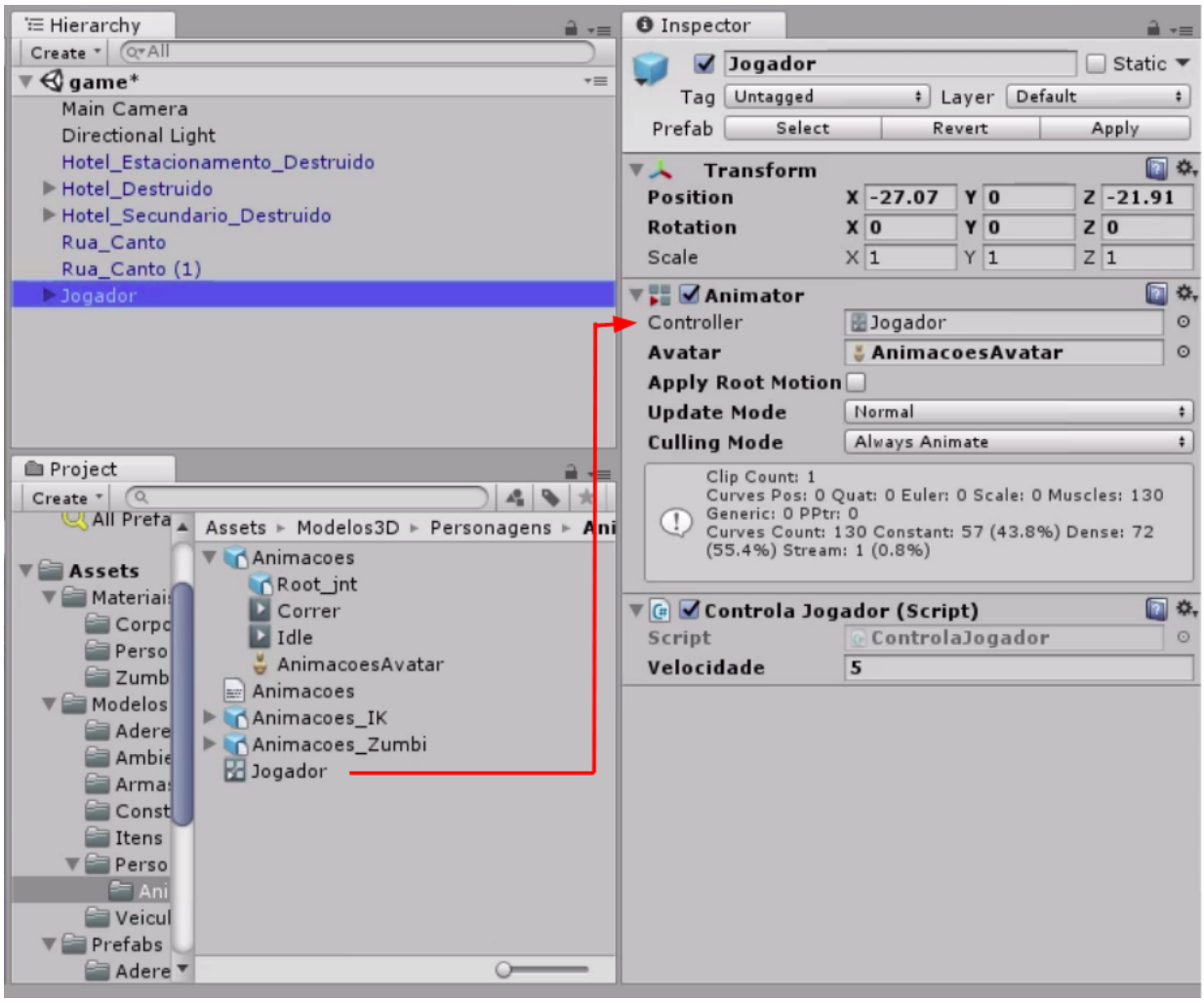
04

Sistema de animações Mecanim

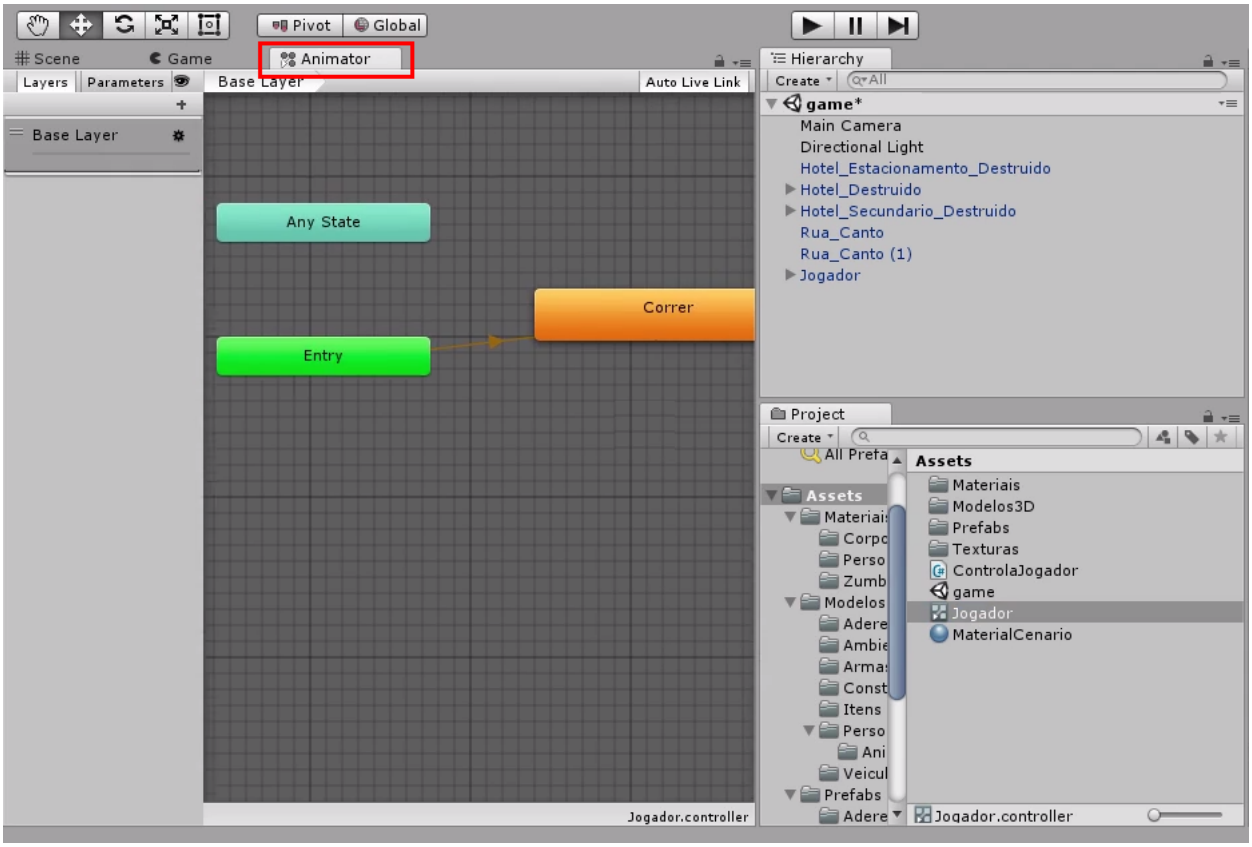
Transcrição

Anteriormente, adicionamos animações à personagem. No entanto, a animação de "Correr" continua rodando, mesmo quando