05

Destruir os zumbis

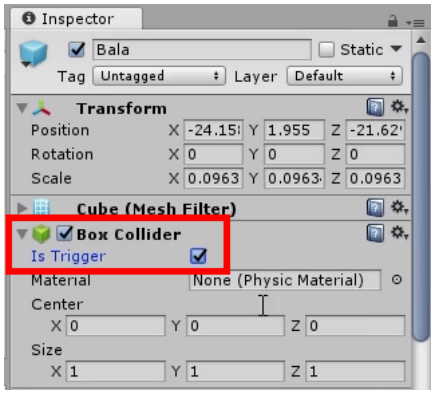
Transcrição

Atualizamos o *script* de `ControlaJogador.cs` para que a personagem principal gire e atire para todos os lados. No entanto, quando as balas são atiradas, ficam dispersas ao encostar nos elementos do cenário, inclusive nos inimigos. Isso acontece porque a física da bala bate nos objetos sem saber o que fazer na sequência, então ela bate e neles, sofre um desvio e segue em frente.

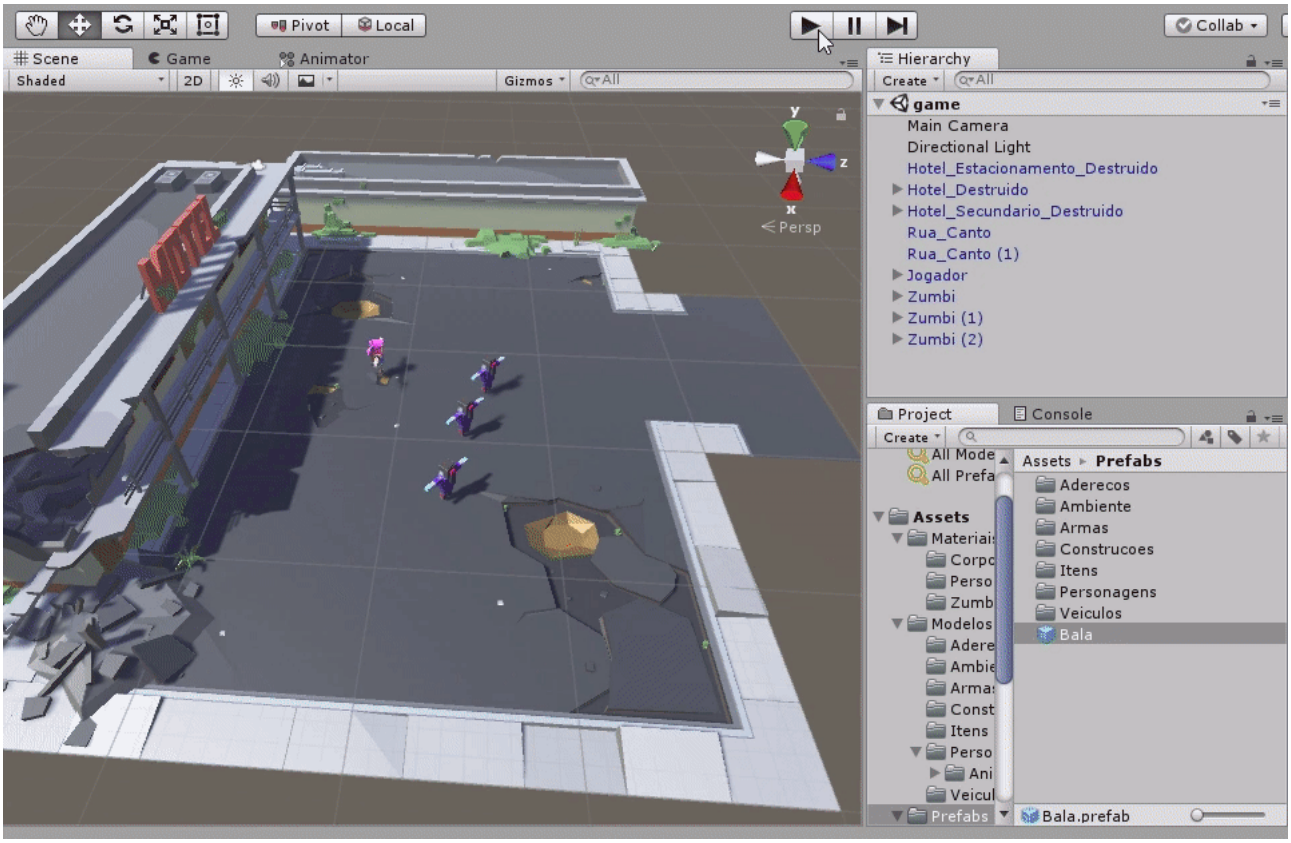
Para corrigir esse problema, teremos que fazer a bala bater nos objetos e ser destruída ou destruir, dependendo do que atingir. Se atingir estruturas do cenário, como paredes, será destruída e irá sumir. Se atingir os inimigos, irá destruí-los.

