Geometry Library Reference

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Document serial number: 000004892117

Table of Contents

Introduction

Library Summary

Library Contents

Item	Description
sce::Geometry	The namespace containing the geometry library.
sce::Geometry::Aos	The namespace containing the array-of-structures geometry types.
sce::Geometry::Aos::Aabb	A 3D axis-aligned bounding box representation.
sce::Geometry::Aos::Bounds	A 3D bounds limit representation.
sce::Geometry::Aos::Capsule	A 3D swept-sphere capsule representation.
sce::Geometry::Aos::Frustum	A 3D frustum representation.
sce::Geometry::Aos::Line	A 3D line representation.
sce::Geometry::Aos::Obb	A 3D oriented bounding box representation.
sce::Geometry::Aos::Plane	A 3D plane representation.
sce::Geometry::Aos::Ray	A 3D ray representation.
sce::Geometry::Aos::Segment	A 3D line segment representation.
sce::Geometry::Aos::Sphere	A 3D sphere representation.

sce::Geometry

Summary

sce::Geometry

The namespace containing the geometry library.

Definition

```
namespace Geometry {}
```

Description

The namespace containing the geometry library.

Inner Classes, Structures, and Namespaces

Item	Description
sce::Geometry::Aos	The namespace containing the array-of-structures geometry types.

Enumerated Types

eFrustumPlanes

The bounding planes of the 3D frustum.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        enum eFrustumPlanes {
            kFrustumFar = 0,
            kFrustumNear = 1,
            kFrustumLeft = 2,
            kFrustumRight = 3,
            kFrustumBottom = 4,
            kFrustumTop = 5,
            kNumFrustumPlanes = 6
        };
    }
}
```

Enumeration Values

Macro	Value	Description
kFrustumFar	0	The far bounding frustum plane.
kFrustumNear	1	The near bounding frustum plane.
kFrustumLeft	2	The left bounding frustum plane.
kFrustumRight	3	The right bounding frustum plane.
kFrustumBottom	4	The bottom bounding frustum plane.
kFrustumTop	5	The top bounding frustum plane.
kNumFrustumPlanes	6	The total number of frustum bounding planes.

Description

The bounding planes of the 3D frustum.

sce::Geometry::Aos

Summary

sce::Geometry::Aos

The namespace containing the array-of-structures geometry types.

Definition

```
namespace Aos {}
```

Description

The namespace containing the array-of-structures geometry types.

Function Summary

Function	Description
closestPoint	Gets the closest point on a Line to a point.
closestPoint	Gets the closest point on a line Ray to a point.
closestPoint	Gets the closest point on a line Segment to a point.
closestPoint	Gets the closest point on a Plane to another point.
closestPoint	Gets the closest point within or upon a Sphere surface to another point.
closestPoint	Gets the closest point within or upon a Bounds surface to another point.
closestPoint	Gets the closest point within or upon an Aabb surface to another point.
closestPoint	Gets the closest point within or upon an Obb surface to another point.
closestPoints	Gets the closest points between two Line objects.
closestPoints	Gets the closest points between a line Ray and a Line object.
closestPoints	Gets the closest points between two line Ray objects.
closestPoints	Gets the closest points between a line Segment and a Line object.
closestPoints	Gets the closest points between a line Segment and a line Ray object.
closestPoints	Gets the closest points between two line Segment objects.
containsPoint	Tests if a point is inside or at the limit of the Bounds.
containsPoint	Tests if a point is inside or at the limit of the Sphere.
containsPoint	Tests if a point is inside or at the limit of the Capsule.
containsPoint	Tests if a point is inside or at the limit of the Aabb.
containsPoint	Tests if a point is inside or at the limit of the Obb.
containsPoint	Tests if a point is inside or at the limit of the Frustum.
intersectionPoint	Tests if a Line intersects with a Plane.
intersectionPoint	Tests if a Line intersects with a Plane.
intersectionPoint	Tests if a Line intersects with an Aabb.
intersectionPoint	Tests if a Line intersects with an Aabb.
intersectionPoint	Tests if a Line intersects with an Aabb.
intersectionPoint	Tests if a Line intersects with an Obb.
intersectionPoint	Tests if a Line intersects with an Obb.
intersectionPoint	Tests if a Line intersects with an Obb.
intersectionPoint	Tests if a Line intersects with a Sphere.
intersectionPoint	Tests if a Line intersects with a Sphere.
intersectionPoint	Tests if a Line intersects with a Sphere.
intersectionPoint	Tests if a Ray intersects with a Plane.
intersectionPoint	Tests if a Ray intersects with a Plane.
intersectionPoint	Tests if a Ray intersects with an Aabb.
intersectionPoint	Tests if a Ray intersects with an Aabb.

Function	Description
intersectionPoint	Tests if a Ray intersects with an Aabb.
intersectionPoint	Tests if a Ray intersects with an Obb.
intersectionPoint	Tests if a Ray intersects with an Obb.
intersectionPoint	Tests if a Ray intersects with an Obb.
intersectionPoint	Tests if a Ray intersects with a Sphere.
intersectionPoint	Tests if a Ray intersects with a Sphere.
intersectionPoint	Tests if a Ray intersects with a Sphere.
intersectionPoint	Tests if a Segment intersects with a Plane.
intersectionPoint	Tests if a Segment intersects with a Plane.
intersectionPoint	Tests if a Segment intersects with an Aabb.
intersectionPoint	Tests if a Segment intersects with an Aabb.
intersectionPoint	Tests if a Segment intersects with an Aabb.
intersectionPoint	Tests if a Segment intersects with an Obb.
intersectionPoint	Tests if a Segment intersects with an Obb.
intersectionPoint	Tests if a Segment intersects with an Obb.
intersectionPoint	Tests if a Segment intersects with a Sphere.
intersectionPoint	Tests if a Segment intersects with a Sphere.
intersectionPoint	Tests if a Segment intersects with a Sphere.
isContainedWithin	Tests if an Aabb is inside or at the limit of another Aabb.
isIntersecting	Tests if a Sphere intersects with another Sphere.
isIntersecting	Tests if a Sphere intersects with a Capsule.
isIntersecting	Tests if a Sphere intersects with an Aabb.
isIntersecting	Tests if a Sphere intersects with an Obb.
isIntersecting	Tests if a Capsule intersects with another Capsule.
isIntersecting	Tests if an Aabb intersects with another Aabb.
isIntersecting	Tests if an Aabb intersects with an Obb.
isIntersecting	Tests if an Obb intersects with another Obb.
isIntersecting	Tests if a Frustum intersects with a Sphere.
isIntersecting	Tests if a Frustum intersects with an Aabb.
isIntersecting	Tests if a Frustum intersects with an Obb.
isOnOrAbovePlane	Determines if a point is on or above a Plane.
isOnOrBelowPlane	Determines if a point is on or below a Plane.
isOnPlane	Determines if a point is on a Plane.

Inner Classes, Structures, and Namespaces

Item	Description
sce::Geometry::Aos::Aabb	A 3D axis-aligned bounding box representation.
sce::Geometry::Aos::Bounds	A 3D bounds limit representation.
sce::Geometry::Aos::Capsule	A 3D swept-sphere capsule representation.
sce::Geometry::Aos::Frustum	A 3D frustum representation.
sce::Geometry::Aos::Line	A 3D line representation.
sce::Geometry::Aos::Obb	A 3D oriented bounding box representation.
sce::Geometry::Aos::Plane	A 3D plane representation.
sce::Geometry::Aos::Ray	A 3D ray representation.
sce::Geometry::Aos::Segment	A 3D line segment representation.
sce::Geometry::Aos::Sphere	A 3D sphere representation.

Type Definitions

Aabb_arg

A type used when passing an Aabb as a function argument.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            typedef const Aabb &Aabb_arg;
        }
    }
}
```

Description

A type used when passing an Aabb as a function argument.

Bounds_arg

A type used when passing Bounds as a function argument.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            typedef const Bounds &Bounds_arg;
        }
    }
}
```

Description

A type used when passing Bounds as a function argument.

Capsule_arg

A type used when passing a Capsule as a function argument.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            typedef const Capsule &Capsule_arg;
        }
    }
}
```

Description

A type used when passing a Capsule as a function argument.

Frustum_arg

A type used when passing a Frustum as a function argument.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            typedef const Frustum &Frustum_arg;
        }
    }
}
```

Description

A type used when passing a Frustum as a function argument.

Document serial number: 000004892117

Line_arg

A type used when passing a Line as a function argument.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            typedef const Line &Line_arg;
        }
    }
}
```

Description

A type used when passing a Line as a function argument.

Obb_arg

A type used when passing an Obb as a function argument.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            typedef const Obb &Obb_arg;
        }
    }
}
```

Description

A type used when passing an Obb as a function argument.

Plane_arg

A type used when passing a Plane as a function argument.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            typedef const Plane &Plane_arg;
        }
    }
}
```

Description

A type used when passing a Plane as a function argument.

Ray_arg

A type used when passing a Ray as a function argument.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            typedef const Ray &Ray_arg;
        }
    }
}
```

Description

A type used when passing a Ray as a function argument.

Segment_arg

A type used when passing a Segment as a function argument.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            typedef const Segment &Segment_arg;
        }
    }
}
```

Description

A type used when passing a Segment as a function argument.

Sphere_arg

A type used when passing a Sphere as a function argument.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            typedef const Sphere &Sphere_arg;
        }
    }
}
```

Description

A type used when passing a Sphere as a function argument.

Functions

closestPoint

Gets the closest point on a Line to a point.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Point3 closestPoint(
                sce::Geometry::Math::Aos::Point3_arg source,
                Line_arg target,
                sce::Geometry::Math::floatInVec *pSqrDistance
            );
        }
    }
}
```

Arguments

source	The 3D test point.
target	The 3D target Line.
pSqrDistance	Output. The squared distance between test point and closest point on the Line.

Return Values

The closest point on the Line.

Description

Gets the closest point on a Line to a point.

Notes

This function will assert if *pSqrDistance* is not valid.

closestPoint

Gets the closest point on a line Ray to a point.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Point3 closestPoint(
                sce::Geometry::Math::Aos::Point3_arg source,
                Ray_arg target,
                sce::Geometry::Math::floatInVec *pSqrDistance
            );
        }
    }
}
```

Arguments

source	The 3D test point.
target	The 3D target line Ray.
pSqrDistance	Output. The squared distance between the test point and closest point on the line Ray.

Return Values

The closest point on the line Ray.

Description

Gets the closest point on a line Ray to a point.

Notes

This function will assert if *pSqrDistance* is not valid.

closestPoint

Gets the closest point on a line Segment to a point.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Point3 closestPoint(
                sce::Geometry::Math::Aos::Point3_arg source,
                Segment_arg target,
                sce::Geometry::Math::floatInVec *pSqrDistance
            );
        }
    }
}
```

Arguments

source	The 3D test point.
target	The 3D target line Segment.
pSqrDistance	Output. The squared distance between the test point and closest point on the line Segment.

Return Values

The closest point on the line Segment.

Description

Gets the closest point on a line Segment to a point.

