

# Utiliser les moteurs de jeu et Github en tant qu'artiste



## Édition Godot



22/06/2024

Version du moteur: Godot 4.2.2

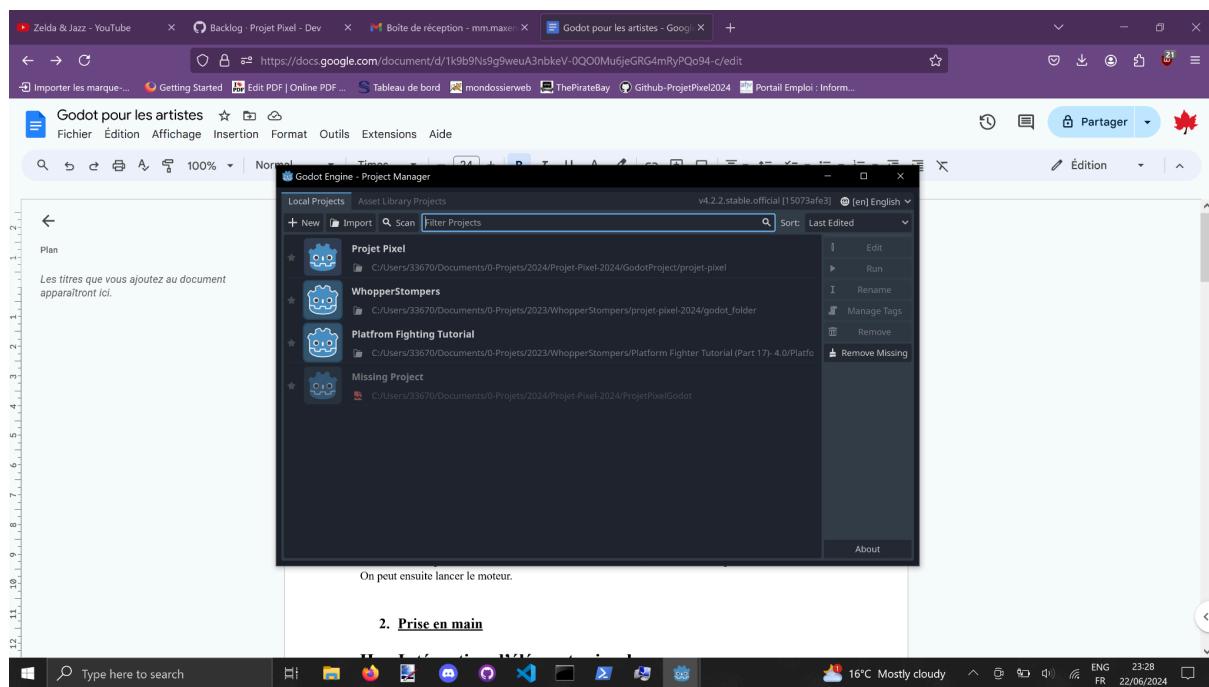
Lien de téléchargement: <https://godotengine.org/download/windows/>

## I. Prise en main de Godot:

### 1. Téléchargement

Le téléchargement se fait directement sur le [site officiel de Godot](#).

Ensuite, on dézip et on a l'exécutable. Pas besoin d'installateur, le moteur entier pèse environ 100 Mo. On peut ensuite lancer le moteur.

