expected to be unit-length

Return Values

The constructed 3x4 transformation matrix

Description

Construct a 3x4 transformation matrix to rotate around a unit-length 3-D vector by the specified radians angle.

rotation

Construct a rotation matrix from a unit-length quaternion.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     static inline const Transform3 rotation(
                            const Quat &unitQuat
                     );
              }
        }
}
```

Arguments

unitQuat	Quaternion, expected to be unit-length

Return Values

The constructed 3x4 transformation matrix

Description

Construct a 3x4 transformation matrix that applies the same rotation as the specified unit-length quaternion.

rotationX

Construct a 3x4 transformation matrix to rotate around the x axis.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     static inline const Transform3 rotationX(
                            vec_float4 radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value

Return Values

The constructed 3x4 transformation matrix

Description

Construct a 3x4 transformation matrix to rotate around the x axis by the specified radians angle.

rotationY

Construct a 3x4 transformation matrix to rotate around the y axis.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     static inline const Transform3 rotationY(
                            vec_float4 radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value

Return Values

The constructed 3x4 transformation matrix

Description

Construct a 3x4 transformation matrix to rotate around the y axis by the specified radians angle.

rotationZ

Construct a 3x4 transformation matrix to rotate around the z axis.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     static inline const Transform3 rotationZ(
                            vec_float4 radians
                     );
              }
        }
}
```

Arguments

radians	Scalar value

Return Values

The constructed 3x4 transformation matrix

Description

Construct a 3x4 transformation matrix to rotate around the z axis by the specified radians angle.

rotationZYX

Construct a 3x4 transformation matrix to rotate around the x, y, and z axes.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     static inline const Transform3 rotationZYX(
                            const Vector3 &radiansXYZ
                     );
              }
        }
}
```

Arguments

radiansXYZ	3-D vector

Return Values

The constructed 3x4 transformation matrix

Description

Construct a 3x4 transformation matrix to rotate around the x, y, and z axes by the radians angles contained in a 3-D vector. Equivalent to *rotationZ(radiansXYZ.getZ()) * rotationY(radiansXYZ.getY()) * rotationX(radiansXYZ.getX())*.

scale

Construct a 3x4 transformation matrix to perform scaling.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     static inline const Transform3 scale(
                            const Vector3 &scaleVec
                     );
              }
        }
}
```

Arguments

scaleVec	3-D vector

Return Values

The constructed 3x4 transformation matrix

Description

Construct a 3x4 transformation matrix to perform scaling, in which the non-diagonal elements are zero and the diagonal elements are set to the elements of *scaleVec*.

translation

Construct a 3x4 transformation matrix to perform translation.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     static inline const Transform3 translation(
                            const Vector3 &translateVec
                     );
              }
        }
}
```

Arguments

translateVec	3-D vector

Return Values

The constructed 3x4 transformation matrix

Description

Construct a 3x4 transformation matrix to perform translation, which is an identity matrix except for the translation component, with coordinates equal to those in `translateVec`.

Public Instance Methods

get4Aos

Extract four AoS 3x4 transformation matrices.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline void get4Aos(
                            Aos::Transform3 &result0,
                            Aos::Transform3 &result1,
                            Aos::Transform3 &result2,
                            Aos::Transform3 &result3
                     );
              }
        }
}
```

Arguments

result0	An output AoS 3x4 transformation matrix
result1	An output AoS 3x4 transformation matrix
result2	An output AoS 3x4 transformation matrix
result3	An output AoS 3x4 transformation matrix

Return Values

None

Description

Extract four AoS 3x4 transformation matrices from four slots of an SoA 3x4 transformation matrix (transpose the data format).

getCol

Get the column of a 3x4 transformation matrix referred to by the specified index.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline const Vector3 getCol(
                            int col
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-3

Return Values

The column referred to by the specified index

Description

Get the column of a 3x4 transformation matrix referred to by the specified index.