Notes

This function will assert if *pSqrDistance* is not valid.

closestPoint

Gets the closest point on a Plane to another point.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Point3 closestPoint(
                sce::Geometry::Math::Aos::Point3_arg source,
                Plane_arg target,
                sce::Geometry::Math::floatInVec *pSqrDistance
            );
        }
    }
}
```

Arguments

source	The 3D test point.
target	The 3D target Plane.
pSqrDistance	Output. The squared distance between the test point and closest point on the Plane.

Return Values

The closest point on the Plane.

Description

Gets the closest point on a Plane to another point.

Notes

This function will assert if *pSqrDistance* is not valid.

closestPoint

Gets the closest point within or upon a Sphere surface to another point.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Point3 closestPoint(
                sce::Geometry::Math::Aos::Point3_arg source,
                Sphere_arg target,
                sce::Geometry::Math::floatInVec *pSqrDistance
            );
        }
    }
}
```

Arguments

source	The 3D test point.
target	The 3D target Sphere.
pSqrDistance	Output. The squared distance between the test point and closest point within or upon the Sphere surface.

Return Values

The closest point within or upon the Sphere surface.

Description

Gets the closest point within or upon a Sphere surface to another point.

Notes

This function will assert if *pSqrDistance* is not valid.

closestPoint

Gets the closest point within or upon a Bounds surface to another point.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Point3 closestPoint(
                sce::Geometry::Math::Aos::Point3_arg source,
                Bounds_arg target,
                sce::Geometry::Math::floatInVec *pSqrDistance
            );
        }
    }
}
```

Arguments

source	The 3D test point.
target	The 3D target Bounds.
pSqrDistance	Output. The squared distance between the test point and the closest point within or upon the Bounds surface.

Return Values

The closest point within or upon the Bounds surface.

Description

Gets the closest point within or upon a Bounds surface to another point.

Notes

This function will assert if *pSqrDistance* is not valid.

closestPoint

Gets the closest point within or upon an Aabb surface to another point.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Point3 closestPoint(
                sce::Geometry::Math::Aos::Point3_arg source,
                Aabb_arg target,
                sce::Geometry::Math::floatInVec *pSqrDistance
            );
        }
    }
}
```

Arguments

source	The 3D test point.
target	The 3D target Aabb surface.
pSqrDistance	Output. The squared distance between the test point and the closest point within or upon the Aabb surface.

Return Values

The closest point within or upon the Aabb surface.

Description

Gets the closest point within or upon an Aabb surface to another point.

Notes

This function will assert if *pSqrDistance* is not valid.

closestPoint

Gets the closest point within or upon an Obb surface to another point.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Point3 closestPoint(
                sce::Geometry::Math::Aos::Point3_arg source,
                Obb_arg target,
                sce::Geometry::Math::floatInVec *pSqrDistance
            );
        }
    }
}
```

Arguments

source	The 3D test point.
target	The 3D target Obb surface.
pSqrDistance	Output. The squared distance between the test point and the closest point within or upon the Obb surface.

Return Values

The closest point within or upon the Obb surface.

Description

Gets the closest point within or upon an Obb surface to another point.

Notes

This function will assert if *pSqrDistance* is not valid.

closestPoints

Gets the closest points between two Line objects.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::floatInVec closestPoints(
                Line_arg source,
                Line_arg target,
                sce::Geometry::Math::Aos::Point3 &sourcePt,
                sce::Geometry::Math::Aos::Point3 &targetPt
            );
        }
    }
}
```

Arguments

source	The 3D source Line.
target	The 3D target Line.
sourcePt	Output. The closest point on the source Line.
targetPt	Output. The closest point on the target Line.

Return Values

The minimum squared distance between two lines.

Description

Gets the closest points between two Line objects.

closestPoints

Gets the closest points between a line Ray and a Line object.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::floatInVec closestPoints(
                Ray arg source,
                Line arg target,
                sce::Geometry::Math::Aos::Point3 &sourcePt,
                sce::Geometry::Math::Aos::Point3 &targetPt
            );
        }
    }
}
```

Arguments

source	The 3D source line Ray.
target	The 3D target Line.
sourcePt	Output. The closest point on the source line Ray.
targetPt	Output. The closest point on the target Line.

Return Values

The minimum squared distance between the line Ray and the Line.

Description

Gets the closest points between a line Ray and a Line object.

closestPoints

Gets the closest points between two line Ray objects.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::floatInVec closestPoints(
                Ray arg source,
                Ray arg target,
                sce::Geometry::Math::Aos::Point3 &sourcePt,
                sce::Geometry::Math::Aos::Point3 &targetPt
            );
        }
    }
}
```

Arguments

source	The 3D source line Ray.
target	The 3D target line Ray.
sourcePt	Output. The closest point on the source line Ray.
targetPt	Output. The closest point on the target line Ray.

Return Values

The minimum squared distance between two line Ray objects.

Description

Gets the closest points between two line Ray objects.

closestPoints

Gets the closest points between a line Segment and a Line object.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::floatInVec closestPoints(
                Segment arg source,
                Line arg target,
                sce::Geometry::Math::Aos::Point3 &sourcePt,
                sce::Geometry::Math::Aos::Point3 &targetPt
            );
        }
    }
}
```

Arguments

source	The 3D source line Segment.
target	The 3D target Line.
sourcePt	Output. The closest point on the source line Segment.
targetPt	Output. The closest point on the target Line.

Return Values

The minimum squared distance between the line Segment and the Line.

Description

Gets the closest points between a line Segment and a Line object.

closestPoints

Gets the closest points between a line Segment and a line Ray object.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::floatInVec closestPoints(
                Segment arg source,
                Ray arg target,
                sce::Geometry::Math::Aos::Point3 &sourcePt,
                sce::Geometry::Math::Aos::Point3 &targetPt
            );
        }
    }
}
```

Arguments

source	The 3D source line Segment.
target	The 3D target line Ray.
sourcePt	Output. The closest point on the source line Segment.
targetPt	Output. The closest point on the target line Ray.

Return Values

The minimum squared distance between the line Segment and the line Ray.

Description

Gets the closest points between a line Segment and a line Ray object.

closestPoints

Gets the closest points between two line Segment objects.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::floatInVec closestPoints(
                Segment arg source,
                Segment arg target,
                sce::Geometry::Math::Aos::Point3 &sourcePt,
                sce::Geometry::Math::Aos::Point3 &targetPt
            );
        }
    }
}
```

Arguments

source	The 3D source line Segment.
target	The 3D target line Segment.
sourcePt	Output. The closest point on the source line Segment.
targetPt	Output. The closest point on the target line Segment.

Return Values

The minimum squared distance between two line Segment Objects.

Description

Gets the closest points between two line Segment objects.

containsPoint

Tests if a point is inside or at the limit of the Bounds.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec containsPoint(
                sce::Geometry::Math::Aos::Point3_arg source,
                Bounds_arg target
            );
        }
    }
}
```

Arguments

source	The 3D test point.
target	The 3D target Bounds.

Return Values

Returns true if the point is inside or at the limit of the Bounds. Returns false if it is not.

Description

Tests if a point is inside or at the limit of the Bounds.

containsPoint

Tests if a point is inside or at the limit of the Sphere.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec containsPoint(
                sce::Geometry::Math::Aos::Point3_arg source,
                Sphere_arg target
            );
        }
    }
}
```

Arguments

source	The 3D test point.
target	The 3D target Sphere.

Return Values

Returns true if the point is inside or at the limit of the Sphere. Returns false if it is not.

Description

Tests if a point is inside or at the limit of the Sphere.

containsPoint

Tests if a point is inside or at the limit of the Capsule.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec containsPoint(
                sce::Geometry::Math::Aos::Point3_arg source,
                Capsule_arg target
            );
        }
    }
}
```

Arguments

source	The 3D test point.
target	The 3D target Capsule.

Return Values

Returns true if the point is inside or at the limit of the Capsule. Returns false if it is not.

Description

Tests if a point is inside or at the limit of the Capsule.

containsPoint

Tests if a point is inside or at the limit of the Aabb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec containsPoint(
                sce::Geometry::Math::Aos::Point3_arg source,
                Aabb_arg target
            );
        }
    }
}
```

Arguments

source	The 3D test point.
target	The 3D target Aabb.

Return Values

Returns true if the point is inside or at the limit of the Aabb. Returns false if it is not.

Description

Tests if a point is inside or at the limit of the Aabb.

containsPoint

Tests if a point is inside or at the limit of the Obb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec containsPoint(
                sce::Geometry::Math::Aos::Point3_arg source,
                Obb_arg target
            );
        }
    }
}
```

Arguments

source	The 3D test point.
target	The 3D target Obb.

Return Values

Returns true if the point is inside or at the limit of the Obb. Returns false if it is not.

Description

Tests if a point is inside or at the limit of the Obb.

containsPoint

Tests if a point is inside or at the limit of the Frustum.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec containsPoint(
                sce::Geometry::Math::Aos::Point3_arg source,
                Frustum_arg target,
                sce::Geometry::Math::boolInVec_arg ignoreFarPlane =
                sce::Geometry::Math::boolInVec(false)
            );
        }
    }
}
```

Arguments

source	The 3D test point.
target	The 3D target Frustum.
ignoreFarPlane	A flag that indicates whether the far plane should be ignored in the test.

Return Values

Returns true if the point is inside or at the limit of the Frustum. Returns false if it is not.

Description

Tests if a point is inside or at the limit of the Frustum.

intersectionPoint

Tests if a Line intersects with a Plane.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Line_arg source,
                Plane_arg target,
                sce::Geometry::Math::floatInVec *pMinT
            );
        }
    }
}
```

Arguments

source	The 3D test Line.
target	The 3D target Plane.
pMinT	Output. Receives the Line parametric value in the event of a collision. If a collision does not occur, *pMinT* is undefined.

Return Values

Returns true if the Line intersects with the Plane. Returns false if it does not.

Description

Tests if a Line intersects with a Plane.

Notes

This function will assert if *pMinT* is not valid.

intersectionPoint

Tests if a Line intersects with a Plane.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Line_arg source,
                Plane_arg target,
                sce::Geometry::Math::Aos::Point3 *pCollision
            );
        }
    }
}
```

Arguments

source	The 3D test Line.
target	The 3D target Plane.
pCollision	Output. Receives the 3D collision point in the event of a collision. If a collision does not occur, *pCollision* is undefined.

Return Values

Returns true if the Line intersects with the Plane. Returns false if it does not.

Description

Tests if a Line intersects with a Plane.

Notes

This function will assert if *pCollision* is not valid.

intersectionPoint

Tests if a Line intersects with an Aabb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Line_arg source,
                Aabb_arg target,
                sce::Geometry::Math::floatInVec *pMinT
            );
        }
    }
}
```

Arguments

source	The 3D test Line.
target	The 3D target Aabb.
pMinT	Output. Receives the minimum Line parametric value in the event of a collision. If a collision does not occur, *pMinT* is undefined.

Return Values

Returns true if the Line intersects with the Aabb. Returns false if it does not.

Description

Tests if a Line intersects with an Aabb.

Notes

This function will assert if *pMinT* is not valid.

intersectionPoint

Tests if a Line intersects with an Aabb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Line_arg source,
                Aabb_arg target,
                sce::Geometry::Math::Aos::Point3 *pCollision
            );
        }
    }
}
```

Arguments

source	The 3D test Line.
target	The 3D target Aabb.
pCollision	Output. Receives the minimum parametric 3D collision point along the Line, in the event of a collision. If a collision does not occur, *pCollision* is undefined.

Return Values

Returns true if the Line intersects with the Aabb. Returns false if it does not.

Description

Tests if a Line intersects with an Aabb.

Notes

This function will assert if *pCollision* is not valid.

intersectionPoint

Tests if a Line intersects with an Aabb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Line_arg source,
                Aabb_arg target,
                sce::Geometry::Math::Aos::Point3 *pCollision,
                sce::Geometry::Math::Aos::Vector3 *pNormal
            );
        }
    }
}
```

Arguments

source	The 3D test Line.
target	The 3D target Aabb.
pCollision	Output. Receives the minimum parametric 3D collision point along the Line, in the event of a collision. If a collision does not occur, *pCollision* is undefined.
pNormal	Output. Receives the 3D collision contact normal in the event of a collision. If a collision does not occur, *pNormal* is undefined.