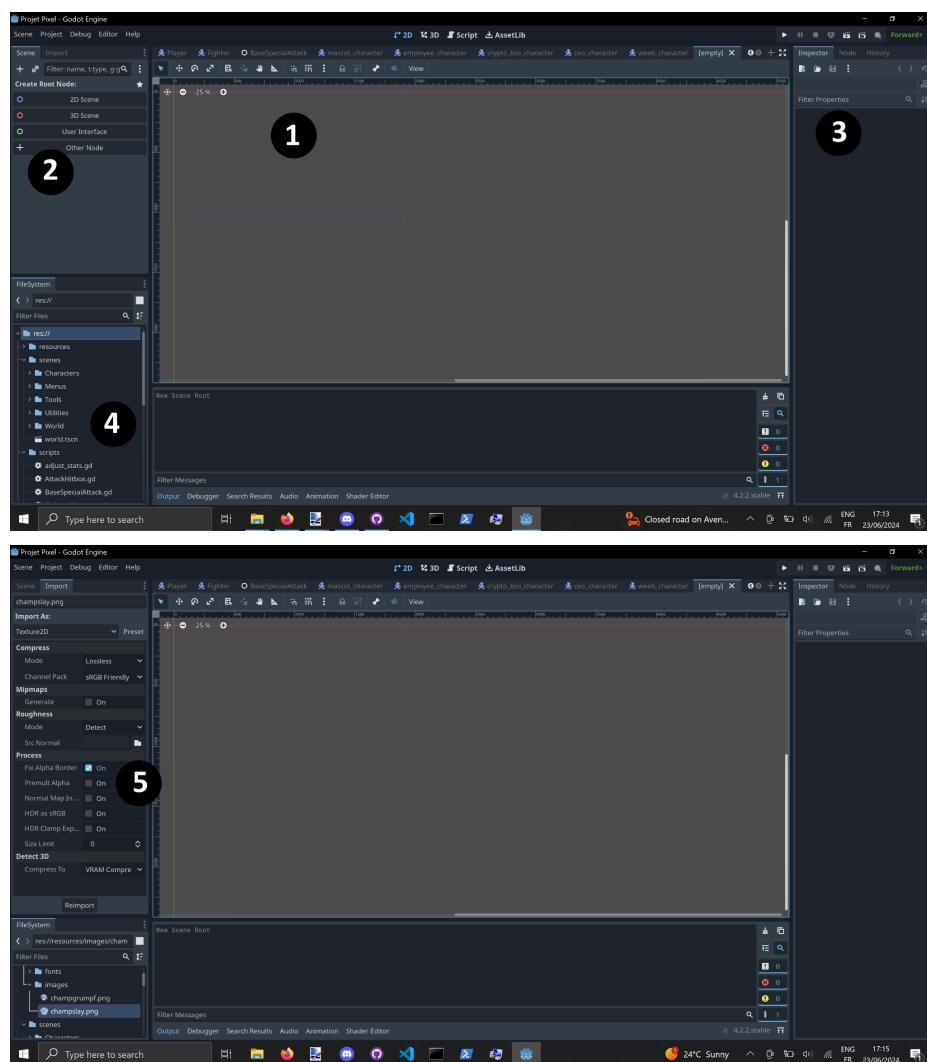


On tombe sur cette fenêtre, où on peut soit ouvrir un projet récent, soit en **Import** un qui est sur votre ordinateur (pour ça il faut naviguer jusqu'au dossier et double-cliquer le fichier **project.godot**).

## 2. Utilisation

### Vocabulaire:

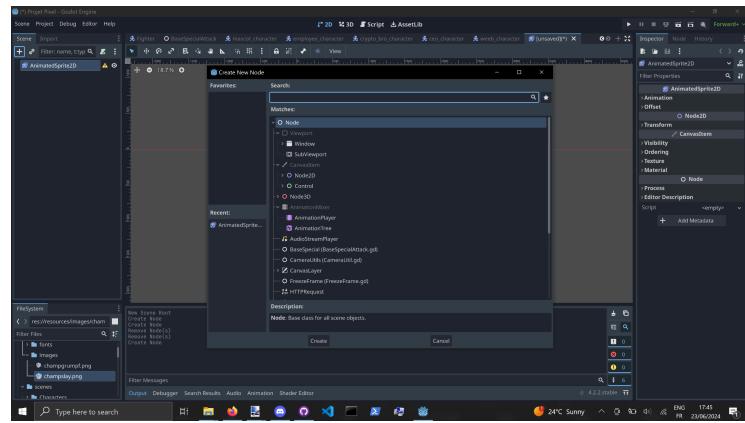
- **Scenes:** fichiers ‘.tscn’. Ce sont les objets qu’on crée dans Godot: les niveaux, les personnages, les attaques, les effets de particules, les menus, etc... Peuvent contenir du code et des ressources (sprites, fichiers audios, effets de particules, autres scènes...)
- **Scripts:** fichiers codes (.gd). On peut les attacher à des scènes.
- **Node:** objet minimal de Godot, élément minimal d’une scène. Une node peut être constituée de plusieurs Nodes. Les scènes sont constituées de Nodes, et sont elles-mêmes des Nodes. Différentes Nodes ont différentes propriétés (afficher un sprite, avoir une zone de collision, jouer un fichier audio, contenir des éléments de code...)



### Fenêtres principales de Godot:

- 1- **Viewport:** fenêtre principale, où on peut voir/éditer les scènes/nodes 2D, les scènes/nodes 3D, et où l’on a l’éditeur de code.
- 2- **Scene:** “arborescence” de la scène. Liste des Nodes dans la scène actuelle.
- 3- **Inspector:** détail d’une Node sélectionnée dans la scène actuelle. Ici on peut voir et changer les propriétés d’un objet précis de la scène.
- 4- **FileSystem:** explorateur de fichiers du projet.

**5- Import:** détails d'un fichier sélectionné dans le FileSystem. Ici, on peut changer les caractéristiques d'importation du fichier. Exemple important: pour les fichiers en pixel art, on peut désactiver l'option "Filter", qui arrondit les images et cache les pixels individuels.