ela está parada. Veremos como fazê-la parar quando nenhum *input* estiver acionado.

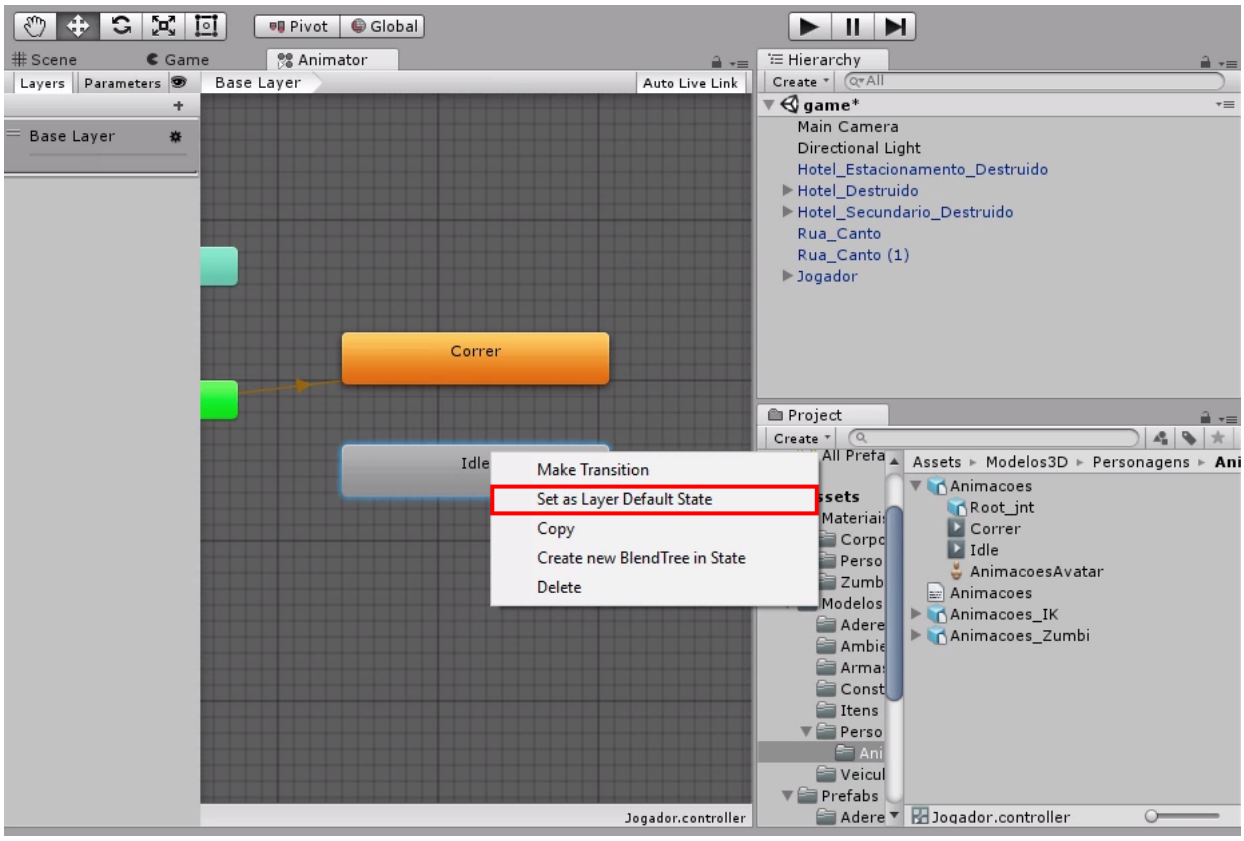
Se clicarmos em "Jogador", em "Hierarchy", notem que "Controller" está preenchido com "Jogador", em "Animator".



