PHYSICS

View online: https://www.construct.net/en/make-games/manuals/construct-3/behavior-reference/physics

The **Physics behavior** simulates realistic object physics. It is powered by Box2D. Construct comes with several examples demonstrating what the Physics behavior can do; filter by the *Physics* behavior in the Example Browser to find them.

The Physics behavior is relatively complex. The following tutorials are recommended to gain a basic understanding of how to use the Physics behavior and some important points to know before beginning to use it:

- Physics in Construct: The basics
- Physics in Construct: Forces, impulses, torque and joints

This manual section will not repeat the information in these tutorials. Instead it will describe each feature of the Physics behavior. The tutorials describe how Physics engines work, what the different types of joints are, the difference between impulses and forces, and so on in case you're not already familiar with them.

Scripting

When using JavaScript or TypeScript coding, the features of this behavior can be accessed via the IPhysicsBehaviorInstance script interface.

Using Physics in Construct

The Physics behavior simulates physics separately to the Construct layout. Construct will try to keep the Physics and Construct "worlds" synchronised if one changes but not the other, but this can be unpredictable. For example, setting an object's position or angle will cause Construct to teleport the corresponding object in the physics simulation to the object's new position, which does not always properly take in to account collisions. The same is true of using other Construct behaviors at the same time as Physics.

Therefore it is highly recommended to control Physics objects entirely via the Physics behavior (by setting forces, impulses, torques etc.), rather than trying to manipulate objects by *Set position*, *Set angle* etc.

Another consequence is Physics won't respond to objects with the *Solid* or *Jumpthru* behaviors. These behaviors are totally redundant when using Physics and have no effect. Instead, use the *Immovable* property.

Physics properties

Immovable

If enabled, simulate the object having infinite mass. Its density is ignored and it will never move.

Collision mask

How to handle physics collisions for this object. The options are:

- **Use collision polygon** uses the object's collision polygon from the Animations editor for physics collisions. If it doesn't have a collision polygon it will use the object's bounding box.
- **Bounding box** ignores the object's collision polygon if any, and for the purposes of Physics collisions considers the object to be a rectangle.
- **Circle** ignores the object's collision polygon if any, and for the purposes of Physics collisions considers the object to be a circle. This allows objects to smoothly roll along (like for example barrels). This is especially useful since object's collision polygons is made out of straight lines, so a smooth circle cannot be created that way.

Collision filter tags

Filter collisions with other objects with the Physics behavior by specifying their instance tags. The way these tags are used depends on the *Collision filter mode* property.

Collision filter mode

Specifies how the collision filter tags are checked against instance tags. In *Inclusive* mode, collisions are only enabled with objects that match any of the given tags; if no collision filter tags are specified, then all collisions are disabled. In *Exclusive* mode, collisions are disabled with objects that match any of the given tags; if no collision filter tags are specified, then all collisions are enabled (the default).

Prevent rotation

Prevents the object from rotating when colliding with other Physics objects. Note this does not completely prevent the object from rotating - it only stops rotation caused by Physics collisions.

Density

The density of the physics object. Only used if *Immovable* is disabled. The object mass is calculated as its density multiplied by the area of its collision mask. The exact density values used are not important and have no specific units - only the relative density is significant (i.e. an object with density 6 will be twice as dense as an object with density 3).

Friction

The friction coefficient of the physics object from 0 (no friction) to 1 (maximum friction). This adjusts how easily objects move against each other while touching.

Elasticity

The elasticity (also known as *restitution* or *bounciness*) of the physics object, from 0 (inelastic, like a rock) to 1 (maximum elasticity, like a rubber ball). This affects how high objects bounce when hitting the floor.

Linear damping

The rate the object slows down over time while moving, from 0 (no slowdown at all) to 1 (maximum slowdown).

Angular damping

The rate the object slows down over time while spinning, from 0 (no slowdown at all) to 1 (maximum slowdown).

Bullet

Enable enhanced collision detection for fast-moving objects. This can affect performance, so do not enable it unless the object moves so fast the physics engine's standard collision detection is unreliable.

Enabled

Whether the physics simulation is initially enabled or disabled. If disabled, no physics is processed for the object, and other physics objects can pass through the object as if it were empty space.

Physics conditions

Compare angular velocity

Compare the current angular velocity of the physics body, in degrees per second. A positive value indicates clockwise rotation and a negative value indicates anticlockwise rotation.

Compare mass

Compare the mass of the physics body. This is determined by multiplying the *Density* by the area of the object's collision polygon.

Compare velocity

Compare the current velocity (speed) of the physics body, in pixels per second. The velocity can be compared on an individual axis, such as just the X axis to compare the horizontal motion, or the overall velocity can be used.

Is enabled

True if the physics behavior is currently enabled. When disabled the physics body is completely removed from the physics simulation, so other physics objects will pass through the object.

Is immovable

True if the physics behavior is currently set to immovable (having an infinite mass and so never moving).

Is sleeping

True if the object has been at rest and not moved or been disturbed for a while, so that the physics engine can stop processing it. Note objects can still be moving imperceptibly which can prevent them from being asleep even when they appear to be stopped.

