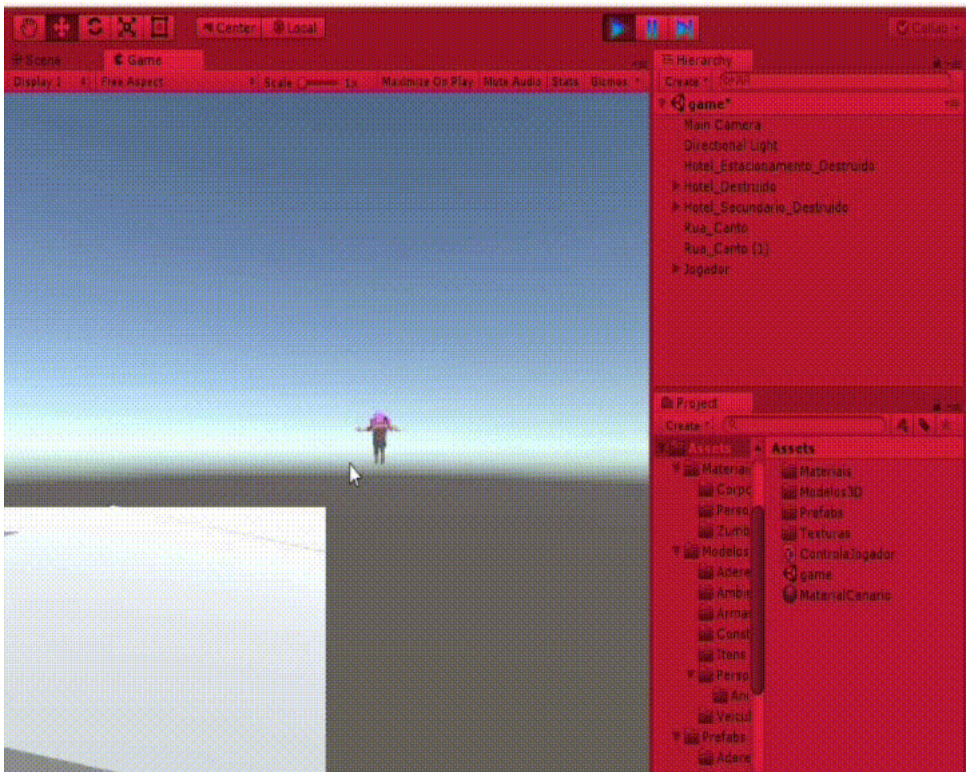
17

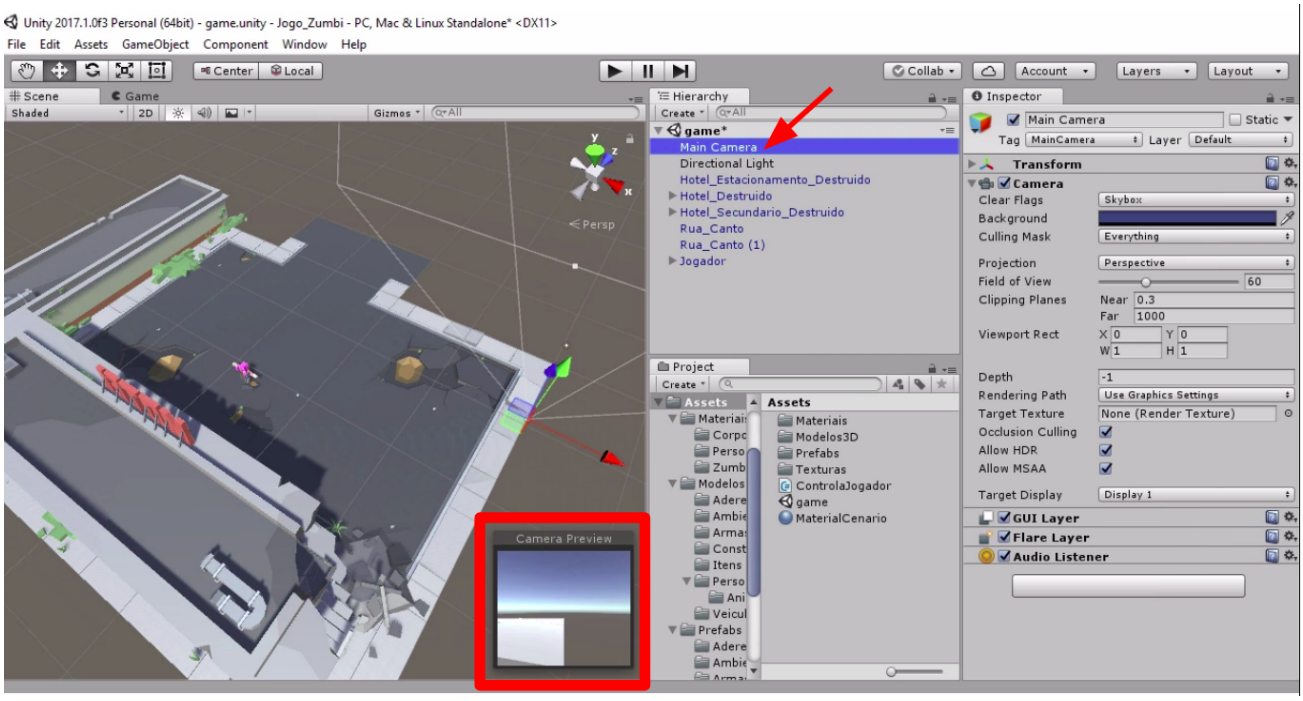
Movimentação por segundo

Transcrição

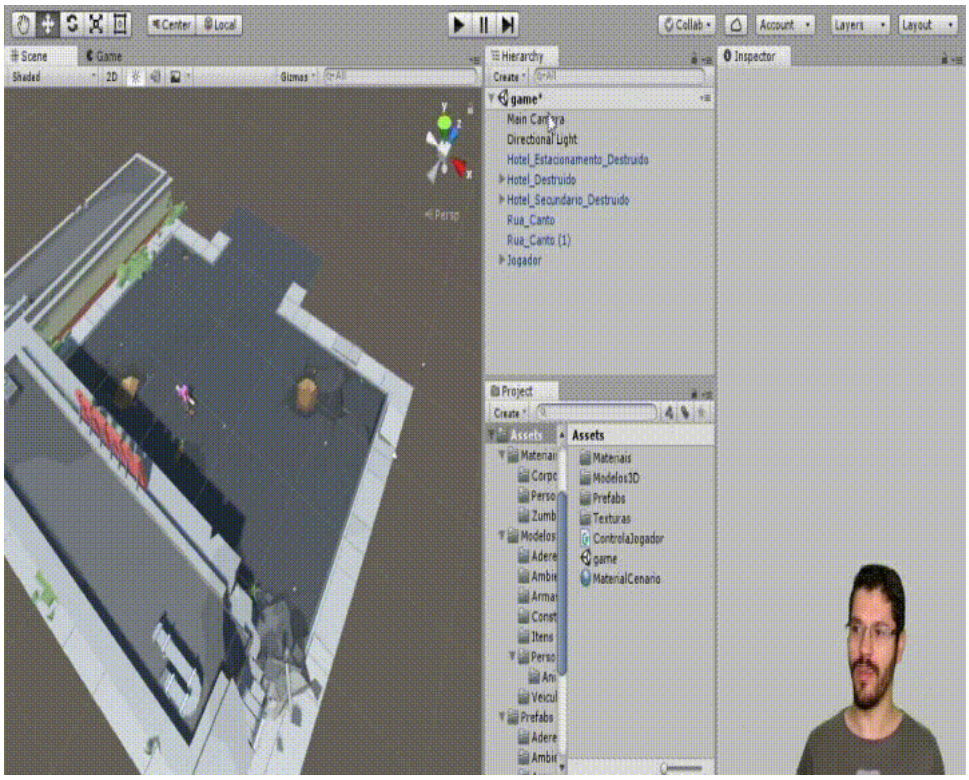
Ao clicarmos em "Play", notem que é difícil localizar a personagem na janela "Game", que funciona como amostra do funcionamento do jogo e, no canto inferior esquerdo, está a borda do cenário. Se mudarmos para a aba "Scene", a visualização é muito melhor.



