LINE-OF-SIGHT BEHAVIOR SCRIPT INTERFACE

View online: https://www.construct.net/en/make-games/manuals/construct-3/scripting/scripting-reference/behavior-interfaces/line-of-sight

---

The `ILOSBehaviorInstance` interface derives from IBehaviorInstance to add APIs specific to the Line-of-sight behavior.

An additional `ILOSBehaviorRay` interface is returned by the `castRay()` method.

Examples

See the Scripting raycasting example for a demonstration of using the `castRay()` method to perform raycasting.

Line-of-sight behavior APIs

---

range

Set or get the maximum distance in pixels that line-of-sight can reach. If an object is further away than this distance, the object will never have line-of-sight to it, even if the intervening space is clear.

---

coneOfView

Set or get the angle in radians of the cone of view in which the object can have line-of-sight to other objects, relative to the current angle of the object.

---

addObstacle(iObjectClass)

If the *Obstacles* property is *Custom*, adds the given IObjectClass as another kind of object to count as an obstruction to line-of-sight.

> *Note that while this is a method for the instance, it affects the entire behavior.*

---

clearObstacles(iObjectClass)

If the *Obstacles* property is *Custom*, clears all obstacles added with the `addObstacle()` method.

> *Note that while this is a method for the instance, it affects the entire behavior.*

hasLOStoPosition(x, y)

Returns a boolean indicating if the object currently has line-of-sight to a position in layout co-ordinates, respecting the range and cone of view.

hasLOSBetweenPositions(fromX, fromY, fromAngle, toX, toY)

Returns a boolean indicating if there is line-of-sight between any two positions in the layout, instead of using the object's own position. This respects the range and cone of view, based on *fromAngle*, which is in radians.

castRay(fromX, fromY, toX, toY, useCollisionCells = true)

Check for obstacle intersection between any two positions in the layout, returning a `ILOSBehaviorRay` interface representing the result. Check the `didCollide` property of the returned interface to identify if an intersection was found. The other properties of the interface indicate the hit position, normal and reflection angle, if an intersection was found. For more information see the documentation of the `ILOSBehaviorRay` interface below. The *useCollisionCells* parameter specifies whether to use the collision cells optimisation when testing line of sight. Usually this is faster, but in some cases over extremely long distances it can be slower.

> *This method ignores the range and cone of view, to allow raycasting anywhere in the layout.*

ray

Returns the `ILOSBehaviorRay` interface representing the result of the last `castRay` method call.

ILOSBehaviorRay interface

This interface is used to represent the result of a call to `castRay()`. All its properties are read-only.

didCollide

Read-only boolean indicating whether an intersection was found.

> *The rest of the properties on this interface are only set if `didCollide` is `true`.*

hitX

hitY

getHitPosition()

If *didCollide* is true, the read-only position of the first obstacle the ray intersected, in layout co-ordinates. The method returns both values at the same time.

---

hitDistance

If *didCollide* is true, the read-only distance between the ray start point and the hit position.

---

hitUid

If *didCollide* is true, the read-only UID of the instance that was the first obstacle the ray intersected.

---

getNormalX(length)

getNormalY(length)

getNormal(length)

If *didCollide* is true, returns a position at a given distance along the surface normal vector. The `getNormal()` variant returns both values at the same time.

---

normalAngle

If *didCollide* is true, the read-only angle of the surface normal at the point of intersection, in radians.

---

getReflectionX(length)

getReflectionY(length)

getReflection(length)

If *didCollide* is true, returns a position at a given distance along the reflection vector. The `getReflection()` variant returns both values at the same time.

---

reflectionAngle

If *didCollide* is true, the read-only angle of the reflection at the point of intersection, in radians.