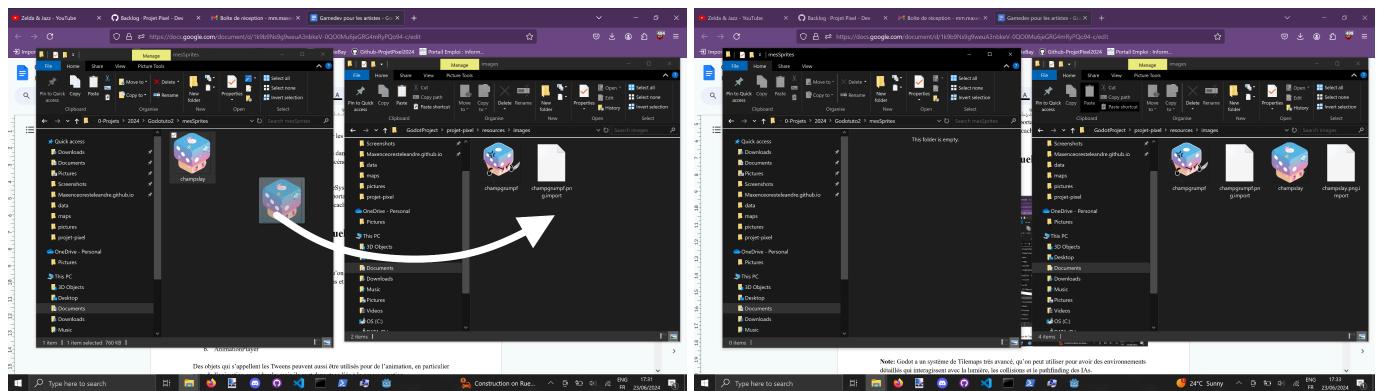
Si vous voulez ajouter une Node à une Scene, cliquez sur le bouton "+" en haut à gauche, et la liste des Nodes de Godot apparaîtra à l'écran. Ensuite, vous pourrez sélectionner la Node dont vous avez besoin (un sprite, un objet 3D, un élément d'interface, etc...).

## II. Intégration d'éléments visuels

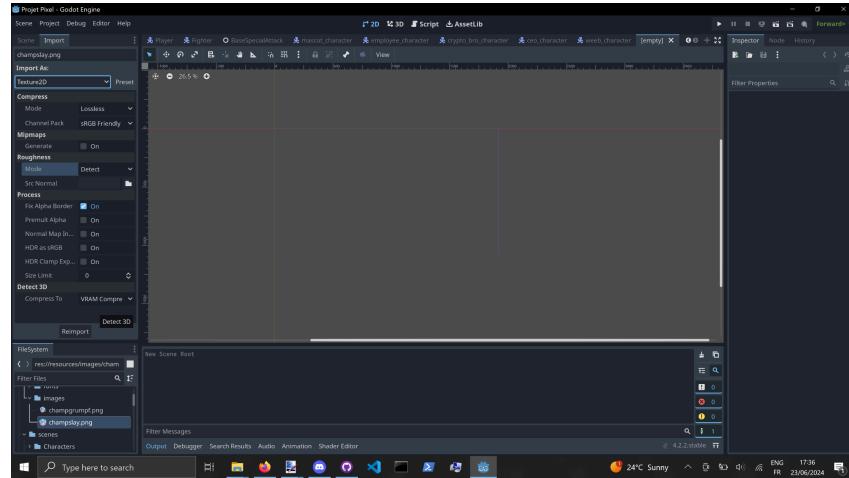
### 1. Importer et utiliser des sprites

#### a. Importer un Sprite:

Pour importer un sprite dans Godot, il suffit de le glisser dans un dossier du projet, depuis l'explorateur de fichier.



Quand vous ouvrez godot, ça créera un fichier "*NomDeLaRessource.import*" automatiquement.

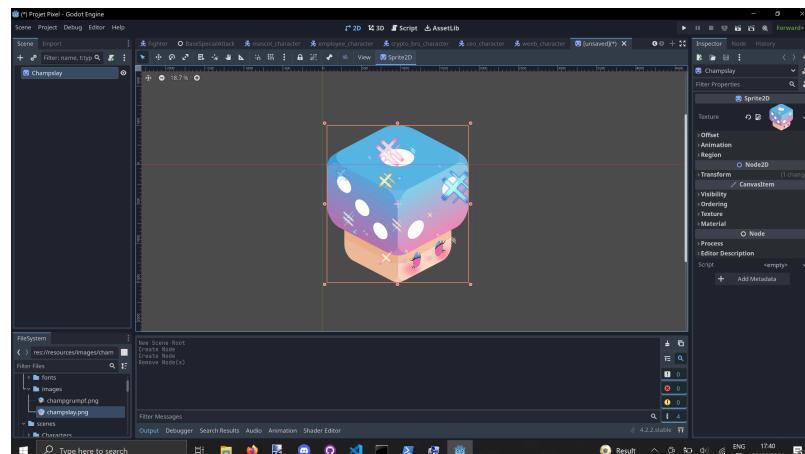


Après, vous pouvez modifier les paramètres d’importation de la ressource en la sélectionnant dans **FileSystem** et en ouvrant la fenêtre **Import**, mais la plupart du temps ce n'est pas nécessaire.

