Godot Quick Reference

Keyboard Shortcuts

Key	Use
F5	Run current project
F6	Run current scene
F7	Resume after pause
F8	Stop
F9	Toggle breakpoint
F10	Step out
F11	Step into
Ctrl \	Show hide recently opened files
Ctrl S	Save
Ctrl K	Comment a line
Ctrl R	Search/Replace current file
Ctrl F	Search current file
Shift Ctrl F	Find in files
Shift Ctrl R	Replace in files
Ctrl Shift F11	Max space for editing
Ctrl + A	Add new node
Ctrl + Shift + A	Instantiate new node
F	Focus on the selected node in the 3D scene view
Ctrl + F1	Switch to 2D
Ctrl + F2	Switch to 3D
Ctrl + F3	Switch to Code

Useful Nodes

Node	Purpose
Node3D	Node with a transform
XROrigin3D	Origin of the world in VR

Node	Purpose
XRCamera3D	Tracked Camera in VR
DirectionalLight	
StaticBody3D	World rigid body
CollisionShape3D	Required to respond to collisions. Set the Shape property
MeshInstance3D	3D mesh renderer
RigidBody3D	Rigid body
CharacterBody3D	Kinematic rigid body
Timer	Node that send signals on an interval
Camera3D	3D Camera
Node2D	2D transform node. The 2D transform has position, rotation (float)

Transforms

To do	Use
Movement	translate, move_and_slide, move_and_collide
Setting the position	position =, transform.origin =, global_transform.origin =
Rotating	rotate, rotate_x, rotate_y, rotate_z
Setting the rotation	rotation = Vector3(x, y, z). This is in radians. transform.basis = transform.basis.rotated(), global_transform.basis = global_transform.basis.rotated(), or Basis (from) - where from is a quaternion
Setting the scale	scale, transform.basis.scale, global_transform.basis.scale n

Particle Systems

Property	Meaning
ProcessMaterial	A shader that will process the particles. This is where the particle system is configured
DrawPass	Draws one Particle. Has a material
Amount	How many particles in the system
Emission Shapes	
Lifetime	How long each one lives for
One shot	Just fire once and stop

Property	Meaning
Preprocess	Wind the particle system forward this amount before starting
Explosiveness	Explodes them all out semi randomly
Randomness	How randomly they emit

Referencing other nodes

```
$"..".add_child(bullet)
$CharacterBody3D/Turret/bulletSpawn.global_transform.basis
$Timer.start(1.0 / fireRate)

get_parent()
find_child()
@onready var path1:Path3D=get_node("../Path3D")
@onready var path2:Path3D=$../Path3D
get_tree().root
get_tree().quit()
```

GDScript Reference

Code	Description
func _ready():	
if condition: else:	
if condition: elif:	
for i in range(length):	
while condition:	
var i = 0:int	
var f = 0.0:float	
var v = Vector3(1, 2, 3)	
@export var bulletPrefab:PackedScene	Give a node a reference to a packedscene (prefab) that can be instantiated later
var bullet = bulletPrefab.instantiate()	Create a new node from a packedscene
class_name MyClass extends Node: ...	Create a named class
var n = Something.new()	Instantiate a new object

Code	Description
func my_method():	Create a function
get_node("/path/to/node").get_node("MyComponent")	Get a node using path string
var rigidbody = get_node("/path/to/node").get_node("RigidBody")	
yield(get_tree().create_timer(duration), "timeout")	This is a coroutine. Timers are better
Input.is_action_pressed("ui_accept")	Check for an action
delta *or* get_process_delta_time()	time since last frame
global_transform.looking_at(boid.global_transform.origin, Vector3.UP)	
a.dot(b)	Dot product of two vectors. Used to calculate infront/behind or angle between the vectors, or for lighting
a.cross(b)	Cross product of two vectors
v.normalized()	Make of length 1. Preserve the direction
v.length()	Magnitude of the vector
a.distance_to(b)	Euclidian distance
from.angle_to(to)	
v.clamped(max)	Limit the magnitude
a.linear_interpolate(b, t)	lerp
inDirection.reflect(inNormal)	Reflect
Vector3.UP	World UP vector
Vector3.RIGHT	
Vector3.FORWARD	
rand_range() *In Godot, call randomize() once in your program to set the random seed*	
basis.slerp or quat.slerp	Slerp a basis vectror or quaternion
basis.xform()	Transform a vector
DebugDraw3D.draw_sphere(target.global_transform.origin, slowing_radius, Color.aquamarine)	Draw a sphere
DebugDraw3D.draw_line(boid.global_transform.origin, feeler.hit_target, Color.chartreuse) *or* DebugDraw.draw_arrow_line(feeler.hit_target, feeler.hit_target + feeler.normal, Color.blue, 0.1)	Draw a line

Code	Description
@tool	
@export	
@onready	