ncollide Reference Guide

Compilation and usage

Compilation

Usage

Geometric primitives

List of primitives

The Ball

Mathematically speaking, the `Ball` structure describes a closed ball on the *n*-dimensional euclidean space. In two dimensions, this is a disk, and in three dimensions, this is a sphere, centered at the origin.

Description	Accessors	Value
The radius of the ball	`b.radius()`	User-defined with `Ball::new`

2D and 3D example:

```
let ball = Ball::new(1.0);
assert!(ball.radius() == 1.0);
```

The Cuboid

The `Cuboid` structure describes a rectangle in two dimensions, or a cuboid in three dimensions. A cuboid is defined by its *half extents* − that is − its half length along each coordinate axis. For performance and accuracy reasons, each cuboid also has an *internal margin*. This means that the actual geometry used for collision detection is a cuboid with rounded corners with a radius equal to the margin.

Description	Accessors	Value
The half extents of the cuboid	`c.half_extents()`	User-defined by `Cuboid::new`
The internal margin of the cuboid	`c.margin()`	`0.04` or user-defined by `Cuboid::new_with_margin`

3D example:

```
let cuboid = Cuboid::new(Vec3::new(1.0, 2.0, 3.0));

assert!(cuboid.margin() == 0.04); // default margin.
```

2D example:

```
let cuboid = Cuboid::new_with_margin(Vec2::new(1.0, 2.0), 0.2);
assert!(cuboid.margin() == 0.2); // user-defined margin
```

The Cylinder

The `Cylinder` structure describes a rectangle in two dimensions (use `Cuboid` instead), or a cylinder in three dimensions. The principal axis is the positive `y` axis.

Description	Accessors	Value
The half height of the cylinder	`c.half_height()`	User-defined by `Cylinder::new`
The radius of the cylinder basis	`c.radius()`	User-defined by `Cylinder::new`
The margin of the cylinder	`c.margin()`	`0.04` or user-defined by `Cylinder::new_with_margin`

3D example:

```
let cylinder1 = Cylinder::new(1.0, 0.5);
let cylinder2 = Cylinder::new_with_margin(1.0, 0.5, 0.1);

assert!(cylinder1.margin() == 0.04); // default margin
assert!(cylinder2.margin() == 0.1);  // user-defined margin
```

The Cone

The `Cone` structure describes an isosceles triangle in two dimensions, or a cone of revolution in tree dimensions. A cone is defined by the *radius* of its basis and its *half height* − the half distance between the basis and the apex. The principal axis is the positive `y` axis.

Description	Accessors	Value
The half height of the cone	`c.half_height()`	User-defined by `Cone::new`
The radius of the cone basis	`c.radius()`	User-defined by `Cone::new`
The margin of the cone	`c.margin()`	`0.04` or user-defined by `Cone::new_with_margin`

3D example:

```
let cone1 = Cone::new(1.0, 0.5);
let cone2 = Cone::new_with_margin(1.0, 0.5, 0.1);

assert!(cone1.margin() == 0.04); // default margin
assert!(con2.margin() == 0.1);   // user-defined margin
```

The Capsule

The `Capsule` structure describes the minkowski sum of a segment and a ball. In other words, this is a cylinder with its flat extremities replaced by balls. A capsule is defined by its *half height* and the *radius* of its extremities. The principal axis is the positive `y` axis.

Description	Accessors	Value
The half height of the capsule	`c.half_height()`	User-defined by `Capsule::new`
The radius of the capsule extremities	`c.radius()`	User-defined by `Capsule::new`

2D and 3D example:

```
let capsule = Capsule::new(1.0, 0.5);

assert!(capsule.half_height() == 1.0);
assert!(capsule.radius() == 0.5);
```

The Plane

The `Plane` structure describes a solid closed half-space. A plane is defined by its *normal*. Every point that has a negative or zero dot product with the plane normal is considered *inside* of the plane. Other points are considered *outside* of the plane.

Description	Accessors	Value
The normal of the plane	`p.normal()`	User-defined by `Plane::new`

2D and 3D example:

```
let plane = Plane::new(Vec2::new(1.0, 0.0));

assert!(plane.normal().x == 1.0);
assert!(plane.normal().y == 0.0);
```

The Mesh

The Compound

Defining your own primitive

Collision detection

Narrow phase

Broad phase

Time of Impact

Procedural generation

Primitives

Paths

Ray casting

Bounding volumes

AABB

Bounding Sphere

Traits

Q&A

Table des matières

Introduction 1
Compilation and usage 1
- Compilation 1
- Usage 1

Geometric primitives 1
- List of primitives 6
- Defining your own primitive 6

Collision detection 6
- Narrow phase 6
- Broad phase 6
- Time of Impact 6

Procedural generation 6
- Primitives 6
- Paths 6

Ray casting 6
Bounding volumes 6
- AABB 6
- Bounding Sphere 6

Traits 6
Q&A 6