### b. Ajouter un Sprite à une scène

Pour les scènes 2D (personnages, environnement), vous pouvez juste sélectionner le fichier et le glisser dans le **Viewport** pour créer une **Node** Sprite2D.

En cliquant dessus, vous pouvez voir et modifier ses propriétés dans l'**Inspector**.



Vous pouvez aussi cliquer sur une **Node** Sprite2D qui existe déjà, et glisser-déposer le fichier image que vous voulez utiliser sur sa propriété Texture dans l'**Inspector**.

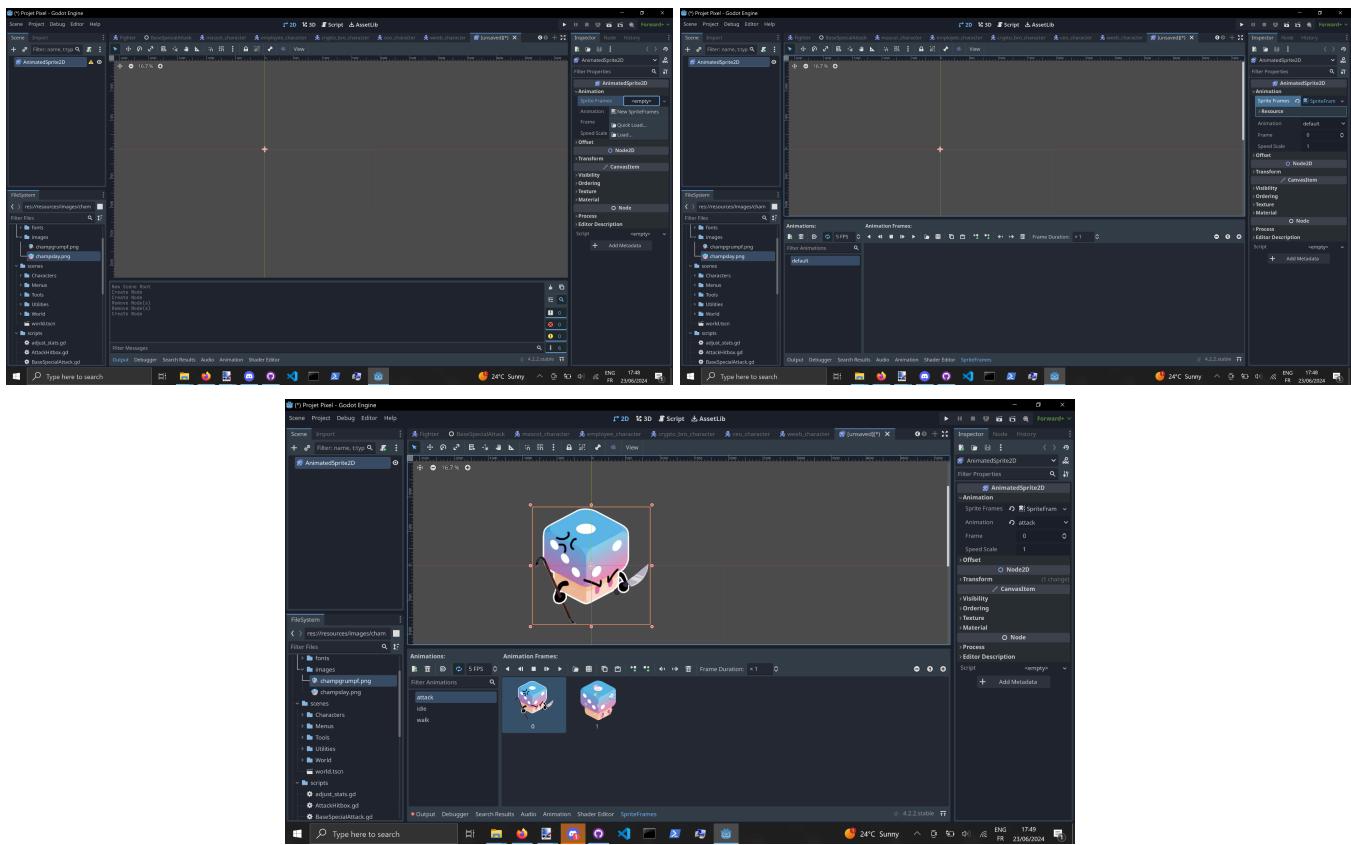
On ne va pas toucher aux éléments d’interface ici, mais c’est un processus un peu différent.

