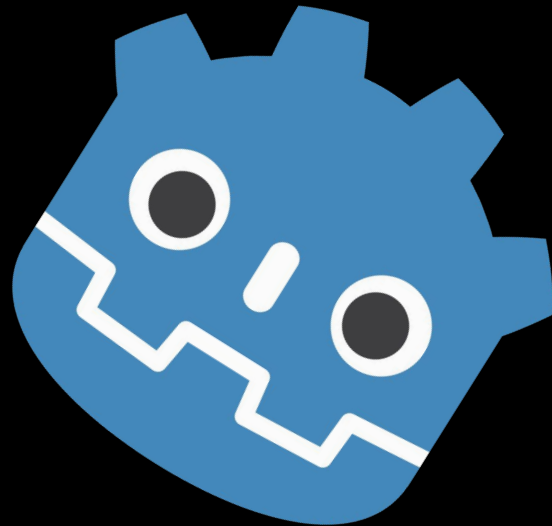




GODOT !!!!



# Plan :

- Présentation (chiant) de Godot.
- Les noeuds, c'est pas que dans le cerveau.
- Les noeuds principalement principaux.
- On programme que ici
- C'est l'heure de construire soit même
- Le début après la fin (:

**Godot** est un [moteur de jeu multiplateforme](#), c'est-à-dire un [logiciel](#) permettant de créer des [jeux vidéo](#) qui est compatible avec différents [systèmes d'exploitation](#). Il comporte entre autres un [moteur 2D](#), un [moteur 3D](#), un [moteur physique](#), un gestionnaire d'[animations](#), et des [langages de script](#) pour programmer des comportements. Il est depuis janvier 2014 ouvert au public et disponible sous [licence MIT](#), ce qui fait de lui un [logiciel libre](#). En plus des nombreux contributeurs [bénévoles](#) qui participent au projet, quelques développeurs rémunérés par don mensuel y travaillent à temps plein.

# **Plus sérieusement**

Godot logiciel de création de jeu

Pratique car facile d'utilisation

Logiciel complètement libre d'accès

...




# Premier principe


3 Dossiers principaux :

- Scènes
- Scripts
- Ressources

Après, vous faites concrètement comme vous voulez

## Classe : CharacterBody2D

Hérite de :  PhysicsBody2D <  CollisionObject2D <  Node2D <  CanvasItem <  Node <  Object

Héritée par :  "Script/player.gd"

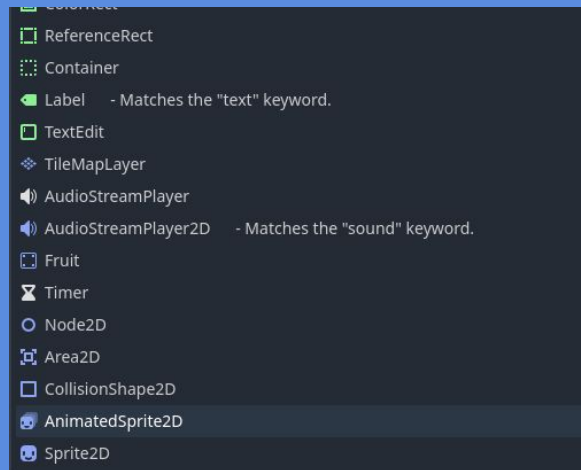
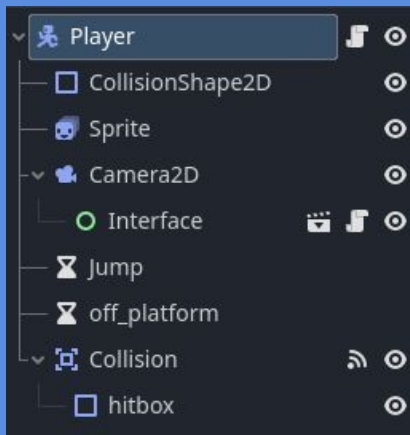
Un corps physique 2D spécialisé pour les personnages déplacés par script.