Return Values

Returns true if the Line intersects with the Aabb. Returns false if it does not.

Description

Tests if a Line intersects with an Aabb.

Notes

This function will assert if *pCollision* or *pNormal* are not valid.

The collision normal for a valid collision will always point outward from the Aabb.

intersectionPoint

Tests if a Line intersects with an Obb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Line_arg source,
                Obb_arg target,
                sce::Geometry::Math::floatInVec *pMinT
            );
        }
    }
}
```

Arguments

source	The 3D test Line.
target	The 3D target Obb.
pMinT	Output. Receives the minimum Line parametric value in the event of a collision. If a collision does not occur, *pMinT* is undefined.

Return Values

Returns true if the Line intersects with the Obb. Returns false if it does not.

Description

Tests if a Line intersects with an Obb.

Notes

This function will assert if *pMinT* is not valid.

intersectionPoint

Tests if a Line intersects with an Obb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Line_arg source,
                Obb_arg target,
                sce::Geometry::Math::Aos::Point3 *pCollision
            );
        }
    }
}
```

Arguments

source	The 3D test Line.
target	The 3D target Obb.
pCollision	Output. Receives the minimum parametric 3D collision point along the Line, in the event of a collision. If a collision does not occur, *pCollision* is undefined.

Return Values

Returns true if the Line intersects with the Obb. Returns false if it does not.

Description

Tests if a Line intersects with an Obb.

Notes

This function will assert if *pCollision* is not valid.

intersectionPoint

Tests if a Line intersects with an Obb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Line_arg source,
                Obb_arg target,
                sce::Geometry::Math::Aos::Point3 *pCollision,
                sce::Geometry::Math::Aos::Vector3 *pNormal
            );
        }
    }
}
```

Arguments

source	The 3D test Line.
target	The 3D target Obb.
pCollision	Output. Receives the minimum parametric 3D collision point along the Line, in the event of a collision. If a collision does not occur, *pCollision* is undefined.
pNormal	Output. Receives the 3D collision contact normal in the event of a collision. If a collision does not occur, *pNormal* is undefined.

Return Values

Returns true if the Line intersects with the Obb. Returns false if it does not.

Description

Tests if a Line intersects with an Obb.

Notes

This function will assert if *pCollision* or *pNormal* are not valid.

The collision normal for a valid collision will always point outward from the Obb.

intersectionPoint

Tests if a Line intersects with a Sphere.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Line_arg source,
                Sphere_arg target,
                sce::Geometry::Math::floatInVec *pMinT
            );
        }
    }
}
```

Arguments

source	The 3D test Line.
target	The 3D target Sphere.
pMinT	Output. Receives the minimum Line parametric value in the event of a collision. If a collision does not occur, *pMinT* is undefined.

Return Values

Returns true if the Line intersects with the Sphere. Returns false if it does not.

Description

Tests if a Line intersects with a Sphere.

Notes

This function will assert if *pMinT* is not valid.

intersectionPoint

Tests if a Line intersects with a Sphere.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Line_arg source,
                Sphere_arg target,
                sce::Geometry::Math::Aos::Point3 *pCollision
            );
        }
    }
}
```

Arguments

source	The 3D test Line.
target	The 3D target Sphere.
pCollision	Output. Receives the minimum parametric 3D collision point along the Line, in the event of a collision. If a collision does not occur, *pCollision* is undefined.

Return Values

Returns true if the Line intersects with the Sphere. Returns false if it does not.

Description

Tests if a Line intersects with a Sphere.

Notes

This function will assert if *pCollision* is not valid.

intersectionPoint

Tests if a Line intersects with a Sphere.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Line_arg source,
                Sphere_arg target,
                sce::Geometry::Math::Aos::Point3 *pCollision,
                sce::Geometry::Math::Aos::Vector3 *pNormal
            );
        }
    }
}
```

Arguments

source	The 3D test Line.
target	The 3D target Sphere.
pCollision	Output. Receives the minimum parametric 3D collision point along the Line, in the event of a collision. If a collision does not occur, *pCollision* is undefined.
pNormal	Output. Receives the 3D collision contact normal in the event of a collision. If a collision does not occur, *pNormal* is undefined.

Return Values

Returns true if the Line intersects with the Sphere. Returns false if it does not.

Description

Tests if a Line intersects with a Sphere.

Notes

This function will assert if *pCollision* or *pNormal* are not valid.

The collision normal for a valid collision will always point outward from the Sphere.

intersectionPoint

Tests if a Ray intersects with a Plane.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Ray_arg source,
                Plane_arg target,
                sce::Geometry::Math::floatInVec *pMinT
            );
        }
    }
}
```

Arguments

source	The 3D test Ray.
target	The 3D target Plane.
pMinT	Output. Receives the Ray parametric value in the event of a collision. If a collision does not occur, *pMinT* is undefined.

Return Values

Returns true if the Ray intersects with the Plane. Returns false if it does not.

Description

Tests if a Ray intersects with a Plane.

Notes

This function will assert if *pMinT* is not valid.

intersectionPoint

Tests if a Ray intersects with a Plane.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Ray_arg source,
                Plane_arg target,
                sce::Geometry::Math::Aos::Point3 *pCollision
            );
        }
    }
}
```

Arguments

source	The 3D test Ray.
target	The 3D target Plane.
pCollision	Output. Receives the 3D collision point in the event of a collision. If a collision does not occur, *pCollision* is undefined.

Return Values

Returns true if the Ray intersects with the Plane. Returns false if it does not.

Description

Tests if a Ray intersects with a Plane.

Notes

This function will assert if *pCollision* is not valid.

intersectionPoint

Tests if a Ray intersects with an Aabb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Ray_arg source,
                Aabb_arg target,
                sce::Geometry::Math::floatInVec *pMinT
            );
        }
    }
}
```

Arguments

source	The 3D test Ray.
target	The 3D target Aabb.
pMinT	Output. Receives the minimum Ray parametric value in the event of a collision. If a collision does not occur, *pMinT* is undefined.

Return Values

Returns true if the Ray intersects with the Aabb. Returns false if it does not.

Description

Tests if a Ray intersects with an Aabb.

Notes

This function will assert if *pMinT* is not valid.

intersectionPoint

Tests if a Ray intersects with an Aabb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Ray_arg source,
                Aabb_arg target,
                sce::Geometry::Math::Aos::Point3 *pCollision
            );
        }
    }
}
```

Arguments

source	The 3D test Ray.
target	The 3D target Aabb.
pCollision	Output. Receives the minimum parametric 3D collision point along the Ray, in the event of a collision. If a collision does not occur, *pCollision* is undefined.

Return Values

Returns true if the Ray intersects with the Aabb. Returns false if it does not.

Description

Tests if a Ray intersects with an Aabb.

Notes

This function will assert if *pCollision* is not valid.

intersectionPoint

Tests if a Ray intersects with an Aabb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Ray_arg source,
                Aabb_arg target,
                sce::Geometry::Math::Aos::Point3 *pCollision,
                sce::Geometry::Math::Aos::Vector3 *pNormal
            );
        }
    }
}
```

Arguments

source	The 3D test Ray.
target	The 3D target Aabb.
pCollision	Output. Receives the minimum parametric 3D collision point along the Ray, in the event of a collision. If a collision does not occur, *pCollision* is undefined.
pNormal	Output. Receives the 3D collision contact normal in the event of a collision. If a collision does not occur, *pNormal* is undefined.

Return Values

Returns true if the Ray intersects with the Aabb. Returns false if it does not.

Description

Tests if a Ray intersects with an Aabb.

Notes

This function will assert if *pCollision* or *pNormal* are not valid.

If the Ray begins within the Aabb, the collision normal for a valid collision will point inward into the Aabb. The normal will point outward if the Ray does not begin within the Aabb.

intersectionPoint

Tests if a Ray intersects with an Obb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Ray_arg source,
                Obb_arg target,
                sce::Geometry::Math::floatInVec *pMinT
            );
        }
    }
}
```

Arguments

source	The 3D test Ray.
target	The 3D target Obb.
pMinT	Output. Receives the minimum Ray parametric value in the event of a collision. If a collision does not occur, *pMinT* is undefined.

Return Values

Returns true if the Ray intersects with the Obb. Returns false if it does not.

Description

Tests if a Ray intersects with an Obb.

Notes

This function will assert if *pMinT* is not valid.

intersectionPoint

Tests if a Ray intersects with an Obb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Ray_arg source,
                Obb_arg target,
                sce::Geometry::Math::Aos::Point3 *pCollision
            );
        }
    }
}
```

Arguments

source	The 3D test Ray.
target	The 3D target Obb.
pCollision	Output. Receives the minimum Ray parametric 3D collision point in the event of a collision. If a collision does not occur, *pCollision* is undefined.

Return Values

Returns true if the Ray intersects with the Obb. Returns false if it does not.

Description

Tests if a Ray intersects with an Obb.

Notes

This function will assert if *pCollision* is not valid.

intersectionPoint

Tests if a Ray intersects with an Obb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Ray_arg source,
                Obb_arg target,
                sce::Geometry::Math::Aos::Point3 *pCollision,
                sce::Geometry::Math::Aos::Vector3 *pNormal
            );
        }
    }
}
```

Arguments

source	The 3D test Ray.
target	The 3D target Obb.
pCollision	Output. Receives the minimum parametric 3D collision point along the Ray, in the event of a collision. If a collision does not occur, *pCollision* is undefined.
pNormal	Output. Receives the 3D collision contact normal in the event of a collision. If a collision does not occur, *pNormal* is undefined.

Return Values

Returns true if the Ray intersects with the Obb. Returns false if it does not.

Description

Tests if a Ray intersects with an Obb.

Notes

This function will assert if *pCollision* or *pNormal* are not valid.

If the Ray begins within the Obb, the collision normal for a valid collision will point inward into the Obb. The normal will point outward if the Ray does not begin within the Obb.

intersectionPoint

Tests if a Ray intersects with a Sphere.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Ray_arg source,
                Sphere_arg target,
                sce::Geometry::Math::floatInVec *pMinT
            );
        }
    }
}
```

Arguments

source	The 3D test Ray.
target	The 3D target Sphere.
pMinT	Output. Receives the minimum Ray parametric value in the event of a collision. If a collision does not occur, *pMinT* is undefined.

Return Values

Returns true if the Ray intersects with the Sphere. Returns false if it does not.

Description

Tests if a Ray intersects with a Sphere.

Notes

This function will assert if *pMinT* is not valid.

intersectionPoint

Tests if a Ray intersects with a Sphere.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Ray_arg source,
                Sphere_arg target,
                sce::Geometry::Math::Aos::Point3 *pCollision
            );
        }
    }
}
```

Arguments

source	The 3D test Ray.
target	The 3D target Sphere.
pCollision	Output. Receives the minimum parametric 3D collision point along the Ray, in the event of a collision. If a collision does not occur, *pCollision* is undefined.

Return Values

Returns true if the Ray intersects with the Sphere. Returns false if it does not.

Description

Tests if a Ray intersects with a Sphere.

