05

Fazer o zumbi vagar

Agora que estamos gerando a posição aleatórias, temos que restringir essas posições de serem calculadas o tempo todo. Para isso, podemos utilizar um contador.

Vamos então criar duas variáveis no script do Gerador que serão usadas no nosso contador: uma para de fato ser o tempo do contador, e outra para termos um padrão de quanto em quanto tempo queremos recalcular nossa posição aleatória.

```
private float contadorVagar;
private float tempoEntrePosicoesAleatorias = 4;
```

Agora temos que contar o tempo e limitar a aleatorização de uma nova posição para acontecer somente quando o contador zerar (ao invés de acontecer o tempo todo).

Lembre-se: depois de aleatorizar uma posição, você precisa aumentar novamente o tempo do contador para ele não voltar a acontecer o tempo todo!

```
Vagar()
{
    contadorVagar -= Time.deltaTime;

    if (contadorVagar <= 0)
    {
        posicaoAleatoria = AleatorizarPosicao();
        contadorVagar += tempoEntrePosicoesAleatorias;
    }

    direcao = posicaoAleatoria - transform.position;
    movimentaInimigo.Movimentar(direcao, statusInimigo.Velocidade);
}
```

Agora o nosso personagem vai até a posição, mas quando chega lá ele fica travando. Para resolver isto temos dois passos: o primeiro é parar o nosso zumbi quando ele chegar e o segundo é ele parar também na animação.

Vamos começar pelo primeiro. Temos que calcular a distância entre o zumbi a posição que geramos aleatoriamente, e verificar se elas estão próximas para parar o zumbi.

Nem sempre é sábio testar para ver se as posições são exatamente iguais, mas sim se uma está bem próxima da outra, pois valores exatamente iguais neste caso serão bem difíceis. Então vamos testar a distância entre os dois é bem pequena.

Para quebrar melhor o problema, podemos salvar este resultado num variável `bool` e testa-la em seguida.

```
Vagar()
{
    contadorVagar -= Time.deltaTime;
```

```
    if(contadorVagar <= 0)
    {
        posicaoAleatoria = AleatorizarPosicao();
        contadorVagar += tempoEntrePosicoesAleatorias.
    }
    bool ficouPertoOSuficiente = Vector3.Distance(transform.position, posicaoAleatorias) <= 0.0

    direcao = posicaoAleatoria - transform.position;
    movimentaInimigo.Movimentar(direcao, statusInimigo.Velocidade);
}
```

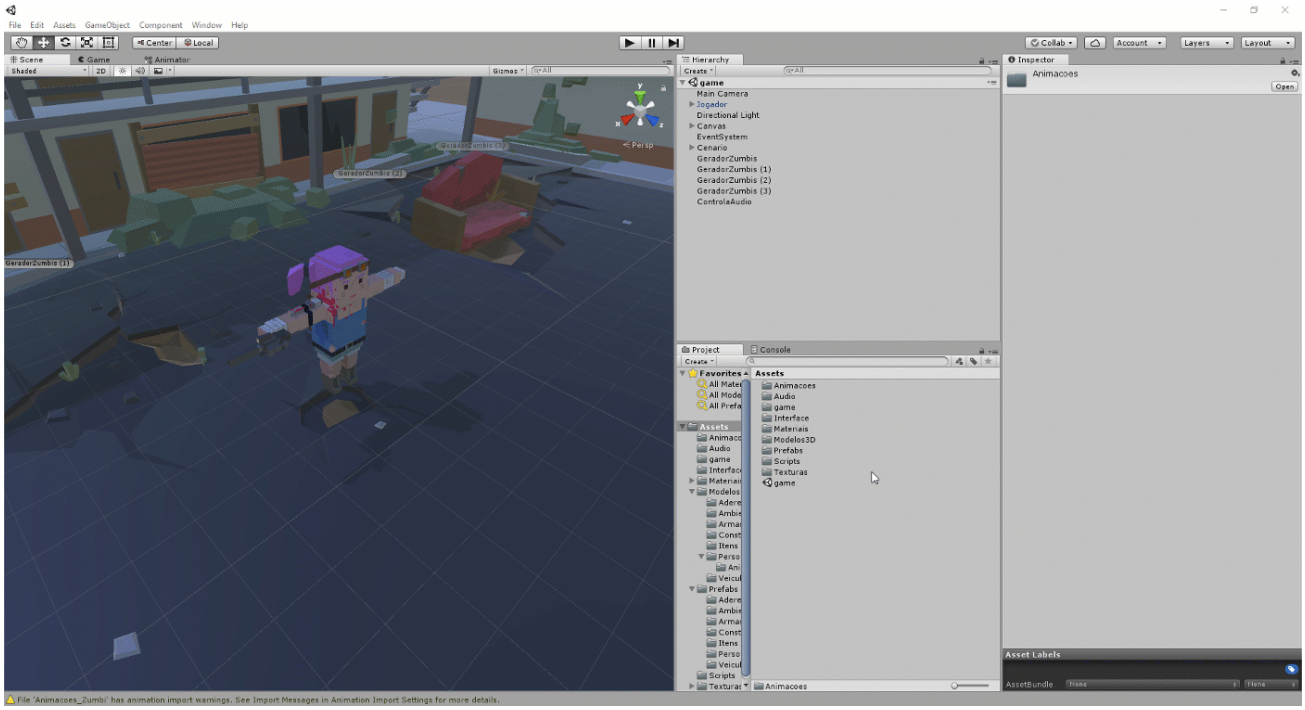
Agora é só testar esse `bool` num `if` para saber se ele é verdadeiro e se nosso personagem chegou na posição.

```
Vagar()
{
    contadorVagar -= Time.deltaTime;

    if(contadorVagar <= 0)
    {
        posicaoAleatoria = AleatorizarPosicao();
        contadorVagar += tempoEntrePosicoesAleatorias.
    }
    bool ficouPertoOSuficiente = Vector3.Distance(transform.position, posicaoAleatorias) <= 0.0

    if(ficouPertoOSuficiente == false)
    {
        direcao = posicaoAleatoria - transform.position;
        movimentaInimigo.Movimentar(direcao, statusInimigo.Velocidade);
    }
}
```