### Description

**CharacterBody2D** est une classe spécialisée pour les corps physiques qui sont destinés à être contrôlés par l'utilisateur. Ils ne sont pas affectés par la physique du tout, mais ils affectent les autres corps physiques sur leur chemin. Ils sont principalement utilisés pour fournir à l'API de haut niveau un moyen de déplacer des objets avec de la détection de mur et de pente (la méthode `move_and_slide()`) en plus de la détection générale de collisions fournie par `PhysicsBody2D.move_and_collide()`. Cela la rend utile pour les corps physiques hautement configurables qui doivent se déplacer de manière spécifique et se entrer en collision avec le monde, comme c'est souvent le cas avec des personnages contrôlés par l'utilisateur.

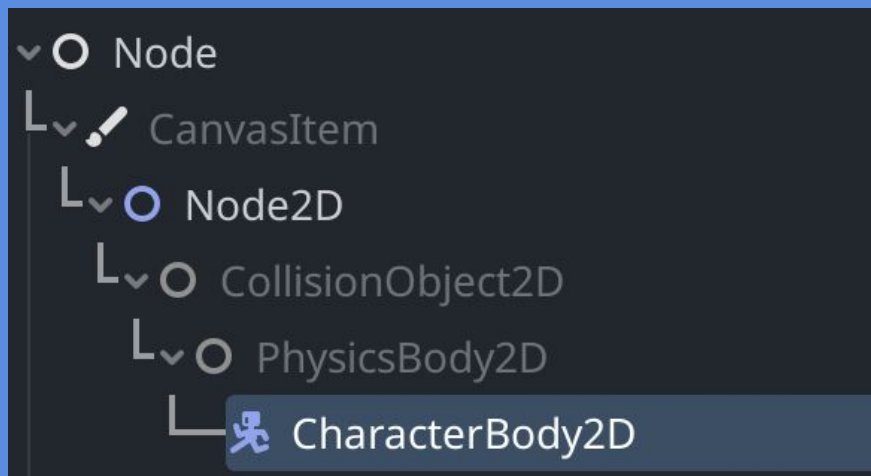
Pour les objets de jeu qui ne nécessitent pas de détection de mouvement ou de collision complexe, comme des plates-formes mobiles, `AnimatableBody2D` est plus simple à configurer.