Notes

This function will assert if *pCollision* is not valid.

intersectionPoint

Tests if a Ray intersects with a Sphere.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Ray_arg source,
                Sphere_arg target,
                sce::Geometry::Math::Aos::Point3 *pCollision,
                sce::Geometry::Math::Aos::Vector3 *pNormal
            );
        }
    }
}
```

Arguments

source	The 3D test Ray.
target	The 3D target Sphere.
pCollision	Output. Receives the minimum parametric 3D collision point along the Ray, in the event of a collision. If a collision does not occur, *pCollision* is undefined.
pNormal	Output. Receives the 3D collision contact normal in the event of a collision. If a collision does not occur, *pNormal* is undefined.

Return Values

Returns true if the Ray intersects with the Sphere. Returns false if it does not.

Description

Tests if a Ray intersects with a Sphere.

Notes

This function will assert if *pCollision* or *pNormal* are not valid.

If the Ray begins within the Sphere, the collision normal for a valid collision will point inward into the Sphere. The normal will point outward if the Ray does not begin within the Sphere.

intersectionPoint

Tests if a Segment intersects with a Plane.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Segment_arg source,
                Plane_arg target,
                sce::Geometry::Math::floatInVec *pMinT
            );
        }
    }
}
```

Arguments

source	The 3D test Segment.
target	The 3D target Plane.
pMinT	Output. Receives the Segment parametric value in the event of a collision. If a collision does not occur, *pMinT* is undefined.

Return Values

Returns true if the Segment intersects with the Plane. Returns false if it does not.

Description

Tests if a Segment intersects with a Plane.

Notes

This function will assert if *pMinT* is not valid.

intersectionPoint

Tests if a Segment intersects with a Plane.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Segment_arg source,
                Plane_arg target,
                sce::Geometry::Math::Aos::Point3 *pCollision
            );
        }
    }
}
```

Arguments

source	The 3D test Segment.
target	The 3D target Plane.
pCollision	Output. Receives the 3D collision point in the event of a collision. If a collision does not occur, *pCollision* is undefined.

Return Values

Returns true if the Segment intersects with the Plane. Returns false if it does not.

Description

Tests if a Segment intersects with a Plane.

Notes

This function will assert if *pCollision* is not valid.

intersectionPoint

Tests if a Segment intersects with an Aabb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Segment_arg source,
                Aabb_arg target,
                sce::Geometry::Math::floatInVec *pMinT
            );
        }
    }
}
```

Arguments

source	The 3D test Segment.
target	The 3D target Aabb.
pMinT	Output. Receives the minimum Segment parametric value in the event of a collision. If a collision does not occur, *pMinT* is undefined.

Return Values

Returns true if the Segment intersects with the Aabb. Returns false if it does not.

Description

Tests if a Segment intersects with an Aabb.

Notes

This function will assert if *pMinT* is not valid.

intersectionPoint

Tests if a Segment intersects with an Aabb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Segment_arg source,
                Aabb_arg target,
                sce::Geometry::Math::Aos::Point3 *pCollision
            );
        }
    }
}
```

Arguments

source	The 3D test Segment.
target	The 3D target Aabb.
pCollision	Output. Receives the minimum parametric 3D collision point along the Segment, in the event of a collision. If a collision does not occur, *pCollision* is undefined.

Return Values

Returns true if the Segment intersects with the Aabb. Returns false if it does not.

Description

Tests if a Segment intersects with an Aabb.

Notes

This function will assert if *pCollision* is not valid.

Document serial number: 000004892117

intersectionPoint

Tests if a Segment intersects with an Aabb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Segment_arg source,
                Aabb_arg target,
                sce::Geometry::Math::Aos::Point3 *pCollision,
                sce::Geometry::Math::Aos::Vector3 *pNormal
            );
        }
    }
}
```

Arguments

source	The 3D test Segment.
target	The 3D target Aabb.
pCollision	Output. Receives the minimum parametric 3D collision point along the Segment, in the event of a collision. If a collision does not occur, *pCollision* is undefined.
pNormal	Output. Receives the 3D collision contact normal in the event of a collision. If a collision does not occur, *pNormal* is undefined.

Return Values

Returns true if the Segment intersects with the Aabb. Returns false if it does not.

Description

Tests if a Segment intersects with an Aabb.

Notes

This function will assert if *pCollision* or *pNormal* are not valid.

If the Segment begins within the Aabb, the collision normal for a valid collision will point inward into the Aabb. The normal will point outward if the Segment does not begin within the Aabb.

intersectionPoint

Tests if a Segment intersects with an Obb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Segment_arg source,
                Obb_arg target,
                sce::Geometry::Math::floatInVec *pMinT
            );
        }
    }
}
```

Arguments

source	The 3D test Segment.
target	The 3D target Obb.
pMinT	Output. Receives the minimum Segment parametric value in the event of a collision. If a collision does not occur, *pMinT* is undefined.

Return Values

Returns true if the Segment intersects with the Obb. Returns false if it does not.

Description

Tests if a Segment intersects with an Obb.

Notes

This function will assert if *pMinT* is not valid.

intersectionPoint

Tests if a Segment intersects with an Obb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Segment_arg source,
                Obb_arg target,
                sce::Geometry::Math::Aos::Point3 *pCollision
            );
        }
    }
}
```

Arguments

source	The 3D test Segment.
target	The 3D target Obb.
pCollision	Output. Receives the minimum parametric 3D collision point along the Segment, in the event of a collision. If a collision does not occur, *pCollision* is undefined.

Return Values

Returns true if the Segment intersects with the Obb. Returns false if it does not.

Description

Tests if a Segment intersects with an Obb.

Notes

This function will assert if *pCollision* is not valid.

intersectionPoint

Tests if a Segment intersects with an Obb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Segment_arg source,
                Obb_arg target,
                sce::Geometry::Math::Aos::Point3 *pCollision,
                sce::Geometry::Math::Aos::Vector3 *pNormal
            );
        }
    }
}
```

Arguments

source	The 3D test Segment.
target	The 3D target Obb.
pCollision	Output. Receives the minimum parametric 3D collision point along the Segment, in the event of a collision. If a collision does not occur, *pCollision* is undefined.
pNormal	Output. Receives the 3D collision contact normal in the event of a collision. If a collision does not occur, *pNormal* is undefined.

Return Values

Returns true if the Segment intersects with the Obb. Returns false if it does not.

Description

Tests if a Segment intersects with an Obb.

Notes

This function will assert if *pCollision* or *pNormal* are not valid.

If the Segment begins within the Obb, the collision normal for a valid collision will point inward into the Obb. The normal will point outward if the Segment does not begin within the Obb.

intersectionPoint

Tests if a Segment intersects with a Sphere.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Segment_arg source,
                Sphere_arg target,
                sce::Geometry::Math::floatInVec *pMinT
            );
        }
    }
}
```

Arguments

source	The 3D test Segment.
target	The 3D target Sphere.
pMinT	Output. Receives the minimum Segment parametric value in the event of a collision. If a collision does not occur, *pMinT* is undefined.

Return Values

Returns true if the Segment intersects with the Sphere. Returns false if it does not.

Description

Tests if a Segment intersects with a Sphere.

Notes

This function will assert if *pMinT* is not valid.

intersectionPoint

Tests if a Segment intersects with a Sphere.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Segment_arg source,
                Sphere_arg target,
                sce::Geometry::Math::Aos::Point3 *pCollision
            );
        }
    }
}
```

Arguments

source	The 3D test Segment.
target	The 3D target Sphere.
pCollision	Output. Receives the minimum parametric 3D collision point along the Segment, in the event of a collision. If a collision does not occur, *pCollision* is undefined.

Return Values

Returns true if the Segment intersects with the Sphere. Returns false if it does not.

Description

Tests if a Segment intersects with a Sphere.

Notes

This function will assert if *pCollision* is not valid.

intersectionPoint

Tests if a Segment intersects with a Sphere.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            intersectionPoint(
                Segment_arg source,
                Sphere_arg target,
                sce::Geometry::Math::Aos::Point3 *pCollision,
                sce::Geometry::Math::Aos::Vector3 *pNormal
            );
        }
    }
}
```

Arguments

source	The 3D test Segment.
target	The 3D target Sphere.
pCollision	Output. Receives the minimum parametric 3D collision point along the Segment, in the event of a collision. If a collision does not occur, *pCollision* is undefined.
pNormal	Output. Receives the 3D collision contact normal in the event of a collision. If a collision does not occur, *pNormal* is undefined.

Return Values

Returns true if the Segment intersects with the Sphere. Returns false if it does not.

Description

Tests if a Segment intersects with a Sphere.

Notes

This function will assert if *pCollision* or *pNormal* are not valid.

If the Segment begins within the Sphere, the collision normal for a valid collision will point inward into the Sphere. The normal will point outward if the Segment does not begin within the Sphere.

isContainedWithin

Tests if an Aabb is inside or at the limit of another Aabb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            isContainedWithin(
                Aabb_arg container,
                Aabb_arg containee
            );
        }
    }
}
```

Arguments

container	The 3D test Aabb.
containee	The 3D target Aabb.

Return Values

Returns true if the source Aabb is inside or at the limit of the target Aabb. Returns false if it does not.

Description

Tests if an Aabb is inside or at the limit of another Aabb.

isIntersecting

Tests if a Sphere intersects with another Sphere.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec isIntersecting(
                Sphere_arg source,
                Sphere_arg target
            );
        }
    }
}
```

Arguments

source	The 3D test Sphere.
target	The 3D target Sphere.

Return Values

Returns true if the source Sphere intersects with the target Sphere. Returns false if it does not.

Description

Tests if a Sphere intersects with another Sphere.

isIntersecting

Tests if a Sphere intersects with a Capsule.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec isIntersecting(
                Sphere_arg source,
                Capsule_arg target
            );
        }
    }
}
```

Arguments

source	The 3D test Sphere.
target	The 3D target Capsule.

Return Values

Returns true if the Sphere intersects with the Capsule. Returns false if it does not.

Description

Tests if a Sphere intersects with a Capsule.

isIntersecting

Tests if a Sphere intersects with an Aabb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec isIntersecting(
                Sphere_arg source,
                Aabb_arg target
            );
        }
    }
}
```

Arguments

source	The 3D test Sphere.
target	The 3D target Aabb.

Return Values

Returns true if the Sphere intersects with the Aabb. Returns false if it does not.

Description

Tests if a Sphere intersects with an Aabb.

isIntersecting

Tests if a Sphere intersects with an Obb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec isIntersecting(
                Sphere_arg source,
                Obb_arg target
            );
        }
    }
}
```

Arguments

source	The 3D test Sphere.
target	The 3D target Obb.

Return Values

Returns true if the Sphere intersects with the Obb. Returns false if it does not.

Description

Tests if a Sphere intersects with an Obb.

isIntersecting

Tests if a Capsule intersects with another Capsule.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec isIntersecting(
                Capsule_arg source,
                Capsule_arg target
            );
        }
    }
}
```

Arguments

source	The 3D test Capsule.
target	The 3D target Capsule.

Return Values

Returns true if the source Capsule intersects with the target Capsule. Returns false if it does not.

Description

Tests if a Capsule intersects with another Capsule.

isIntersecting

Tests if an Aabb intersects with another Aabb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec isIntersecting(
                Aabb_arg source,
                Aabb_arg target
            );
        }
    }
}
```

Arguments

source	The 3D test Aabb.
target	The 3D target Aabb.

Return Values

Returns true if the source Aabb intersects with the target Aabb. Returns false if it does not.

Description

Tests if an Aabb intersects with another Aabb.

isIntersecting

Tests if an Aabb intersects with an Obb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec isIntersecting(
                Aabb_arg source,
                Obb_arg target
            );
        }
    }
}
```

Arguments

source	The 3D test Aabb.
target	The 3D target Obb.