getCol0

Get column 0 of a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline const Vector3 getCol0();
              }
        }
}
```

Arguments

None

Return Values

Column 0

Description

Get column 0 of a 3x4 transformation matrix.

getCol1

Get column 1 of a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline const Vector3 getCol1();
              }
        }
}
```

Arguments

None

Return Values

Column 1

Description

Get column 1 of a 3x4 transformation matrix.

getCol2

Get column 2 of a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline const Vector3 getCol2();
              }
        }
}
```

Arguments

None

Return Values

Column 2

Description

Get column 2 of a 3x4 transformation matrix.

getCol3

Get column 3 of a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline const Vector3 getCol3();
              }
        }
}
```

Arguments

None

Return Values

Column 3

Description

Get column 3 of a 3x4 transformation matrix.

getElem

Get the element of a 3x4 transformation matrix referred to by column and row indices.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline vec_float4 getElem(
                            int col,
                            int row
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-3
row	Index, expected in the range 0-2

Return Values

Element selected by *col* and *row*

Description

Get the element of a 3x4 transformation matrix referred to by column and row indices.

getRow

Get the row of a 3x4 transformation matrix referred to by the specified index.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline const Vector4 getRow(
                            int row
                     );
              }
        }
}
```

Arguments

row	Index, expected in the range 0-2

Return Values

The row referred to by the specified index

Description

Get the row of a 3x4 transformation matrix referred to by the specified index.

getTranslation

Get the translation component of a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline const Vector3 getTranslation();
              }
        }
}
```

Arguments

None

Return Values

Translation component

Description

Get the translation component of a 3x4 transformation matrix.

getUpper3x3

Get the upper-left 3x3 submatrix of a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline const Matrix3 getUpper3x3();
              }
        }
}
```

Arguments

None

Return Values

Upper-left 3x3 submatrix

Description

Get the upper-left 3x3 submatrix of a 3x4 transformation matrix.

setCol

Set the column of a 3x4 transformation matrix referred to by the specified index.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline Transform3 &setCol(
                            int col,
                            const Vector3 &vec
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-3
vec	3-D vector

Return Values

A reference to the resulting 3x4 transformation matrix

Description

Set the column of a 3x4 transformation matrix referred to by the specified index.

setCol0

Set column 0 of a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
             class Transform3 {
                    inline Transform3 &setCol0(
                           const Vector3 &col0
                    );
             }
        }
}
```

Arguments

col0	3-D vector

Return Values

A reference to the resulting 3x4 transformation matrix

Description

Set column 0 of a 3x4 transformation matrix.

setCol1

Set column 1 of a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline Transform3 &setCol1(
                            const Vector3 &col1
                     );
              }
        }
}
```

Arguments

col1	3-D vector

Return Values

A reference to the resulting 3x4 transformation matrix

Description

Set column 1 of a 3x4 transformation matrix.

setCol2

Set column 2 of a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline Transform3 &setCol2(
                            const Vector3 &col2
                     );
              }
        }
}
```

Arguments

col2	3-D vector

Return Values

A reference to the resulting 3x4 transformation matrix

Description

Set column 2 of a 3x4 transformation matrix.

setCol3

Set column 3 of a 3x4 transformation matrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline Transform3 &setCol3(
                            const Vector3 &col3
                     );
              }
        }
}
```

Arguments

col3	3-D vector

Return Values

A reference to the resulting 3x4 transformation matrix

Description

Set column 3 of a 3x4 transformation matrix.

setElem

Set the element of a 3x4 transformation matrix referred to by column and row indices.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline Transform3 &setElem(
                            int col,
                            int row,
                            vec_float4 val
                     );
              }
        }
}
```

Arguments

col	Index, expected in the range 0-3
row	Index, expected in the range 0-2
val	Scalar value

Return Values

A reference to the resulting 3x4 transformation matrix

Description

Set the element of a 3x4 transformation matrix referred to by column and row indices.

