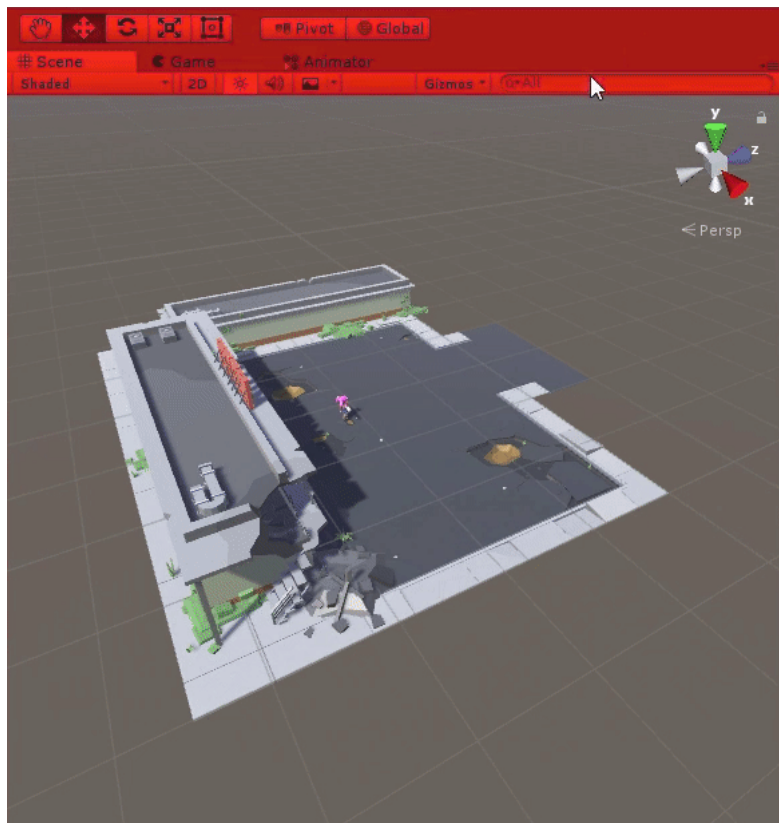


08

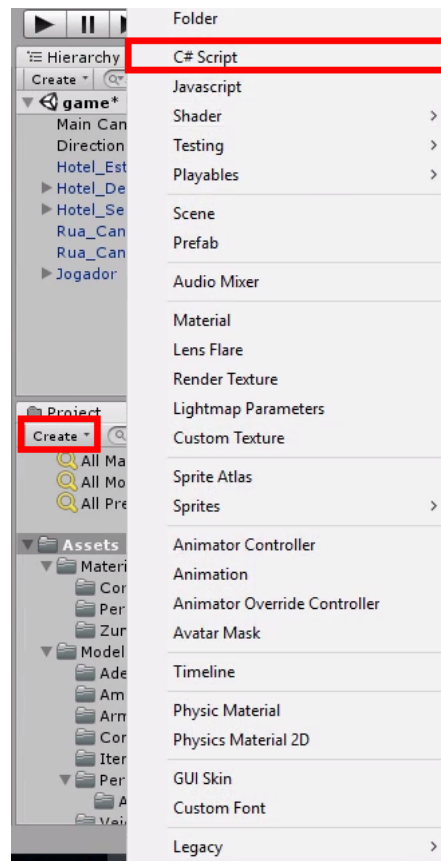
## Movimentando a câmera

### Transcrição

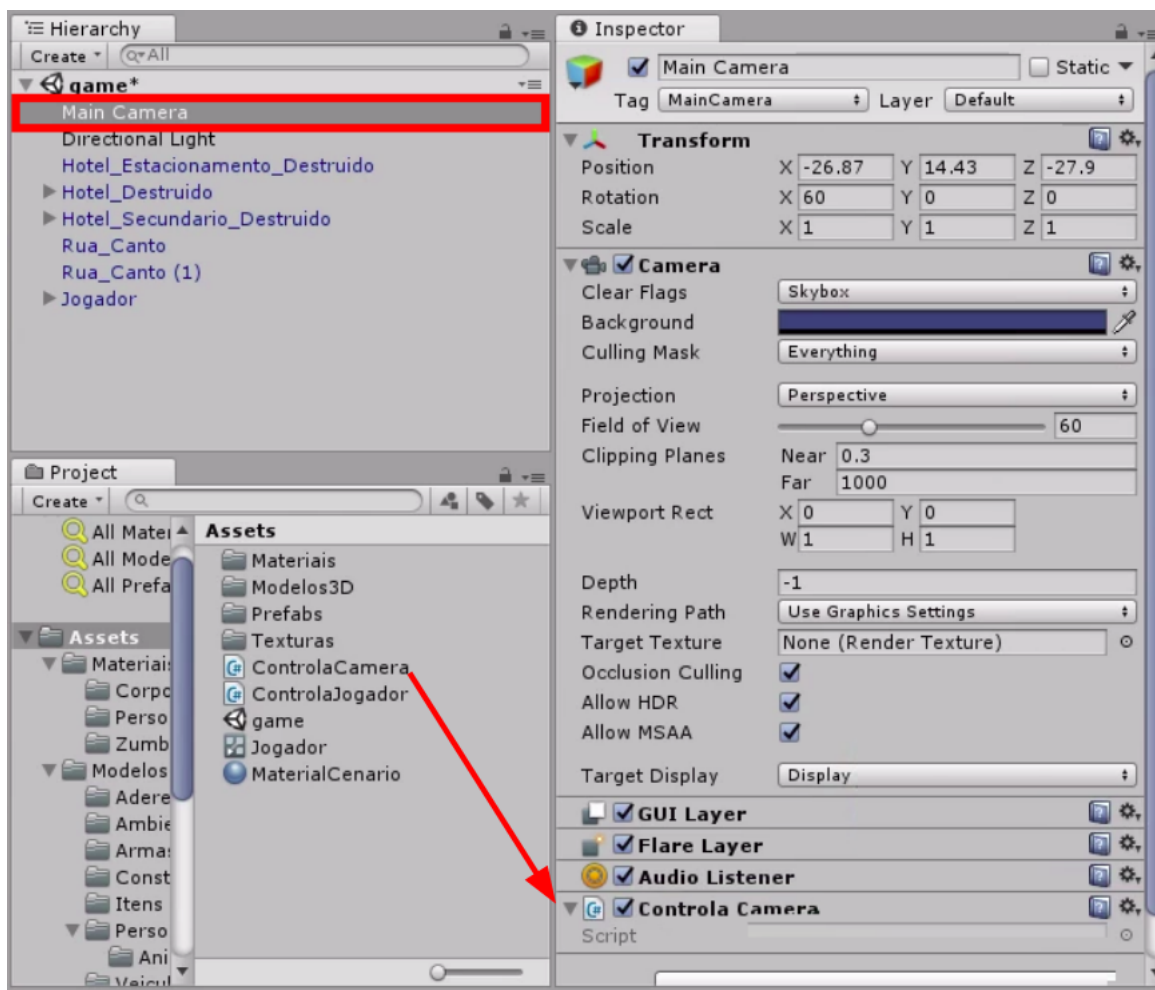
Continuaremos o desenvolvimento do jogo. Notem que ao acionar o "Play", a personagem está se movimentando, mas a partir de um ponto a perdemos de vista.



A câmera que registra as imagens do jogo está fixa na imagem inicial e não acompanha o jogador. Queremos que ela o siga. Para isso, criaremos um *script*, um código, clicando em "Create" abaixo de "Project" e selecionando a opção "C# Script".



Nomearemos o novo código como "ControlaCamera". É importante lembrar de não inserir espaço e acentos no nome. Clicaremos em "Main Camera", em "Hierarchy", e arrastaremos o código de "Assets" para o "Inspector" dela. Dessa forma, ele estará pronto para funcionar após salvarmos as edições.