Return Values

Returns true if the Aabb intersects with the Obb. Returns false if it does not.

Description

Tests if an Aabb intersects with an Obb.

isIntersecting

Tests if an Obb intersects with another Obb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec isIntersecting(
                Obb_arg source,
                Obb_arg target
            );
        }
    }
}
```

Arguments

source	The 3D test Obb.
target	The 3D target Obb.

Return Values

Returns true if the source Obb intersects with the target Obb. Returns false if it does not.

Description

Tests if an Obb intersects with another Obb.

Document serial number: 000004892117

isIntersecting

Tests if a Frustum intersects with a Sphere.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec isIntersecting(
                Frustum_arg source,
                Sphere_arg target,
                sce::Geometry::Math::boolInVec_arg ignoreFarPlane =
                sce::Geometry::Math::boolInVec(false)
            );
        }
    }
}
```

Arguments

source	The 3D test Frustum.
target	The 3D target Sphere.
ignoreFarPlane	A flag that indicates whether the far plane should be ignored in the test.

Return Values

Returns true if the Frustum intersects with the Sphere. Returns false if it does not.

Description

Tests if a Frustum intersects with a Sphere.

isIntersecting

Tests if a Frustum intersects with an Aabb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec isIntersecting(
                Frustum arg source,
                Aabb arg target,
                sce::Geometry::Math::boolInVec_arg ignoreFarPlane =
                sce::Geometry::Math::boolInVec(false)
            );
        }
    }
}
```

Arguments

source	The 3D test Frustum.
target	The 3D target Aabb.
ignoreFarPlane	A flag that indicates whether the far plane should be ignored in the test.

Return Values

Returns true if the Frustum intersects with the Aabb. Returns false if it does not.

Description

Tests if a Frustum intersects with an Aabb.

isIntersecting

Tests if a Frustum intersects with an Obb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec isIntersecting(
                Frustum_arg source,
                Obb_arg target,
                sce::Geometry::Math::boolInVec_arg ignoreFarPlane =
                sce::Geometry::Math::boolInVec(false)
            );
        }
    }
}
```

Arguments

source	The 3D test Frustum.
target	The 3D target Obb.
ignoreFarPlane	A flag that indicates whether the far plane should be ignored in the test.

Return Values

Returns true if the Frustum intersects with the Obb. Returns false if it does not.

Description

Tests if a Frustum intersects with an Obb.

isOnOrAbovePlane

Determines if a point is on or above a Plane.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            isOnOrAbovePlane(
                sce::Geometry::Math::Aos::Point3_arg source,
                Plane_arg target
            );
        }
    }
}
```

Arguments

source	The 3D test point.
target	The 3D target Plane.

Return Values

Returns true if a point is on or above a Plane. Returns false if it is not.

Description

Determines if a point is on or above a Plane.

isOnOrBelowPlane

Determines if a point is on or below a Plane.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec
            isOnOrBelowPlane(
                sce::Geometry::Math::Aos::Point3_arg source,
                Plane_arg target
            );
        }
    }
}
```

Arguments

source	The 3D test point.
target	The 3D target Plane.

Return Values

Returns true if a point is on or below a Plane. Returns false if it is not.

Description

Determines if a point is on or below a Plane.

isOnPlane

Determines if a point is on a Plane.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            SCE_GEOMETRY_INLINE sce::Geometry::Math::boolInVec isOnPlane(
                sce::Geometry::Math::Aos::Point3_arg source,
                Plane_arg target
            );
        }
    }
}
```

Arguments

source	The 3D test point.
target	The 3D target Plane.

Return Values

Returns true if a point is on a Plane. Returns false if it is not.

Description

Determines if a point is on a Plane.

sce::Geometry::Aos::Aabb

Summary

sce::Geometry::Aos::Aabb

A 3D axis-aligned bounding box representation.

Definition

```
#include <sce_geometry.h>
class Aabb {};
```

Description

A 3D axis-aligned bounding box represented by a center point (origin) and the half-widths.

Methods Summary

Methods	Description
Aabb	The default constructor. This does no initialization.
Aabb	Constructs a 3D Aabb from bounds.
Aabb	Constructs a 3D Aabb from two points.
Aabb	Constructs a 3D Aabb from a center and half-widths.
getHalfWidths	Gets the half-widths of the 3D Aabb.
getMaximum	Gets the maximum bounds of the 3D Aabb.
getMinimum	Gets the minimum bounds of the 3D Aabb.
getOrigin	Gets the center (origin) of the 3D Aabb.
setHalfWidths	Sets the half-widths of the 3D Aabb.
setOrigin	Sets the center (origin) of the 3D Aabb.

Constructors and Destructors

Aabb

The default constructor. This does no initialization.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Aabb {
                SCE_GEOMETRY_ALWAYS_INLINE Aabb();
            }
        }
    }
}
```

Arguments

None

Return Values

None

Description

The default constructor. This does no initialization.

Aabb

Constructs a 3D Aabb from bounds.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Aabb {
                explicit SCE_GEOMETRY_INLINE Aabb(
                    Bounds_arg bounds
                );
            }
        }
    }
}
```

Arguments

bounds	The bounds limits.

Return Values

None

Description

Constructs a 3D Aabb from bounds.

Aabb

Constructs a 3D Aabb from two points.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Aabb {
                SCE_GEOMETRY_INLINE Aabb(
                    sce::Geometry::Math::Aos::Point3_arg pnt0,
                    sce::Geometry::Math::Aos::Point3_arg pnt1
                );
            }
        }
    }
}
```

Arguments

pnt0	The first point.
pnt1	The second point.

Return Values

None

Description

Constructs a 3D Aabb from two points.

Aabb

Constructs a 3D Aabb from a center and half-widths.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Aabb {
                SCE_GEOMETRY_INLINE Aabb(
                    sce::Geometry::Math::Aos::Point3_arg center,
                    sce::Geometry::Math::Aos::Vector3_arg halfWidths
                );
            }
        }
    }
}
```

Arguments

center	The center point (origin) of the 3D Aabb.
halfWidths	The half-widths of the 3D Aabb.

Return Values

None

Description

Constructs a 3D Aabb from a center and half-widths.

Public Instance Methods

getHalfWidths

Gets the half-widths of the 3D Aabb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Aabb {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Vector3
                getHalfWidths() const;
            }
        }
    }
}
```

Arguments

None

Return Values

The half-widths of the 3D Aabb.

Description

Gets the half-widths of the 3D Aabb.

Document serial number: 000004892117

getMaximum

Gets the maximum bounds of the 3D Aabb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Aabb {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Point3
                getMaximum() const;
            }
        }
    }
}
```

Arguments

None

Return Values

The maximum bounds of the 3D Aabb.

Description

Gets the maximum bounds of the 3D Aabb.

getMinimum

Gets the minimum bounds of the 3D Aabb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Aabb {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Point3
                getMinimum() const;
            }
        }
    }
}
```

Arguments

None

Return Values

The minimum bounds of the 3D Aabb.

Description

Gets the minimum bounds of the 3D Aabb.

getOrigin

Gets the center (origin) of the 3D Aabb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Aabb {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Point3
                getOrigin() const;
            }
        }
    }
}
```

Arguments

None

Return Values

The origin of the 3D Aabb.

Description

Gets the center (origin) of the 3D Aabb.

setHalfWidths

Sets the half-widths of the 3D Aabb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Aabb {
                SCE_GEOMETRY_INLINE Aabb &setHalfWidths(
                    sce::Geometry::Math::Aos::Vector3_arg halfWidths
                );
            }
        }
    }
}
```

Arguments

halfWidths	The half-widths of the 3D Aabb.

Return Values

A reference to the resulting 3D Aabb.

Description

Sets the half-widths of the 3D Aabb.

setOrigin

Sets the center (origin) of the 3D Aabb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Aabb {
                SCE_GEOMETRY_INLINE Aabb &setOrigin(
                    sce::Geometry::Math::Aos::Point3_arg origin
                );
            }
        }
    }
}
```

Arguments

origin	The Aabb origin.

Return Values

A reference to the resulting 3D Aabb.

Description

Sets the center (origin) of the 3D Aabb.

sce::Geometry::Aos::Bounds

Summary

sce::Geometry::Aos::Bounds

A 3D bounds limit representation.

Definition

```
#include <sce_geometry.h>
class Bounds {};
```

Description

A 3D bounds limit represented by points at the minimum and maximum extent in each dimension.

Methods Summary

Methods	Description
Bounds	The default constructor. This does no initialization.
Bounds	Constructs bounds limits from one point.
Bounds	Constructs bounds limits from two points.
extend	Extends the bounds limits to envelop a point.
extend	Extends the bounds limits to envelop another 3D bounds.
getMaximum	Gets the maximum bounds of the 3D bounds.
getMinimum	Gets the minimum bounds of the 3D bounds.
setMaximum	Sets the maximum bounds of the 3D bounds.
setMinimum	Sets the minimum bounds of the 3D bounds.

Constructors and Destructors

Bounds

The default constructor. This does no initialization.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Bounds {
                SCE_GEOMETRY_ALWAYS_INLINE Bounds();
            }
        }
    }
}
```

Arguments

None

Return Values

None

Description

The default constructor. This does no initialization.

Bounds

Constructs bounds limits from one point.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Bounds {
                explicit SCE_GEOMETRY_INLINE Bounds(
                    sce::Geometry::Math::Aos::Point3_arg pnt
                );
            }
        }
    }
}
```

Arguments

pnt	The point to construct the bounds limits from.

Return Values

None

Description

Constructs bounds limits from one point.

Bounds

Constructs bounds limits from two points.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Bounds {
                SCE_GEOMETRY_INLINE Bounds(
                    sce::Geometry::Math::Aos::Point3_arg pnt0,
                    sce::Geometry::Math::Aos::Point3_arg pnt1
                );
            }
        }
    }
}
```

Arguments

pnt0	The first point.
pnt1	The second point.

Return Values

None

Description

Constructs bounds limits from two points.

Public Instance Methods

extend

Extends the bounds limits to envelop a point.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Bounds {
                SCE_GEOMETRY_INLINE Bounds &extend(
                    sce::Geometry::Math::Aos::Point3_arg pnt
                );
            }
        }
    }
}
```

Arguments

pnt	The point to be enveloped.

Return Values

A reference to the resulting 3D bounds.

Description

Extends the bounds limits to envelop a point.

extend

Extends the bounds limits to envelop another 3D bounds.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Bounds {
                SCE_GEOMETRY_INLINE Bounds &extend(
                    Bounds_arg bounds
                );
            }
        }
    }
}
```

Arguments

bounds	The other 3D bounds to be enveloped.

Return Values

A reference to the resulting 3D bounds.

Description

Extends the bounds limits to envelop another 3D bounds.

getMaximum

Gets the maximum bounds of the 3D bounds.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Bounds {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Point3
                getMaximum() const;
            }
        }
    }
}
```

Arguments

None

Return Values

The maximum bounds of the 3D bounds.

Description

Gets the maximum bounds of the 3D bounds.

getMinimum

Gets the minimum bounds of the 3D bounds.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Bounds {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Point3
                getMinimum() const;
            }
        }
    }
}
```

Arguments

None

Return Values

The minimum bounds of the 3D bounds.