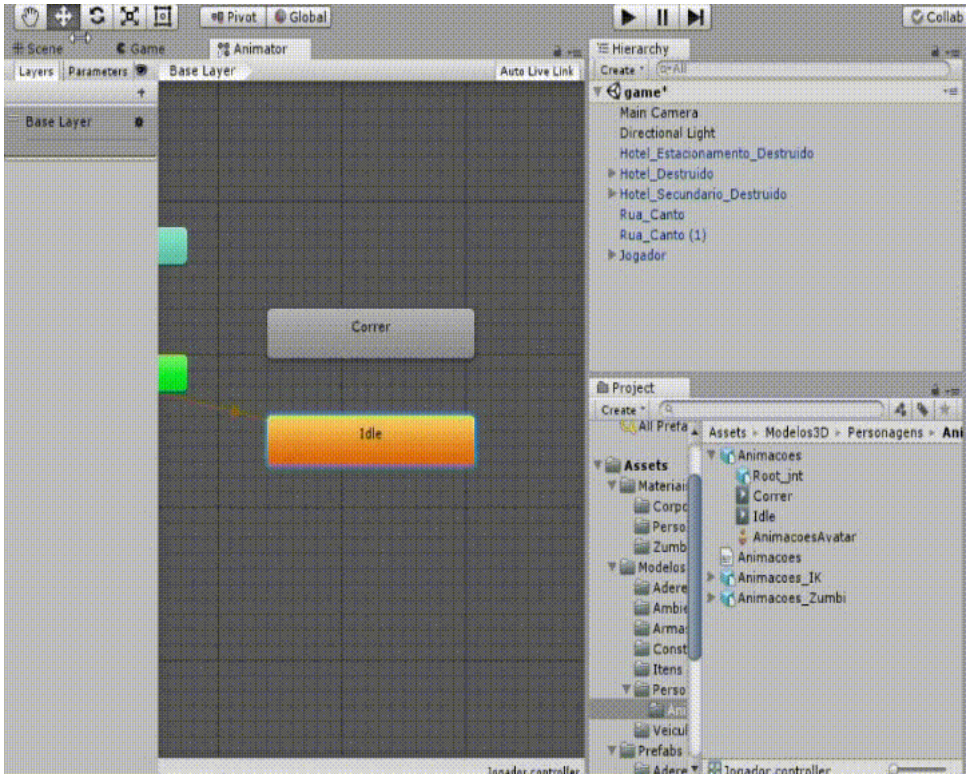
Esse "Jogador" está em "Project > Assets > Modelos3D > Personagens > Animacoes". Inclusive, arrastaremos para "Assets" para que fique mais fácil localizá-lo. Clicaremos duas vezes nele e, na janela de "Scene", abrirá uma nova aba "Animator" para tratar as animações, na qual "Correr" é a única.



Acrescentaremos `Idle` em "Animator", arrastando-a de "Animacoes". Assim, **administraremos a troca** de uma animação para outra. Na sequência, clicaremos com o botão direito do mouse em cima de `Idle` e selecionaremos a opção "Set as Layer Default State", para defini-la como estado inicial, identificado com a cor laranja.



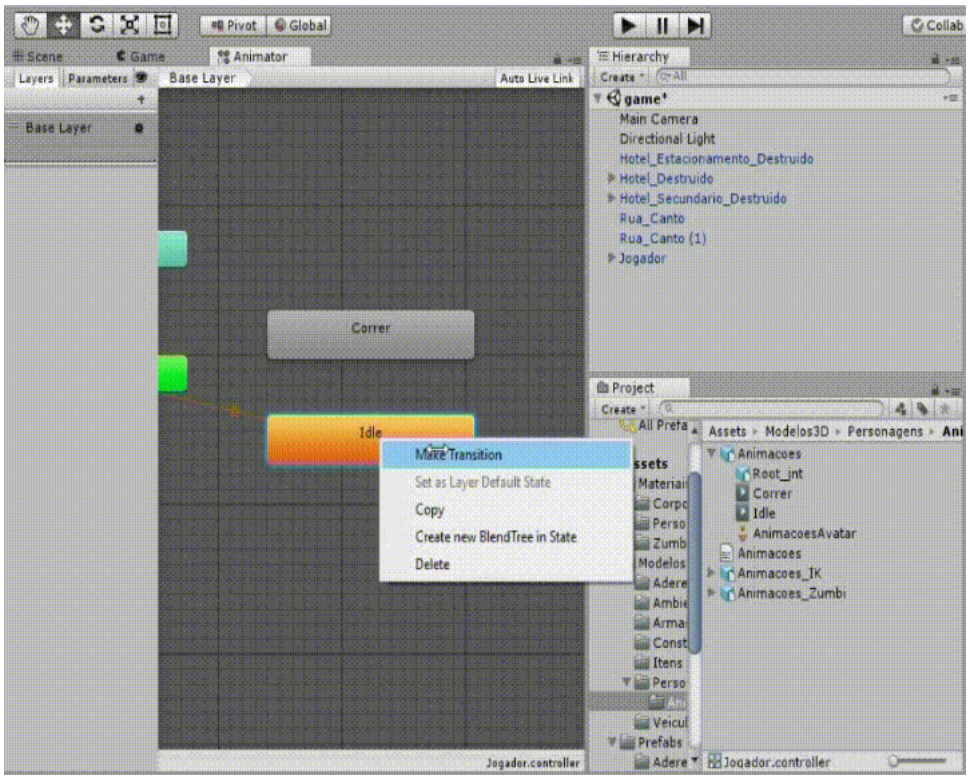
Aplicada a alteração, `Correr` que estava laranja, ficou cinza e `Idle` que estava cinza, ficou laranja, pois agora é o estado padrão. O objetivo é que, ao ativarmos "Play", a personagem fique inicialmente parada, respirando, e que ela se movimente, no momento que pressionarmos as teclas. Se olharmos a janela "Game", com "Play" ativado, veremos que quando pressionamos as teclas de movimento, ela continua só respirando.



Para que as animações sejam ativadas de acordo com o pressionar das teclas:

- voltaremos a "Animator";
- clicaremos com o botão direito em cima de `Idle` ;
- selecionaremos a opção "Make Transition", para fazer a **transição** de uma animação para outra;
- direcionaremos a seta que aparecerá em `Idle` para `Correr` .