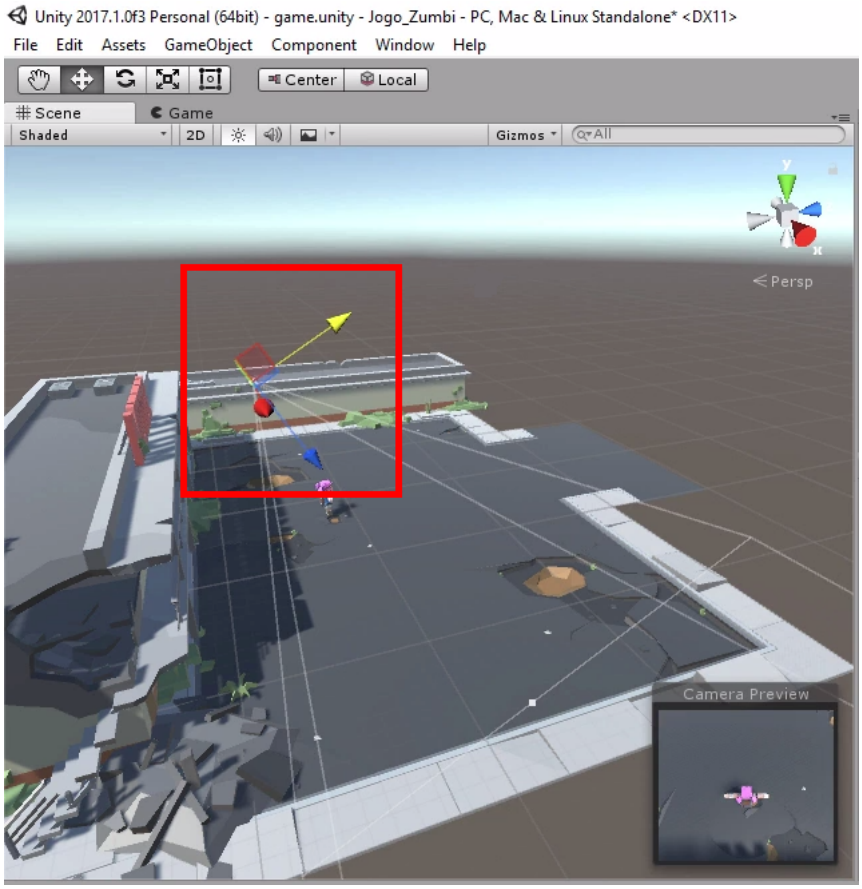
A janela "Game" funciona por meio de uma câmera que visualiza a cena e a transforma em jogo. Em "Hierarchy", essa câmera é um dos primeiros itens da lista ("Main Camera"), que vem como um padrão quando começamos a desenvolver "Scene".



Se clicarmos nela, veremos que podemos movê-la e que, no canto inferior direito da janela "Scene", há uma pré visualização para ver o que ela está capturando. Movimentaremos, levando-a para o centro da cena, próximo a personagem. Editaremos "Rotation X", em "Transform", substituindo `0` por `60`, dessa forma, veremos o jogo de cima.

