Godot Pet Game – Clean Project Skeleton (Nintendogs-style)

Core principles

- Keep **assets** (raw) separate from **scenes** (ready-to-place). - Drive balance & variations via **Resource (.tres)** in `data/`. - Reuse logic via **scripts/** modules; avoid mixing heavy logic into scenes. - Use **Autoload** for: SceneRouter, GameState, Save, GameDB. - Hide experimental/unused stuff in `quarantine/` and add `.gdignore`.

Folder tree (baseline)

```
res://
  assets/
    characters/    pets/    buildings/    props/    ui/    audio/    shaders/
  data/
    breeds/    items/    rooms/    tuning/
  scenes/
    _base/
      pet/    actor/    prop/    ui/
    pets/    characters/    rooms/    props/    ui/    flow/
  scripts/
    core/    pet/    ai/    interact/    ui/    systems/    util/
  ui/
  addons/
  quarantine/ (.gdignore)
  project.godot
```

Autoloads

```
SceneRouter.gd   (scripts/core/SceneRouter.gd)
GameState.gd     (scripts/core/GameState.gd)
Save.gd          (scripts/core/Save.gd)       # placeholder (stub)
GameDB.gd        (scripts/core/GameDB.gd)
```

Minimum flow scenes

- `scenes/flow/Splash.tscn` → fades then SceneRouter.goto(MainMenu) - `scenes/flow/MainMenu.tscn` → New/Continue/Settings/Quit - `scenes/rooms/Home.tscn` → where up to 3 pets are spawned - `scenes/ui/HUD.tscn` → top bar: time/date/money, player stat; pet cards 1–3

Key scripts (snippets)

```
# scripts/core/SceneRouter.gd
extends Node
func goto(scene_path: String) -> void:
    var next := load(scene_path).instantiate()
    var root := get_tree().root
    root.call_deferred("add_child", next)
    for c in root.get_children():
        if c != next: c.queue_free()
```

```
# scripts/core/GameState.gd (signals for HUD)
extends Node
class_name GameState
signal time_changed(hour:int, minute:int, day:int, season:String, year:int)
signal money_changed(amount:int)
signal pets_changed(active_ids:Array)
signal player_stat_changed(stat:Dictionary)
var hour := 8; var minute := 0; var day := 1; var season := "Spring"; var year := 1
var money := 0
var active_pets: Array[String] = []
var player_stat := {"hp":100, "energy":80, "mood":"ok"}
func tick_minutes(delta_min:int=1):
    minute += delta_min
    if minute >= 60: minute -= 60; hour = (hour+1)%24
    emit_signal("time_changed", hour, minute, day, season, year)
func add_money(v:int): money += v; emit_signal("money_changed", money)
func set_active_pets(ids:Array[String]): active_pets = ids.slice(0,3); emit_signal("pets_changed", active_
```

```
func set_player_stat(d:Dictionary): player_stat = d; emit_signal("player_stat_changed", player_stat)
```

```
# scripts/pet/PetBrain.gd (no human speech; intents -> emotes)
extends Node
class_name PetBrain
signal need_changed(name: String, value: float)
signal mood_changed(mood: String)
signal intent_fired(intent: String, intensity: float)
@export var breed: Resource
var needs := {"hunger":0.2,"thirst":0.1,"energy":0.7,"affection":0.6}
var current_mood := "neutral"
func _process(dt):
    for k in needs.keys(): needs[k] = clamp(needs[k] - 0.01*dt, 0.0, 1.0)
    if needs["hunger"] < 0.25: react("hungry", 0.7)
func react(intent:String, intensity:float=0.5) -> void:
    emit_signal("intent_fired", intent, intensity)
```

```
# scripts/pet/EmoteController.gd
extends Node
class_name EmoteController
@export var config_dir := "res://data/pets/emotes"
@onready var anim: AnimationTree = $"../AnimationTree"
@onready var sfx  := $"../SfxController"
var table := {}
func _ready():
    table = _load_configs(config_dir)
    get_parent().get_node("PetBrain").intent_fired.connect(play)
func play(intent:String, intensity:float) -> void:
    if not table.has(intent): return
    var cfg = table[intent]
    if cfg.anim_name != "": anim.set("parameters/%s/transition_request" % cfg.anim_name, true)
    _spawn_bubble(_pick(cfg.bubble_texts))
    sfx.play_any(cfg.sfx_tags, intensity)
func _load_configs(path:String)->Dictionary:
    var d := {}; for f in DirAccess.get_files_at(path):
        if f.ends_with(".tres"): var r := load(path + "/" + f); if r: d[r.intent] = r;
    return d
func _pick(arr): return arr[randi()%arr.size()] if arr.size()>0 else ""
func _spawn_bubble(text:String) -> void:
    var b := preload("res://scenes/ui/Bubble.tscn").instantiate()
    b.set_text(text); get_tree().current_scene.add_child(b); b.follow_node(get_parent())
```

```
# scripts/pet/SfxController.gd
extends Node
class_name SfxController
@export var bark_soft: Array[AudioStream]
@export var bark_mid:  Array[AudioStream]
@export var whine:     Array[AudioStream]
@export var whine_soft:Array[AudioStream]
@export var purr:      Array[AudioStream]
@onready var player := AudioStreamPlayer3D.new()
func _ready(): add_child(player)
func play_any(tags:Array[String], intensity:float):
    if tags.is_empty(): return
    var tag = tags[randi()%tags.size()]; var bank: Array = get(tag)
    if bank.is_empty(): return
    player.stream = bank[randi()%bank.size()]
    player.volume_db = lerp(-8.0, 0.0, clamp(intensity,0,1))
    player.pitch_scale = lerp(0.9, 1.1, randf())
    player.play()
```

HUD wiring (3 pets max)

```
HUD.tscn (Control FullRect) with TimeBox, MoneyBox, PlayerStat, PetCard1..3
HUD.gd listens to GameState signals and binds each PetCard to active pets (id-based).
```

Export tips

- Use Release + Strip Debug; do NOT export all resources; consider Encrypt Pack. - Move experimental files into `quarantine/` and add `.gdignore`. - Use Orphan Resource Explorer to remove unused assets before shipping.