setRow

Set the row of a 3x4 transformation matrix referred to by the specified index.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline Transform3 &setRow(
                            int row,
                            const Vector4 &vec
                     );
              }
        }
}
```

Arguments

row	Index, expected in the range 0-2
vec	4-D vector

Return Values

A reference to the resulting 3x4 transformation matrix

Description

Set the row of a 3x4 transformation matrix referred to by the specified index.

setTranslation

Set translation component.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline Transform3 &setTranslation(
                            const Vector3 &translateVec
                     );
              }
        }
}
```

Arguments

translateVec	3-D vector

Return Values

A reference to the resulting 3x4 transformation matrix

Description

Set the translation component of a 3x4 transformation matrix equal to the specified 3-D vector.

setUpper3x3

Set the upper-left 3x3 submatrix.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Transform3 {
                     inline Transform3 &setUpper3x3(
                            const Matrix3 &mat3
                     );
              }
        }
}
```

Arguments

mat3	3x3 matrix

Return Values

A reference to the resulting 3x4 transformation matrix

Description

Set the upper-left 3x3 submatrix elements of a 3x4 transformation matrix equal to the specified 3x3 matrix.

Vectormath::Soa::Vector3

Summary

Vectormath::Soa::Vector3

A set of four 3-D vectors in structure-of-arrays format.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
class Vector3;
```

Description

A class representing a set of four 3-D vectors stored in structure-of-arrays (SoA) format.

Methods Summary

Methods	Description
get4Aos	Extract four AoS 3-D vectors.
getElem	Get an x, y, or z element of a 3-D vector by index.
getX	Get the x element of a 3-D vector.
getY	Get the y element of a 3-D vector.
getZ	Get the z element of a 3-D vector.
operator *	Multiply a 3-D vector by a scalar.
operator *=	Perform compound assignment and multiplication by a scalar.
operator+	Add two 3-D vectors.
operator+	Add a 3-D vector to a 3-D point.
operator+=	Perform compound assignment and addition with a 3-D vector.
operator-	Subtract a 3-D vector from another 3-D vector.
operator-	Negate all elements of a 3-D vector.
operator-=	Perform compound assignment and subtraction by a 3-D vector.
operator/	Divide a 3-D vector by a scalar.
operator/=	Perform compound assignment and division by a scalar.
operator=	Assign one 3-D vector to another.
operator[]	Subscripting operator to set or get an element.
operator[]	Subscripting operator to get an element.
setElem	Set an x, y, or z element of a 3-D vector by index.
setX	Set the x element of a 3-D vector.
setY	Set the y element of a 3-D vector.
setZ	Set the z element of a 3-D vector.
Vector3	Default constructor; does no initialization.
Vector3	Copy a 3-D vector.
Vector3	Construct a 3-D vector from x, y, and z elements.
Vector3	Copy elements from a 3-D point into a 3-D vector.
Vector3	Set all elements of a 3-D vector to the same scalar value.
Vector3	Replicate an AoS 3-D vector.
Vector3	Insert four AoS 3-D vectors.
xAxis	Construct x axis.
yAxis	Construct y axis.

Methods	Description
zAxis	Construct z axis.

Constructors and Destructors

Vector3

Default constructor; does no initialization.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline Vector3();
              }
        }
}
```

Arguments

None

Return Values

None

Description

Default constructor; does no initialization.

Vector3

Copy a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline Vector3(
                            const Vector3 &vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

None

Description

Construct a copy of a 3-D vector.

Vector3

Construct a 3-D vector from x, y, and z elements.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline Vector3(
                            vec_float4 x,
                            vec_float4 y,
                            vec_float4 z
                     );
              }
        }
}
```

Arguments

x	Scalar value
y	Scalar value
z	Scalar value

Return Values

None

Description

Construct a 3-D vector containing the specified x, y, and z elements.

Vector3

Copy elements from a 3-D point into a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     explicit inline Vector3(
                            const Point3 &pnt
                     );
              }
        }
}
```

Arguments

pnt	3-D point

Return Values

None

Description

Construct a 3-D vector containing the x, y, and z elements of the specified 3-D point.

Vector3

Set all elements of a 3-D vector to the same scalar value.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     explicit inline Vector3(
                            vec_float4 scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

None

Description

Construct a 3-D vector with all elements set to the scalar value argument.

Vector3

Replicate an AoS 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline Vector3(
                            Aos::Vector3 vec
                     );
              }
        }
}
```

Arguments

vec	AoS 3-D vector

Return Values

None

Description

Replicate an AoS 3-D vector in all four slots of an SoA 3-D vector.