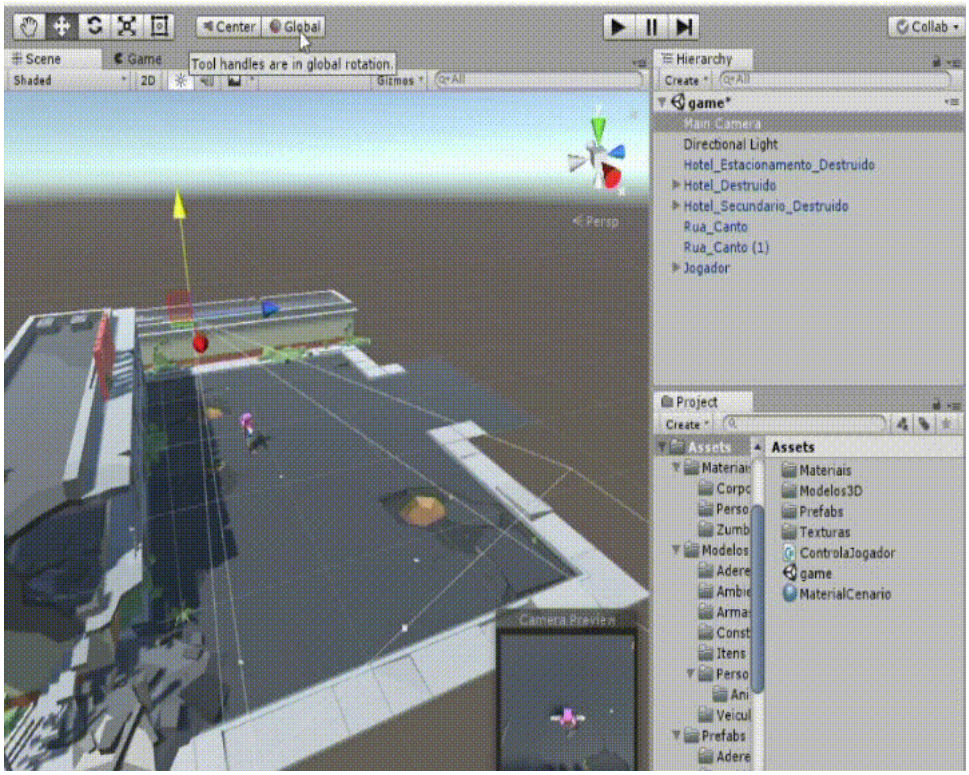


Reparem que ao aplicar essa substituição, os eixos de movimentação, que ativamos por meio da tecla "W", trocaram de posição, pois ao rotacionarmos a câmera, o eixo foi trocado para o de rotação.



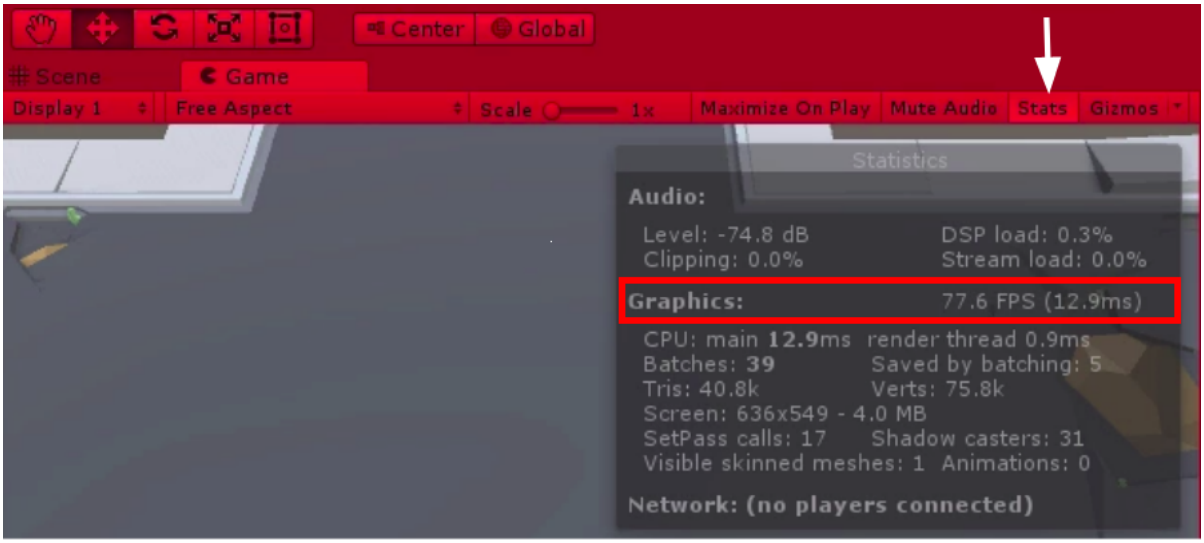
Então, o eixo Y que estava para cima, após a rotação, ficou na diagonal. Para ajustar, clicaremos no botão "Local" para que fique no modo "Global", na barra superior da Unity.

