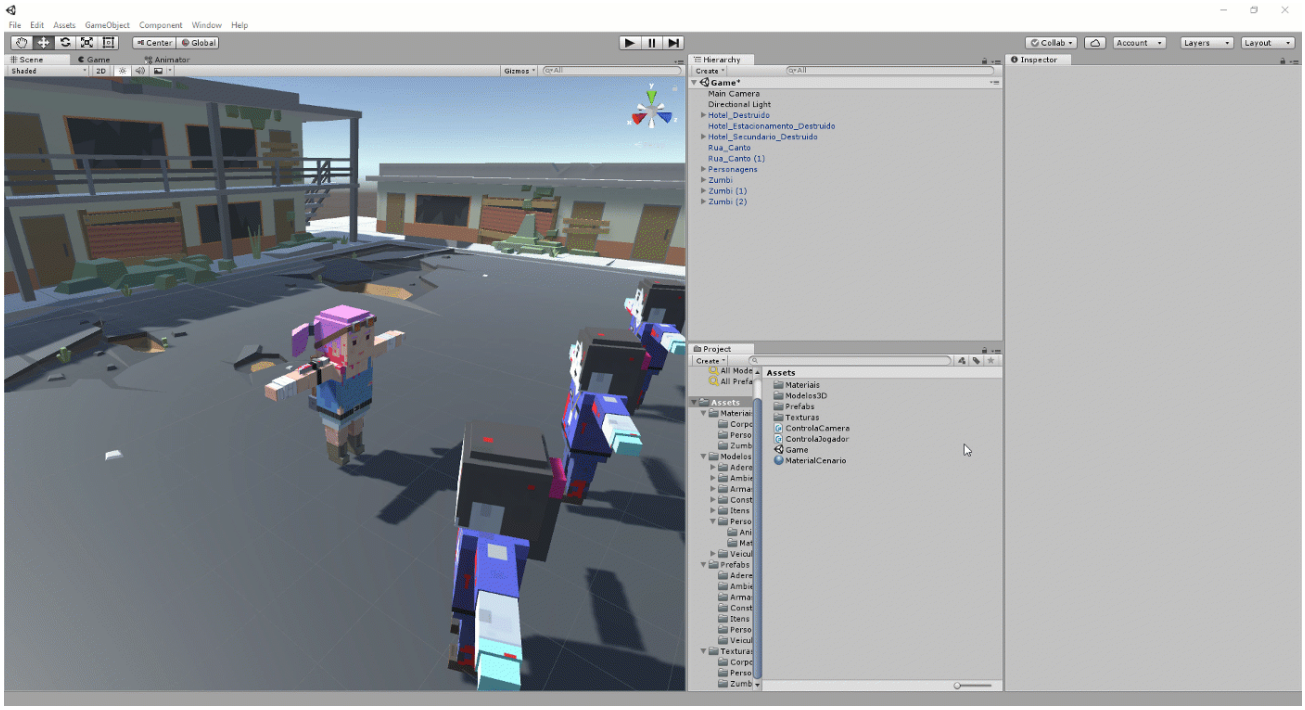
04

Zumbi perseguindo o Jogador

Já trabalhamos bastante com o jogador vamos começar a trabalhar com o Inimigo?

