Nœuds, Ressources et Références

TD n°2

M2 GAMAGORA

25 septembre 2025

Plan de la présentation

Les Nœuds dans Godot

Comment référencer un nœud ?

Méthodes disponibles :

- Avec `$`
- Avec `get_node()`
- Avec `@onready`
- Avec `%`
- Avec `@export`

Méthode 1 : Avec $

L'opérateur $ est un raccourci pour `get_node()`. Rapide et concis pour les références directes. Il peut cependant etre plutot enervant en cas de refactorisation de l'arborescence.

```
extends CharacterBody2D

func _ready():
    # Equivalent a get_node("Sprite2D")
    var sprite = $Sprite2D

    # Noeud avec un chemin plus complexe
    var ui_label = "/UI/Label"
```

Méthode 2 : Avec $

L'operateur % est similaire à $ mais utilise des nom unique dans la scène. Il est indépendant du placement du nœud dans la hiérarchie. Plus efficace que $ et moins sujet aux problèmes de refactorisation. Il est tout de meme sensible au changement de nom du noeud. Il demande de plus d'etre le seul noeud avec ce nom etant marqué unique.

```
extends CharacterBody2D

func _ready():
    # Equivalent a get_node("Sprite2D")
    var sprite = %Sprite2D

    # Independemment du placement du noeud cible et du noeud sur
        lequel est le script le chemin ne change pas
    var ui_label = %Label
```

Méthode 3 : Avec get_node()

Méthode explicite permettant de fournir le chemin sous forme de string ou de NodePath. Dans le cas d'un string hardcodé la performance est la même que $. Dans le cas d'une variable le chemin peut etre dynamique mais la performance est moindre.

```
extends Node2D

func _ready():
    # Recuperer un noeud profond
    var health_bar = get_node("../UI/Label")

    # recupere un noeud avec nom unique
    var unique_node = get_node("%UniqueNodeName")

    # Gestion d'erreur
    if has_node("OptionalNode"):
        var optional = get_node("OptionalNode")
```

Méthode 4 : Avec @onready

definition en début de script, initialisation automatique une fois le nœud prêt. Réutilisation simple sous forme de variable. Assez peu sensible au problème de refactoration car le chemin est indiqué qu'une seule fois. Facile a mettre en place (drag and drop) et très efficace en terme de performance.

```
extends Node2D

# References declarees avec @onready
@onready var player: CharacterBody2D = $Player
@onready var health_bar: ProgressBar = $UI/HealthBar

func _ready():
    # Les variables sont deja initialisees
    player.position = Vector2(100, 100)
    health_bar.max_value = 100
```

Méthode 5 : Avec @export

definition en début de script, initialisation automatique une fois le nœud prêt.
Réutilisation simple sous forme de variable.

```
extends Node2D

# References declarees avec @onready
@export var player: CharacterBody2D
@export var health_bar: ProgressBar

func _ready():
    # Les variables sont deja initialisees
    player.position = Vector2(100, 100)
    health_bar.max_value = 100
```

Comparaison des méthodes de référencement

Méthode	Lisibilité	Performance	Flexibilité	Refactorisation
`$Node/Node`	Excellente	Bonne	Mauvais	Mauvais
`%Node`	Excellente	Très bon	Moyenne	Bonne
`get_node("Node/Node")`	Moyenne	Bonne	Bonne	Mauvais
`get_node("%Node")`	Moyenne	Bonne	Bonne	Bonne
`get_node(path)`	Moyenne	Mauvaise	Excellente	Bonne
`@onready`	Très bon	Excellente	Excellent	Bon/très bon
`@export`	Très bon	Excellente	Excellente	Excellent

Les Ressources

Comment référencer une ressource ?

- Avec `preload("res://path/to/resource.tres")`
- Avec `load("res://path/to/resource.tres")`
- Avec `CustomResource.new()`
- Avec une variable `@export var resource: Resource`
- Avec une constante `const resource = preload("res://path/to/resource.tres")`

Cycle de vie des ressources

- Chargement `load()` ou `preload()`
- tant qu'une référence existe, la ressource reste en mémoire
- Partage entre toutes les instances
- Lorsque les références tombent à 0, la ressource est libérée automatiquement
- Lorsque la ressource est déchargé de la mémoire, toutes les modifications sur la resource sont perdues

Modification perdues au déchargement

```
extends Node2D


func _ready() -> void:
  var x = load("res://new_resource.tres")
  print(x.value) # prints default value
  x.value = 3
  print(x.value) # prints 3
  x = null
  x = load("res://new_resource.tres")
  print(x.value) # prints default value
```