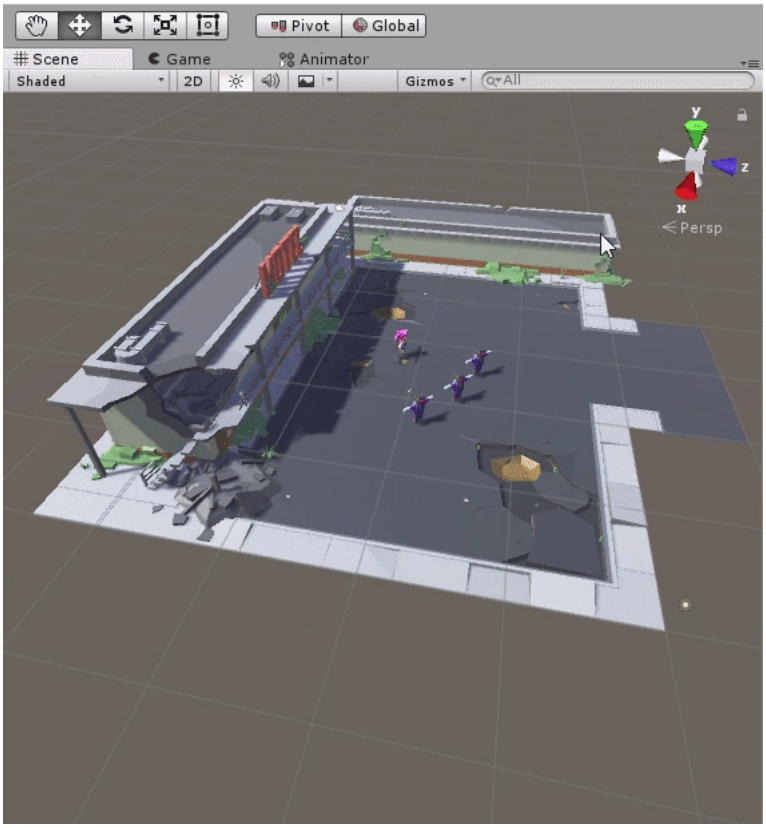
13

Sistema de física

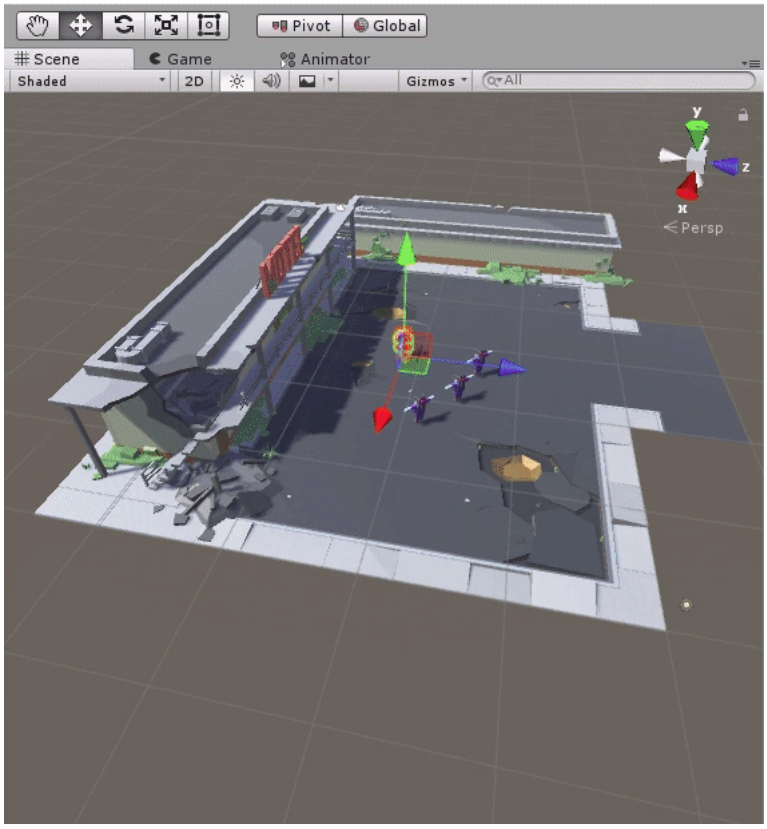
Transcrição

Anteriormente, adicionamos colisores aos objetos. Ligaremos o "Play" para testar e veremos que a personagem ainda atravessa as paredes.