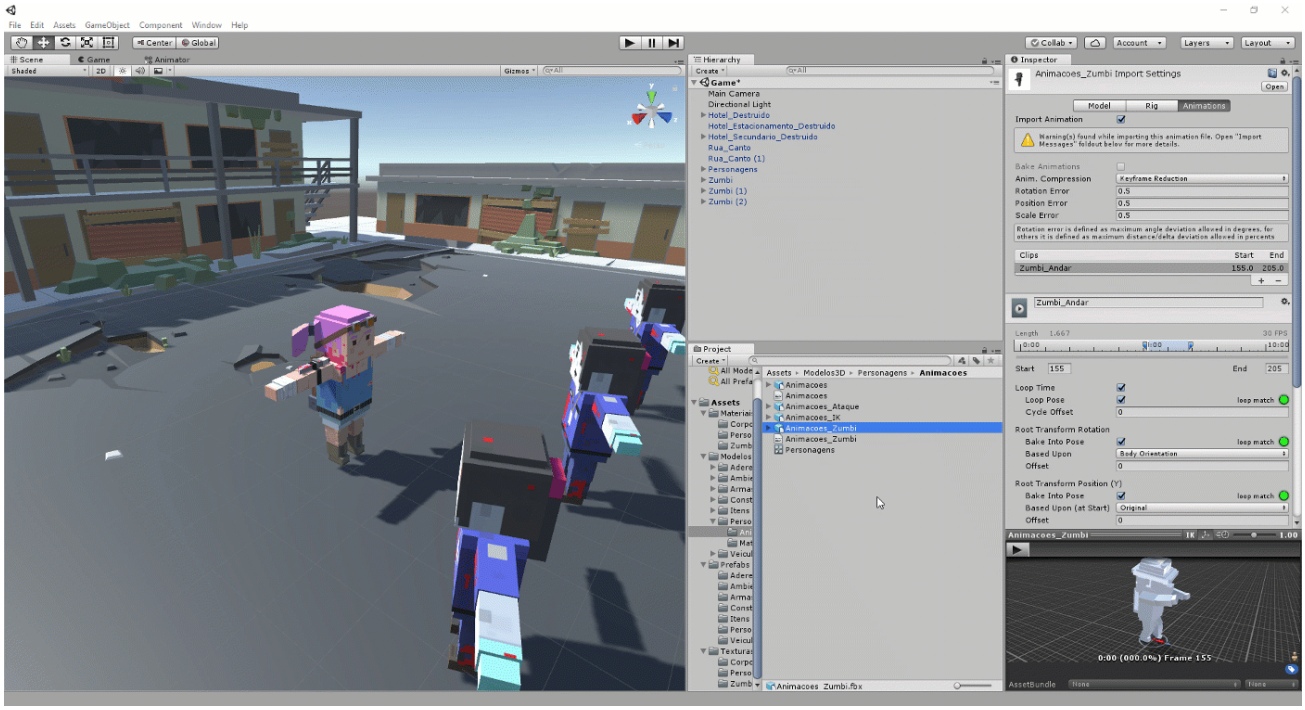
Nosso inimigo fica parado no cenário sem fazer nada o legal seria ele perseguir o nosso jogador, certo? Vamos fazer isto!

No inimigo vamos iniciar pela animação ao invés da movimentação então selecione o objeto que tem as animações navegando até a pasta **Assets > Modelos3D > Personagens > Animacoes** e clique no objeto `Animacoes_Zumbi .`

