Club de codage Saint Gatien - M.Grondin - https://github.com/MaxGrondin

- Tutoriel de **développeur** + **animateur**
- Niveau **medium**

Animer un personnage 2D avec *AnimatedSprite*

Rappels :

- Ce tutoriel vous permet de cocher une case d'expérience dans votre **carnet de progression** en tant que **développeur** ET d'**animateur**.
- Le vocabulaire propre au moteur GODOT est écrit en *italique*.

Objectifs du tutoriel :

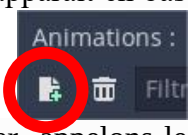
- Animer un personnage dans un jeu 2D : son image changera selon ses actions, comme marcher, sauter, attendre.
- Ce tutoriel s'adresse surtout aux jeux « plateformer », mais il est facilement adaptable pour tout autre jeu 2D.

Prérequis obligatoire :

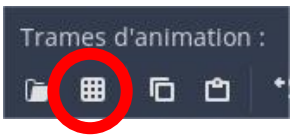
- Avoir déjà conçu un niveau 2D avec gestion des collisions
- Avoir déjà conçu un personnage jouable en 2D (ayant donc son *Sprite*, sa *CollisionShape2D* et son script).
- Avoir conçu une *SpriteSheet* de personnage (par exemple sur Piskel) et avoir placé son image dans le dossier de votre projet GODOT.

Etapes à suivre :

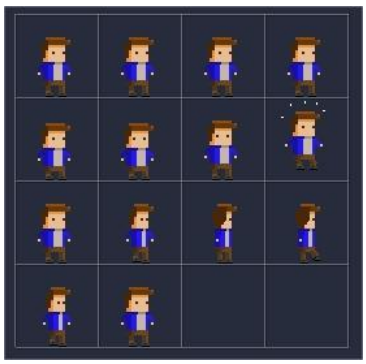
1. Ouvrez votre projet Godot
2. Supprimer le nœud *Sprite* de votre personnage.
3. Créez à sa place un nœud nommé *AnimatedSprite* (enfant au *KinematicBody2D*).
4. Dans l'inspecteur de l'*AnimatedSprite*, cliquez sur [vide] à côté de « Frames » et choisissez « Nouveau Spritesframes ».
5. Au même emplacement, cliquez sur « SpriteFrames », un nouvel onglet d'animation apparait en bas de l'écran.
6. Dans cet onglet, cliquez sur l'icône du parchemin avec un « + » vert pour ajouter une nouvelle animation (voir image de droite).



7. Par défaut elle aura le nom « New Anim », double-cliquez dessus pour le modifier, appelons-le « WalkRight » (« marche à droite » en anglais).
8. Cliquez ensuite sur le symbole de grille en dessous de « Trames d'animation » (voir image de droite), ce qui permet de choisir dans le dossier votre image qui servira pour les animations.



9. Dans la page qui s'ouvre, choisissez votre image dans les ressources du dossier.



10. Votre image doit désormais apparaitre dans une grille (voir image de gauche). Si vous avez dessiné vos *Sprites* de personnages en 64x64 pixels, la grille devrait être adaptée à chaque image. Si vous constatez un décalage, vous pouvez modifier les paramètres en haut de la page (décalage de la grille, nombre de pixels par image).
11. Comme nous voulons ici faire une animation de marche, sélectionnez en cliquant dessus les images qui font parti de cette animation (dans le bon ordre). Cliquez ensuite sur « Ajouter … Trame(s) »

12. Les images que vous avez choisies apparaissent dans l'onglet en bas. Si elles sont dans le désordre, vous pouvez le modifier avec les flèches située sous « Trames d'animation : ».