**Note:** Godot a un système de Tilemaps très avancé, qu’on peut utiliser pour avoir des environnements détaillés qui interagissent avec la lumière, les collisions et le pathfinding des IAs.

## 2. Créer des animations

### a. AnimatedSprite2D

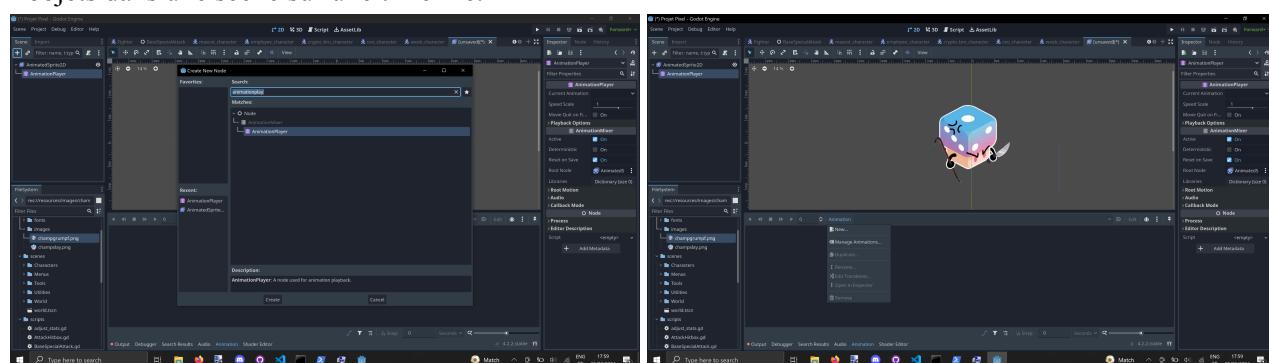
Il y a une Node qui permet de faire de l’animation traditionnelle sous Godot: l’AnimatedSprite2D.