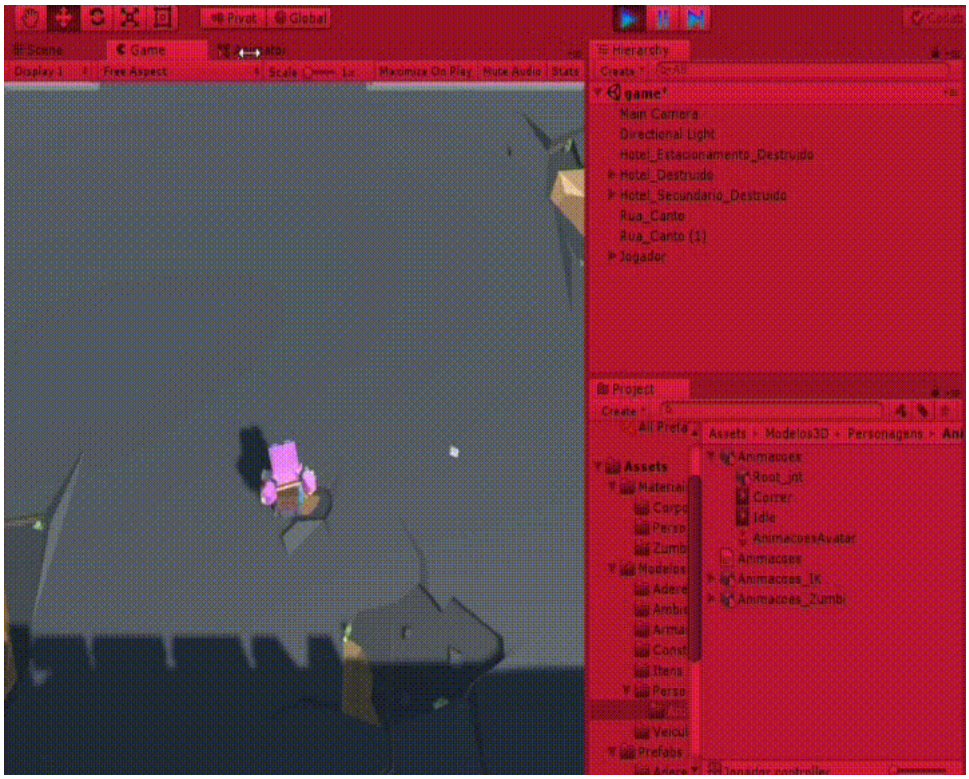
Se ativarmos o "Play", veremos que a personagem está inicialmente parada e, em seguida, passou a correr. Da mesma forma que fizemos com `Idle` , aplicaremos em `Correr` a transição para `Idle` . Ao ligarmos o "Play",veremos que a transição está funcionando.



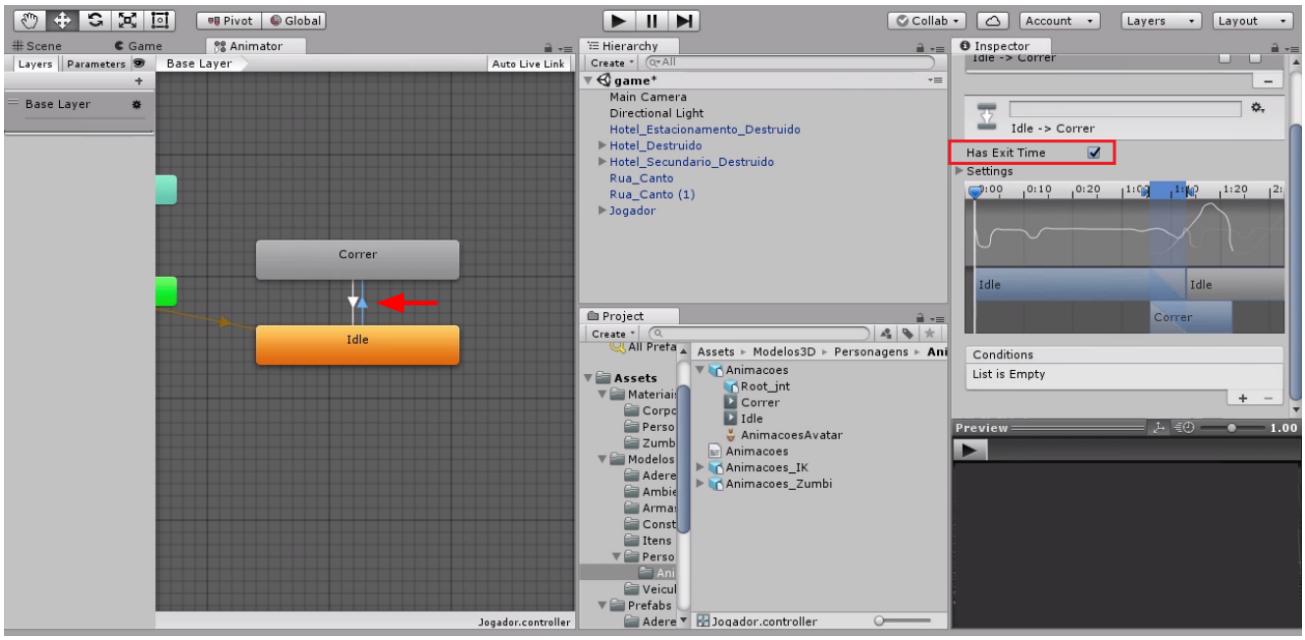
Para visualizar como funciona a transição, podemos:

- arrastar a janela "Animator" para baixo de "Game";
- clicar em "Jogador" em "Hierarchy".

Veremos as trocas em "Animator". Quando a animação `Idle` termina, `Correr` é inciada e vice-versa.



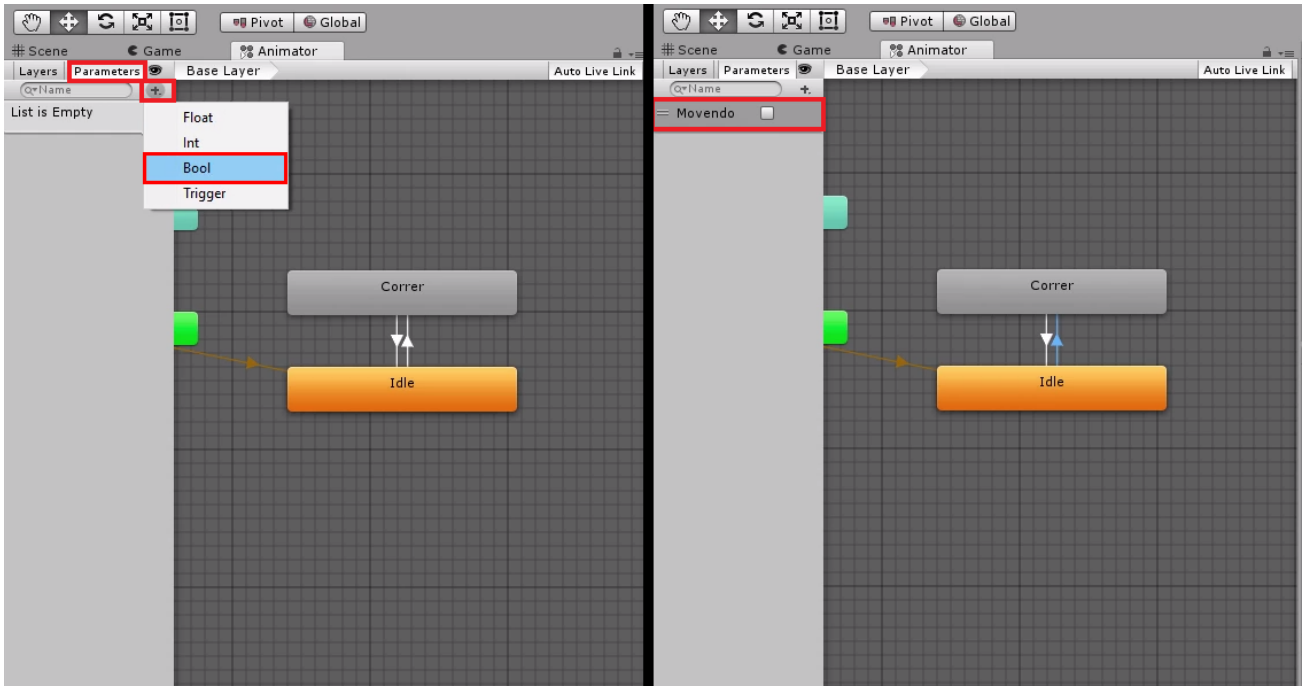
Não é bem o que queremos. Voltaremos a janela "Animator" à posição inicial, à direita de "Game" e estabeleceremos o nosso **objetivo** que é a **transição de movimentos** de acordo com o **pressionar das teclas**. Para isso, clicaremos na seta que vai de `Idle` para `Correr` . Assim, veremos as propriedades da troca, em "Inspector".