# La Base des Bases : Les noeuds

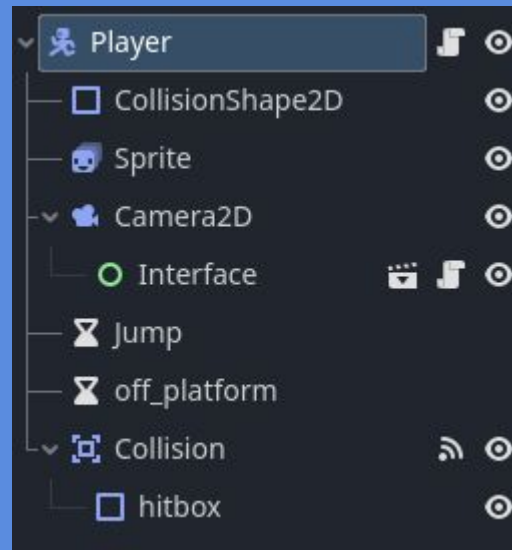


# Comment ke ça fonctionne?

## Principe de classe

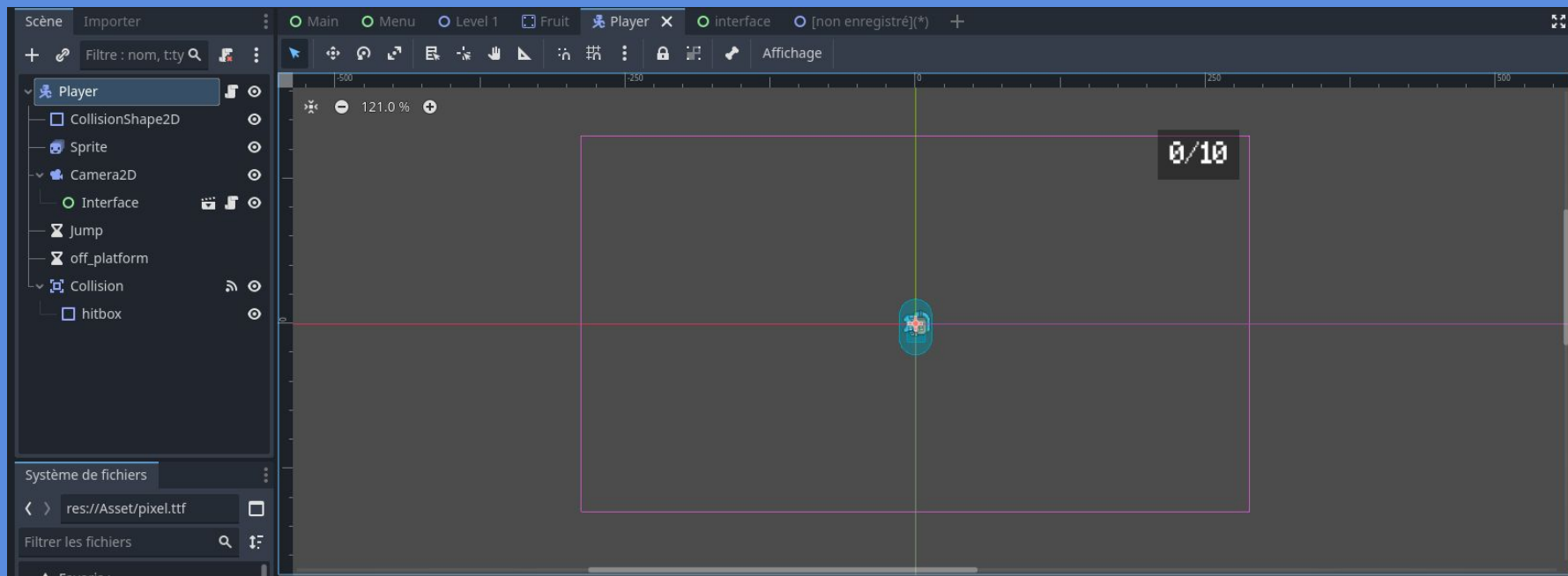


## Noeud Parent -> Noeud Enfant



# Comment ke ça fonctionne?

## Regroupement de noeuds : scène





# Les noeuds principaux

# Les noeuds principaux



# On ajouter les noeuds !!

- Character body 2D
- Collision Shape 2D
- Sprite
- Area 2D
- TileMap
- Camera

```

AnimatedSprite2D = $Sprite
t_death : Timer = $Timer
und_collected : AudioStreamPlayer2D = $SoundEffects

when the node enters the scene tree for the first time:
ready() -> void:
var skin = randi() % 4

match skin :
    0 : sprite.play("Apple")
    1 : sprite.play("Banana")
    2 : sprite.play("Cherry")
    3 : sprite.play("Kiwi")

func disappear() -> void :
    collision_mask = 0
    collision_layer = 0
    sprite.play("Disappear")
    sound_collected.play()
    collected.emit()
    timer_death.start()

```

# Nos scripts

```

func _ready() -> void :
    gravity_scale = 0

func _process(delta: float) -> void:
    if deplace :
        position.x += 2*cos(tourne)
        position.y += 2*sin(tourne)
        tourne += 0.005

    if change :
        var decision = randi()%decisiond

        if decision == 0 :
            tourne = -tourne
            decisiond = randi()%400

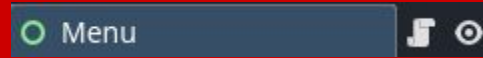
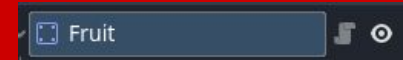
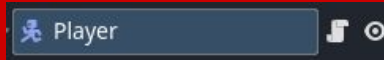
    if queue != null :
        if position.distance_to(queue.position) > 32 :
            queue.suit_action()

    z_index = position.y