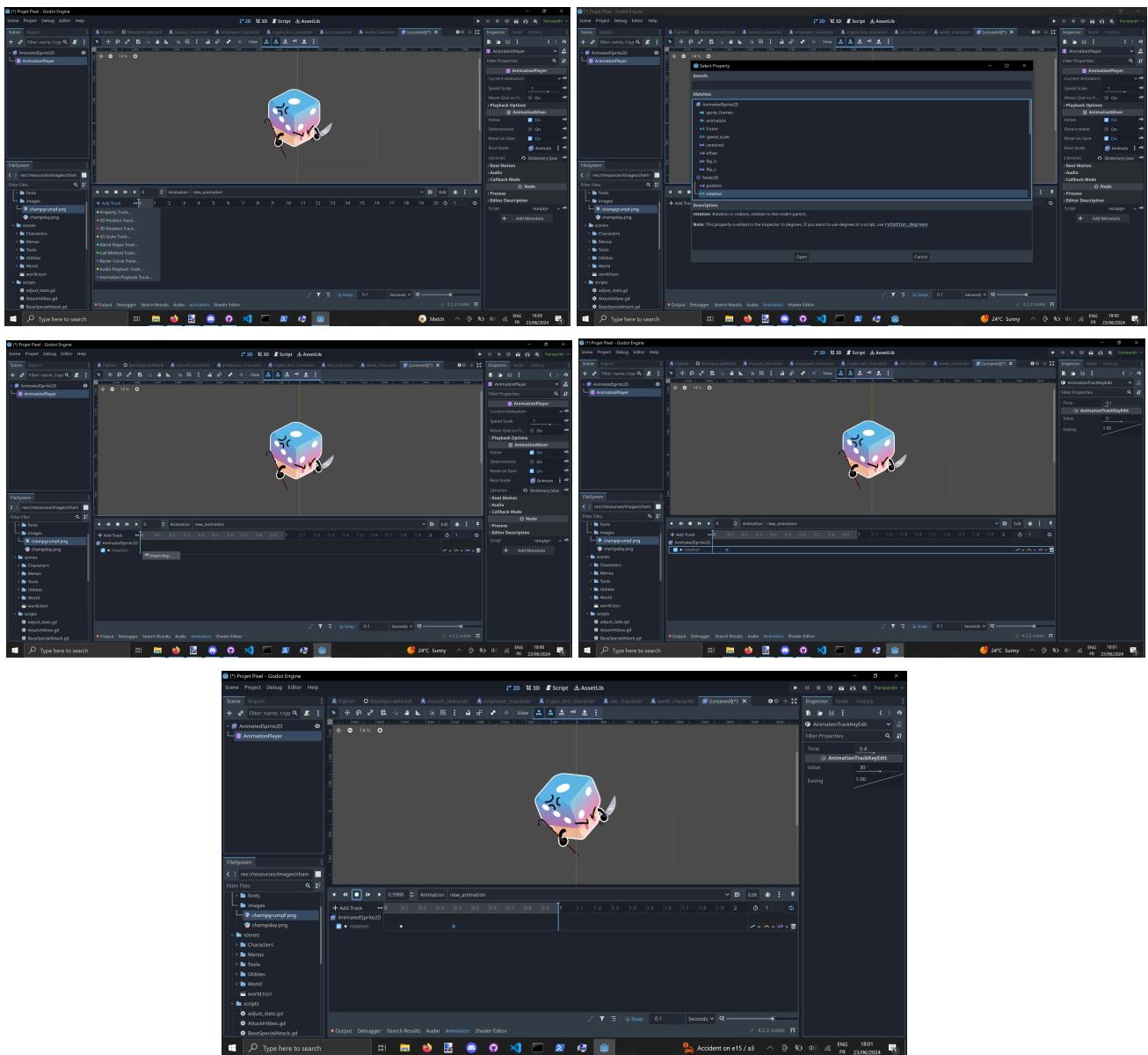


- (1) Pour utiliser l'AnimatedSprite2D, il faut d'abord l'ajouter à la Scene ('+' en haut à gauche, puis chercher AnimatedSprite2D dans la liste des Nodes).
- (2) Ensuite, il faut cliquer sur la propriété SpriteFrames de l'AnimatedSprite2D dans l'Inspector, et sélectionner New SpriteFrames. Si on reclique sur la propriété, ça ouvre le menu SpriteFrames, qui permet d'éditer les animations de cette Node (au milieu-bas de l'écran).
- (3) Dans ce menu (fenêtre milieu-bas), on peut ajouter différentes animations ('+' en haut à gauche de la fenêtre), en supprimer (poubelle en haut à gauche), changer les FPS des animations ou si elles bougent ou pas (en haut à gauche toujours). Pour ajouter des images à une animation, il suffit de les glisser-déposer dans l'espace vide au milieu de la fenêtre. Vous pouvez aussi renommer ou éditer les différentes animations en cliquant sur leur nom (à gauche).

## b. AnimationPlayer

Cette Node permet “d’animer”, en faisant manuellement bouger la position, rotation et taille des objets dans une scène sur une timeline.

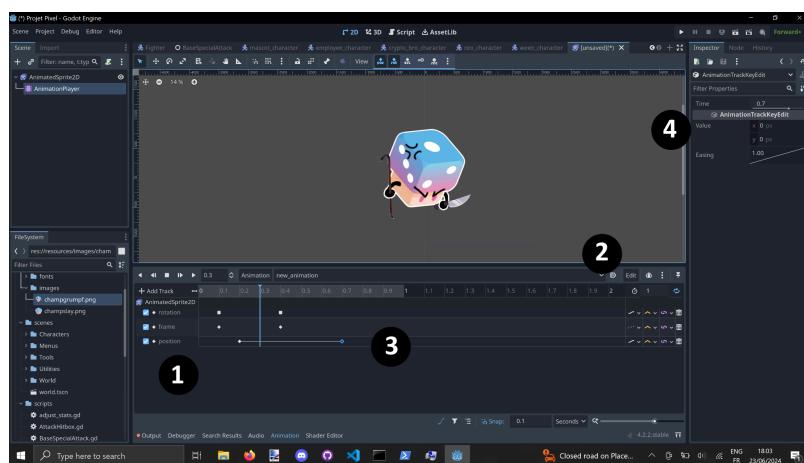




- (1) Pour ajouter un AnimationPlayer dans la scène où vous avez ajouté la Node AnimatedSprite2D, ajoutez une Node et cherchez “AnimationPlayer” dans la liste des Nodes de Godot.
- (2) Quand on clique sur la Node AnimationPlayer qu'on a ajoutée, ça ouvre le menu Animation (fenêtre au milieu-bas de l'écran). Ici, on peut cliquer sur le bouton Animation, et New pour créer une nouvelle animation.
- (3) Ça ouvre l'éditeur d'animations. On peut cliquer sur New Track pour ajouter une Track. Les Tracks représentent l'animation d'une propriété de la Node.  
Exemple: si on a une track sur la propriété Position d'un sprite, avec deux clés ayant pour valeurs (0, 0) et (100, 0), ça va interpler de force la position du sprite entre ces deux valeurs (donc le sprite va faire des aller-retours horizontaux si l'animation boucle, ou faire un aller simple si l'animation ne boucle pas).
- (4) Ajoutez une Track, cliquez sur la Node AnimatedSprite2D que vous aviez créée plus tôt, et sélectionnez la propriété Rotation. On va faire tourner notre Sprite.

