

## N3DIntersectLinePlane()

**SUMMARY** Returns a point representing the intersection of a line segment and a plane

**DECLARED IN** 3Dkit/next3d.h

**SYNOPSIS** void **N3DIntersectLinePlane**(RtPoint \**endPoints*, RtPoint *planeNormal*, RtPoint *planePoint*, RtPoint \**intersection*)

**DESCRIPTION** This function accepts two points defining a line segment and two points defining a plane. It calculates and returns by reference the point where the line and the plane intersect.

*endPoints* is an array of two points defining the line. *planeNormal* and *planePoint* are two points that define a vector normal (perpendicular) to the plane. *planePoint* is on the plane itself, *planeNormal* is a point in space. The line segment between *planePoint* and *planeNormal* is perpendicular to (and thus defines) the plane whose intersection is being tested.

This method treats *endPoints* as the two points defining a line and tests for the intersection of that line with the plane. Thus *intersection* doesn't necessarily represent a point between the points in *endPoints*.

**RETURN** **N3DIntersectLinePlane()** returns in *intersection* the point where the line defined by *endPoints* intersects the plane defined by *planeNormal* and *planePoint*. If the line and plane are parallel, this function returns NaN for all three values of *intersection*.

## N3DInvertMatrix(), N3DMultiplyMatrix()

**SUMMARY** Perform standard matrix manipulations

**DECLARED IN** 3Dkit/next3d.h

**SYNOPSIS** void **N3DMultiplyMatrix**(RtMatrix *preTransform*, RtMatrix *postTransform*, RtMatrix *resultTransform*)  
float **N3DInvertMatrix**(RtMatrix *theTransform*, RtMatrix *theInverse*)

**DESCRIPTION** **N3DMultiplyMatrix()** accepts a *preTransform* matrix, a *postTransform* matrix, and a *resultTransform* matrix. It multiplies *preTransform* by *postTransform* and returns the resulting matrix.

**N3DInvertMatrix()** accepts *theTransform* matrix and returns its inverse.

**RETURN** **N3DMultiplyMatrix()** returns the product of *preTransform* and *postTransform* in *resultTransform*.

**N3DInvertMatrix()** returns the determinant of the matrix and, by reference, the inverse of *theTransform* in *inverseTransform*.

**N3DMultiplyMatrix()** → See **N3DInvertMatrix()**

## N3DMult3DPoint(), N3DMult3DPoints()

**SUMMARY** Transform points between coordinate systems

**DECLARED IN** 3Dkit/next3d.h

**SYNOPSIS** void **N3DMult3DPoint**(RtPoint *thePoint*, RtMatrix *theTransform*, RtPoint *newPoint*)  
void **N3DMult3DPoints**(RtPoint *\*thePoints*, int *pointCount*, RtMatrix *theTransform*, RtPoint *\*newPoints*)

**DESCRIPTION** These functions transform a 3D point or array of 3D points to the coordinate system represented by a 3D matrix.

**N3DMult3DPoint()** accepts *thePoint*, a single point; *theTransform*, a matrix by which to multiply this point; and *newPoint*, a point in which to place the result.

**N3DMult3DPoints()** accepts *thePoints*, an array of points; *pointCount*, the number of points in the array; *theTransform*, a matrix by which to multiply thePoints; and *newPoints*, an array of points in which to place the results.

**RETURN** **N3DMult3DPoint()** returns by reference in *newPoint* the transformation of *thePoint* from its coordinate system to the coordinate system represented by *theTransform*.

**N3DMult3DPoints()** returns by reference in *newPoints* the transformation of *thePoint* from its coordinate system to the coordinate system represented by *theTransform*.

## N3D\_ConvertBoundToPoints(), N3D\_ConvertPointsToBound()

**SUMMARY** Convert between bounding boxes and points

**DECLARED IN** 3Dkit/next3d.h

**SYNOPSIS** void **N3D\_ConvertBoundToPoints**(RtBound *theBound*, RtPoint *\*thePoints*)  
void **N3D\_ConvertPointsToBound**(RtPoint *\*thePoints*, RtBound *theBound*)

**DESCRIPTION** These macros convert between the RtBound and RtPoint data types. *theBound* is a three-dimensional bounding box; *thePoints* is an array of two points.

**RETURN** **N3D\_ConvertBoundToPoints()** returns in *thePoints[0]* the origin of *theBound* and in *thePoints[1]* the extent of *theBound*.

**N3D\_ConvertPointsToBound()** returns in *theBound* a bounding box whose origin is at *thePoints[0]* and whose extent is at *thePoints[1]*.

**N3D\_ConvertPointsToBound()** → See **N3D\_ConvertBoundToPoints()**

## N3D\_CopyBound(), N3D\_CopyMatrix(), N3D\_CopyPoint()

**SUMMARY** Copy data from one 3D data structure to another

**DECLARED IN**     3Dkit/next3d.h

**SYNOPSIS** void **N3D\_CopyBound**(RtBound *sourceBounds*, RtBound *destBounds*)  
             void **N3D\_CopyMatrix**(RtMatrix *sourceMatrix*, RtMatrix *destMatrix*)  
             void **N3D\_CopyPoint**(RtPoint *sourcePoint*, RtPoint *destPoint*)

**DESCRIPTION**     These macros efficiently copy the contents of one 3D data structure to another.

**RETURN**     **N3D\_CopyBound()** returns, in *destBound*, a copy of the values in *sourceBound*.

**N3D\_CopyMatrix()** returns, in *destMatrix*, a copy of the values in *sourceMatrix*.

**N3D\_CopyPoint()** returns, in *destPoint*, a copy of the values in *sourcePoint*.

**N3D\_CopyMatrix()** → See **N3D\_CopyBound()**

**N3D\_CopyPoint()** → See **N3D\_CopyBound()**

**N3D\_WComp()** → See **N3D\_XComp()**

**N3D\_XComp(), N3D\_YComp(), N3D\_ZComp(), N3D\_WComp()**

**SUMMARY** Returns the components of a 3D data structure

**DECLARED IN**     3Dkit/next3d.h

**SYNOPSIS** RtFloat **N3D\_XComp**(RtFloat *\*theVector*)  
             RtFloat **N3D\_YComp**(RtFloat *\*theVector*)  
             RtFloat **N3D\_ZComp**(RtFloat *\*theVector*)  
             RtFloat **N3D\_WComp**(RtFloat *\*theVector*)

**DESCRIPTION**     These macros return the components of a 3D point or submatrix.

             If *theVector* is an RtPoint type, use the macros **N3D\_XComp()**, **N3D\_YComp()**, and **N3D\_ZComp()** to retrieve its elements.

             If *theVector* is a row in an RtMatrix (for example, **myMatrix[2]**), use **N3D\_XComp()**, **N3D\_YComp()**, **N3D\_ZComp()**, and **N3D\_WComp()** to retrieve its elements.

**RETURN**     **N3D\_XComp()** returns the x-component of *theVector*.

**N3D\_YComp()** returns the y-component of *theVector*.

**N3D\_ZComp()** returns the z-component of *theVector*.

**N3D\_WComp()** returns the w-component of *theVector*.

**N3D\_YComp()** → See **N3D\_XComp()**

**N3D\_ZComp()** → **N3D\_XComp()**