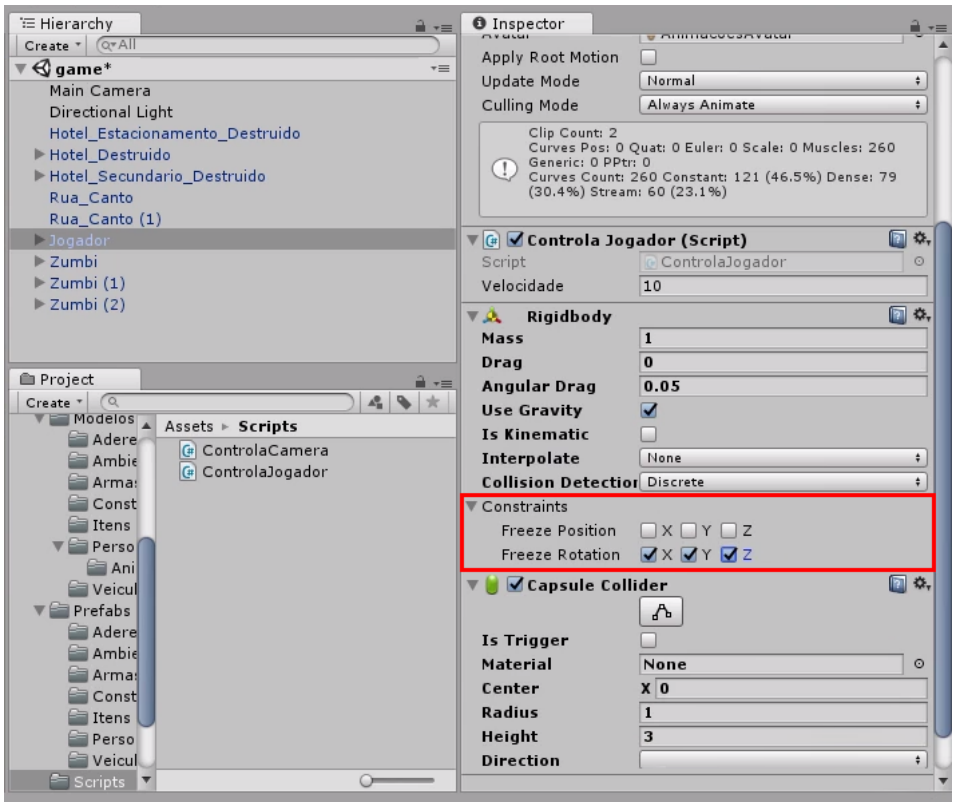


Na Unity, quando queremos que os colisores se batam, um deles deve ter **física** para computar que um objeto está batendo no outro. Levando em consideração que "Jogador" é o objeto que anda e bate nos outros, aplicaremos a física nele. Ela é um componente, então iremos a "Inspector" e clicaremos em "Add Component > Physics > Rigidbody". "Rigidbody" é o componente de corpo rígido que a Unity diponibiliza para uso.