Description

Gets the minimum bounds of the 3D bounds.

setMaximum

Sets the maximum bounds of the 3D bounds.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Bounds {
                SCE_GEOMETRY_INLINE Bounds &setMaximum(
                    sce::Geometry::Math::Aos::Point3_arg maximum
                );
            }
        }
    }
}
```

Arguments

maximum	The maximum bounds.

Return Values

A reference to the resulting 3D bounds.

Description

Sets the maximum bounds of the 3D bounds.

setMinimum

Sets the minimum bounds of the 3D bounds.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Bounds {
                SCE_GEOMETRY_INLINE Bounds &setMinimum(
                    sce::Geometry::Math::Aos::Point3_arg minimum
                );
            }
        }
    }
}
```

Arguments

minimum	The minimum bounds.

Return Values

A reference to the resulting 3D bounds.

Description

Sets the minimum bounds of the 3D bounds.

sce::Geometry::Aos::Capsule

Summary

sce::Geometry::Aos::Capsule

A 3D swept-sphere capsule representation.

Definition

```
#include <sce_geometry.h>
class Capsule {};
```

Description

A 3D swept-sphere capsule represented by two points and the radius of the hemispherical cap.

Methods Summary

Methods	Description
Capsule	The default constructor. This does no initialization.
Capsule	Constructs a 3D capsule from start and end points and a radius.
getPoint0	Gets the start point of the 3D capsule.
getPoint1	Gets the end point of the 3D capsule.
getRadius	Gets the radius of the 3D capsule.
setPoint0	Sets the start point of the 3D capsule.
setPoint1	Sets the end point of the 3D capsule.
setRadius	Sets the radius of the 3D capsule.

Constructors and Destructors

Capsule

The default constructor. This does no initialization.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Capsule {
                SCE_GEOMETRY_ALWAYS_INLINE Capsule();
            }
        }
    }
}
```

Arguments

None

Return Values

None

Description

The default constructor. This does no initialization.

Capsule

Constructs a 3D capsule from start and end points and a radius.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Capsule {
                SCE_GEOMETRY_INLINE Capsule(
                    sce::Geometry::Math::Aos::Point3_arg pnt0,
                    sce::Geometry::Math::Aos::Point3_arg pnt1,
                    sce::Geometry::Math::floatInVec_arg radius
                );
            }
        }
    }
}
```

Arguments

pnt0	The start point.
pnt1	The end point.
radius	The capsule radius.

Return Values

None

Description

Constructs a 3D capsule from start and end points and a radius.

Notes

This function will assert if the radius has a negative value.

Public Instance Methods

getPoint0

Gets the start point of the 3D capsule.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Capsule {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Point3
                getPoint0() const;
            }
        }
    }
}
```

Arguments

None

Return Values

The start point of the 3D capsule.

Description

Gets the start point of the 3D capsule.

getPoint1

Gets the end point of the 3D capsule.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Capsule {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Point3
                getPoint1() const;
            }
        }
    }
}
```

Arguments

None

Return Values

The end point of the 3D capsule.

Description

Gets the end point of the 3D capsule.

getRadius

Gets the radius of the 3D capsule.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Capsule {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::floatInVec getRadius()
                const;
            }
        }
    }
}
```

Arguments

None

Return Values

The radius of the 3D capsule.

Description

Gets the radius of the 3D capsule.

setPoint0

Sets the start point of the 3D capsule.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Capsule {
                SCE_GEOMETRY_INLINE Capsule &setPoint0(
                    sce::Geometry::Math::Aos::Point3_arg pnt0
                );
            }
        }
    }
}
```

Arguments

pnt0	The start point.

Return Values

A reference to the resulting 3D capsule.

Description

Sets the start point of the 3D capsule.

setPoint1

Sets the end point of the 3D capsule.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Capsule {
                SCE_GEOMETRY_INLINE Capsule &setPoint1(
                    sce::Geometry::Math::Aos::Point3_arg pnt1
                );
            }
        }
    }
}
```

Arguments

pnt1	The end point.

Return Values

A reference to the resulting 3D capsule.

Description

Sets the end point of the 3D capsule.

setRadius

Sets the radius of the 3D capsule.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Capsule {
                SCE_GEOMETRY_INLINE Capsule &setRadius(
                    sce::Geometry::Math::floatInVec_arg radius
                );
            }
        }
    }
}
```

Arguments

radius	The radius of the capsule.

Return Values

A reference to the resulting 3D capsule.

Description

Sets the radius of the 3D capsule.

Notes

This function will assert if the radius has a negative value.

sce::Geometry::Aos::Frustum

Summary

sce::Geometry::Aos::Frustum

A 3D frustum representation.

Definition

```
#include <sce_geometry.h>
class Frustum {};
```

Description

A 3D frustum represented by six bounding planes.

Methods Summary

Methods	Description
Frustum	The default constructor. This does no initialization.
Frustum	Constructs a 3D frustum from a projection matrix.
Frustum	Constructs a 3D frustum from a perspective projection camera.
Frustum	Constructs a 3D frustum from axis limits.
Frustum	Constructs a 3D frustum from the bounding planes.
getPlane	Gets the specified bounding plane of the 3D frustum.
setPlane	Sets the specified bounding plane of the 3D frustum.

Constructors and Destructors

Frustum

The default constructor. This does no initialization.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Frustum {
                SCE_GEOMETRY_ALWAYS_INLINE Frustum();
            }
        }
    }
}
```

Arguments

None

Return Values

None

Description

The default constructor. This does no initialization.

Frustum

Constructs a 3D frustum from a projection matrix.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Frustum {
                explicit SCE_GEOMETRY_INLINE Frustum(
                    sce::Geometry::Math::Aos::Matrix4_arg projectionMatrix
                );
            }
        }
    }
}
```

Arguments

projectionMatrix The projection matrix.

Return Values

None

Description

Constructs a 3D frustum from a projection matrix.

Frustum

Constructs a 3D frustum from a perspective projection camera.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Frustum {
                SCE_GEOMETRY_INLINE Frustum(
                    sce::Geometry::Math::floatInVec_arg fovyRadians,
                    sce::Geometry::Math::floatInVec_arg aspect,
                    sce::Geometry::Math::floatInVec_arg zNear,
                    sce::Geometry::Math::floatInVec_arg zFar
                );
            }
        }
    }
}
```

Arguments

fovyRadians	The vertical field of view (radians).
aspect	The horizontal:vertical aspect ratio.
zNear	The minimum z value.
zFar	The maximum z value.

Return Values

None

Description

Constructs a 3D frustum from a perspective projection camera.

Notes

This function will assert if *zNear* is not less than *zFar*.

Frustum

Constructs a 3D frustum from axis limits.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Frustum {
                SCE_GEOMETRY_INLINE Frustum(
                    sce::Geometry::Math::floatInVec_arg zFar,
                    sce::Geometry::Math::floatInVec_arg zNear,
                    sce::Geometry::Math::floatInVec_arg xLeft,
                    sce::Geometry::Math::floatInVec_arg xRight,
                    sce::Geometry::Math::floatInVec_arg yBottom,
                    sce::Geometry::Math::floatInVec_arg yTop
                );
            }
        }
    }
}
```

Arguments

zFar	The maximum z value.
zNear	The minimum z value.
xLeft	The minimum x value.
xRight	The maximum x value.
yBottom	The minimum y value.
yTop	The maximum y value.

Return Values

None

Description

Constructs a 3D frustum from axis limits.

Frustum

Constructs a 3D frustum from the bounding planes.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Frustum {
                SCE_GEOMETRY_INLINE Frustum(
                    Plane_arg plnFar,
                    Plane_arg plnNear,
                    Plane_arg plnLeft,
                    Plane_arg plnRight,
                    Plane_arg plnBottom,
                    Plane_arg plnTop
                );
            }
        }
    }
}
```

Arguments

plnFar	The far bounding frustum plane.
plnNear	The near bounding frustum plane.
plnLeft	The left bounding frustum plane.
plnRight	The right bounding frustum plane.
plnBottom	The bottom bounding frustum plane.
plnTop	The top bounding frustum plane.

Return Values

None

Description

Constructs a 3D frustum from the bounding planes.

Public Instance Methods

getPlane

Gets the specified bounding plane of the 3D frustum.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Frustum {
                SCE_GEOMETRY_INLINE Plane getPlane(
                    eFrustumPlanes planeID
                ) const;
            }
        }
    }
}
```

Arguments

planeID	Identifies the bounding frustum plane.

Return Values

The specified bounding frustum plane.

Description

Gets the specified bounding plane of the 3D frustum.

setPlane

Sets the specified bounding plane of the 3D frustum.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Frustum {
                SCE_GEOMETRY_INLINE Frustum &setPlane(
                    Plane_arg plane,
                    eFrustumPlanes planeID
                );
            }
        }
    }
}
```

Arguments

plane	The bounding frustum plane.
planeID	Identifies the bounding frustum plane.

Return Values

A reference to the resulting 3D frustum.

Description

Sets the specified bounding plane of the 3D frustum.

sce::Geometry::Aos::Line

Summary

sce::Geometry::Aos::Line

A 3D line representation.

Definition

```
#include <sce_geometry.h>
class Line {};
```

Description

A 3D line represented by a point on the line and a normalized direction along it.

Methods Summary

Methods	Description
getDirection	Gets the normalized direction of the 3D line.
getOrigin	Gets the origin point that the 3D line passes through.
getPointOnLine	Gets a parametric point on the 3D line.
Line	The default constructor. This does no initialization.
Line	Constructs a 3D line from an origin and a direction.
Line	Constructs a 3D line from two points that exist on the line.
setDirection	Sets the normalized direction of the 3D line.
setOrigin	Sets the origin point that the 3D line passes through.

Constructors and Destructors

Line

The default constructor. This does no initialization.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Line {
                SCE_GEOMETRY_ALWAYS_INLINE Line();
            }
        }
    }
}
```

Arguments

None

Return Values

None

Description

The default constructor. This does no initialization.

Line

Constructs a 3D line from an origin and a direction.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Line {
                SCE_GEOMETRY_INLINE Line(
                    sce::Geometry::Math::Aos::Point3_arg origin,
                    sce::Geometry::Math::Aos::Vector3_arg direction
                );
            }
        }
    }
}
```

Arguments

origin	A point on the line.
direction	A normalized direction along the line.

Return Values

None

Description

Constructs a 3D line from an origin and a direction.

Notes

This function will assert if the direction is not normalized.

Line

Constructs a 3D line from two points that exist on the line.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Line {
                SCE_GEOMETRY_INLINE Line(
                    sce::Geometry::Math::Aos::Point3_arg pnt0,
                    sce::Geometry::Math::Aos::Point3_arg pnt1
                );
            }
        }
    }
}
```

Arguments

pnt0	The first point on the line.
pnt1	The second point on the line, which should be different from the first.

Return Values

None

Description

Constructs a 3D line from two points that exist on the line.

Notes

The result is undefined if the points are equal.

Public Instance Methods

getDirection

Gets the normalized direction of the 3D line.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Line {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Vector3
                getDirection() const;
            }
        }
    }
}
```

Arguments

None

Return Values

The normalized direction of the 3D line.

Description

Gets the normalized direction of the 3D line.

getOrigin

Gets the origin point that the 3D line passes through.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Line {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Point3
                getOrigin() const;
            }
        }
    }
}
```

Arguments

None

Return Values

The origin point that the 3D line passes through.

Description

Gets the origin point that the 3D line passes through.

getPointOnLine

Gets a parametric point on the 3D line.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Line {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Point3
                getPointOnLine(
                    sce::Geometry::Math::floatInVec_arg t
                ) const;
            }
        }
    }
}
```

Arguments

t	The parametric scalar value.

Return Values

A point on the line.

Description

Gets a parametric point on the 3D line.

setDirection

Sets the normalized direction of the 3D line.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Line {
                SCE_GEOMETRY_INLINE Line &setDirection(
                    sce::Geometry::Math::Aos::Vector3_arg direction
                );
            }
        }
    }
}
```

Arguments

direction	The normalized direction of the line.

