

Handling collisions in your game

Most games need to detect and handle collisions between objects. Detecting and handling collisions can be done to make objects stick on platforms and detect collisions between bullets and players or bullets and other objects.

GDevelop provides several different ways to handle collisions. You can detect collisions using Event Editor conditions and actions, or you can use Object behaviors.

Make objects solids: use the "Separate objects" action (good for top-down games, RPG...)



Separate two objects

Common actions for all objects/Position

You can use the “**Separate objects**” action to move objects manually. The “Separate objects” action can also move objects with “forces” or set the object's position.

“**Separate objects.**” This action takes two object names as the argument.

- The first object name is the *object moving* (the player, enemy, etc.).
- The second object name is an *object (or a group of objects) that are solid*. For example, these objects can be the walls.

The action will then iterate over all of the objects given. It will ensure that if an object of the first kind is colliding with an object of the second kind, the object will be moved away. This is done using an algorithm called the *SAT algorithm*.

Add condition

This action will be launched in every frame. In an event without conditions, the action is already doing the collision checks. Avoid launching this action multiple times. Doing so could reduce game performance.

Detect collisions

Using “Separate objects” is a good way to ensure that your objects can't move into other solid objects. This action checks collisions between objects. For example, if the game object “player” is touching a wall, this action, when used with the condition called “**Collision,**” will trigger damages to the player.

The sequence is important.

1. Run the condition called “**Collision.**”

2. Add appropriate actions.
3. Add the “Separate Objects” action.

After running the “Separate objects” action, objects are moved. *Collisions between objects will no longer be able to be checked.*

You can find usage of these conditions and actions in the examples:

See it in action! 🎮

Open this example online: [Bomb the Crate Example](#)



Platformer games: use the Platformer character and Platform behaviors

If you're making a platformer game, it's a good idea to use the "[Platformer character](#)" behavior. It's a ready-made platform game engine that is highly customizable. The “Platformer character” behavior handles the gravity and the collisions with the platforms.

Detect collisions in a platformer game?

In a platformer game with the “Platformer character” behavior, collisions with platforms are handled for you.

- You can still use the **Collision condition** to check for collisions between an object and other objects (for example, between the player and enemies) and react accordingly.
- You can use the “Is on Floor” condition to check if an object is on a platform.

PlayerHitBox is on floor	Add action
Add condition	
PlayerHitBox is moving	Do =0 to the number
Add condition	Add action
PlayerHitBox is moving	Do =2 to the number
Add condition	Add action

See it in action! 🎮

Open this example online: [Platformer Example](#)

Game with physics? Use the Physics behavior

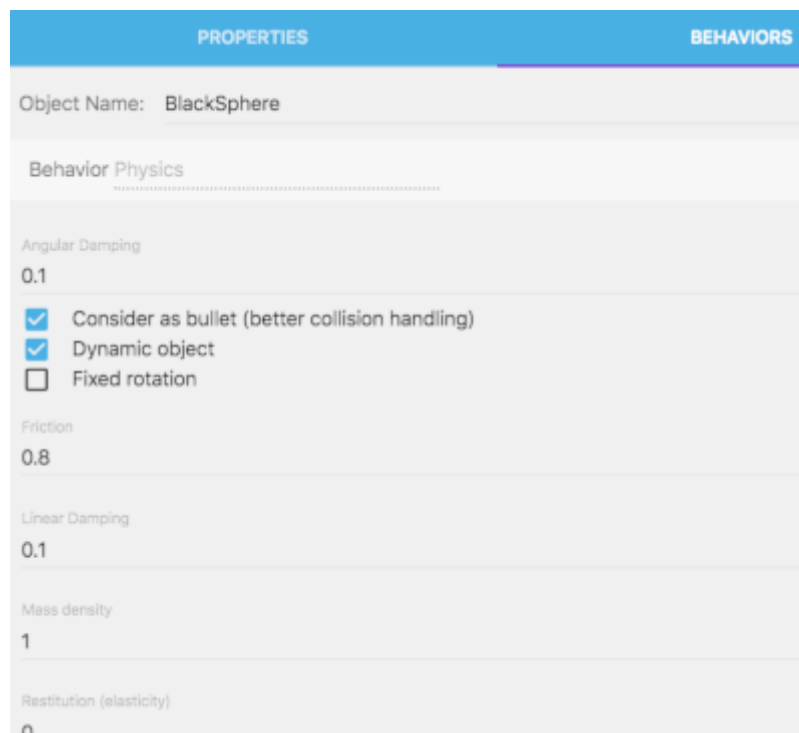
Use [Physics behavior](#) in order to achieve realistic physical behavior in your game. Attach “Physics” behavior to your objects. The objects will then behave as though they are alive in the game world. Some examples of real-world behavior include bouncing balls, falling, jumping, etc.