Primeiro, faremos com que os zumbis sumam após a colisão, pois quando são atingidos pelas balas, são empurrados para trás. Queremos que eles sumam ao serem atingidos pelas balas.

Na janela "Project", acessaremos "Bala" na pasta "Assets > Prefabs". Em "Inspector", ativaremos a opção "Is Trigger" de "Box Colider".



Assim, transformamos a bala em um gatilho ("trigger" em inglês), que gera colisão sem colidir fisicamente com o objeto. Ou seja, atravessará o objeto.



Agora, notem que ela está atravessando os zumbis e as paredes do hotel. A Unity está computando a colisão, mas não a física. Temos o gatilho de colisão, mas não a temos fisicamente.

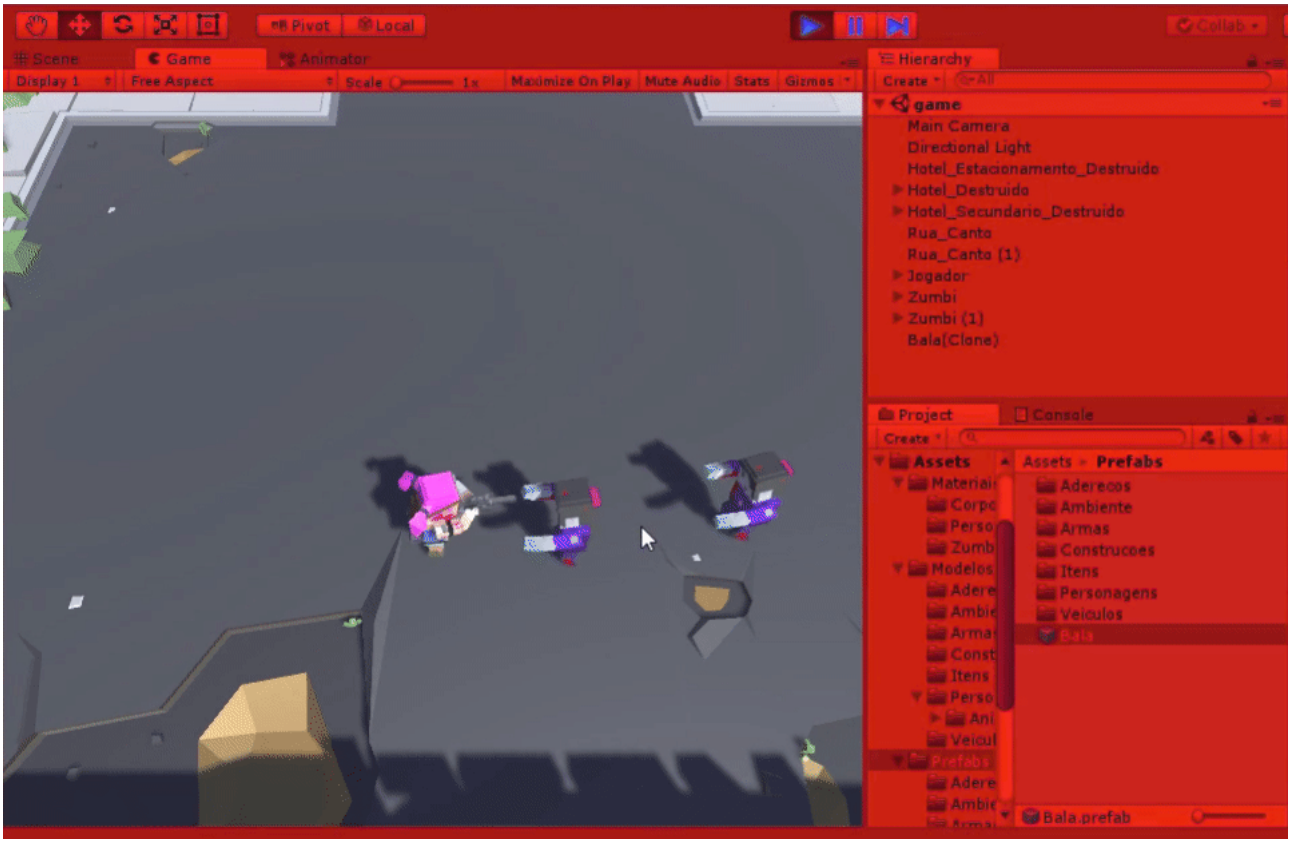
Para determinar se a bala destruirá ou será destruída, no *script* dela (`Bala.cs`), abaixo de `FixedUpdate` , acrecentaremos:

```
void OnTriggerEnter(Collider objetoDeColisao)
{
    Destroy(objetoDeColisao.gameObject);
}
```

Por meio desse método, estabelecemos no *script* que quando um *trigger*, no caso a bala, bate em um objeto, algo acontece. Quando eles colidirem, saberemos o objeto com o qual ele entrou em contato por meio da variável `objetoDeColisao` .