13. Vérifiez une nécessité : la première et la dernière image de l'animation doivent être la même image du personnage statique.
14. Voilà, vous avez créé votre première animation, bravo !
15. Ajoutez ensuite d'autres animations, par exemple « Idle » (attente) et « Jump » (saut) en reprenant depuis l'étape 6 (pour l'animation de saut, enlevez l'option « Boucle » dans l'onglet animation).
16. Quand vous avez ajouté toutes vos animations, cliquez sur le nœud *AnimatedSprite*, vous pourrez ensuite dans l'*Inspecteur* choisir l'image par défaut de votre personnage à côté de « Animation » (choisissez « Idle »). Vous pouvez également voir les animations être jouées si vous activez la case « Playing » de l'*Inspecteur* (mais désactivez-la avant de tester le jeu).
17. Allez maintenant dans le script du *KinematicBody2D*.
18. Dans le code, parmi vos lignes de *variables*, ajoutez la ligne suivante :

```
onready var animated_sprite = get_node("AnimatedSprite")
```

19. Ecrivez maintenant les lignes suivantes au sein de votre fonction de déplacement « *func get_input(): »* en respectant les indentations (attention, vérifiez que la fonction utilise le mêmes *input* comme `ui_right` par exemple. Vérifiez également que les noms des animations dans le code comme « WalkRight » correspondent à ceux que vous avez choisi lors de leur création).

```
→ if Input.is_action_pressed('ui_right') and is_on_floor():
→  → animated_sprite.play ("WalkRight")
→  → $AnimatedSprite.flip_h = false
→ elif Input.is_action_pressed('ui_left') and is_on_floor():
→  → $AnimatedSprite.flip_h = true
→  → animated_sprite.play ("WalkRight")
→ elif Input.is_action_pressed('ui_right') and Input.is_action_pressed ('ui_select'):
→  → animated_sprite.play ("Jump")
→ elif Input.is_action_pressed('ui_left') and Input.is_action_pressed ('ui_select'):
→  → animated_sprite.play ("Jump")
→ elif Input.is_action_pressed('ui_select') :
→  → animated_sprite.play ("Jump")
→ else:
→  → animated_sprite.play ("Idle")
```

(les "→" représentent une indentation, obtenue avec la touche TAB du clavier).

20. Lancez votre jeu, les animations de marche, de saut et d'attente devraient être jouées.
21. Passons aux explications sur vos lignes de code à partir de quelques exemples.

Exemple 1 :

```
onready var animated_sprite = get_node("AnimatedSprite")
```

- Cette ligne ajoute une variable (« var ») que le jeu va activer dès son lancement (« onready », ce qui signifie « déjà prêt »).
- Nous indiquons ensuite qu'une variable nommée « animated_sprite », lorsqu'elle sera présente dans le code, fera appel (*get*) au nœud (*node*) « AnimatedSprite », celui dans lequel nous avons créé nos animations plus tôt.

Exemple 2 :

```
→ if Input.is_action_pressed('ui_right') and is_on_floor():
```

- Dans cet exemple, la première ligne demande une condition (« if » = si)
- `Input.is_action_pressed('ui_right') =` signifie que la touche « flèche droite » (*ui_right*) du clavier doit être appuyée.
- « and » commande une 2e condition obligatoire qui est `is_on_floor()` (ce qui signifie « est sur le sol »). La première ligne demande donc 2 conditions simultanées.
- La ligne se termine par « : », ce signe fait que les effets de ses conditions sont définis dans les lignes indentées en dessous d'elle.

Exemple 3 :

```
→  → animated_sprite.play ("WalkRight")
```

- Le code appliquera au personnage l'animation « WalkRight », qui sera visible en jeu.

Exemple 4 :

```
→  → $AnimatedSprite.flip_h = true
```

- Ici les animations du personnage sont retournées (« flip ») comme devant un miroir.
- C'est avec ce type de ligne que l'on gère le retournement du personnage vers la gauche (rappelez-vous que vous n'avez dessiné que des animations tournées vers la droite dans l'*AnimatedSprite*).

Exemple 5 :

```
→ else:
→  → animated_sprite.play (“Idle”)
```

- « Else » signifie ici "autrement". Placée à la fin d'une liste de « if », elle indique donc ce qui se passera automatiquement si aucune condition notées plus haut dans le code n'est remplie.
- L'effet est de jouer sur le personnage l'animation « Idle »

22. Si vous souhaitez ajouter d'autres animations à cette liste, vous connaissez maintenant les codes et conditions nécessaires.
23. Bravo, vous avez terminé ce tutoriel et gagné un point d'expérience de **développeur** ET d'**animateur**!