Após ajustar a câmera, ativaremos "Play" e veremos que a visualização melhorou bastante.



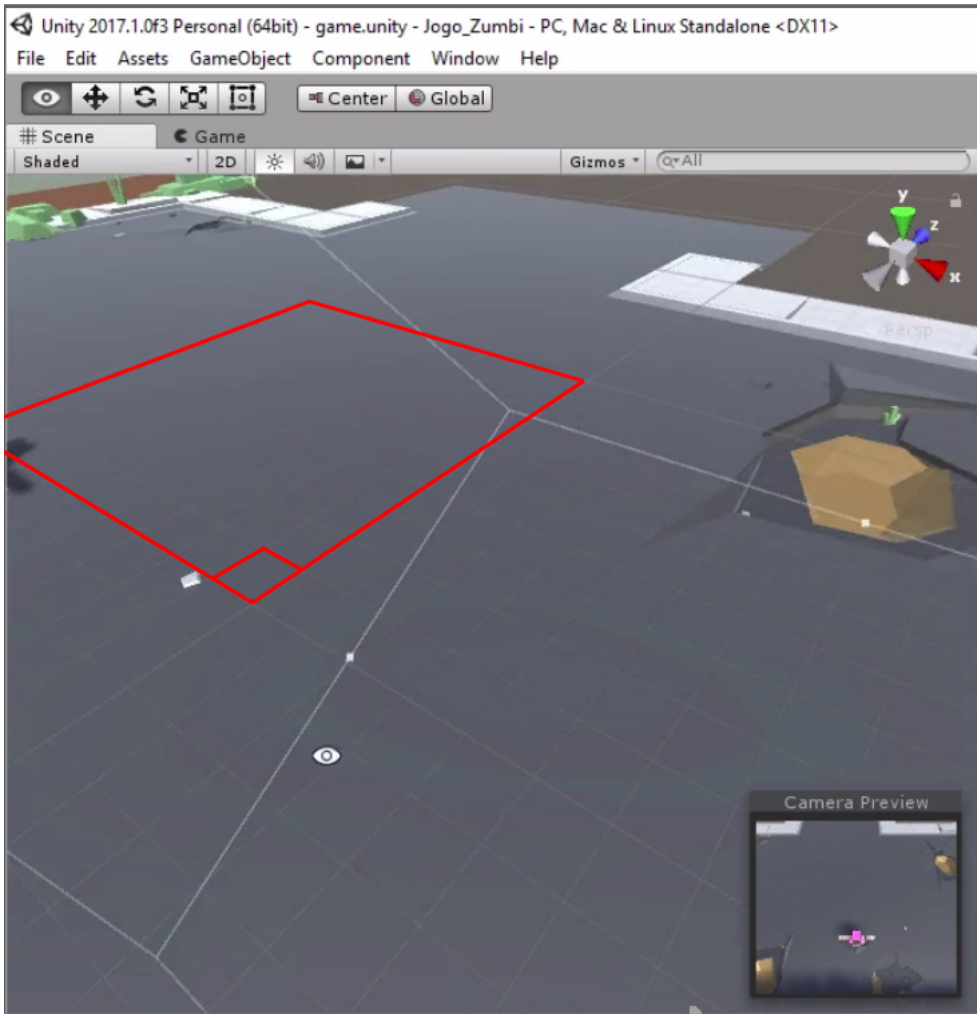
Porém, testando os movimentos, vemos que a personagem está deslizando muito rápido. Como em filmes, o jogo é uma sequência de imagens, que dão a impressão de movimento. Mas, em jogos, o número de imagens rodadas por segundo é muito maior.

Com "Play" ativado, se clicarmos em "Status", localizado na barra superior da janela "Game", veremos o número de imagens rodadas por segundo, que é de, aproximadamente, 80 FPS ("Frames Per Second" ou " Quadros Por Segundo" em português).