Return Values

A reference to the resulting 3D line.

Description

Sets the normalized direction of the 3D line.

Notes

This function will assert if the direction is not normalized.

setOrigin

Sets the origin point that the 3D line passes through.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Line {
                SCE_GEOMETRY_INLINE Line &setOrigin(
                    sce::Geometry::Math::Aos::Point3_arg origin
                );
            }
        }
    }
}
```

Arguments

origin	A point on the line.

Return Values

A reference to the resulting 3D line.

Description

Sets the origin point that the 3D line passes through.

sce::Geometry::Aos::Obb

Summary

sce::Geometry::Aos::Obb

A 3D oriented bounding box representation.

Definition

```
#include <sce_geometry.h>
class Obb {};
```

Description

A 3D oriented bounding box represented by a center (origin) point/orientation as a transform and the half-widths.

Methods Summary

Methods	Description
getHalfWidths	Gets the half-widths of the 3D Obb.
getLocalToWorld	Gets the local-space center (origin) transform of the 3D Obb.
getOrigin	Gets the world-space center point (origin) of the 3D Obb.
getWorldToLocal	Gets the world-space center (origin) transform of the 3D Obb.
Obb	The default constructor. This does no initialization.
Obb	Constructs a 3D Obb from a transform and half-widths.
Obb	Constructs a 3D Obb from a transform and an Aabb.
Obb	Constructs a 3D Obb from a transform and an Obb.
Obb	Constructs a 3D Obb from bounds.
Obb	Constructs a 3D Obb from an Aabb.
setHalfWidths	Sets the half-widths of the 3D Obb.
setLocalToWorld	Sets the local-space center (origin) transform of the 3D Obb.
setOrigin	Sets the world-space center point (origin) of the 3D Obb.
setWorldToLocal	Sets the world-space center (origin) transform of the 3D Obb.

Constructors and Destructors

Obb

The default constructor. This does no initialization.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Obb {
                SCE_GEOMETRY_ALWAYS_INLINE Obb();
            }
        }
    }
}
```

Arguments

None

Return Values

None

Description

The default constructor. This does no initialization.

Obb

Constructs a 3D Obb from a transform and half-widths.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Obb {
                SCE_GEOMETRY_INLINE Obb(
                    sce::Geometry::Math::Aos::Transform3_arg trfm,
                    sce::Geometry::Math::Aos::Vector3_arg halfWidths
                );
            }
        }
    }
}
```

Arguments

trfm	The center (origin) local2World point/orientation of the 3D Obb.
halfWidths	The half-widths of the 3D Obb.

Return Values

None

Description

Constructs a 3D Obb from a transform and half-widths.

Notes

This function will assert if the transform axes are not orthogonal.

Obb

Constructs a 3D Obb from a transform and an Aabb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Obb {
                SCE_GEOMETRY_INLINE Obb(
                    sce::Geometry::Math::Aos::Transform3_arg trfm,
                    Aabb_arg aabb
                );
            }
        }
    }
}
```

Arguments

trfm	The transform to apply to the 3D Aabb.
aabb	The Aabb.

Return Values

None

Description

Constructs a 3D Obb from a transform and an Aabb.

Notes

This function will assert if the transform axes are not orthogonal.

Obb

Constructs a 3D Obb from a transform and an Obb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Obb {
                SCE_GEOMETRY_INLINE Obb(
                    sce::Geometry::Math::Aos::Transform3_arg trfm,
                    Obb_arg obb
                );
            }
        }
    }
}
```

Arguments

trfm	The transform to apply to the 3D Obb.
obb	The Obb.

Return Values

None

Description

Constructs a 3D Obb from a transform and an Obb.

Notes

This function will assert if the transform axes are not orthogonal.

Obb

Constructs a 3D Obb from bounds.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Obb {
                explicit SCE_GEOMETRY_INLINE Obb(
                    Bounds_arg bounds
                );
            }
        }
    }
}
```

Arguments

bounds	The bounds limits.

Return Values

None

Description

Constructs a 3D Obb from bounds.

Obb

Constructs a 3D Obb from an Aabb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Obb {
                explicit SCE_GEOMETRY_INLINE Obb(
                    Aabb_arg aabb
                );
            }
        }
    }
}
```

Arguments

aabb	The Aabb.

Return Values

None

Description

Constructs a 3D Obb from an Aabb.

Public Instance Methods

getHalfWidths

Gets the half-widths of the 3D Obb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Obb {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Vector3
                getHalfWidths() const;
            }
        }
    }
}
```

Arguments

None

Return Values

The half-widths of the 3D Obb.

Description

Gets the half-widths of the 3D Obb.

Document serial number: 000004892117

getLocalToWorld

Gets the local-space center (origin) transform of the 3D Obb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Obb {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Transform3
                getLocalToWorld() const;
            }
        }
    }
}
```

Arguments

None

Return Values

The local-space center (origin) transform of the 3D Obb.

Description

Gets the local-space center (origin) transform of the 3D Obb.

getOrigin

Gets the world-space center point (origin) of the 3D Obb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Obb {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Point3
                getOrigin() const;
            }
        }
    }
}
```

Arguments

None

Return Values

The world-space center point (origin) of the 3D Obb.

Description

Gets the world-space center point (origin) of the 3D Obb.

getWorldToLocal

Gets the world-space center (origin) transform of the 3D Obb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Obb {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Transform3
                getWorldToLocal() const;
            }
        }
    }
}
```

Arguments

None

Return Values

The world-space center (origin) transform of the 3D Obb.

Description

Gets the world-space center (origin) transform of the 3D Obb.

setHalfWidths

Sets the half-widths of the 3D Obb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Obb {
                SCE_GEOMETRY_INLINE Obb &setHalfWidths(
                    sce::Geometry::Math::Aos::Vector3_arg halfWidths
                );
            }
        }
    }
}
```

Arguments

halfWidths	The half-widths of the 3D Obb.

Return Values

A reference to the resulting 3D Obb.

Description

Sets the half-widths of the 3D Obb.

setLocalToWorld

Sets the local-space center (origin) transform of the 3D Obb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Obb {
                SCE_GEOMETRY_INLINE Obb &setLocalToWorld(
                    sce::Geometry::Math::Aos::Transform3 arg localToWorld
                );
            }
        }
    }
}
```

Arguments

localToWorld The Obb transform.

Return Values

A reference to the resulting 3D Obb.

Description

Sets the local-space center (origin) transform of the 3D Obb.

Notes

This function will assert if the transform axes are not orthogonal.

setOrigin

Sets the world-space center point (origin) of the 3D Obb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Obb {
                SCE_GEOMETRY_INLINE Obb &setOrigin(
                    sce::Geometry::Math::Aos::Point3_arg origin
                );
            }
        }
    }
}
```

Arguments

origin	The center point of the 3D Obb.

Return Values

A reference to the resulting 3D Obb.

Description

Sets the world-space center point (origin) of the 3D Obb.

setWorldToLocal

Sets the world-space center (origin) transform of the 3D Obb.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Obb {
                SCE_GEOMETRY_INLINE Obb &setWorldToLocal(
                    sce::Geometry::Math::Aos::Transform3 arg worldToLocal
                );
            }
        }
    }
}
```

Arguments

worldToLocal	The Obb transform.

Return Values

A reference to the resulting 3D Obb.

Description

Sets the world-space center (origin) transform of the 3D Obb.

Notes

This function will assert if the transform axes are not orthogonal.

sce::Geometry::Aos::Plane

Summary

sce::Geometry::Aos::Plane

A 3D plane representation.

Definition

```
#include <sce_geometry.h>
class Plane {};
```

Description

A 3D plane represented by the plane normal vector and the plane dot-product scalar value.

Methods Summary

Methods	Description
getNormal	Gets the normal of the 3D plane.
getScalar	Gets the scalar of the 3D plane.
Plane	The default constructor. This does no initialization.
Plane	Constructs a 3D plane from a Vector4.
Plane	Constructs a 3D plane from the normal and a point on the plane.
Plane	Constructs a 3D plane from the normal and scalar.
Plane	Constructs a plane from three points that exist on the plane.
setNormal	Sets the normal of the 3D plane.
setScalar	Sets the scalar of the 3D plane.

Constructors and Destructors

Plane

The default constructor. This does no initialization.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Plane {
                SCE_GEOMETRY_ALWAYS_INLINE Plane();
            }
        }
    }
}
```

Arguments

None

Return Values

None

Description

The default constructor. This does no initialization.

Plane

Constructs a 3D plane from a Vector4.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Plane {
                explicit SCE_GEOMETRY_INLINE Plane(
                    sce::Geometry::Math::Aos::Vector4_arg plane
                );
            }
        }
    }
}
```

Arguments

plane	The vector containing the XYZ normal and W scalar.

Return Values

None

Description

Constructs a 3D plane from a Vector4.

Notes

This function will assert if the XYZ component is not normalized.

Plane

Constructs a 3D plane from the normal and a point on the plane.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Plane {
                SCE_GEOMETRY_INLINE Plane(
                    sce::Geometry::Math::Aos::Point3_arg pnt,
                    sce::Geometry::Math::Aos::Vector3_arg normal
                );
            }
        }
    }
}
```

Arguments

pnt	A point on the plane.
normal	A plane normal vector.

Return Values

None

Description

Constructs a 3D plane from the normal and a point on the plane.

Notes

This function will assert if the normal is not normalized.

Plane

Constructs a 3D plane from the normal and scalar.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Plane {
                SCE_GEOMETRY_INLINE Plane(
                    sce::Geometry::Math::Aos::Vector3_arg normal,
                    sce::Geometry::Math::floatInVec_arg scalar
                );
            }
        }
    }
}
```

Arguments

normal	A plane normal vector.
scalar	A plane scalar value.

Return Values

None

Description

Constructs a 3D plane from the normal and scalar.

Notes

Will assert if normal is not normalized

Plane

Constructs a plane from three points that exist on the plane.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Plane {
                SCE_GEOMETRY_INLINE Plane(
                    sce::Geometry::Math::Aos::Point3_arg pnt0,
                    sce::Geometry::Math::Aos::Point3_arg pnt1,
                    sce::Geometry::Math::Aos::Point3_arg pnt2
                );
            }
        }
    }
}
```

Arguments

pnt0	The first point on the plane.
pnt1	The second point on the plane.
pnt2	The third point on the plane.

Return Values

None

Description

Constructs a plane from three points that exist on the plane.

Notes

The result is undefined if the points are collinear or any are equal.

Public Instance Methods

getNormal

Gets the normal of the 3D plane.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Plane {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Vector3
                getNormal() const;
            }
        }
    }
}
```

Arguments

None

Return Values

The normal of the 3D plane.

Description

Gets the normal of the 3D plane.

getScalar

Gets the scalar of the 3D plane.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Plane {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::floatInVec getScalar()
                const;
            }
        }
    }
}
```

Arguments

None

Return Values

The scalar of the 3D plane.

Description

Gets the scalar of the 3D plane.

Document serial number: 000004892117

setNormal

Sets the normal of the 3D plane.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Plane {
                SCE_GEOMETRY_INLINE Plane &setNormal(
                    sce::Geometry::Math::Aos::Vector3_arg normal
                );
            }
        }
    }
}
```

Arguments

normal	The plane normal.

Return Values

A reference to the resulting 3D plane.

Description

Sets the normal of the 3D plane.

Notes

This function will assert if the normal is not normalized.

setScalar

Sets the scalar of the 3D plane.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Plane {
                SCE_GEOMETRY_INLINE Plane &setScalar(
                    sce::Geometry::Math::floatInVec_arg scalar
                );
            }
        }
    }
}
```

Arguments

scalar	The plane scalar.