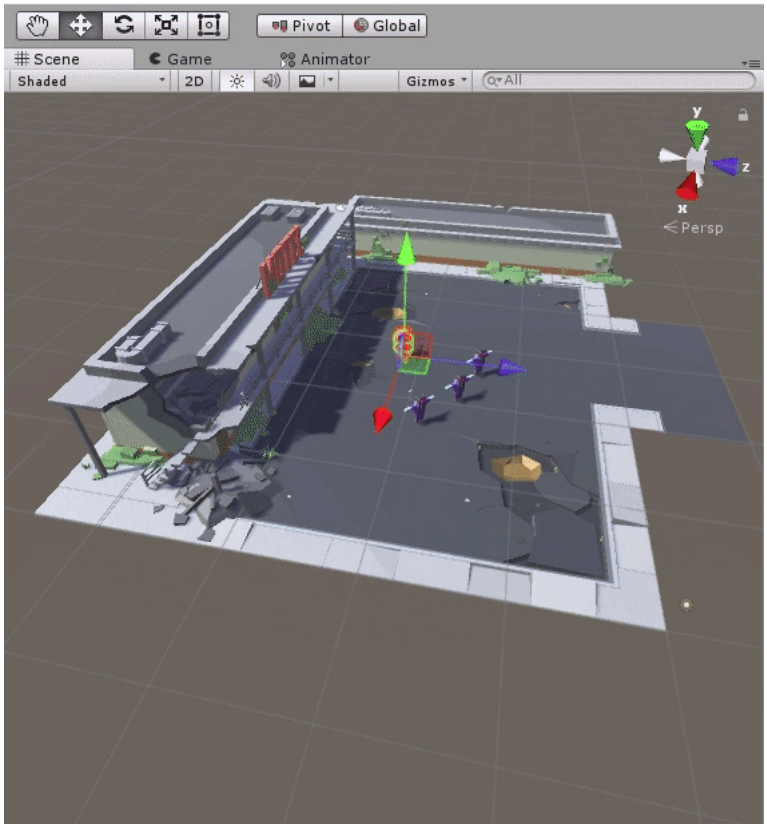
Ligaremos o "Play" novamente e veremos que a personagem não atravessa mais as paredes, mas ao insistir em passar por algum objeto, ela tomba e começa a rodar aleatoriamente.



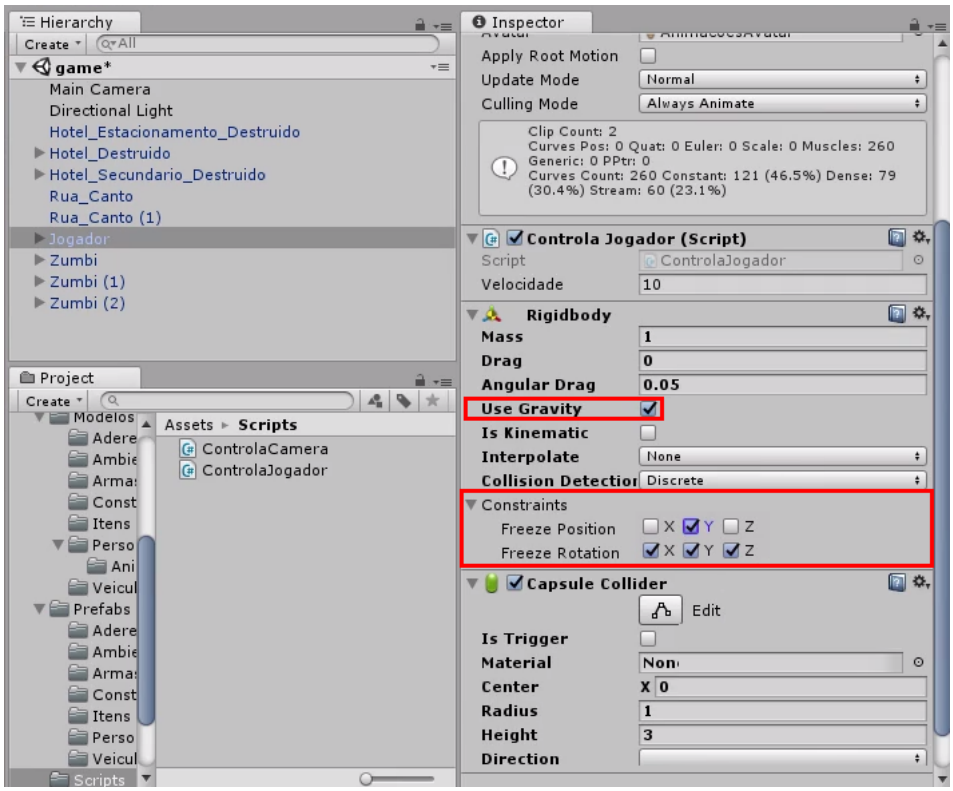
Quando batemos nos zumbis, a personagem tentou subir neles desencadeando esse tombo. Limitaremos a rotação para que ela não fique rodando aleatoriamente. Em "Inspector", configuraremos a parte de "Constraints" em "Rigidbody", marcando a caixa de seleção dos três eixos em "Freeze Rotation" para impedir que a física rotacione "Jogador".