Ajustaremos a velocidade de movimento no *script*. No código, o trecho de `Update` roda uma vez para cada imagem, ou seja, se estamos rodando aproximadamente 80 imagens por segundo, o `Update` está rodando 80 vezes. E, se especificamos em `Input` que o jogador pode mover a personagem `1` para esquerda ou para a direita e `1` para frente

ou para trás, então o `1` é movido 80 vezes em 1 segundo. Considerando que `Update` roda 80 vezes em 1 segundo, a personagem está andando 80 quadradinhos da cena.



É uma distância muito grande, pois cada quadrado grande da Unity é formado por 10 pequenos. Ou seja, 80 desses equivalem a 8 quadrados grandes, perímetro que vai além da extensão do estacionamento. Então, em 1 segundo conseguimos andar o estacionamento inteiro.

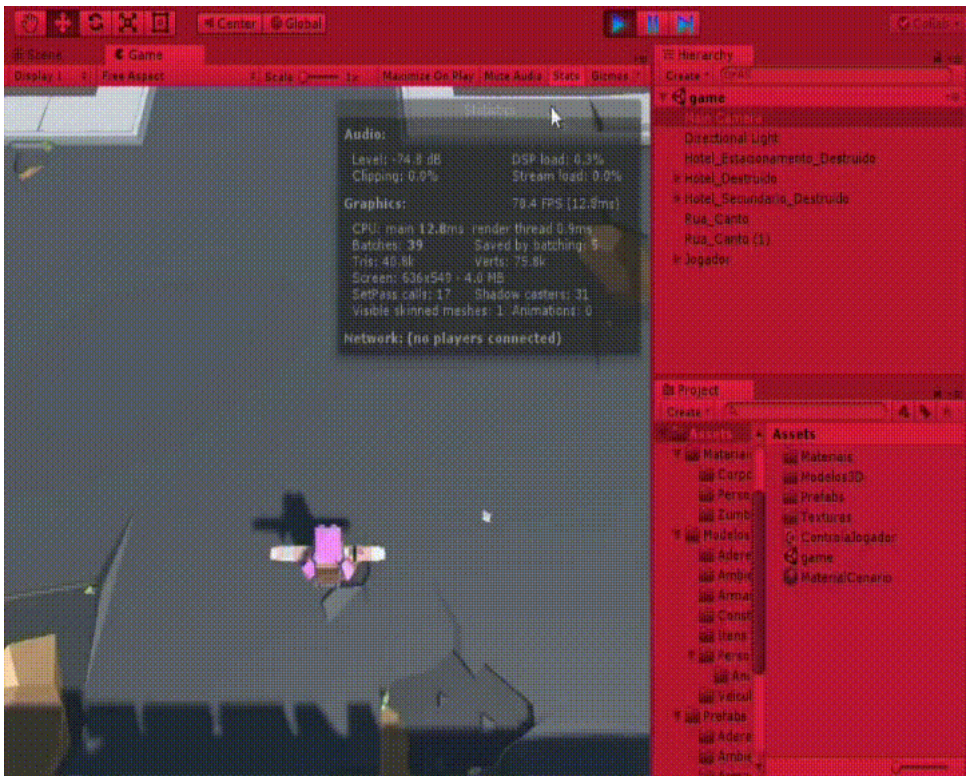
Para **normalizar a velocidade**, em vez rodar **um** `Update` **por quadro**, rodaremos **um quadro por segundo**. Assim, ajustaremos a disparidade da velocidade que está oscilando entre 60 e 80 imagens por segundo e o movimento que está na velocidade do jogo, passará a ser velocidade por segundo, ou seja, andará 1 quadro por segundo.

Para isso, no código, em `transform`, multiplicaremos (`*`) `direcao`, que faz a movimentação nos intervalos de `0` a `1` para os quatro lados, por `Time.deltaTime`, para estabelecer que o **movimento será por segundo**:

```
public class ControlaJogador : MonoBehaviour {


    // 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 * Time.deltaTime);
```

```
    }
}
```

O `Time.deltaTime` pega o tempo que a Unity demora para rodar cada quadro e, no final, normaliza para velocidade por segundo. Salvas as alterações, desativaremos e ativaremos "Play".



