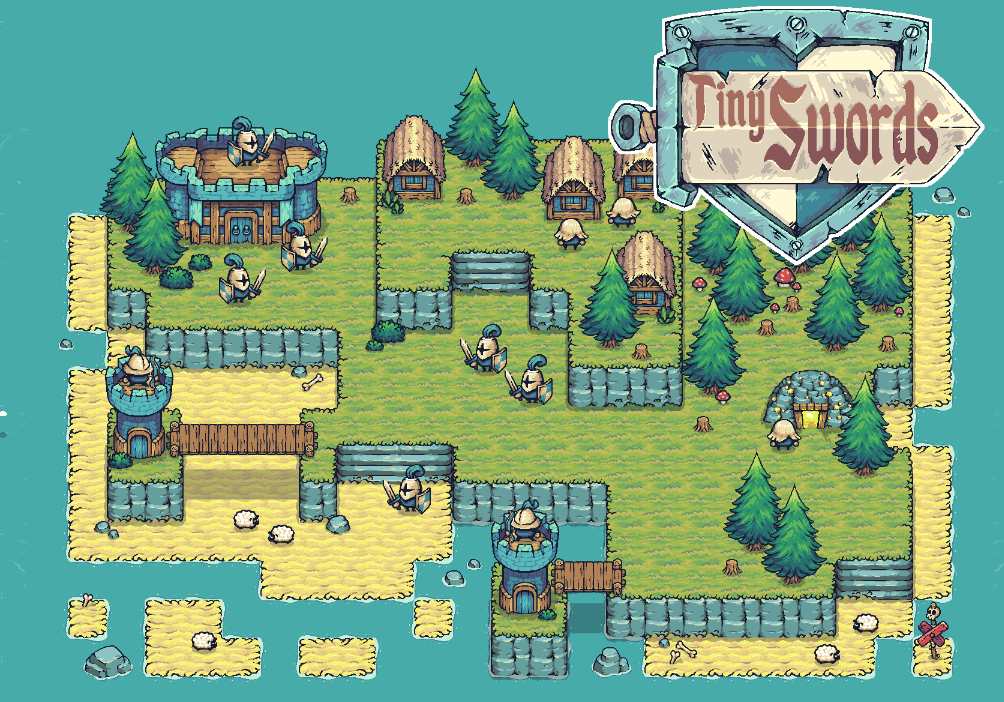
Vamos aprender usando o pacote Tiny Swords ([Espadas minúsculas por Pixel Frog (itch.io)](https://pixelfrog-assets.itch.io/tiny-swords))

O pacote é imenso, existe um mar de possibilidades!



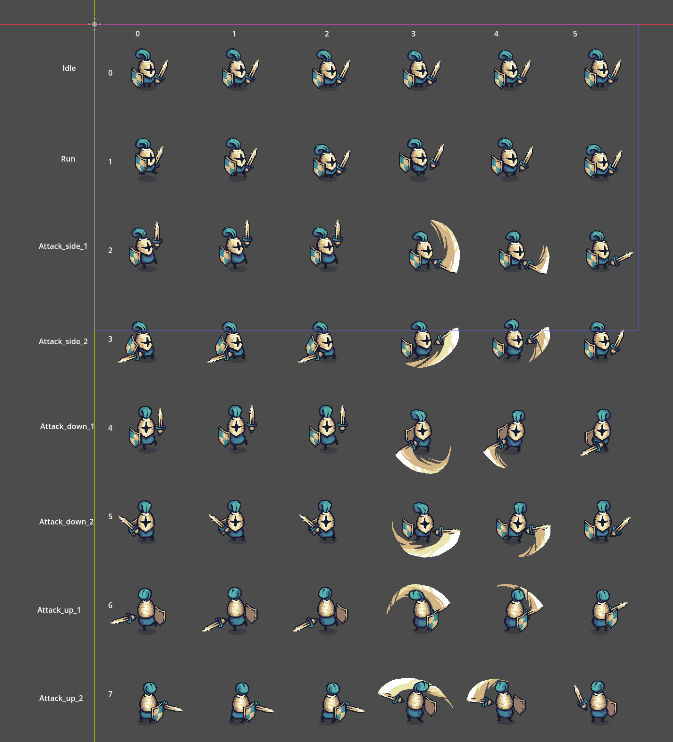
**Iremos primeiro criar o jogador**

Usando a pasta Factions para escolher. Iremos usar o **Warrior** como personagem.