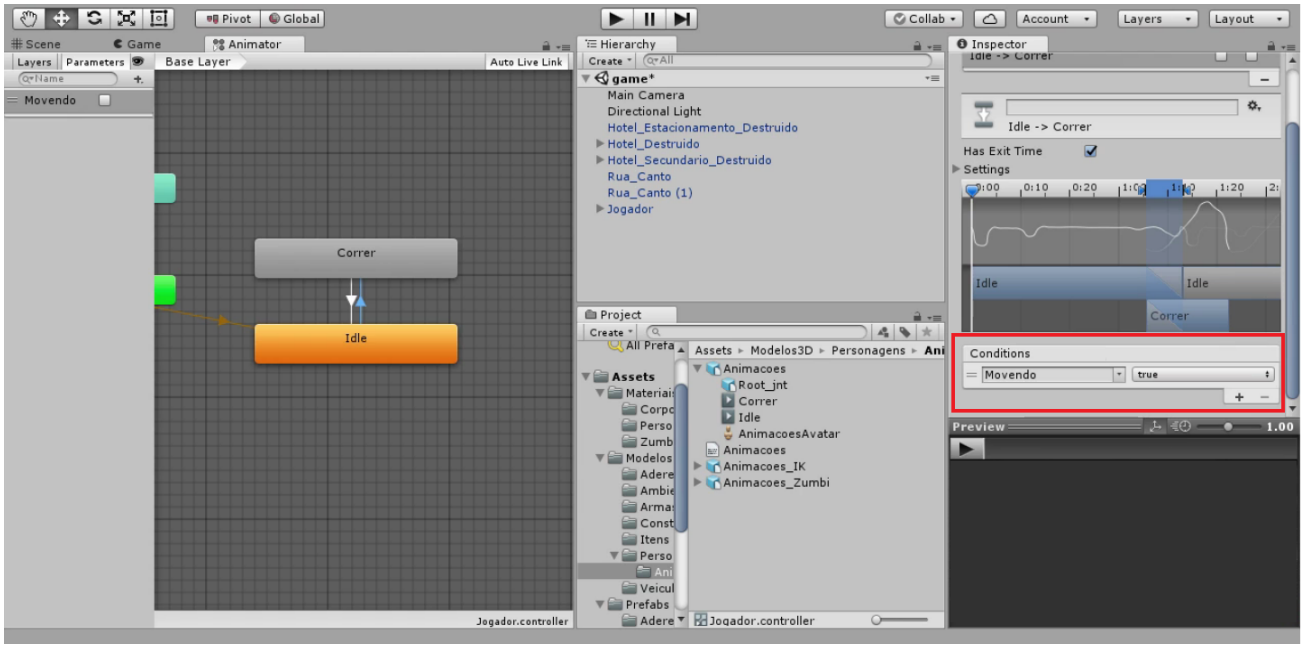
No momento, a troca é feita por meio da propriedade "Exit Time", ou seja, quando `Idle` termina, `Correr` é iniciada e vice-versa. Para que a transição ocorra ao pressionar as teclas, em "Animator":

- clicaremos na aba "Parameters" para criar um **parâmetro**, que são como as variáveis, de troca entre as animações;

