

the many ways of instantiating/duplicating/spawning a scene in Godot 4

@Kolja Sam, kolja.luemer.com, v1.0
Up to date for Godot 4.3, mistakes are mine :)

Original Scene



car.tscn



car.gd

```
class_name Car extends Node2D
@export var top_speed = 60
```

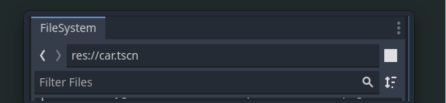
Enable Editable Children

- how?**
 - Right click your instantiated child scene and click `Editable Children`
- what will happen?**
 - You will now see all child nodes of your instantiated scene expanded in the scene tree
 - You can now edit those children's properties
 - Changes to the original scene will still be reflected
- when and why to use this?**
 - You want to keep the ability to have one parent object where you can make sweeping changes, but be able to make small changes on the children objects
- example:**
 - You decide that in your `offroad_race.tscn`, it would look better to replace the `texture` of the `Sprite` of the wheels of your car as well as change their size
- note:**
 - Beware: It is not always intuitive what you can and cannot edit with `Editable Children`: for example, you cannot delete any child of the original scene, but you can `hide` them
 - Beware: It is not always clear when a change in the original scene is going to silently overwrite changes you made in instantiated scenes with `Editable Children`

Make Local

- how?**
 - Right click your instantiated child scene and click `Make Local`
- what will happen?**
 - You will now see all child nodes of your instantiated scene expanded in the scene tree
 - You can now edit those children's properties
 - Changes to the original scene will *not* be reflected, this scene is now entirely independent of the original scene
- when and why to use this?**
 - You want to reuse some of the ideas of another scene once to make a completely new and independent scene
- example:**
 - You instantiate your `car.tscn` but decide in the end to go with a motorbike, for which you will steal some `Nodes` (e.g. the wheels) but which is overall completely different.
- note:**
 - Like `Duplicate Scene File`: When you use this to create a lot of similar scenes, you will have to manually change all of the scenes when you make a change that should affect all of them
 - If you "break" inheritance this way, it's probably a sign that some abstraction failed you and should be redesigned

Things you do in the `FileSystem` Dock



Things you do in a `.gd` Script

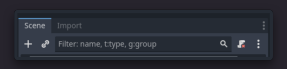
2D 3D Script AssetLib

Call `.instantiate()`

- how?**
 - requires your original scene to be loaded as a `PackedScene`, on which you call `.instantiate()`
 - for example:

```
1. const CAR := preload("res://car.tscn")
2. var car_inst := CAR.instantiate()
3. add_child(car_inst)
```
- what will happen?**
 - the same as in `Instantiate Child Scene`, only you are doing this in a script, so this will usually happen while your game is running
- when and why to use this?**
 - Whenever you want to spawn `Nodes` dynamically, for example as reaction to player input, on a timer, at the beginning of a level, etc.
- example:**
 - You make a `busy_highway.tscn` where you attach `busy_highway.gd` which you use to spawn a new `car.tscn` every two seconds

Things you do in the `Scene` Dock



Instantiate Child Scene

- how?**
 - Right click a `Node` in your `Scene Tree` and select the `chain Icon` (or use `Ctrl+Shift+A`)
- what will happen?**
 - The complete original scene will appear as a single `Node` in your scene tree
 - There will be an icon to jump to the original scene
 - If you make changes to the original scene, they will be reflected
- when and why to use this?**
 - Whenever you want to use a scene you've made in another scene (this will be very often)
- example:**
 - You make a `garage.tscn` scene and you instantiate `car.tscn` in it, so there is a car in the garage