Ativaremos o "Play" para ver o que mudou após a configuração em "Rigidbody". A personagem não inicia a rotação aleatória ao passar pelos zumbis, mas ao sair do perímetro de "Hotel_Estacionamento_Destruido", ela cai.



Para que isso não aconteça, precisamos marcar a caixa de seleção do eixo Y em "Freeze Position", considerando que "Use Gravity" está ativada em "Rigidbody". Dessa forma, a personagem não cairá mais pelo cenário.



Ligaremos o "Play" e veremos que ela não cai mais ao ao sair do perímetro de "Hotel_Estacionamento_Destruido". No entanto, quando ela encosta na parede do hotel, notem que a câmera treme.

Especificamos que "Jogador" se move por meio de `transform.Translate`, então ele está sendo jogado para dentro da parede do colisor, ao mesmo tempo em que a física impede que isso aconteça. Ou seja, a física contradiz o `transform.Translate`, desencadeando esse efeito de travamento na tela do jogo.

Solucionaremos esse problema utilizando a física para mover a personagem em vez do `Transform`. Dessa forma, quando colidir com a parede, a personagem ficará parada. No "Inspector" de "Jogador", abriremos o código clicando duas vezes em "ControlaJogador", à direita de "Script".

Abaixo de `Vector3`, acrescentaremos um trecho de código para movimentar a personagem por meio da física. Porém a Unity garante por meio de `transform` que todo objeto de jogo possui posição no espaço. Levando isso em consideração, utilizaremos:

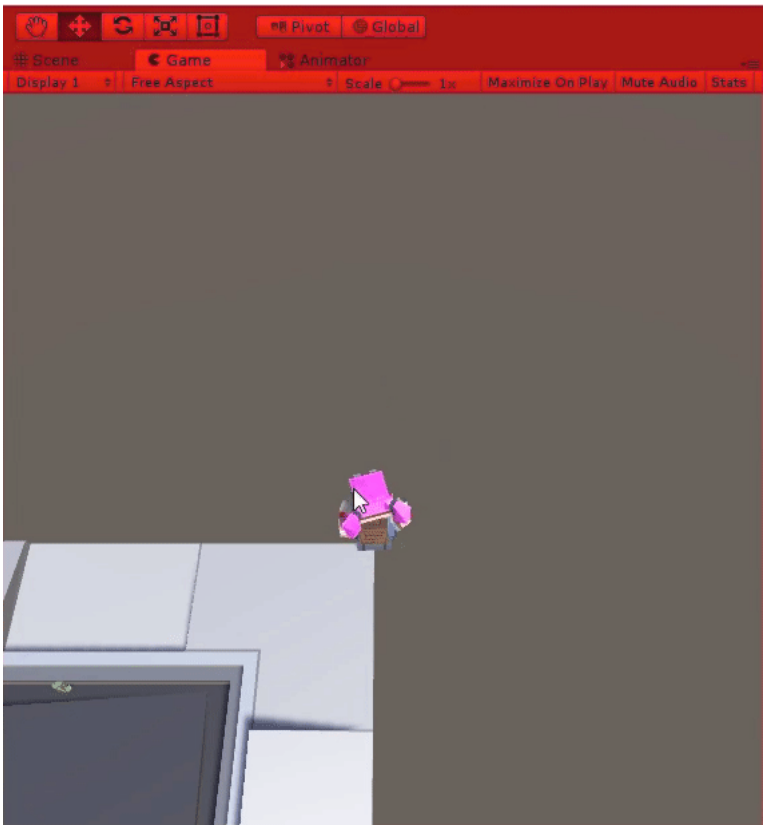
- digitando `GetComponent<Rigidbody>()` para pegar o componente "Rigidbody" e movimentar a personagem por meio da física;
- ponto (`.`) para acessá-lo;
- `MovePosition` para mover a posição do objeto a partir da física;
- `direcao * Velocidade * Time.deltaTime` entre parênteses (`()`),
- ponto e vírgula (`;`) para concluir a linha.