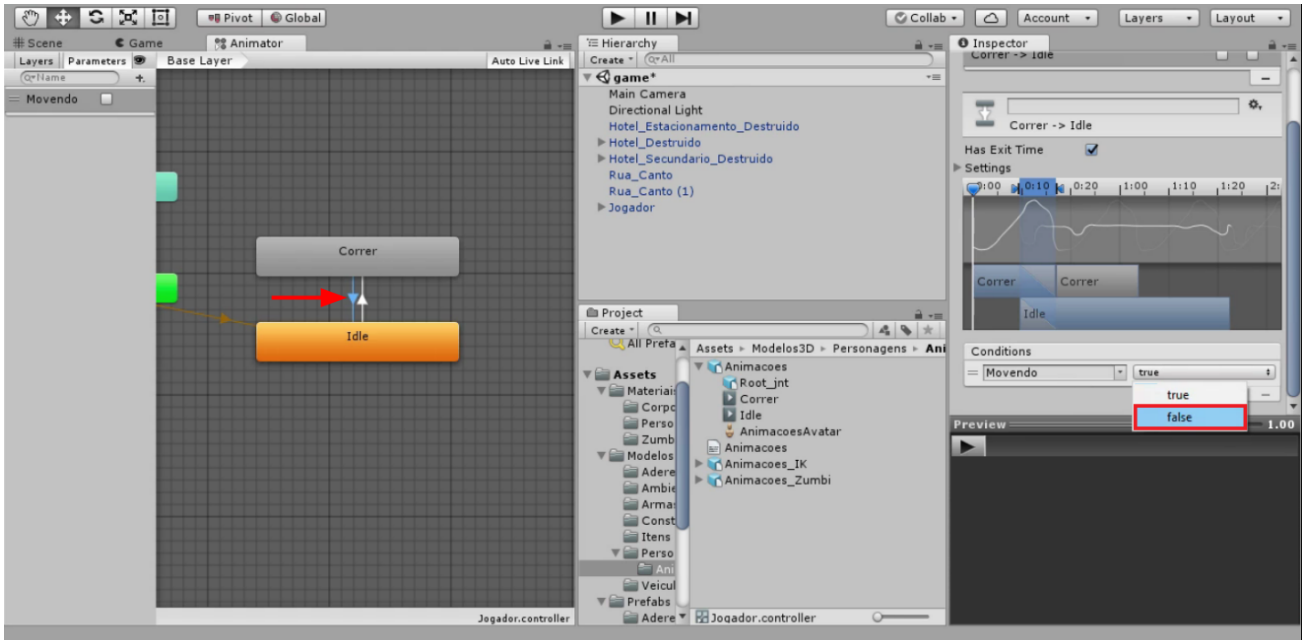
- clicaremos no sinal de soma (+) e nas opções veremos "Float", variável que utilizamos anteriormente;
- selecionaremos a opção "Boot", variável lógica que contém dois valores:
  - verdadeiro: correr;
  - falso: ocioso;
- nomearemos o parâmetro, substituindo "New Boot" por "Movendo";
- marcaremos a caixa de seleção que aparecerá à direita de "Movendo" quando a personagem estiver correndo (verdadeiro) e desmarcaremos quando estiver ociosa (falso).



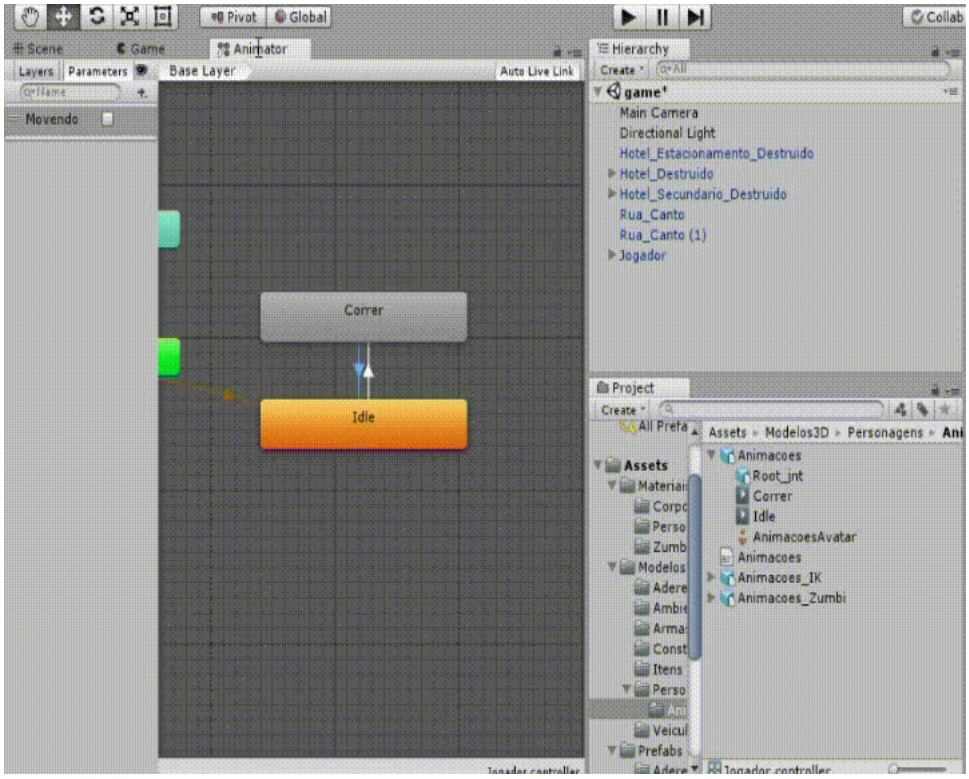
Clicaremos na seta que vai de `Idle` para `Correr`, e em "Inspector, iremos a "Conditions" para escolher uma condição para que a transição entre as animações ocorra, clicando no sinal de soma (+).



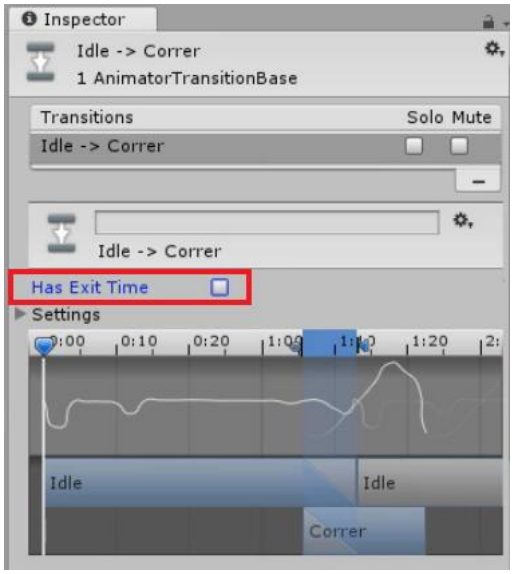
Para aplicar a transição de `Correr` para `Idle`, clicaremos na outra seta e, em "Inspector", estabeleceremos a condição ("Conditions") de que "Movendo" é "false", ou seja, quando a personagem não estiver correndo.