Vector3

Insert four AoS 3-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline Vector3(
                            Aos::Vector3 vec0,
                            Aos::Vector3 vec1,
                            Aos::Vector3 vec2,
                            Aos::Vector3 vec3
                     );
              }
        }
}
```

Arguments

vec0	AoS 3-D vector
vec1	AoS 3-D vector
vec2	AoS 3-D vector
vec3	AoS 3-D vector

Return Values

None

Description

Insert four AoS 3-D vectors into four slots of an SoA 3-D vector (transpose the data format).

Operator Methods

operator *

Multiply a 3-D vector by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline const Vector3 operator *(
                            vec_float4 scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

Product of the specified 3-D vector and scalar

Description

Multiply a 3-D vector by a scalar.

operator *=

Perform compound assignment and multiplication by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline Vector3 &operator *=(
                            vec_float4 scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

A reference to the resulting 3-D vector

Description

Perform compound assignment and multiplication by a scalar.

operator+

Add two 3-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline const Vector3 operator+(
                            const Vector3 &vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

Sum of the specified 3-D vectors

Description

Add two 3-D vectors.

operator+

Add a 3-D vector to a 3-D point.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline const Point3 operator+(
                            const Point3 &pnt
                     );
              }
        }
}
```

Arguments

pnt	3-D point

Return Values

Sum of the specified 3-D vector and 3-D point

Description

Add a 3-D vector to a 3-D point.

operator+=

Perform compound assignment and addition with a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline Vector3 &operator+=(
                            const Vector3 &vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

A reference to the resulting 3-D vector

Description

Perform compound assignment and addition with a 3-D vector.

operator-

Subtract a 3-D vector from another 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline const Vector3 operator-(
                            const Vector3 &vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

Difference of the specified 3-D vectors

Description

Subtract a 3-D vector from another 3-D vector.

operator-

Negate all elements of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline const Vector3 operator-();
              }
        }
}
```

Arguments

None

Return Values

3-D vector containing negated elements of the specified 3-D vector

Description

Negate all elements of a 3-D vector.

operator-=

Perform compound assignment and subtraction by a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline Vector3 &operator-=(
                            const Vector3 &vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

A reference to the resulting 3-D vector

Description

Perform compound assignment and subtraction by a 3-D vector.

operator/

Divide a 3-D vector by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline const Vector3 operator/(
                            vec_float4 scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

Quotient of the specified 3-D vector and scalar

Description

Divide a 3-D vector by a scalar.

operator/=

Perform compound assignment and division by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline Vector3 &operator/=(
                            vec_float4 scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

A reference to the resulting 3-D vector

Description

Perform compound assignment and division by a scalar.

operator=

Assign one 3-D vector to another.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline Vector3 &operator=(
                            const Vector3 &vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

A reference to the resulting 3-D vector

Description

Assign one 3-D vector to another.

operator[]

Subscripting operator to set or get an element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline vec_float4 &operator[](
                            int idx
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-2

Return Values

A reference to indexed element

Description

Subscripting operator invoked when applied to non-const Vector3.

operator[]

Subscripting operator to get an element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline vec_float4 operator[](
                            int idx
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-2

Return Values

Indexed element

Description

Subscripting operator invoked when applied to const Vector3.

Public Static Methods

xAxis

Construct x axis.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     static inline const Vector3 xAxis();
              }
        }
}
```

Arguments

None

Return Values

The constructed 3-D vector

Description

Construct a 3-D vector equal to (1,0,0).

yAxis

Construct y axis.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     static inline const Vector3 yAxis();
              }
        }
}
```

Arguments

None

Return Values

The constructed 3-D vector

Description

Construct a 3-D vector equal to (0,1,0).

zAxis

Construct z axis.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     static inline const Vector3 zAxis();
              }
        }
}
```

Arguments

None

Return Values

The constructed 3-D vector

Description

Construct a 3-D vector equal to (0,0,1).