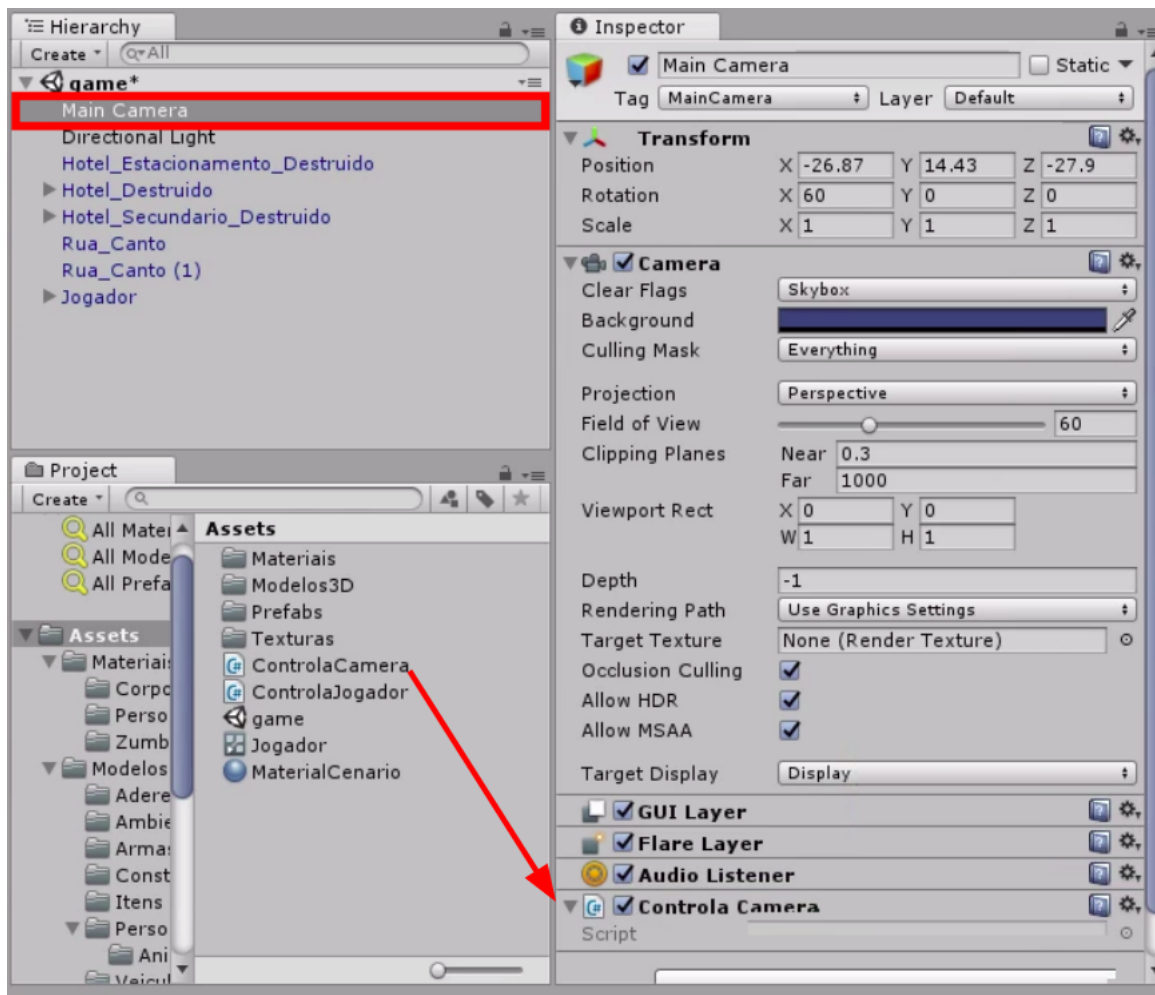
Para que a câmera acompanhe o jogador, precisamos utilizar a posição dela, obtida em "Transform", em "Position" do "Jogador". Aplicaremos isso no código, por meio de:

- `transform.position` , para captar a posição da câmera no código, considerando que o código está nela, esse trecho aponta a posição dela;
- `=` , para jogar a posição dela no "Jogador", igualaremos as posições;
- `public GameObject` antes de `Update` , para obter a posição do "Jogador". Já utilizamos variáveis do tipo `float` e `Vector3` . "Jogador" será uma variável do tipo `GameObject` , ou seja, objeto de jogo. Qualquer objeto que está no jogo é um `GameObject` ;
- `Jogador` como nome da variável.