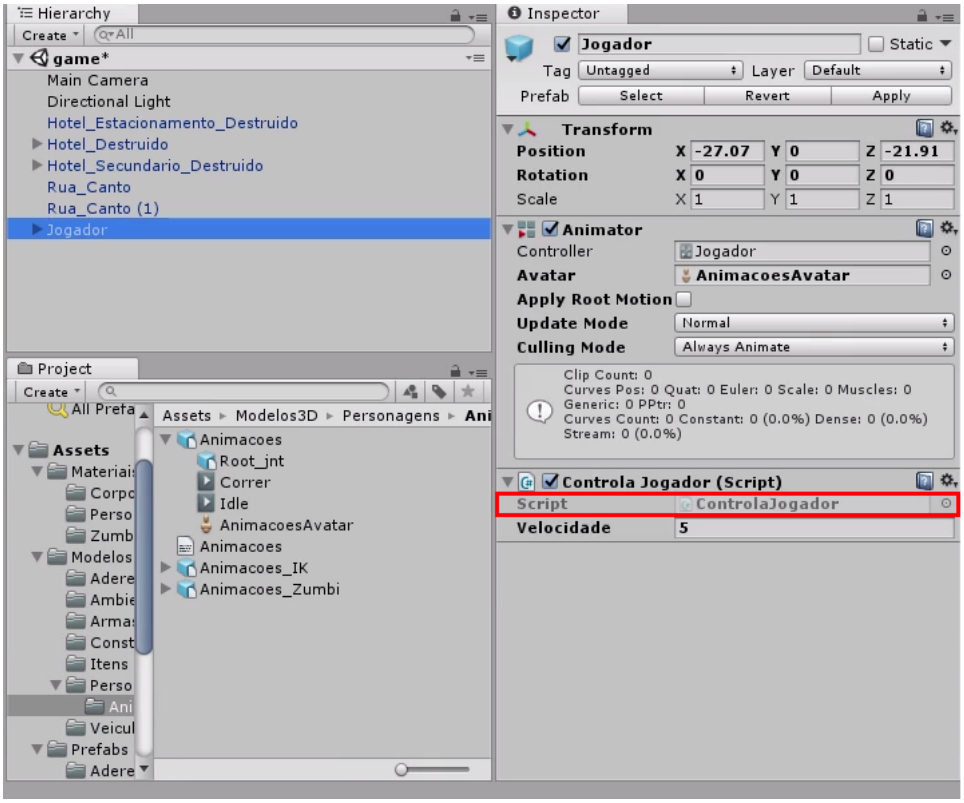
Arrastaremos a janela "Animator" para baixo de "Game", novamente, e ligaremos o "Play". Notem que enquanto a personagem está parada, somente a barra de `Idle` é carregada. Se marcarmos a caixa de seleção de "Movendo", o carregamento de `Idle` será interrompido e o de `Correr` iniciará, segundos depois. Se desmarcarmos a caixa de seleção, `Idle` voltará a carregar e `Correr` pausará.



Funcionou. Ao marcar "Movendo", a personagem se movimenta e ao desmarcar, ela para. Voltaremos a janela "Animator" à posição inicial, à direita de "Game", para editar o tempo de transição entre as animações, que está um pouco lento em função de "Exit Time", que está ativado. Assim, para que uma animação comece, a outra precisa terminar, mesmo que "Movendo" seja verdadeiro. Desmarcaremos a caixa de "Exit Time" nas propriedades de `Idle` e `Correr`.



Pronto, temos uma forma de trocar de uma animação para outra por meio do parâmetro "Movendo". Veremos como utilizar essas aplicações no *script*, pois é no código que estabelecemos os movimentos da personagem de acordo com as teclas do teclado. Dessa forma, o jogo ficará coerente, pois não faz sentido precisarmos marcar ou desmarcar a caixa de seleção de "Movendo" toda vez que quisermos fazer a transição de uma animação para outra. Abriremos o código que está em "Hierarchy > Jogador > ControlaJogador"



Dentro de `Update`, acrescentaremos um trecho para determinar que:

- quando a personagem se movimentar, além de transladar (`Translate`), rode a animação;
- quando nenhuma tecla estiver pressionada, a personagem pare.

Para isso, no código, utilizaremos a as expressões **condicionais**, que testem se o código está correto ou não, por meio de:

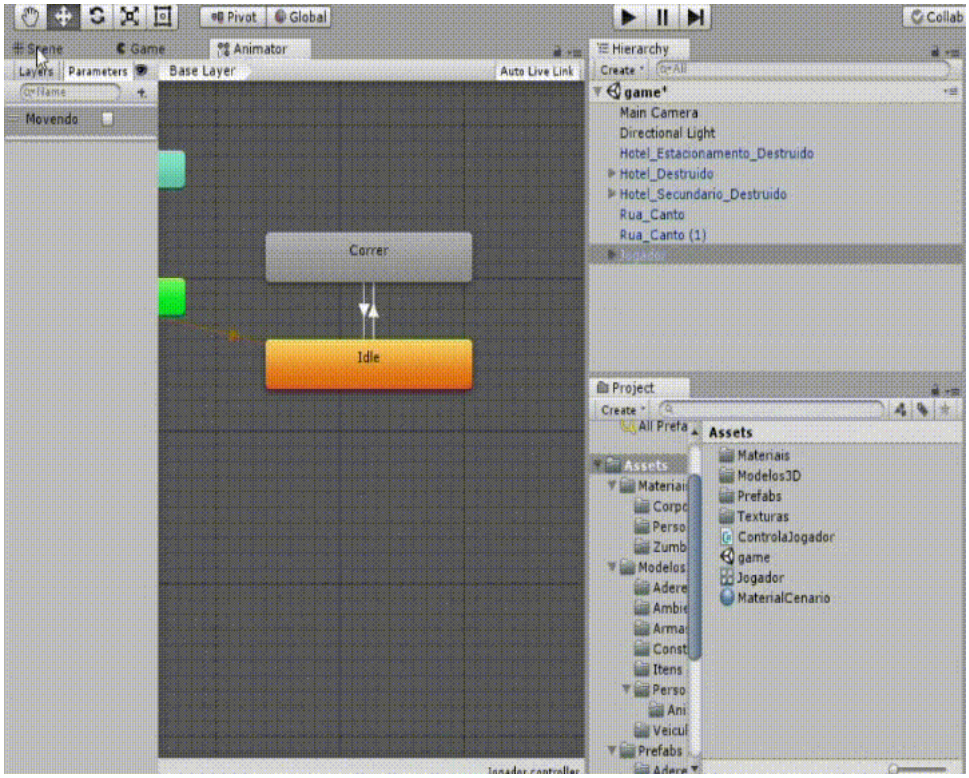
- `if` , estabeleceremos que **se** a personagem estiver se movendo, a animação será executada;
- `direcao` , indicaremos se há movimento ou não, pois quando o `eixoX` ou `eixoZ` for diferente (`1` , `-1`) de `0` , haverá movimento;