Public Instance Methods

get4Aos

Extract four AoS 3-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline void get4Aos(
                            Aos::Vector3 &result0,
                            Aos::Vector3 &result1,
                            Aos::Vector3 &result2,
                            Aos::Vector3 &result3
                     );
              }
        }
}
```

Arguments

result0	An output AoS 3-D vector
result1	An output AoS 3-D vector
result2	An output AoS 3-D vector
result3	An output AoS 3-D vector

Return Values

None

Description

Extract four AoS 3-D vectors from four slots of an SoA 3-D vector (transpose the data format).

getElem

Get an x, y, or z element of a 3-D vector by index.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline vec_float4 getElem(
                            int idx
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-2

Return Values

Element selected by the specified index

Description

Get an x, y, or z element of a 3-D vector by specifying an index of 0, 1, or 2, respectively.

getX

Get the x element of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline vec_float4 getX();
              }
        }
}
```

Arguments

None

Return Values

x element of a 3-D vector

Description

Get the x element of a 3-D vector.

getY

Get the y element of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline vec_float4 getY();
              }
        }
}
```

Arguments

None

Return Values

y element of a 3-D vector

Description

Get the y element of a 3-D vector.

getZ

Get the z element of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline vec_float4 getZ();
              }
        }
}
```

Arguments

None

Return Values

z element of a 3-D vector

Description

Get the z element of a 3-D vector.

setElem

Set an x, y, or z element of a 3-D vector by index.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline Vector3 &setElem(
                            int idx,
                            vec_float4 value
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-2
value	Scalar value

Return Values

A reference to the resulting 3-D vector

Description

Set an x, y, or z element of a 3-D vector by specifying an index of 0, 1, or 2, respectively.

setX

Set the x element of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline Vector3 &setX(
                            vec_float4 x
                     );
              }
        }
}
```

Arguments

x	Scalar value

Return Values

A reference to the resulting 3-D vector

Description

Set the x element of a 3-D vector to the specified scalar value.

setY

Set the y element of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline Vector3 &setY(
                            vec_float4 y
                     );
              }
        }
}
```

Arguments

y Scalar value

Return Values

A reference to the resulting 3-D vector

Description

Set the y element of a 3-D vector to the specified scalar value.

setZ

Set the z element of a 3-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector3 {
                     inline Vector3 &setZ(
                            vec_float4 z
                     );
              }
        }
}
```

Arguments

z	Scalar value

Return Values

A reference to the resulting 3-D vector

Description

Set the z element of a 3-D vector to the specified scalar value.

Vectormath::Soa::Vector4

Summary

Vectormath::Soa::Vector4

A set of four 4-D vectors in structure-of-arrays format.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
class Vector4;
```

Description

A class representing a set of four 4-D vectors stored in structure-of-arrays (SoA) format.

Methods Summary

Methods	Description
get4Aos	Extract four AoS 4-D vectors.
getElem	Get an x, y, z, or w element of a 4-D vector by index.
getW	Get the w element of a 4-D vector.
getX	Get the x element of a 4-D vector.
getXYZ	Get the x, y, and z elements of a 4-D vector.
getY	Get the y element of a 4-D vector.
getZ	Get the z element of a 4-D vector.
operator *	Multiply a 4-D vector by a scalar.
operator *=	Perform compound assignment and multiplication by a scalar.
operator+	Add two 4-D vectors.
operator+=	Perform compound assignment and addition with a 4-D vector.
operator-	Subtract a 4-D vector from another 4-D vector.
operator-	Negate all elements of a 4-D vector.
operator-=	Perform compound assignment and subtraction by a 4-D vector.
operator/	Divide a 4-D vector by a scalar.
operator/=	Perform compound assignment and division by a scalar.
operator=	Assign one 4-D vector to another.
operator[]	Subscripting operator to set or get an element.
operator[]	Subscripting operator to get an element.
setElem	Set an x, y, z, or w element of a 4-D vector by index.
setW	Set the w element of a 4-D vector.
setX	Set the x element of a 4-D vector.
setXYZ	Set the x, y, and z elements of a 4-D vector.
setY	Set the y element of a 4-D vector.
setZ	Set the z element of a 4-D vector.
Vector4	Default constructor; does no initialization.
Vector4	Copy a 4-D vector.
Vector4	Construct a 4-D vector from x, y, z, and w elements.
Vector4	Construct a 4-D vector from a 3-D vector and a scalar.
Vector4	Copy x, y, and z from a 3-D vector into a 4-D vector, and set w to 0.