Configure game walls or solid objects that should not move with “static” behavior.

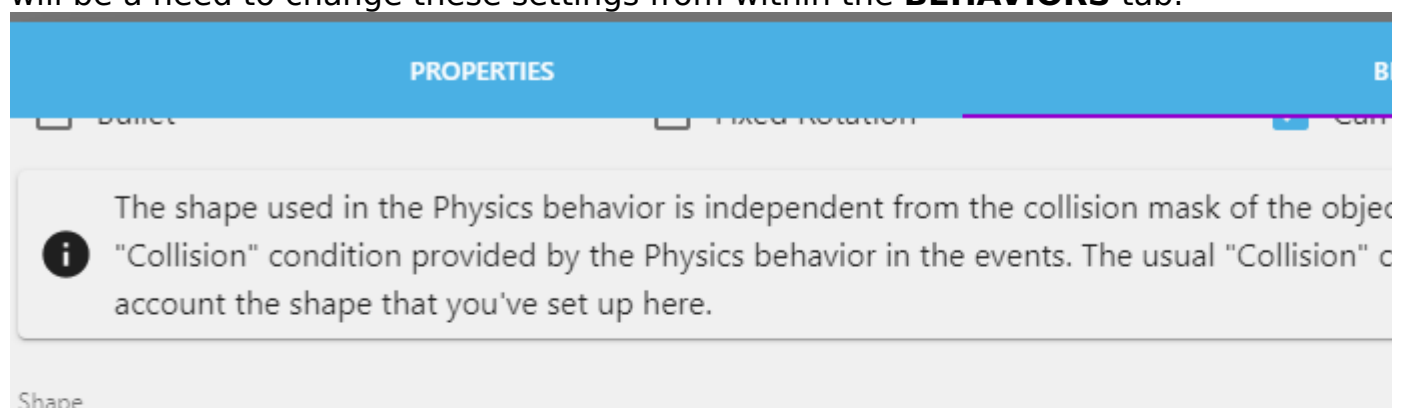
Detect collisions with the Physics behavior

When you're using the **Physics** behavior, *do not use* the **Collision** condition that is in the **Features for all objects** category. *The physics engine will manage all collisions by itself.* The Collision condition won't correctly detect when objects are touching.

Instead, use the Collision condition *inside the Physics behavior category*, which properly uses the physics engine to simulate the collisions.






Additionally, objects with the **Physics** behavior ignore the object's Collision Masks and instead use the collision information on the **BEHAVIORS** tab of the object itself. This defaults to a box that is the full dimension of the object. In most cases, there will be a need to change these settings from within the **BEHAVIORS** tab.




Box ▾


Width	Height	Shape Offset X	Shape Offset Y
0	0	0	0



Unable to load the image

 A temporary image to help you visualize the shape/polygon

Density	Gravity Scale
1	1
Friction	Restitution
0.3	0.1
Linear Damping	Angular Damping

 HELP

See it in action! 🎮Open this example online: [Physics Example](#)