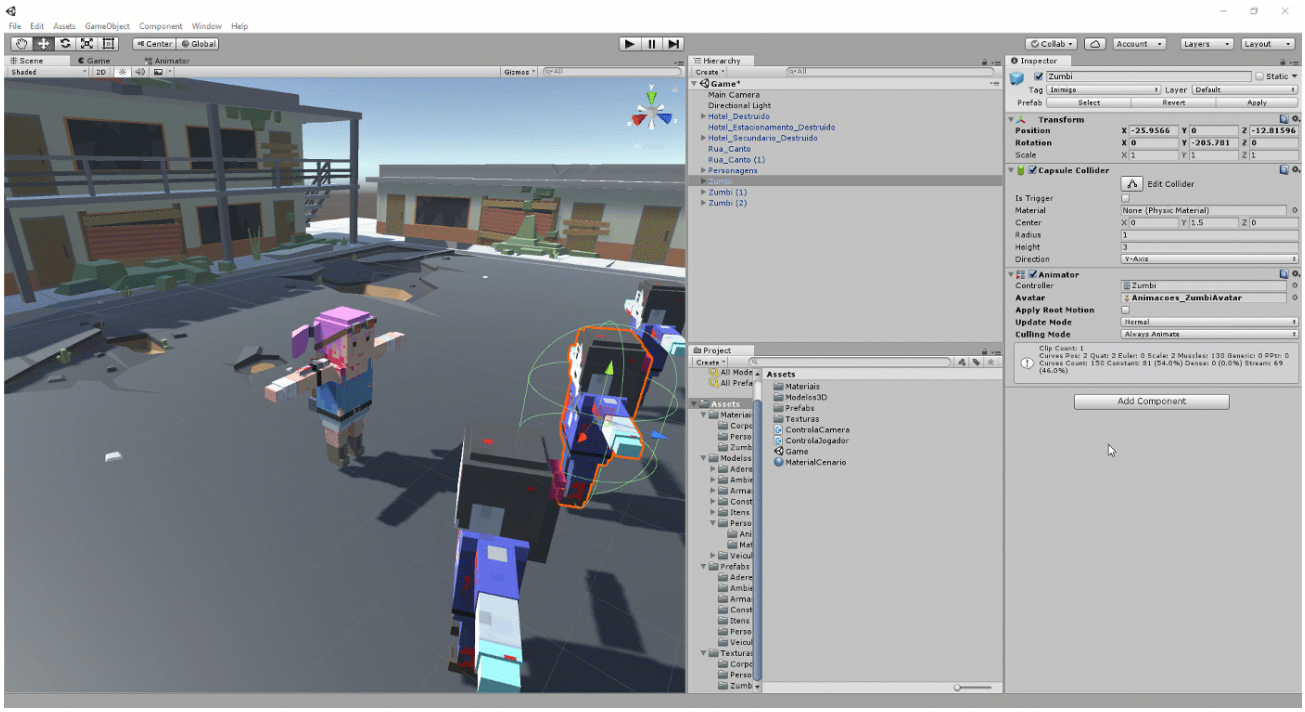


Na aba de animações já temos ela separadas então não vamos ter esta necessidade, mas o modelo tem outras animações que você pode separar, os *Quadros* dessa animação estão no arquivo de texto `Animacoes_Zumbi.txt .`

Vamos arrastar a animação **Zumbi_Andar** e o *Avatar* para o nosso Zumbi inimigo.



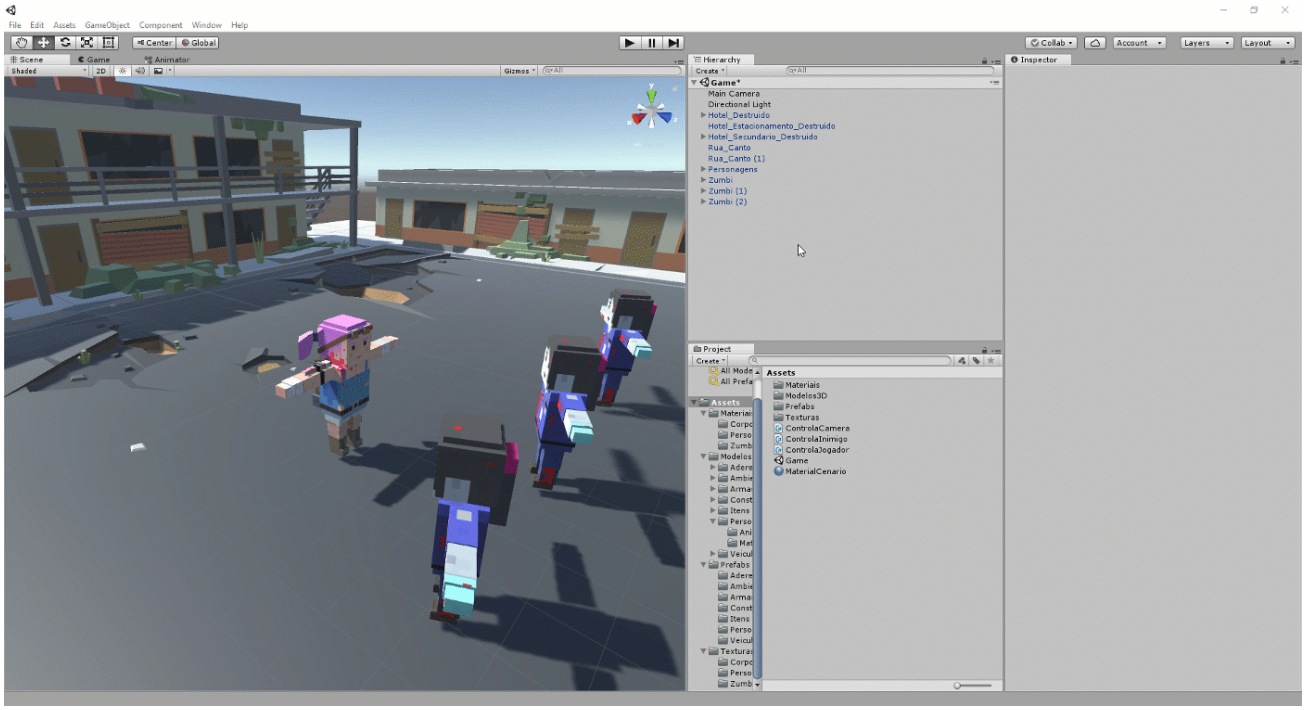
Agora já temos nosso Zumbi animado correndo, vamos então trabalhar na movimentação dele? Para fazer isto vamos fazer algo similar ao nosso jogador então vamos começar adicionando um **Rigidbody** ao nosso inimigo, **Congelando (Freeze)** na posição em Y e rotação em X, Y e Z neste Rigidbody e vamos criar um *Script* com o nome `ControlaInimigo` e também adicionar a ele.