Return Values

A reference to the resulting 3D plane.

Description

Sets the scalar of the 3D plane.

Document serial number: 000004892117

sce::Geometry::Aos::Ray

Summary

sce::Geometry::Aos::Ray

A 3D ray representation.

Definition

```
#include <sce_geometry.h>
class Ray {};
```

Description

A 3D ray represented by the starting point (origin) of the ray and a normalized positive direction along it.

Methods Summary

Methods	Description
getDirection	Gets the normalized direction of the 3D ray.
getOrigin	Gets the origin point that the 3D ray starts from.
getPointOnRay	Gets a parametric point on the 3D ray.
Ray	The default constructor. This does no initialization.
Ray	Constructs a 3D ray from an origin and a direction.
Ray	Constructs a 3D ray from two points that exist on the ray.
setDirection	Sets the normalized direction of the 3D ray.
setOrigin	Sets the origin point that the 3D ray starts from.

Constructors and Destructors

Ray

The default constructor. This does no initialization.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Ray {
                SCE_GEOMETRY_ALWAYS_INLINE Ray();
            }
        }
    }
}
```

Arguments

None

Return Values

None

Description

The default constructor. This does no initialization.

Ray

Constructs a 3D ray from an origin and a direction.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Ray {
                SCE_GEOMETRY_INLINE Ray(
                    sce::Geometry::Math::Aos::Point3_arg origin,
                    sce::Geometry::Math::Aos::Vector3_arg direction
                );
            }
        }
    }
}
```

Arguments

origin	The starting point on the ray.
direction	The normalized direction along the ray.

Return Values

None

Description

Constructs a 3D ray from an origin and a direction.

Notes

This function will assert if the direction is not normalized.

Ray

Constructs a 3D ray from two points that exist on the ray.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Ray {
                SCE_GEOMETRY_INLINE Ray(
                    sce::Geometry::Math::Aos::Point3_arg pnt0,
                    sce::Geometry::Math::Aos::Point3_arg pnt1
                );
            }
        }
    }
}
```

Arguments

pnt0	The starting point of the ray.
pnt1	The second point on the ray, which should be different from the first.

Return Values

None

Description

Constructs a 3D ray from two points that exist on the ray.

Notes

The result is undefined if the points are equal.

Public Instance Methods

getDirection

Gets the normalized direction of the 3D ray.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Ray {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Vector3
                getDirection() const;
            }
        }
    }
}
```

Arguments

None

Return Values

The normalized direction of the 3D ray.

Description

Gets the normalized direction of the 3D ray.

getOrigin

Gets the origin point that the 3D ray starts from.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Ray {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Point3
                getOrigin() const;
            }
        }
    }
}
```

Arguments

None

Return Values

The origin point that the 3D ray starts from.

Description

Gets the origin point that the 3D ray starts from.

getPointOnRay

Gets a parametric point on the 3D ray.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Ray {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Point3
                getPointOnRay(
                    sce::Geometry::Math::floatInVec_arg t
                ) const;
            }
        }
    }
}
```

Arguments

t	The parametric scalar value.

Return Values

A point on the ray.

Description

Gets a parametric point on the 3D ray.

Notes

This function will assert if parameter *t* is less than zero.

setDirection

Sets the normalized direction of the 3D ray.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Ray {
                SCE_GEOMETRY_INLINE Ray &setDirection(
                    sce::Geometry::Math::Aos::Vector3_arg direction
                );
            }
        }
    }
}
```

Arguments

direction	The normalized direction of the ray.

Return Values

A reference to the resulting 3D ray.

Description

Sets the normalized direction of the 3D ray.

Notes

This function will assert if the direction is not normalized.

setOrigin

Sets the origin point that the 3D ray starts from.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Ray {
                SCE_GEOMETRY_INLINE Ray &setOrigin(
                    sce::Geometry::Math::Aos::Point3_arg origin
                );
            }
        }
    }
}
```

Arguments

origin	The starting point on the ray.

Return Values

A reference to the resulting 3D line.

Description

Sets the origin point that the 3D ray starts from.

sce::Geometry::Aos::Segment

Summary

sce::Geometry::Aos::Segment

A 3D line segment representation.

Definition

```
#include <sce_geometry.h>
class Segment {};
```

Description

A 3D line segment represented by the starting point (origin) of the segment and a span to the end point.

Methods Summary

Methods	Description
getOrigin	Gets the origin point at one end of the 3D segment.
getPointOnSegment	Gets a parametric point on the 3D segment.
getSpan	Gets the span from the origin to the other end of the 3D segment.
Segment	The default constructor. This does no initialization.
Segment	Constructs a 3D segment from a point at one end of the segment and a span to the other end.
Segment	Constructs a 3D segment from the two points that exist at either end of the segment.
setOrigin	Sets the origin point at one end of the 3D segment.
setSpan	Sets the span from the origin to the other end of the 3D segment.

Constructors and Destructors

Segment

The default constructor. This does no initialization.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Segment {
                SCE_GEOMETRY_ALWAYS_INLINE Segment();
            }
        }
    }
}
```

Arguments

None

Return Values

None

Description

The default constructor. This does no initialization.

Document serial number: 000004892117

Segment

Constructs a 3D segment from a point at one end of the segment and a span to the other end.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Segment {
                SCE_GEOMETRY_INLINE Segment(
                    sce::Geometry::Math::Aos::Point3_arg origin,
                    sce::Geometry::Math::Aos::Vector3_arg span
                );
            }
        }
    }
}
```

Arguments

origin	A point at one end of the segment.
span	The span to the other end of the segment.

Return Values

None

Description

Constructs a 3D segment from a point at one end of the segment and a span to the other end.

Notes

The result is undefined if the *span* parameter has zero length.

Segment

Constructs a 3D segment from the two points that exist at either end of the segment.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Segment {
                SCE_GEOMETRY_INLINE Segment(
                    sce::Geometry::Math::Aos::Point3_arg pnt0,
                    sce::Geometry::Math::Aos::Point3_arg pnt1
                );
            }
        }
    }
}
```

Arguments

pnt0	The first point (origin) on the segment.
pnt1	The second point on the segment, which should be different from the first.

Return Values

None

Description

Constructs a 3D segment from the two points that exist at either end of the segment.

Notes

The result is undefined if the points are equal.

Public Instance Methods

getOrigin

Gets the origin point at one end of the 3D segment.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Segment {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Point3
                getOrigin() const;
            }
        }
    }
}
```

Arguments

None

Return Values

The origin point at one end of the 3D segment.

Description

Gets the origin point at one end of the 3D segment.

getPointOnSegment

Gets a parametric point on the 3D segment.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Segment {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Point3
                getPointOnSegment(
                    sce::Geometry::Math::floatInVec_arg t
                ) const;
            }
        }
    }
}
```

Arguments

t	The parametric scalar value.

Return Values

A point on the segment.

Description

Gets a parametric point on the 3D segment.

Notes

This function will assert if parameter *t* is less than zero or greater than one.

getSpan

Gets the span from the origin to the other end of the 3D segment.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Segment {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Vector3 getSpan()
                const;
            }
        }
    }
}
```

Arguments

None

Return Values

The span from the origin to the other end of the 3D segment.

Description

Gets the span from the origin to the other end of the 3D segment.

setOrigin

Sets the origin point at one end of the 3D segment.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Segment {
                SCE_GEOMETRY_INLINE Segment &setOrigin(
                    sce::Geometry::Math::Aos::Point3_arg pnt
                );
            }
        }
    }
}
```

Arguments

pnt	A point at one end of the segment.

Return Values

A reference to the resulting 3D segment.

Description

Sets the origin point at one end of the 3D segment.

setSpan

Sets the span from the origin to the other end of the 3D segment.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Segment {
                SCE_GEOMETRY_INLINE Segment &setSpan(
                    sce::Geometry::Math::Aos::Vector3_arg span
                );
            }
        }
    }
}
```

Arguments

span	The span from the origin to the other end of the 3D segment.

Return Values

A reference to the resulting 3D segment.

Description

Sets the span from the origin to the other end of the 3D segment.

sce::Geometry::Aos::Sphere

Summary

sce::Geometry::Aos::Sphere

A 3D sphere representation.

Definition

```
#include <sce_geometry.h>
class Sphere {};
```

Description

A 3D sphere represented by the center point (origin) and the radius.

Methods Summary

Methods	Description
getOrigin	Gets the origin (center) of the 3D sphere.
getRadius	Gets the radius of the 3D sphere.
setOrigin	Sets the origin (center) of the 3D sphere.
setRadius	Sets the radius of the 3D sphere.
Sphere	The default constructor. This does no initialization.
Sphere	Constructs a 3D sphere from bounds.
Sphere	Constructs a 3D sphere from the origin and radius.

Constructors and Destructors

Sphere

The default constructor. This does no initialization.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Sphere {
                SCE_GEOMETRY_ALWAYS_INLINE Sphere();
            }
        }
    }
}
```

Arguments

None

Return Values

None

Description

The default constructor. This does no initialization.

Sphere

Constructs a 3D sphere from bounds.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Sphere {
                explicit SCE_GEOMETRY_INLINE Sphere(
                    Bounds_arg bounds
                );
            }
        }
    }
}
```

Arguments

bounds	The bounds limits.

Return Values

None

Description

Constructs a 3D sphere from bounds.

Document serial number: 000004892117

Sphere

Constructs a 3D sphere from the origin and radius.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Sphere {
                SCE_GEOMETRY_INLINE Sphere(
                    sce::Geometry::Math::Aos::Point3_arg origin,
                    sce::Geometry::Math::floatInVec_arg radius
                );
            }
        }
    }
}
```

Arguments

origin	The sphere origin.
radius	The sphere radius.

Return Values

None

Description

Constructs a 3D sphere from the origin and radius.

Notes

This function will assert if the radius has a negative value.

Public Instance Methods

getOrigin

Gets the origin (center) of the 3D sphere.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Sphere {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::Aos::Point3
                getOrigin() const;
            }
        }
    }
}
```

Arguments

None

Return Values

The origin of the 3D sphere.

Description

Gets the origin (center) of the 3D sphere.

getRadius

Gets the radius of the 3D sphere.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Sphere {
                SCE_GEOMETRY_INLINE sce::Geometry::Math::floatInVec getRadius()
                const;
            }
        }
    }
}
```

Arguments

None

Return Values

The radius of the 3D sphere.

Description

Gets the radius of the 3D sphere.

setOrigin

Sets the origin (center) of the 3D sphere.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Sphere {
                SCE_GEOMETRY_INLINE Sphere &setOrigin(
                    sce::Geometry::Math::Aos::Point3_arg origin
                );
            }
        }
    }
}
```

Arguments

origin	The sphere origin.

Return Values

A reference to the resulting 3D sphere.

Description

Sets the origin (center) of the 3D sphere.

setRadius

Sets the radius of the 3D sphere.

Definition

```
#include <sce_geometry.h>
namespace sce {
    namespace Geometry {
        namespace Aos {
            class Sphere {
                SCE_GEOMETRY_INLINE Sphere &setRadius(
                    sce::Geometry::Math::floatInVec_arg radius
                );
            }
        }
    }
}
```

Arguments

radius	The sphere radius.

Return Values

A reference to the resulting 3D sphere.

Description

Sets the radius of the 3D sphere.

Notes

This function will assert if the radius has a negative value.