A velocidade reduziu, mas ficou muito lenta. Um quadradinho por segundo é pouco. Para aumentar a velocidade, precisamos aumentar o número de quadrados, multiplicando `direcao` por um valor. Mas, a dinâmica desse processo é ineficiente.

Para que o **ajuste de velocidade** fique **dinâmico** e se ajuste proporcionalmente ao cenário, **multiplicaremos** `direcao` por uma **variável** que permitirá variar o valor dela. Essa variável será declarada dentro da classe, acima de `Update` :

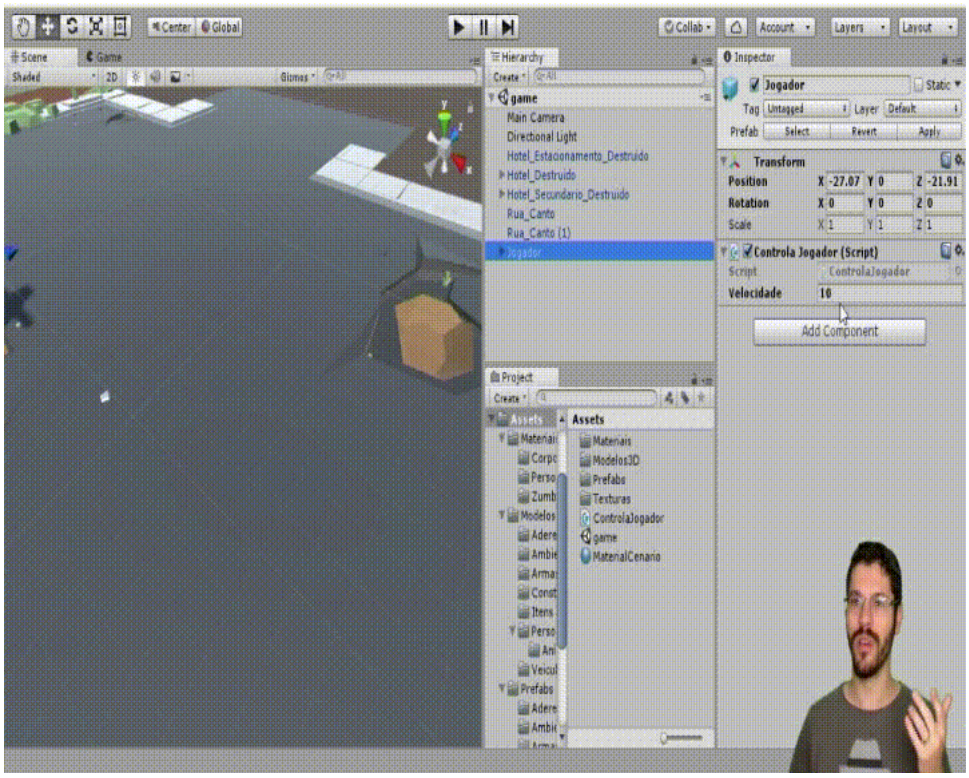
- começando com uma nova palavra `public` ;
- na sequência, `float` por se tratar de um número que pode conter casas decimais;
- nomearemos como `Velocidade` ;
- atribuiremos (`=`) o valor inicial `10` , da mesma forma que atribuímos `Input` nos `eixoX` e `eixoZ` .

Após declarar, adicionaremos `Velocidade` à `transform` :

```
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);

    }
}
```

Mas, nada mudou se ainda tivermos que abri o *script* para alterar o valor em `Velocidade` sempre que necessário. De volta à Unity, se clicarmos em "Jogador", notem que a velocidade (`10`) aparece em "Inspector" como um item editável.



Ao testarmos o movimento, vemos que a velocidade muda de acordo com o valor que inserimos em "Inspector". O que proporciona dinamismo para o desenvolvimento do jogo. Isso acontece em função do `public` que adicionamos antes de `float` .

Assim, a personagem está se movendo de forma mais coerente e a velocidade não depende mais da quantidade de *frames*, agora ela varia por segundo.