- (5) Maintenant que la Track est apparue, faites un Clic droit sur la Timeline, et cliquez sur New Key, pour ajouter une clé.
- (6) Un point ou un carré devrait apparaître sur la timeline. C'est la clé. Cliquez dessus pour la sélectionner, et dans la fenêtre de l'Inspector, vous allez avoir les propriétés de la clé: sa position dans la timeline, sa valeur de propriété et sa valeur de easing (comment est interpolée la valeur de propriété par rapport aux autres clés sur la timeline).
- (7) Changez la valeur de propriété de la clé, ajoutez d'autres clés, et changez leurs valeurs, appuyez sur play, waw, votre sprite tourne ! Vous faites de l'animation :D

## Résumé:



**1- Tracks:** c'est ici que vous pouvez ajouter, supprimer ou sélectionner des Tracks de l'animation en cours. Vous pouvez avoir autant de Tracks que vous voulez. Une même animation peut donc modifier n'importe quelles propriétés de Nodes en même temps.

**2- Animation:** le bouton Animation permet de sélectionner, d'ajouter, de supprimer, dupliquer ou renommer des animations. Un même AnimationPlayer peut avoir plusieurs animations, utile pour passer de Idle à Walk à Hit.

**3- Timeline:** endroit où l'on ajoute les clés d'animation. Les valeurs des propriétés sont interpolées entre les clés, selon les courbes à droite de la fenêtre, et les easings des clés. Les double flèches à droite permettent de déterminer si l'animation boucle ou pas (une animation de Walk devrait boucler mais pas une animation d'attaque).

**4- Key:** fenêtre d'édition des propriétés d'une clé. On peut changer ici sa valeur, son temps (position dans la timeline), et son easing (jouez avec ça c'est marrant de voir les effets qu'ont l.easing sur une animation).

**4bis- Propriétés de l'animationPlayer:** Si vous recliquez sur la Node AnimationPlayer, l'Inspecteur affichera les propriétés de la Node plutôt que celles de la clé. Il y a quelques propriétés utiles là-dedans, mais on s'écarte de l'animation là.

**Note:** Des objets qui s'appellent les Tweens peuvent aussi être utilisés pour de l'animation, en particulier pour faire de l'animation procédurale, mais ils sont davantage liés à la programmation.

## III. Utilisation avec Github

### 1. C'est quoi Github

#### Qu'est ce que Git ?

*C'est l'outil.*

Git c'est un système très pratique pour les projets informatiques, qui permet de gérer les différentes versions des projets pendant leur évolution.

Ça permet de:

- voir exactement les changements effectués entre les différentes sauvegardes (commits) d'un projet
- avoir toutes ces versions sauvegardées en ligne. Ça veut dire que TOUT est sauvegardé, même la première version du projet, donc si à un moment le projet a un problème, on peut revenir à un commit précédent.
- collaborer à plusieurs sur un même projet, en voyant les commits de tout le monde, ce sur quoi différentes personnes travaillent, etc.
- l'open-source ! Ça serait impossible sans git et les projets (repositories) publics

#### Qu'est ce que Github ?

*C'est les serveurs.*

Github c'est un service en ligne qui héberge les projets (repositories) Git dans le cloud.

#### Qu'est ce que Github Desktop ?

*C'est l'interface qui simplifie tout.*

Github Desktop c'est juste un logiciel pour utiliser Github facilement, sans passer par la ligne de commande et en visualisant rapidement les différents commits.

#### Vocabulaire:

- **Commit:** sauvegarde du projet dans un état donné, version du projet.
- **Push:** action de prendre les nouvelles modifications qu'on a faites sur notre PC, et les sauvegarder sur le projet en ligne (repository). L'action de Push crée un Commit.
- **Pull:** action de prendre la dernière version en ligne du projet et la récupérer sur notre PC. En gros, ça regarde si on a les derniers Commits, et si c'est pas le cas ça les récupère.
- **Branch:** un peu compliqué, en gros version parallèle du projet. Vous pouvez voir ça comme une version test du projet, où on fait plusieurs Commits pour ajouter un nouvel élément, et quand l'élément est stable, on fusionne la branche avec la version principale du projet. La version principale du projet, c'est la branche Main.