Physics actions

Forces

Apply force

Apply force at angle

Apply force towards position

Apply a force on the object, either at an angle, towards a position, or with custom X and Y axis forces. Applying a force causes the object to accelerate in the direction of the force. Forces can be applied at an image point, which normally also causes the object to rotate. Using `0` for the image point uses the object's center of mass, which does not cause rotation. Use `-1` to use the object's origin, which may be different to the center of mass and cause rotation.

Apply impulse

Apply impulse at angle

Apply impulse towards position

Apply an impulse on the object, either at an angle, towards a position, or with custom X and Y axis impulses. Applying an impulse simulates the object being struck, e.g. hit by a bat. Impulses can be applied at an image point, which normally also causes the object to rotate. Using `0` for the image point uses the object's center of mass, which does not cause

rotation. Use `-1` to use the object's origin, which may be different to the center of mass and cause rotation.

Set velocity

Set the object's current velocity directly, providing a speed in pixels per second for the X and Y axes.

Teleport

Set the object position preserving the Physics velocity. Normally changing the position of the object will reposition it, but alter the velocity to try to ensure the physics simulation remains realistic even though some external change was made to the object position. Using the *Teleport* action will reposition the object but not alter its Physics velocity in any way, which is sometimes desirable for purposes such as if a Physics object goes through a portal and is meant to appear somewhere else but with the same velocity.

Global settings

These actions affect Physics behaviors in general, not just the one it was set for.

Enable/disable collisions

By default, all Physics objects collide with each other. You can disable collisions between the object and another Physics object so they pass through each other. This affects all instances of both object types. Note: enabling collisions again when objects are overlapping can cause instability in the simulation.

Set stepping iterations

Set the number of velocity iterations and position iterations used in the physics engine. The default is 8 and 3 respectively. Lower values run faster but are less accurate, and higher values can reduce performance but provide a more realistic simulation.

Set stepping mode

Choose whether the Physics time step uses *dt* (delta time, for *Framerate independent*) or a *fixed* value. By default it uses delta time to ensure physics progresses at a regular speed on displays with different refresh rates. However in some circumstances this can produce non-deterministic results, and note the Physics behavior clamps the maximum time step to 1/30 (about 33ms, equivalent to 30 FPS) to prevent the instability that can result from large time steps. Set to *Fixed* to always use a time step of 1/60 (about 16ms) regardless of the framerate, which makes the simulation deterministic, but will cause it to run at different speeds on displays with different refresh rates.

Set world gravity

Set the force of gravity affecting all Physics objects. By default this is a force of 10 downwards.

Joints

Create distance joint

Fix two physics objects at a given distance apart, as if connected by a pole. An image point can be specified to connect to a specific part of the object. Note that an image point of 0 specifies the center of gravity of the object - if you intend to connect to the object origin, use -1.

Create revolute joint

Create limited revolute joint

Hinge two physics objects together, so they can rotate freely as if connected by a pin. Limited revolute joints only allow rotation through a certain range of angles, like the clapper of a bell. An image point can also be specified to connect to a specific part of the object. Note that an image point of 0 specifies the center of gravity of the object - if you intend to connect to the object origin, use -1.

Create prismatic joint

Restrict the movement of two physics objects along a specific axis, given by its angle. *Enable limit* specifies whether there is a lower and upper movement limit; if enabled these are given by the lower and upper translation, otherwise unlimited movement is allowed along the axis. A motor can also be enabled to provide a continuous force along the axis.

Remove all joints

Remove all joints from the object. Any objects this object was attached to via joints is also affected. Note some joints automatically disable collisions between the objects, so you may want to manually disable collisions again after removing joints otherwise overlapping objects will "teleport" apart (as the physics engine will try to prevent them overlapping).

Object settings

These set the corresponding properties. For more information, see *Physics properties*. The one exception is the **Set awake** action.

Set awake

Physics simulations are relatively CPU intensive, requiring a lot of calculations. To save processor time, the Physics engine will make objects that have come to a complete stop go in to "sleep" mode so they no longer require processing. However sometimes changes like repositioning an adjacent object will leave the object in "sleep" mode so it will not respond

properly. In this situation the *Set awake* action can be used to force a sleeping object to resume simulation. (It can also be used to force an object to go in to "sleep" mode, but note this is normally done automatically when possible.)

Torque

Apply torque

Apply torque towards angle

Apply torque towards position

Apply a torque (rotational acceleration) to the object, either directly, or towards an angle or position.

Set angular velocity

Set the angular velocity (rate of rotation) directly, in degrees per second.

Physics expressions

AngularVelocity

The current angular velocity (rate of rotation) of the physics object, in degrees per second.

CenterOfMassX

CenterOfMassY

The position of the center of mass of the physics object, as calculated by the physics engine. This depends on the *collision mask* property, and is not necessarily in the middle of the object.

ContactCount

Return the number of locations the physics engine has identified this object as touching other physics objects.

ContactXAt(index)

ContactYAt(index)

Return the position of a contact with another physics object, in layout co-ordinates, given by the zero-based index of the contact.

Mass

The mass of the physics object, as calculated by the physics engine. This is the area of the object's collision mask multiplied by its density.

VelocityX

VelocityY

The current speed of the physics object, in pixels per second.

AngularDamping

Density

Elasticity

Friction

LinearDamping

These return the corresponding properties. For more information, see *Physics properties*.