Antes de escrever o código da movimentação vamos pensar um pouco no que queremos fazer, no jogador tínhamos a variável `direcao` para controlar o movimento a partir das teclas apertadas mas no inimigo a direção que queremos é a posição do *Jogador* então vamos criar no script uma variável do tipo `GameObject` que valha este jogador?

public GameObject Jogador;

Lembre-se de salvar o *Script*, agora podemos preencher esta variável com o objeto arrastando ele da *Hierarquia* para o *Inspetor*.



Agora temos a posição do jogador mas isso não significa que só fazer o Zumbi andar para esta posição porque a nossa direção vai ser a posição posição do Jogador baseada na posição atual do inimigo.

Para isto temos que fazer um cálculo simples de pegar a posição final que queremos chegar que é a posição do Jogador e subtrair a posição inicial que é a posição do Zumbi inimigo. Como vamos movimentar o Zumbi utilizando **Rigidbody** assim como o Jogador vamos já fazer este cálculo no `FixedUpdate` ? Seu código então fica assim:

```
void FixedUpdate ()
{
    Vector3 direcao = Jogador.transform.position - transform.position;
}
```

Agora podemos iniciar a movimentação para isso vamos fazer o mesmo que fizemos com o Jogador utilizando o **Rigidbody** e o método `MovePosition` a partir da posição que estamos no código temos então:

```
void FixedUpdate ()
{
    Vector3 direcao = Jogador.transform.position - transform.position;

    GetComponent<Rigidbody>().MovePosition (GetComponent<Rigidbody>().position + direcao);
}
```

Agora falta fazermos este deslocamento ter uma velocidade que seja possível de trocarmos de forma rápida caso necessário e deslocar pelo tempo.

Para isto vamos criar a variável de velocidade no topo do *Script* pela linha `public float Velocidade = 5;` e é claro que o valor inicial tem que ser menos que o do Jogador senão o jogo ficaria um pouco injusto.

E também vamos multiplicar o deslocamento por `Time.deltaTime` para fazer a velocidade ser por segundo e não por *Frame (Quadro)* como fizemos no Jogador, então temos:

```
void FixedUpdate ()
{
    Vector3 direcao = Jogador.transform.position - transform.position;

    GetComponent<Rigidbody>().MovePosition (GetComponent<Rigidbody>().position + direcao * Velo
}
```

Se testarmos agora vamos notar que o nosso zumbi está muito rápido, por que isso acontece? No jogador quando utilizamos a variável `direcao` por causa de ela ser controlada pelo aperto das teclas ela foi limitada a um valor que vão de -1 a 1 em X e Z.

Porém no nosso inimigo a `direcao` aqui é controlada pela posição final e inicial que nosso inimigo está assumindo então ela é um `Vector3` bem maior, temos que fazer ela também ficar em bases de -1 a 1 para podermos de fato deslocar levando em consideração somente a variável `Velocidade` .

Para isso temos que fazer um processo conhecido como **Normalização** que pega um **Vetor** de algum tamanho e transforma em outro que sempre tem tamanho até no máximo `1` .

A Unity já calcula tudo para nós então basta você pegar a variável `direcao` e colocar `direcao.normalized` que ela é normalizada, tendo nosso código o seguinte aspecto.

```
void FixedUpdate ()
{
    Vector3 direcao = Jogador.transform.position - transform.position;

    GetComponent<Rigidbody>().MovePosition (GetComponent<Rigidbody>().position + direcao.normal
}
```

Pronto! Agora é só salvar o *Script* e aplicar o **Prefab**!