RenderScript Quaternion Functions

Overview

The following functions manipulate quaternions.

Summary

Functions	
rsQuaternionAdd	Add two quaternions
rsQuaternionConjugate	Conjugate a quaternion
rsQuaternionDot	Dot product of two quaternions
rsQuaternionGetMatrixUnit	Get a rotation matrix from a quaternion
rsQuaternionLoadRotate	Create a rotation quaternion
rsQuaternionLoadRotateUnit	Quaternion that represents a rotation about an arbitrary unit vector
rsQuaternionMultiply	Multiply a quaternion by a scalar or another quaternion
rsQuaternionNormalize	Normalize a quaternion
rsQuaternionSet	Create a quaternion
rsQuaternionSlerp	Spherical linear interpolation between two quaternions

Functions

rsQuaternionAdd : Add two quaternions

void rsQuaternionAdd(rs_quaternion* q, const rs_quaternion* rhs);

Parameters

q	Destination quaternion to add to.
rhs	Quaternion to add.

Adds two quaternions, i.e. `*q += *rhs;`

rsQuaternionConjugate : Conjugate a quaternion

void rsQuaternionConjugate(rs_quaternion* q);

Parameters

q	Quaternion to modify.

Conjugates the quaternion.

rsQuaternionDot : Dot product of two quaternions

float rsQuaternionDot(const rs_quaternion* q0, const rs_quaternion* q1);

Parameters

q0 First quaternion.
q1 Second quaternion.

Returns the dot product of two quaternions.

rsQuaternionGetMatrixUnit : Get a rotation matrix from a quaternion

void rsQuaternionGetMatrixUnit(rs_matrix4x4* m, const rs_quaternion* q);

Parameters

m	Resulting matrix.
q	Normalized quaternion.

Computes a rotation matrix from the normalized quaternion.

rsQuaternionLoadRotate : Create a rotation quaternion

void rsQuaternionLoadRotate(rs_quaternion* q, float rot, float x, float y, float z);

Parameters

q	Destination quaternion.
rot	Angle to rotate by.
x	X component of a vector.
y	Y component of a vector.
z	Z component of a vector.

Loads a quaternion that represents a rotation about an arbitrary vector (doesn't have to be unit)

rsQuaternionLoadRotateUnit : Quaternion that represents a rotation about an arbitrary unit vector

void rsQuaternionLoadRotateUnit(rs_quaternion* q, float rot, float x, float y, float z);

Parameters

q	Destination quaternion.
rot	Angle to rotate by, in radians.
x	X component of the vector.
y	Y component of the vector.
z	Z component of the vector.

Loads a quaternion that represents a rotation about an arbitrary unit vector.

rsQuaternionMultiply : Multiply a quaternion by a scalar or another quaternion

void rsQuaternionMultiply(rs_quaternion* q, const rs_quaternion* rhs);

void rsQuaternionMultiply(rs_quaternion* q, float scalar);

Parameters

q	Destination quaternion.
scalar	Scalar to multiply the quaternion by.
rhs	Quaternion to multiply the destination quaternion by.

Multiplies a quaternion by a scalar or by another quaternion, e.g `*q = *q * scalar;` or `*q = *q * *rhs;`.

rsQuaternionNormalize : Normalize a quaternion

void rsQuaternionNormalize(rs_quaternion* q);

Parameters

q	Quaternion to normalize.

Normalizes the quaternion.

rsQuaternionSet : Create a quaternion

```
void rsQuaternionSet(rs_quaternion* q, const rs_quaternion* rhs);
void rsQuaternionSet(rs_quaternion* q, float w, float x, float y, float z);
```

Parameters

q	Destination quaternion.
w	W component.
x	X component.
y	Y component.
z	Z component.
rhs	Source quaternion.

Creates a quaternion from its four components or from another quaternion.

rsQuaternionSlerp : Spherical linear interpolation between two quaternions

```
void rsQuaternionSlerp(rs_quaternion* q, const rs_quaternion* q0, const rs_quaternion* q1, float t);
```

Parameters

q	Result quaternion from the interpolation.
q0	First input quaternion.
q1	Second input quaternion.
t	How much to interpolate by.

Performs spherical linear interpolation between two quaternions.