Apagaremos `transform.Translate()` e salvaremos a edição do código, que ficou 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);

        GetComponent<Rigidbody>().MovePosition(direcao * Velocidade * Time.deltaTime);

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

    }
}
```

Ao ativarmos o "Play", localizaremos a personagem em um canto do jogo.

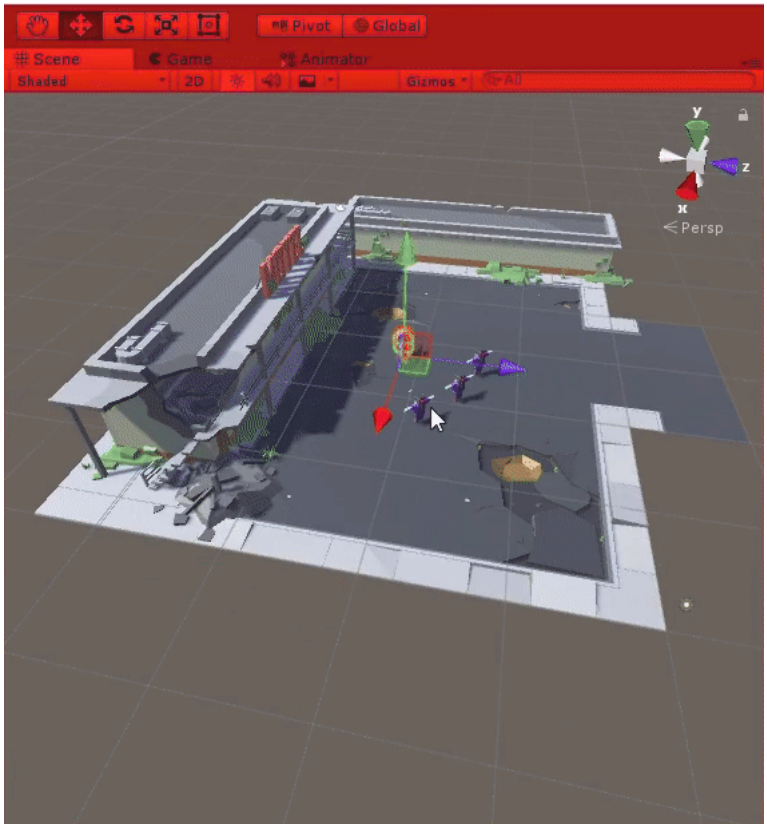


Se tentarmos movê-la, não conseguiremos, porque os comandos para física são diferentes do `transform.Translate`. Nela, precisamos passar a posição em que ela está (`GetComponent<Rigidbody>().position`) e somar (`+`) a ela a direção do movimento (`direcao * Velocidade * Time.deltaTime`). O trecho de código ficará da seguinte forma:

```
        GetComponent<Rigidbody>().MovePosition
            (GetComponent<Rigidbody>().position +
                        (direcao * Velocidade * Time.deltaTime));
```

Acrescentamos parênteses (`()`) para organizar o trecho, lembrando que `direcao * Velocidade * Time.deltaTime` é a direção do movimento. Após conferir se o código está correto, salvaremos e minimizaremos o editor de texto.

Ligaremos o "Play" e veremos que a personagem se move, sem travar tanto na parede.



Resolveremos o travamento no código, transferindo o último trecho que adicionamos para outro método:

- selecionaremos e copiaremos;
- deletaremos;
- após a chave que encerra `Update` adicionaremos um método específico para física `void FixedUpdate()`;
- entre chaves (`{}`) colaremos o trecho que copiamos.

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);

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

    }
```