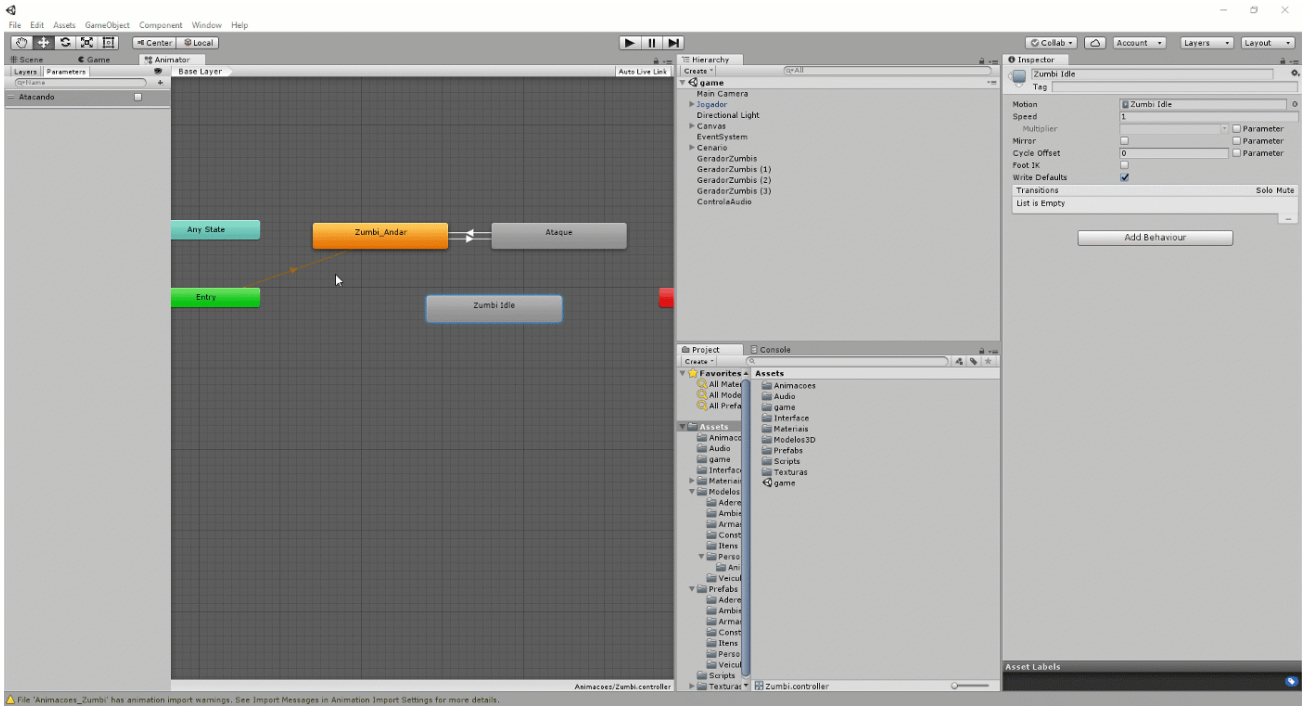
Para solucionar a segunda parte, temos que fazer algumas mudanças no Animator do nosso inimigo; bem parecido com o que fizemos no personagem.

Vamos jogar a animação de Idle do zumbi para o nosso Animator do Zumbi.

Lembre-se de que se você ainda não tiver feito o corte nessa animação, basta cortá-la de acordo com os quadros "arquivo de texto" na pasta das animações



Crie um parâmetro chamado **Movendo** e vamos fazer a transição através deste novo parâmetro, desmarcando o **Has Exit Time** e na condição e testando se é maior ou menor que `0.2` .



Agora é só ir no `FixedUpdate` do nosso inimigo, e incluir a linha que faz ele animar pelo movimento, já que já temos o código de animação nele:

```
animacaoInimigo.Movimentar(direcao.magnitude); .
```