GODOT

Game engine

GODOT CHEAT SHEET / LEARN WITH A POSTER

Version 1.8 (March 2025)

The information was copied mostly from the Godot Engine 4.3 documentation (MIT license)
The data presented here has been selected for a typical beginner's tutorial. It is not intended to be complete.

Editor Default Shortcuts

Q = Select **W** = Move **E** = Rotate **R** = Scale

CTRL + **N** = New Scene
CTRL + **S** = Save Scene
CTRL + **A** = Add Child Node
CTRL + **F** = Frame to Selection
CTRL + **D** = Duplicate

F5 = Run Project
F8 = Stop Running Project

F = Center Selection
Y = Use Snap
T = Use Local Space

3D Nodes

CharacterBody3D
A PhysicsBody3D that is specialized for characters moved by script

Skeleton3D
A node containing a bone hierarchy, used to create a 3D skeletal animation.

MeshInstance3D
A node that takes a mesh resource and adds it to the current scene by creating an instance of it.

HingeJoint3D
A physics joint that restricts the rotation of a PhysicsBody3D around an axis relative to another physics body.

CollisionShape3D
A node that provides a Shape3D to a CollisionObject3D parent and allows to edit it. This can give a detection shape to an Area3D or turn a PhysicsBody3D into a solid object.

RayCast3D
A ray in 3D space, used to find the first CollisionObject3D it intersects.

StaticBody3D
A PhysicsBody3D that can not be moved by external forces. When moved manually, it does not affect other bodies in its path.

Area3D
A region of 3D space that detects other CollisionObject2Ds entering or exiting it.

Camera3D
Camera node for 3D scenes. It displays what is visible from its point of view.

AnimatableBody3D
A PhysicsBody3D that cannot be moved by external forces. When moved by code or by an AnimationPlayer, it affects other bodies in its path. Typically used for moving platforms, doors etc.

DirectionalLight3D
Directional light from a distance that covers the entire scene.

RigidBody3D
A PhysicsBody3D that is moved by the physics simulation. It cannot be controlled directly, instead, you must apply forces to it.

2D Nodes

CharacterBody2D
A PhysicsBody2D that is specialized for characters moved by script

ShapeCast2D
A 2D shape that sweeps a region of space to detect CollisionObject2Ds.

AnimatedSprite2D
Sprite node that contains multiple textures as frames to play for animation

RayCast2D
A ray in 2D space, used to find the first CollisionObject2D it intersects.

CollisionShape2D
A node that provides a Shape2D to a CollisionObject2D parent and allows to edit it. This can give a detection shape to an Area2D or turn a PhysicsBody2D into a solid object.

Area2D
A region of 2D space that detects other CollisionObject2Ds entering or exiting it.

StaticBody2D
A PhysicsBody2D that can not be moved by external forces. When moved manually, it does not affect other bodies in its path.

AnimatableBody2D
A PhysicsBody2D that cannot be moved by external forces. When moved by code or by an AnimationPlayer, it affects other bodies in its path. Typically used for moving platforms, doors etc.

Camera2D
Camera node for 2D scenes. It can move and zoom to create scrollable scenes.

TileMapLayer
A node that uses 2D tiles to build a grid-based map.

Important Control / General Nodes

Button
A themed button that can contain text and an icon.

AnimationPlayer
An animation player is used for general-purpose playback of animations.

Label
A control for displaying plain text.

AudioStreamPlayer
This node plays an audio stream non-positionally. It is ideal for user interfaces, menus, or background music.

LineEdit
An input field for single-line text.

Timer
This node is a countdown timer and is the simplest way to handle time-based logic.

GDScript

```gdscript
# Everything after "#" is a comment.
# A file is a class!

# (optional) icon to show in the editor dialogs:
@icon("res://path/to/optional/icon.svg")

# (optional) class definition:
class_name MyClass

# Inheritance:
extends BaseClass

# Member variables.
var a = 5
var s = "Hello"
var arr = [1, 2, 3]
var dict = {"key": "value", 2: 3}
var other_dict = {key = "value", other_key = 2}
var typed_var: int
var inferred_type := "String"

# Constants.
const ANSWER = 42
const THE_NAME = "Charly"

# Enums.
enum {UNIT_NEUTRAL, UNIT_ENEMY, UNIT_ALLY}
enum Named {THING_1, THING_2, ANOTHER_THING = -1}

# Built-in vector types.
var v2 = Vector2(1, 2)
var v3 = Vector3(1, 2, 3)

# Functions.
func some_function(param1, param2, param3):
    const local_const = 5

    if param1 < local_const:
        print(param1)
    elif param2 > 5:
        print(param2)
    else:
        print("Fail!")

    for i in range(20):
        print(i)

    while param2 != 0:
        param2 -= 1

    match param3:
        3:
            print("param3 is 3!")
        _:
            print("param3 is not 3!")

    var local_var = param1 + 3
    return local_var

# Functions override functions with the same name
# on the base/super class.
# If you still want to call them, use "super":
func something(p1, p2):
    super(p1, p2)

# It's also possible to call another function in the super class:
func other_something(p1, p2):
    super.something(p1, p2)

# Inner class
class Something:
    var a = 10

# Constructor
func _init():
    print("Constructed!")
    var lv = Something.new()
    print(lv.a)
```