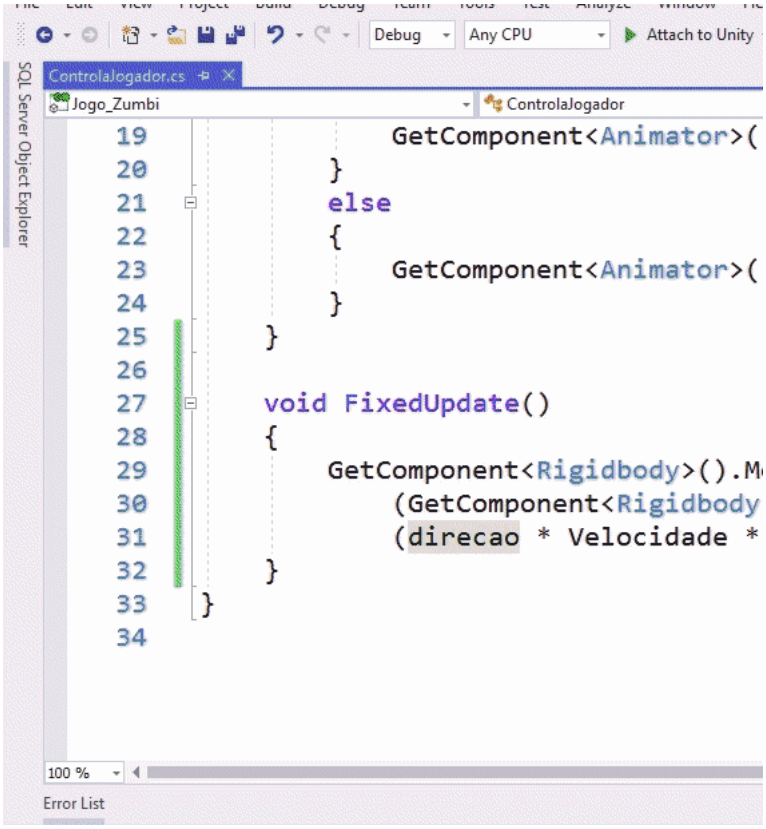
```
    void FixedUpdate()
    {
        GetComponent<Rigidbody>().MovePosition
            (GetComponent<Rigidbody>().position +
            (direcao * Velocidade * Time.deltaTime));
    }
}
```

A diferença de `FixedUpdate` para `Update` é que esse roda o tempo todo, a cada *frame* do jogo e aquele roda em um tempo fixo (`0.02` segundos), padrão da Unity. Assim, a cada 1 segundo, `FixedUpdate` roda 50 vezes, diferente do `Update` que roda todo o *frame* do jogo. Ou seja, `FixedUpdate` roda a cada `0.02` segundos, independente do *frame* de jogo. Isso faz com que a Unity compute a física, mesmo que o jogo não esteja rodando as telas.

Notem que o editor de texto apontará um erro em `direcao` , porque a variável foi declarada em `Update` e a utilizamos fora dele. Para corrigir, faremos a declaração no início do código, inserindo `Vector3 direcao;` abaixo da declaração de `Velocidade` . Não atribuiremos valor inicial a ela, para que a Unity atribua zero aos três eixos (X, Y e Z). Deletaremos `Vector3` que utilizamos para declarar `direcao` em `Update` . Se clicarmos na primeira aparição de `direcao` , veremos que ela fica em destaque em todos os locais em que aprece no código. Salvaremos o código, que agora está assim:

```
public class ControlaJogador : MonoBehaviour {

    public float Velocidade = 10;
    Vector3 direcao;

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

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

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

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

    }

    void FixedUpdate()
    {
        GetComponent<Rigidbody>().MovePosition
            (GetComponent<Rigidbody>().position +
            (direcao * Velocidade * Time.deltaTime));
    }
}
```

Ativaremos "Play" e veremos que funcionou. Quando a personagem colide com a parede, a tela não trava nem treme. Resolvemos a questão da movimentação por meio da física.


Debug
Any CPU
Attach to Unity
ControlaJogador.cs
Jogo_Zumbi
ControlaJogador
SQL Server Object Explorer
GetComponent<Animator>(
}
else
{
GetComponent<Animator>(
}
}
void FixedUpdate()
{
GetComponent<Rigidbody>().M
(GetComponent<Rigidbody
(direcao * Velocidade *
}
}
100 %
Error List