Methods	Description
Vector4	Copy x, y, and z from a 3-D point into a 4-D vector, and set w to 1.
Vector4	Copy elements from a quaternion into a 4-D vector.
Vector4	Set all elements of a 4-D vector to the same scalar value.
Vector4	Replicate an AoS 4-D vector.
Vector4	Insert four AoS 4-D vectors.
wAxis	Construct w axis.
xAxis	Construct x axis.
yAxis	Construct y axis.
zAxis	Construct z axis.

Constructors and Destructors

Vector4

Default constructor; does no initialization.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline Vector4();
              }
        }
}
```

Arguments

None

Return Values

None

Description

Default constructor; does no initialization.

Vector4

Copy a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline Vector4(
                            const Vector4 &vec
                     );
              }
        }
}
```

Arguments

vec	4-D vector

Return Values

None

Description

Construct a copy of a 4-D vector.

Vector4

Construct a 4-D vector from x, y, z, and w elements.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline Vector4(
                            vec_float4 x,
                            vec_float4 y,
                            vec_float4 z,
                            vec_float4 w
                     );
              }
        }
}
```

Arguments

x	Scalar value
y	Scalar value
z	Scalar value
w	Scalar value

Return Values

None

Description

Construct a 4-D vector containing the specified x, y, z, and w elements.

Vector4

Construct a 4-D vector from a 3-D vector and a scalar.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline Vector4(
                            const Vector3 &xyz,
                            vec_float4 w
                     );
              }
        }
}
```

Arguments

xyz	3-D vector
w	Scalar value

Return Values

None

Description

Construct a 4-D vector with the x, y, and z elements of the specified 3-D vector and with the w element set to the specified scalar.

Vector4

Copy x, y, and z from a 3-D vector into a 4-D vector, and set w to 0.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     explicit inline Vector4(
                            const Vector3 &vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

None

Description

Construct a 4-D vector with the x, y, and z elements of the specified 3-D vector and with the w element set to 0.

Vector4

Copy x, y, and z from a 3-D point into a 4-D vector, and set w to 1.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     explicit inline Vector4(
                            const Point3 &pnt
                     );
              }
        }
}
```

Arguments

pnt	3-D point

Return Values

None

Description

Construct a 4-D vector with the x, y, and z elements of the specified 3-D point and with the w element set to 1.

Vector4

Copy elements from a quaternion into a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     explicit inline Vector4(
                            const Quat &quat
                     );
              }
        }
}
```

Arguments

quat	Quaternion

Return Values

None

Description

Construct a 4-D vector containing the x, y, z, and w elements of the specified quaternion.

Vector4

Set all elements of a 4-D vector to the same scalar value.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     explicit inline Vector4(
                            vec_float4 scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

None

Description

Construct a 4-D vector with all elements set to the scalar value argument.

Vector4

Replicate an AoS 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline Vector4(
                            Aos::Vector4 vec
                     );
              }
        }
}
```

Arguments

vec	AoS 4-D vector

Return Values

None

Description

Replicate an AoS 4-D vector in all four slots of an SoA 4-D vector.

Vector4

Insert four AoS 4-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline Vector4(
                            Aos::Vector4 vec0,
                            Aos::Vector4 vec1,
                            Aos::Vector4 vec2,
                            Aos::Vector4 vec3
                     );
              }
        }
}
```

Arguments

vec0	AoS 4-D vector
vec1	AoS 4-D vector
vec2	AoS 4-D vector
vec3	AoS 4-D vector

Return Values

None

Description

Insert four AoS 4-D vectors into four slots of an SoA 4-D vector (transpose the data format).