Add as Child Node

- how?**
 - Use one of the following options:
 - Select a `Node` in your scene tree and select the `+ Icon`
 - Select a `Node` in your scene tree and use `Ctrl+A`
 - Right-click the intended parent `Node` and select `Add Child Node`
 - Then, select your saved scene in the file explorer
- what will happen?**
 - You will get a `Node` of the type of your scene
 - This `Node` will have no children
 - If you attached a script, the added `Node` will also have this script
- when and why to use this?**
 - To add standard `Nodes` from Godot to your scene, like `Node2D`, `Sprite`, `Area2D`, `CenterContainer`, and so on
 - To add simple custom `Nodes` to your scene that either need no child `Nodes` or where you will manually add those child `Nodes`
- example:**
 - All the cars in your game are made in completely different ways, so you use a basic `Car` `Node` which only holds some values like `top_speed` and manually add children like `Sprite` and `CollisionShape2D` for every car you add
- note:**
 - You can only add your custom `Node` in this way if your original scene has a script with a `class_name`
 - Unless you're using Godot classes in an advanced way, you probably don't want to use this to instantiate things that you defined yourself

Call `.new()`

- how:**
 - In any script, call `var car := Car.new()`
 - You will likely also need to add the node to the scene tree with `add_child(car)`
- what will happen?**
 - The same as in `Add as Child Node`, only you are doing this in a script, so this will usually happen while your game is running
- when and why to use this?**
 - To create a basic, pre-defined `Node` in a script (like a `Timer`)
 - To dynamically create a basic, custom `Node` (or abstract `Class`) while your game is running
- example:**
 - You have a dynamic `car_generator` which first creates a simple `Car` node, then adds components to it
- note:**
 - You can only add your custom `Node` in this way if your original scene has a script with a `class_name`
 - Unless you're using Godot classes in an advanced way, you probably don't want to use this to instantiate things that you defined yourself

New Inherited Scene

- how?**
 - Right click your scene in the `FileSystem` dock and select `New Inherited Scene`
- what will happen?**
 - A new scene will open, which will look exactly like the original scene
 - All nodes of the original scene will also be in the new scene, and you can change their properties
 - If you save the scene, you will have another `.tscn` file
- when and why to use this?**
 - You want to build variations of your original scene, while having certain nodes and standard settings which can be edited from a single source (your original scene)
 - This is useful for building levels which share a lot of similar nodes (for example a camera and a player)
- example:**
 - You decide that you want a bunch of different looking car scenes that you can load in. Since all cars are going to have a body and four wheels, you make a couple inherited scenes from `car.tscn` and swap out the textures for body and wheels.
- note:**
 - Beware: Like with `Enable Editable Children`, it's not always clear what you can edit; for example, you cannot delete any child node of the original scene, but you can `hide` them
 - Beware: It is not always clear when a change in the original scene is going to silently overwrite changes you made in inherited scenes.

Duplicate Scene File

- how?**
 - Duplicate the scene file by right-clicking it and selecting `Duplicate`, or just press `Ctrl+D`
 - Duplicate it to a different location by right-clicking it, selecting `Move/Duplicate To...` and choosing a target
- what will happen?**
 - You get another scene with its own file with that is exactly the same as the original
 - Now you can change the scenes completely independently of each other
- when and why to use this?**
 - You want to reuse some of the ideas of another scene once to make a completely new and independent scene
- example:**
 - You duplicate `car.tscn` to make `bike.tscn`, which will have roughly the same nodes, but will go into a different part of your game and work completely differently
- note:**
 - If nodes have scripts attached, they will be shared with the original scene, unless you disconnect them and create new ones
 - When you use this to create a lot of similar scenes, you will have to manually change all of the scenes when you make a change that should affect all of them

Clear Inheritance

- how?**
 - Right click the root node of your inherited scene in the scene tree and select `Clear Inheritance`
- what will happen?**
 - This is exactly like `Make Local`, but for inherited scenes
 - You can freely edit, delete and change all nodes
 - Changes to the original scene will *no longer* be reflected, this scene is now entirely independent of the original scene
 - Doing this has the exact same effect as `Duplicate Scene File`, but with more steps
- when and why to use this?**
 - You thought that you wanted an inherited scene, but you realize that you really want an independent scene
- example:**
 - As you add more and more car types with inherited scenes, you realize that you want a `semi_truck.tscn`, which needs more sprites and more wheels, so the inheritance gets in your way
- note:**
 - Like in #1: When you use this to create a lot of similar nodes, you will have to manually change all of the scenes when you make a change that should affect all of them