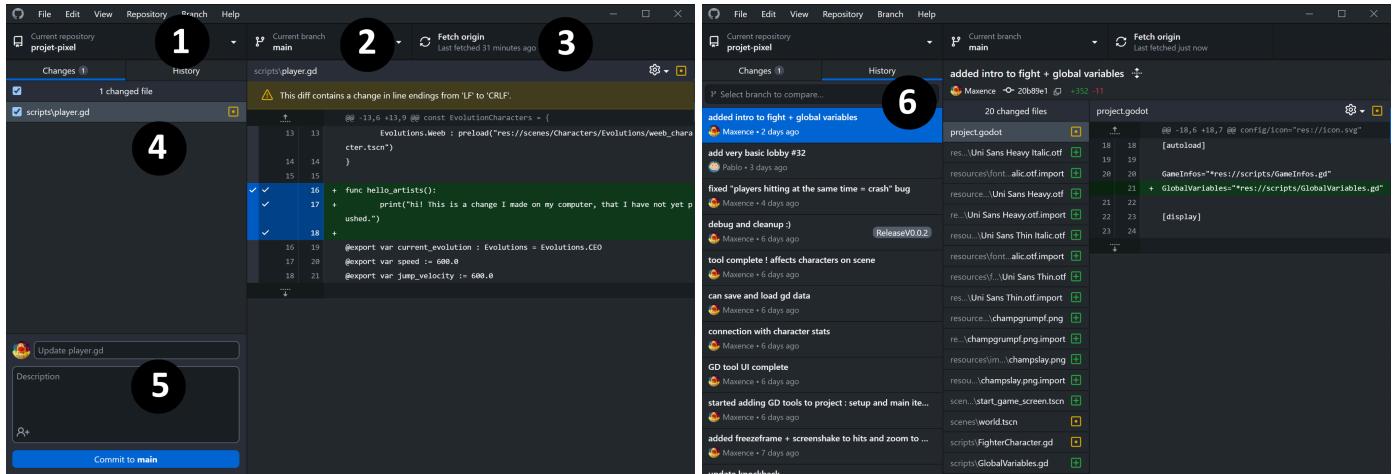
### 2. Setup

Créer un compte Github: <https://github.com/>

Télécharger Github Desktop (optionnel mais pratique): <https://desktop.github.com/>

Lier Github Desktop à votre compte Github.

### 3. Utilisation



**1- Current repository:** projet qu'on est en train de regarder.

**2- Current branch:** branche qu'on est en train de regarder.

**3- Fetch origin:** parfois se transforme en **Pull origin**. C'est le bouton pour regarder s'il y a de nouvelles modifications faites par d'autres personnes, et les récupérer. Quand vous faites un Commit, il se transforme en **Push origin**. C'est ce qui permet d'envoyer vos changements sur le serveur où est sauvegardé le projet.

**4- Changes:** modifications que vous avez faites, mais que vous n'avez pas encore ajoutées au projet (ie que vous n'avez pas encore Commit). C'est bien de vérifier que le projet marche toujours avant de commit. Sinon, clic droit sur les fichiers modifiés et on Discard (= on supprime les modifs).

À gauche, il y a les fichiers que vous avez modifiés, et à droite il y a les modifications faites au fichier (surtout utile pour les fichiers de code où on voit les lignes changées).

**5- Commit:** partie de la fenêtre où vous pouvez sauvegarder sur le projet (commit) vos changements. C'est bien de donner un titre indicatif de ce que vous avez fait et de faire une ptite description gentille.

**6- History:** historique des commits sur la branche actuelle. C'est bien si on veut voir les nouvelles modifications des derniers commits, ou si on a besoin d'aller regarder de vieux commits (ou si on veut regarder le premier commit du projet pour se dire uwu qu'est ce qu'on en a fait du chemin quand même).

## **IV. Bonus :D**

### **Ressources utiles:**

- [godotshaders](#): site sur lequel on peut trouver énormément de shaders (effets visuels jolis) pour Godot. N'hésitez pas à y jeter un œil, ça peut donner des idées!
- [Kenney Game Assets](#): bibliothèque de ressources gratuites et libres de droits pour le jeu vidéo. Il y a de la 2D, de la 3D, de l'interface... Site très sympa, pratique pour le prototypage et les game jams.
- Ressources Godot:
  - [Documentation officielle](#): très pratique, mais plutôt destinée aux devs en vrai
  - [Cours GDQuest](#): comme toujours avec Godot, gratuit et de très bonne qualité uwu. Inclut un tuto interactif destiné aux débutants complets, pas besoin de connaître Unity ou d'avoir déjà fait de la prog pour regarder ça. C'est même présenté sous forme de jeu ! Mais que demande le peuple

### **1. Compiler un jeu et faire un release:**

Les releases, sur Github, sont des exécutables du projet ; des versions compilées qu'on peut faire tester aux utilisateurs.

//TODO

### **2. Jouer avec la lumière**

Les lumières dynamiques en 2D c'est grave cool et ça améliore énormément la qualité visuelle d'un jeu.

//TODO

### **3. Créer des effets de particules**

Il y a des outils de particules cools avec Godot. Pas compliqué à utiliser.

//TODO

### **4. Les shaders**

<https://godotshaders.com/>

C'est joli, ça peut faire plein de trucs, on a besoin de les coder.

//TODO

un peu la flemme, si on voit tout ce qui est écrit au dessus ou si c'est utile à quelqu'un demandez-moi et j'ajouterais le reste ~ Maxence