Operator Methods

operator *

Multiply a 4-D vector by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline const Vector4 operator *(
                            vec_float4 scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

Product of the specified 4-D vector and scalar

Description

Multiply a 4-D vector by a scalar.

operator *=

Perform compound assignment and multiplication by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline Vector4 &operator *=(
                            vec_float4 scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

A reference to the resulting 4-D vector

Description

Perform compound assignment and multiplication by a scalar.

operator+

Add two 4-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline const Vector4 operator+(
                            const Vector4 &vec
                     );
              }
        }
}
```

Arguments

vec	4-D vector

Return Values

Sum of the specified 4-D vectors

Description

Add two 4-D vectors.

operator+=

Perform compound assignment and addition with a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline Vector4 &operator+=(
                            const Vector4 &vec
                     );
              }
        }
}
```

Arguments

vec	4-D vector

Return Values

A reference to the resulting 4-D vector

Description

Perform compound assignment and addition with a 4-D vector.

operator-

Subtract a 4-D vector from another 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline const Vector4 operator-(
                            const Vector4 &vec
                     );
              }
        }
}
```

Arguments

vec	4-D vector

Return Values

Difference of the specified 4-D vectors

Description

Subtract a 4-D vector from another 4-D vector.

operator-

Negate all elements of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline const Vector4 operator-();
              }
        }
}
```

Arguments

None

Return Values

4-D vector containing negated elements of the specified 4-D vector

Description

Negate all elements of a 4-D vector.

operator-=

Perform compound assignment and subtraction by a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
             class Vector4 {
                    inline Vector4 &operator-=(
                           const Vector4 &vec
                    );
             }
        }
}
```

Arguments

vec	4-D vector

Return Values

A reference to the resulting 4-D vector

Description

Perform compound assignment and subtraction by a 4-D vector.

operator/

Divide a 4-D vector by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline const Vector4 operator/(
                            vec_float4 scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

Quotient of the specified 4-D vector and scalar

Description

Divide a 4-D vector by a scalar.

operator/=

Perform compound assignment and division by a scalar.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline Vector4 &operator/=(
                            vec_float4 scalar
                     );
              }
        }
}
```

Arguments

scalar	Scalar value

Return Values

A reference to the resulting 4-D vector

Description

Perform compound assignment and division by a scalar.

operator=

Assign one 4-D vector to another.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline Vector4 &operator=(
                            const Vector4 &vec
                     );
              }
        }
}
```

Arguments

vec	4-D vector

Return Values

A reference to the resulting 4-D vector

Description

Assign one 4-D vector to another.

operator[]

Subscripting operator to set or get an element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline vec_float4 &operator[](
                            int idx
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-3

Return Values

A reference to indexed element

Description

Subscripting operator invoked when applied to non-const Vector4.

operator[]

Subscripting operator to get an element.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline vec_float4 operator[](
                            int idx
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-3

Return Values

Indexed element

Description

Subscripting operator invoked when applied to const Vector4.

Public Static Methods

wAxis

Construct w axis.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     static inline const Vector4 wAxis();
              }
        }
}
```

Arguments

None

Return Values

The constructed 4-D vector

Description

Construct a 4-D vector equal to (0,0,0,1).

xAxis

Construct x axis.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     static inline const Vector4 xAxis();
              }
        }
}
```

Arguments

None

Return Values

The constructed 4-D vector

Description

Construct a 4-D vector equal to (1,0,0,0).

yAxis

Construct y axis.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     static inline const Vector4 yAxis();
              }
        }
}
```

Arguments

None

Return Values

The constructed 4-D vector

Description

Construct a 4-D vector equal to (0,1,0,0).