```

# Kékecé un script ?

## Codes Attachés à un noeud



## Programmé en python ou C#

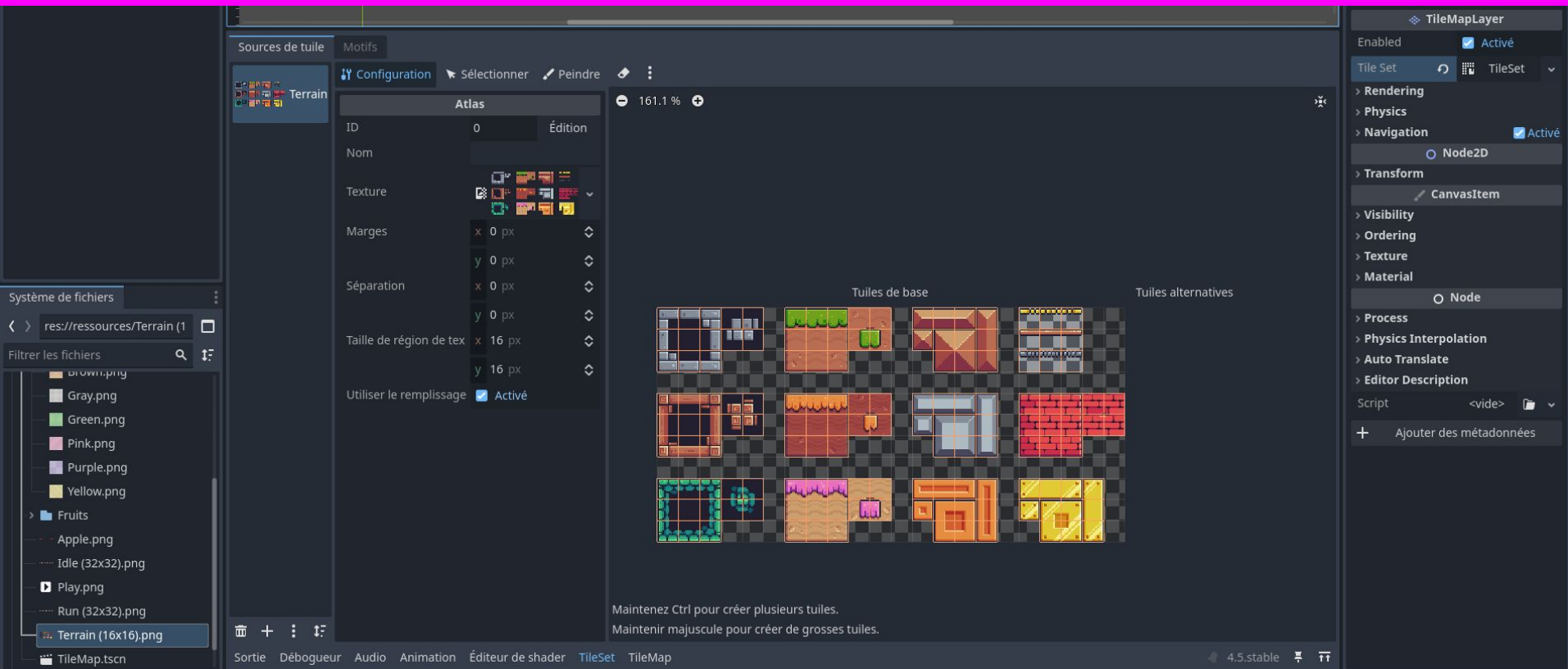
# On écrit nos premiers scripts !!

On veut déplacer notre joueur :

- créer un script
- interagir avec les noeuds
- exporter une valeur
- importer une scène

**TileMapLayer = TileMap => Layer = 0**

# C'est l'heure de construire un niveau





**Faites votre propre niveau**

**C'est pas fini !!**

**Il reste tout une fiche avec plus de truc  
comme:**

- Faire le menu**
- Changer de scène**
- Jouer avec les animations**
- ...**