Cada linha representa uma animação e cada animação é uma ação.

Se perceber não tem **nenhuma animação para a esquerda**, mas é só ir na propriedade offset> e marcar Flip H que ele gira.



Agora vamos dar passivamente vida:

Primeiro iremos criar um novo sprite2d e arrastrar a imagem para la. Agora iremos informar para o godot que é “animado”

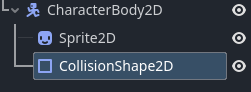
**Iremos em “Animation”:**

Iremos mexer no H e V frames. Como vimos anterior H temos 6 e V temos 8. Assim você vai ver que se transformar em 1 só.

E para mexer na animação que você quer você utiliza o **Frame coords.**



Vamos criar o characterbody2d:



Queremos colocar a origem do objeto seja os “pês” por conta da visão topdown. (movendo o sprite2d para ficar na origem dos pés do characterbody).

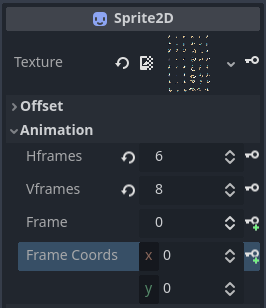
Iremos criar uma cena própria do player. Para personalizar ele sozinho.

**Agora vamos criar animações:**

Criaremos um novo node no player chamado: **AnimationPlayer**.

Nela iremos criar cada Animação separado. Primeiro iremos para a **idle**

Quando fazemos isso vai aparecer uma chave em Animation:

Assim vai clicando na chave que ele vai alterando.



Como saber se é para pular 0.1 por cada cena? Ir no assets e ver o recomendado.

Fazendo as cenas, podemos mexer nela por script. Criando um novo script.



**extends CharacterBody2D:** Esta linha indica que o script está estendendo a classe CharacterBody2D. Isso significa que este script herda todas as propriedades e métodos de CharacterBody2D, que é uma classe no Godot usada para personagens em 2D.

**@onready var animation\_player: AnimationPlayer = $AnimationPlayer:** Esta linha define uma variável animation\_player do tipo AnimationPlayer. O decorador @onready garante que esta variável seja inicializada apenas depois que o nó ao qual está anexado estiver pronto. O $AnimationPlayer busca um nó filho chamado AnimationPlayer e o atribui à variável.

**var is\_running: bool = false:** Define uma variável chamada is\_running do tipo booleano e a inicializa com false. Esta variável será usada para rastrear se o personagem está correndo ou não.

**func \_process(delta: float) -> void**: Define uma função chamada \_process que é chamada em cada frame. O parâmetro delta representa o tempo (em segundos) desde o último frame. O tipo de retorno é void, indicando que a função não retorna nada.

**if Input.is\_action\_just\_pressed("ui\_accept"):** Verifica se a ação "ui\_accept" (geralmente associada à tecla Enter ou Espaço) foi pressionada neste frame. Se a ação foi detectada, o código dentro do bloco if será executado.

**if is\_running:** Dentro do bloco if anterior, esta linha verifica se a variável is\_running é true.

**animation\_player.play("Idle"):** Se is\_running for true, o personagem estava correndo, então a animação Idle é iniciada usando o animation\_player.

**is\_running = false:** Define a variável is\_running para false, indicando que o personagem não está mais correndo.

**else:** Se is\_running for false, este bloco else será executado.

**animation\_player.play("run"):** Inicia a animação run usando o animation\_player, indicando que o personagem começou a correr.

**is\_running = true**: Define a variável is\_running para true, indicando que o personagem agora está correndo.