- `!=` , equivalente a "não igual", ou seja, se `direcao` for **diferente** de `0` ,
- `Vector3.zero` , expressão da Unity para atribuir `0` como valor para os três eixos (X, Y e Z);
- chaves (`{}`) para inserir entre elas, que se `direcao` for diferente de `zero` , a personagem correrá, ou seja, "Movendo" estará ativado;
- `GetComponent<Animator>` para pegar o componente "Animator", considerando que cada parte de "Inspector" é chamada de "Component";
- parênteses (`()`) para adicionar itens a `GetComponent<>` ;
- `SetBool` , após os parênteses;
  - `Set` para marcar, atribuir valor;
  - `Bool` de variável tipo "Bool", de Boolean, que distingue verdadeiro e falso;
- parênteses após `SetBool` ;
- entre parênteses e aspas (`" "`), `Movendo, true` ;
  - `Movendo` , referente ao parâmetro que adicionamos em "Animator";
  - `true` , valor verdadeiro para quando "Movendo" estiver ativado.

O código ficará da seguinte forma:

```
public class ControlaJogador : MonoBehaviour {

    public float Velocidade = 10;

    // Update is called once per frame
    void Update () {

        float eixoX = Input.GetAxis("Horizontal");
        float eixoZ = Input.GetAxis("Vertical");

        Vector3 direcao = new Vector3(eixoX, 0, eixoZ);

        transform.Translate(direcao * Velocidade * Time.deltaTime);

        if(direcao != Vector3.zero)
        {
            GetComponent<Animator>().SetBool("Movendo", true);
        }

    }
}
```

Salvaremos as alterações por meio do atalho "Ctrl + S", voltaremos para "Game" e ligaremos o "Play". Inicialmente, a personagem está parada. Ao teclar as setas, ela se movimenta, mas quando soltamos, ela continua se movimentando.



Para fazê-la parar de se movimentar ao soltarmos as teclas, precisamos mexer na estrutura do código, que testa se a personagem está se movendo ou não. Se `direcao` for diferente de `zero`, a personagem deve se movimentar. Para quando ela estiver parada, utilizaremos `else`:

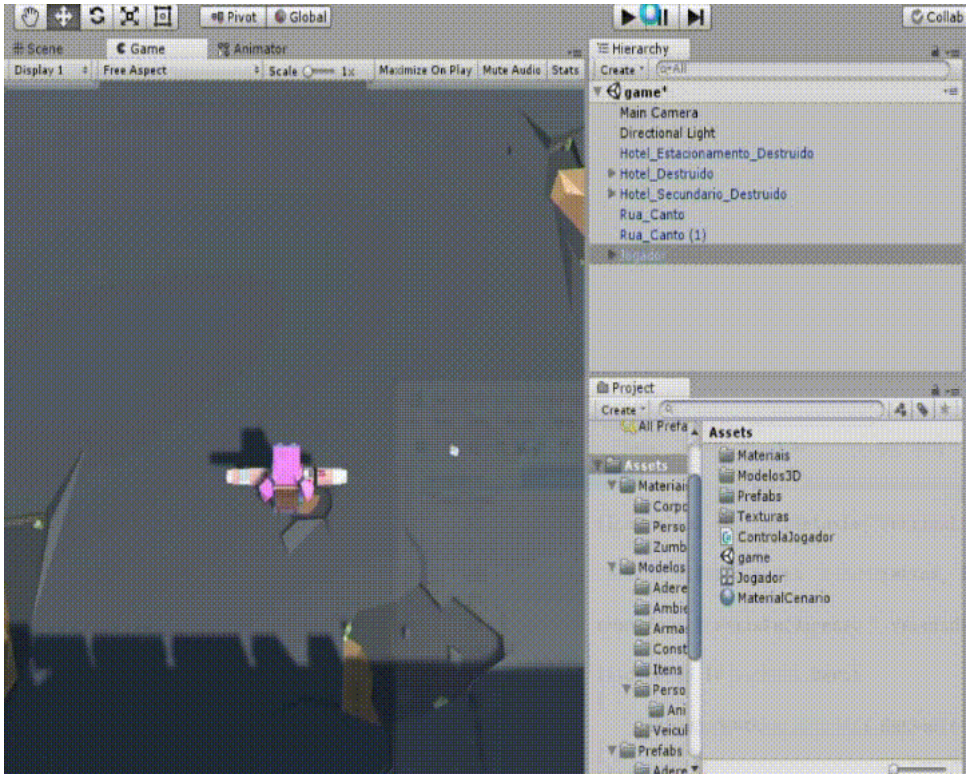
- abriremos chaves (`{}`);
- acrescentaremos entre elas `GetComponent<Animator>().SetBool("Movendo", false);`.

Dessa forma, definimos o que deve acontecer quando nenhuma tecla estiver pressionada. O `else` funciona como "se não", ou seja, se o `if` não acontecer, `else` será executado. Então, se `direcao` for diferente de `zero` e a personagem estiver se movimentando, o trecho de `GetComponent<>` que contém `("Movendo", true)` será executado. Se não, o que contém `("Movendo", false)` será. Salvaremos as alterações no código, que ficará da seguinte forma:

```
public class ControlaJogador : MonoBehaviour {

    public float Velocidade = 10;

    // Update is called once per frame
    void Update () {

        float eixoX = Input.GetAxis("Horizontal");
        float eixoZ = Input.GetAxis("Vertical");

        Vector3 direcao = new Vector3(eixoX, 0, eixoZ);

        transform.Translate(direcao * Velocidade * Time.deltaTime);

        if(direcao != Vector3.zero)
        {
            GetComponent<Animator>().SetBool("Movendo", true);
        }
        else
        {
            GetComponent<Animator>().SetBool("Movendo", false);
```

```
        }

    }
}
```

Ao ligarmos o "Play", em "Game", veremos a personagem parada, inicialmente. Ao teclarmos as teclas de movimento, ela se moverá e ao soltá-las, a personagem para.



O código e o jogo estão ficando bem mais dinâmicos, com as condicionais `if` e `else`.