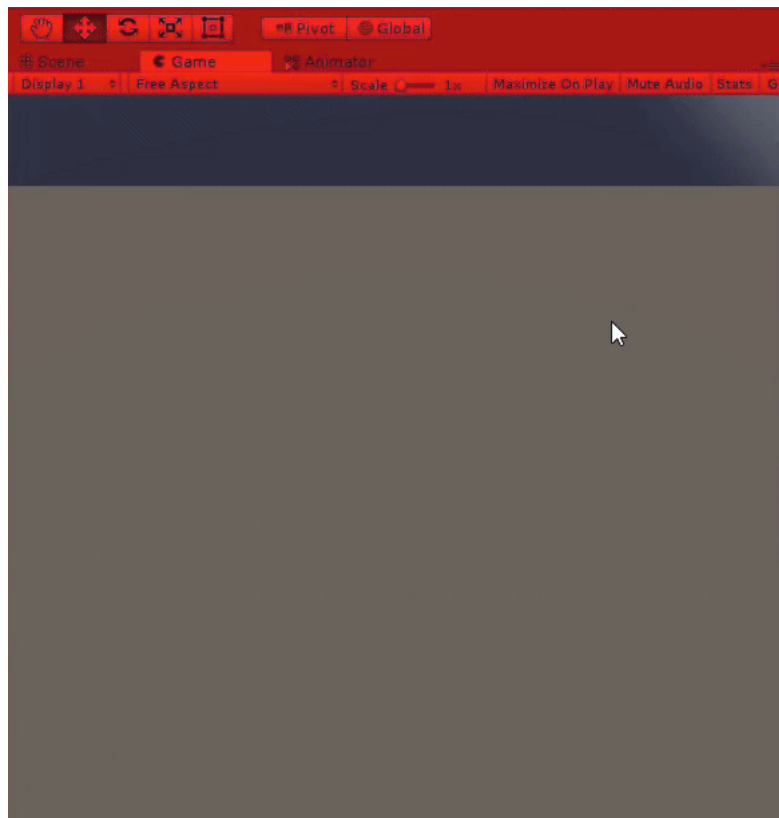
Voltaremos a `transform.position` e igualaremos a `Jogador` . Utilizaremos ponto ( `.` ) para acessar `Jogador` e `transform` para obter a posição ( `position` ). O código ficará da seguinte forma:

```
public class ControlaCamera : MonoBehaviour {  
  
    public GameObject Jogador;  
  
    //Use this for initialization  
    void Start () {  
  
    }  
  
    //Update is called once per frame  
    void Update () {  
        transform.position = Jogador.transform.position;  
    }  
}
```

No entanto, a Unity não entende que `Jogador` refere-se ao "Jogador". Utilizamos a variável pública ( `public` ), pois assim que salvarmos o código, em "Inspector" de "Main Camera", veremos e poderemos preencher com um objeto de jogo a variável `Jogador` , abaixo de "Script". Arrastaremos "Jogador" de "Hierarchy", para "Jogador" em "Inspector". Agora, o "Jogador" está na variável e no código.



Ligaremos o "Play" e veremos algo se mover, mas não o quê. É como se a a câmera estivesse posicionada nos pés dele. Queremos que o registro seja feito de cima, a uma **distância fixa** da personagem.

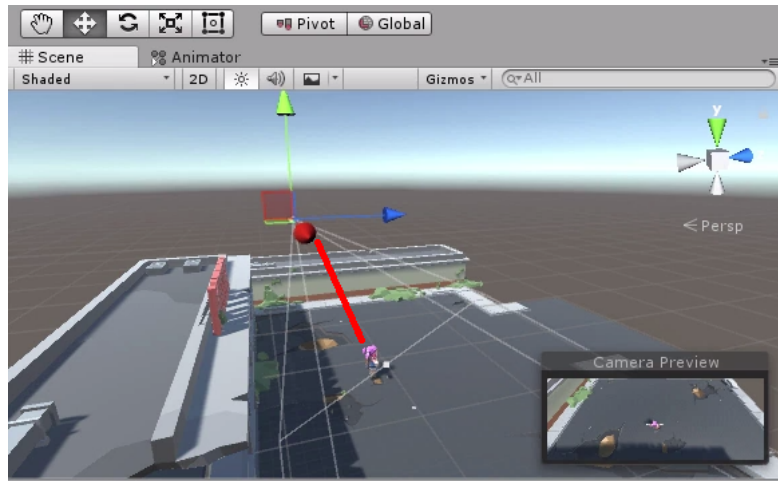


Considerando que será sempre a mesma, precisamos calcular essa distância entre a câmera e o "Jogador" somente **uma vez**. Na Unity, o método executado somente uma vez é o `Start`. Ele roda no início do jogo e, depois, não roda mais.

Assim, adicionaremos nele o cálculo da distância.