zAxis

Construct z axis.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     static inline const Vector4 zAxis();
              }
        }
}
```

Arguments

None

Return Values

The constructed 4-D vector

Description

Construct a 4-D vector equal to (0,0,1,0).

Public Instance Methods

get4Aos

Extract four AoS 4-D vectors.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline void get4Aos(
                            Aos::Vector4 &result0,
                            Aos::Vector4 &result1,
                            Aos::Vector4 &result2,
                            Aos::Vector4 &result3
                     );
              }
        }
}
```

Arguments

result0	An output AoS 4-D vector
result1	An output AoS 4-D vector
result2	An output AoS 4-D vector
result3	An output AoS 4-D vector

Return Values

None

Description

Extract four AoS 4-D vectors from four slots of an SoA 4-D vector (transpose the data format).

getElem

Get an x, y, z, or w element of a 4-D vector by index.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline vec_float4 getElem(
                            int idx
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-3

Return Values

Element selected by the specified index

Description

Get an x, y, z, or w element of a 4-D vector by specifying an index of 0, 1, 2, or 3, respectively.

getW

Get the w element of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline vec_float4 getW();
              }
        }
}
```

Arguments

None

Return Values

w element of a 4-D vector

Description

Get the w element of a 4-D vector.

getX

Get the x element of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline vec_float4 getX();
              }
        }
}
```

Arguments

None

Return Values

x element of a 4-D vector

Description

Get the x element of a 4-D vector.

getXYZ

Get the x, y, and z elements of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline const Vector3 getXYZ();
              }
        }
}
```

Arguments

None

Return Values

3-D vector containing x, y, and z elements

Description

Extract a 4-D vector's x, y, and z elements into a 3-D vector.

getY

Get the y element of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline vec_float4 getY();
              }
        }
}
```

Arguments

None

Return Values

y element of a 4-D vector

Description

Get the y element of a 4-D vector.

getZ

Get the z element of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline vec_float4 getZ();
              }
        }
}
```

Arguments

None

Return Values

z element of a 4-D vector

Description

Get the z element of a 4-D vector.

setElem

Set an x, y, z, or w element of a 4-D vector by index.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline Vector4 &setElem(
                            int idx,
                            vec_float4 value
                     );
              }
        }
}
```

Arguments

idx	Index, expected in the range 0-3
value	Scalar value

Return Values

A reference to the resulting 4-D vector

Description

Set an x, y, z, or w element of a 4-D vector by specifying an index of 0, 1, 2, or 3, respectively.

setW

Set the w element of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline Vector4 &setW(
                            vec_float4 w
                     );
              }
        }
}
```

Arguments

w Scalar value

Return Values

A reference to the resulting 4-D vector

Description

Set the w element of a 4-D vector to the specified scalar value.

setX

Set the x element of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline Vector4 &setX(
                            vec_float4 x
                     );
              }
        }
}
```

Arguments

x	Scalar value

Return Values

A reference to the resulting 4-D vector

Description

Set the x element of a 4-D vector to the specified scalar value.

setXYZ

Set the x, y, and z elements of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline Vector4 &setXYZ(
                            const Vector3 &vec
                     );
              }
        }
}
```

Arguments

vec	3-D vector

Return Values

A reference to the resulting 4-D vector

Description

Set the x, y, and z elements to those of the specified 3-D vector.

Notes

This function does not change the w element.

setY

Set the y element of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline Vector4 &setY(
                            vec_float4 y
                     );
              }
        }
}
```

Arguments

y	Scalar value

Return Values

A reference to the resulting 4-D vector

Description

Set the y element of a 4-D vector to the specified scalar value.

setZ

Set the z element of a 4-D vector.

Definition

```
#include <vectormath/cpp/vectormath_soa.h>
namespace Vectormath {
        namespace Soa {
              class Vector4 {
                     inline Vector4 &setZ(
                            vec_float4 z
                     );
              }
        }
}
```

Arguments

z	Scalar value

Return Values

A reference to the resulting 4-D vector

Description

Set the z element of a 4-D vector to the specified scalar value.