

Introduction

ProjectStack □□□□



[CardTest](#)

[Game](#)

[Game.LoadingProgressChangedEventArgs](#)

[Launcher](#)

[Main](#)

[ResourceEditor](#)

Base class for all UI-related nodes. `Godot.Control` features a bounding rectangle that defines its extents, an anchor position relative to its parent control or the current viewport, and offsets relative to the anchor. The offsets update automatically when the node, any of its parents, or the screen size change.

For more information on Godot's UI system, anchors, offsets, and containers, see the related tutorials in the manual. To build flexible UIs, you'll need a mix of UI elements that inherit from `Godot.Control` and `Godot.Container` nodes.

User Interface nodes and input

Godot propagates input events via viewports. Each `Godot.Viewport` is responsible for propagating `Godot.InputEvents` to their child nodes. As the `Godot.SceneTree.Root` is a `Godot.Window`, this already happens automatically for all UI elements in your game.

Input events are propagated through the `Godot.SceneTree` from the root node to all child nodes by calling `Godot.Node._Input(Godot.InputEvent)`. For UI elements specifically, it makes more sense to override the virtual method `Godot.Control._GuiInput(Godot.InputEvent)`, which filters out unrelated input events, such as by checking z-order, `Godot.Control.MouseFilter`, focus, or if the event was inside of the control's bounding box.

Call `Godot.Control.AcceptEvent()` so no other node receives the event. Once you accept an input, it becomes handled so `Godot.Node._UnhandledInput(Godot.InputEvent)` will not process it.

Only one `Godot.Control` node can be in focus. Only the node in focus will receive events. To get the focus, call `Godot.Control.GrabFocus()`. `Godot.Control` nodes lose focus when another node grabs it, or if you hide the node in focus.

Sets `Godot.Control.MouseFilter` to `Godot.Control.MouseFilterEnum.Ignore` to tell a `Godot.Control` node to ignore mouse or touch events. You'll need it if you place an icon on top of a button.

`Godot.Theme` resources change the `Control`'s appearance. If you change the `Godot.Theme` on a `Godot.Control` node, it affects all of its children. To override some of the theme's parameters, call one of the `add_theme_*_override` methods, like `Godot.Control.AddThemeFontOverride(Godot.StringName, Godot.Font)`. You can override the theme with the Inspector.

Note: Theme items are *not* `Godot.GodotObject` properties. This means you can't access their values using `Godot.GodotObject.Get(Godot.StringName)` and `Godot.GodotObject.Set(Godot.StringName, Godot.Variant)`. Instead, use the `get_theme_*` and `add_theme_*_override` methods provided by this class.

[ResourceEditor.TaskNotifier](#)

A wrapping class that can hold a [Task](#)[↗] value.

[ResourceEditor.TaskNotifier<T>](#)

A wrapping class that can hold a [Task<TResult>](#)[↗] value.