Se declararmos a variável dentro de `Start`, como fizemos em `Update` com `float eixoX`, teremos que utilizá-la somente dentro de `Start`. Da mesma forma que só podemos usar em `Update` as variáveis que declararmos dentro dele. Então, declararemos acima de `Start`, para podermos usá-la tanto em `Start` para cálculo da distância, quanto em `Update` para orientá-la a seguir o "Jogador" considerando o intervalo calculado.

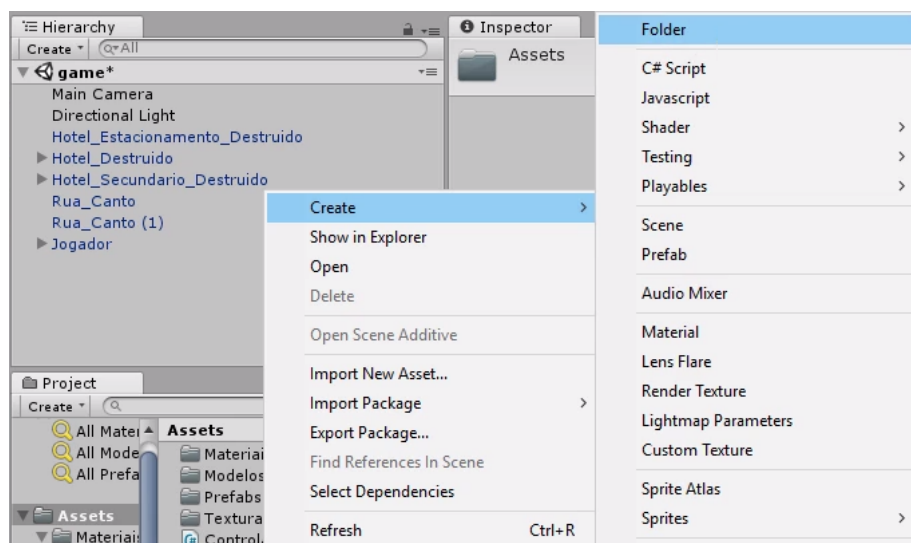
A distância será uma variável do tipo `Vector3`, porque estaremos calculando a diferença de posição entre a câmera e o "Jogador" (outro `Vector3`). Ela não será uma variável pública, pois será utilizada somente nos métodos, então não precisaremos visualizá-la. Nomearemos como `distCompensar` para remeter à função dela, que é compensar a distância entre o "Jogador" e a câmera.



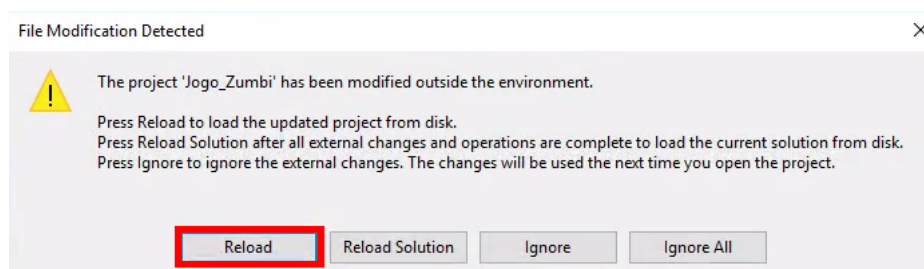
Em `Start`, calcularemos o valor da distância, subtraindo ( - ) a posição do "Jogador" ( `Jogador.transform.position` ) da posição da câmera ( `transform.position` ), que é o objeto com o *script*. Compensaremos a distância do movimento da câmera em `Update`. Então, orientaremos ela a seguir ( `transform.position` ) a posição do "Jogador" ( `Jogador.transform.position` ), considerando a distância calculada em `Start` e somaremos ( + ) a posição ( `distCompensar` ) que ela deve seguir. Salvaremos as alterações e ligaremos o "Play" na Unity. Veremos que a câmera está seguindo a personagem.



Aproveitando que estamos com a mão na massa e temos dois *scripts*, criaremos uma pasta para organizá-los, clicando com o botão direito do mouse em "Assets", selecionaremos "Create > Folder". Nomearemos a pasta como "Script", para armazenar os códigos que criamos.



Se voltarmos ao código, abrirá uma mensagem de detecção de modificação no arquivo, na qual clicaremos na opção "Reload".



Se dermos "Play", tudo continua funcionando normalmente e, se precisarmos acessar os códigos, basta abrirmos a pasta "Scripts" e selecionar.