A parte de `Destroy()` implica na destruição de um objeto, que deve ser inserido entre os parênteses (`()`). No caso, utilizaremos a variável `objetoDeColisao` , do tipo colisor (`Collider`).

Salvo e minimizado o *script* voltaremos a "Game" e ativaremos "Play" para ver como ele roda no jogo. Esperamos que a bala (*trigger*) destrua aquilo em que encostar.

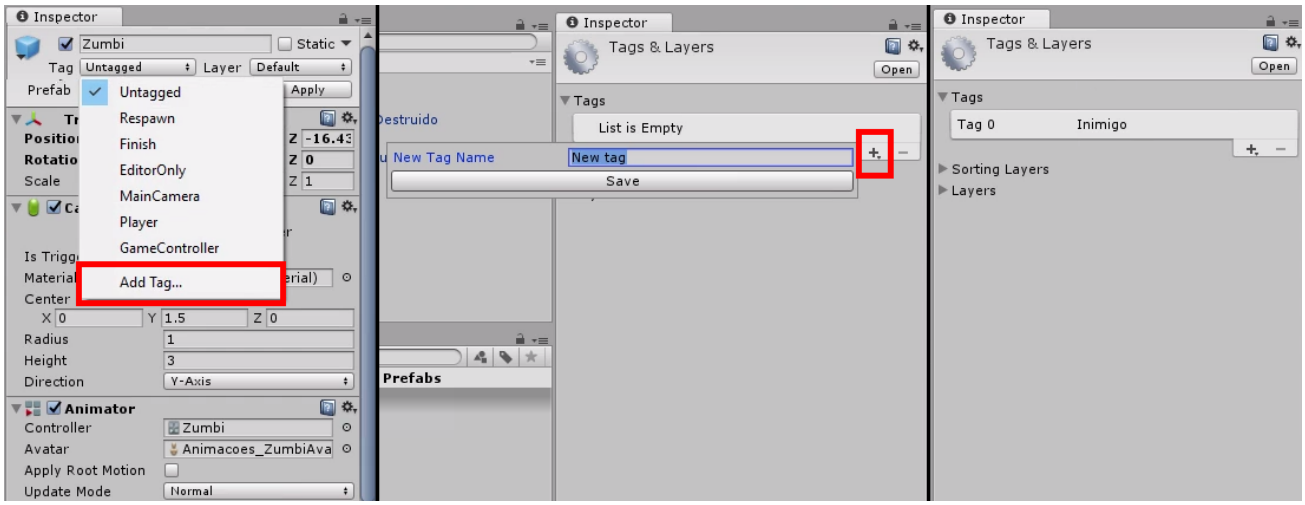


O código está funcionando exatamente da forma esperada. Ao acertar os objetos com as balas, eles somem. Notem que todas as estruturas do cenário somem quando são atingidas pelas balas. Não queremos isso. Queremos que somente os zumbis sumam após serem atingidos e que as balas sejam destruídas ao encostar em objetos que não sejam o inimigo.

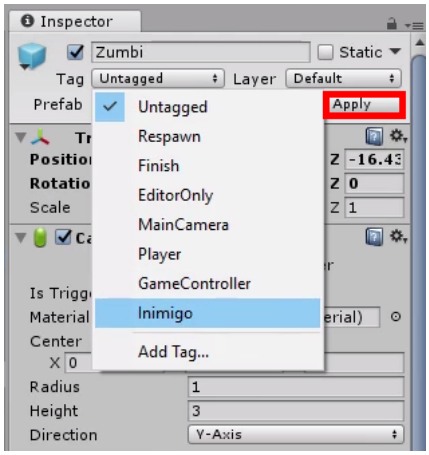
Então, marcaremos os zumbis como inimigos para diferenciá-los do restante do cenário. Em "Inspector", anteriormente, interagimos com a parte de "Layer"; agora, utilizaremos "Tag", à esquerda de "Layer".

Em inglês, "tag" significa "etiqueta". Então, a utilizaremos para etiquetar "Zumbi". É uma forma de marcar o objeto com um padrão e informar ao código quais as balas devem destruir e quais devem destruí-la.

Clicaremos em "Untagged > Add Tag...". Na tela seguinte, clicaremos no botão com sinal de soma (+) e nomearemos como "Inimigo".



Feito isso, em "Hierarchy", selecionaremos "Zumbi" e, em "Inspector", alteraremos a "Tag" de "Untagged" para "Inimigo".



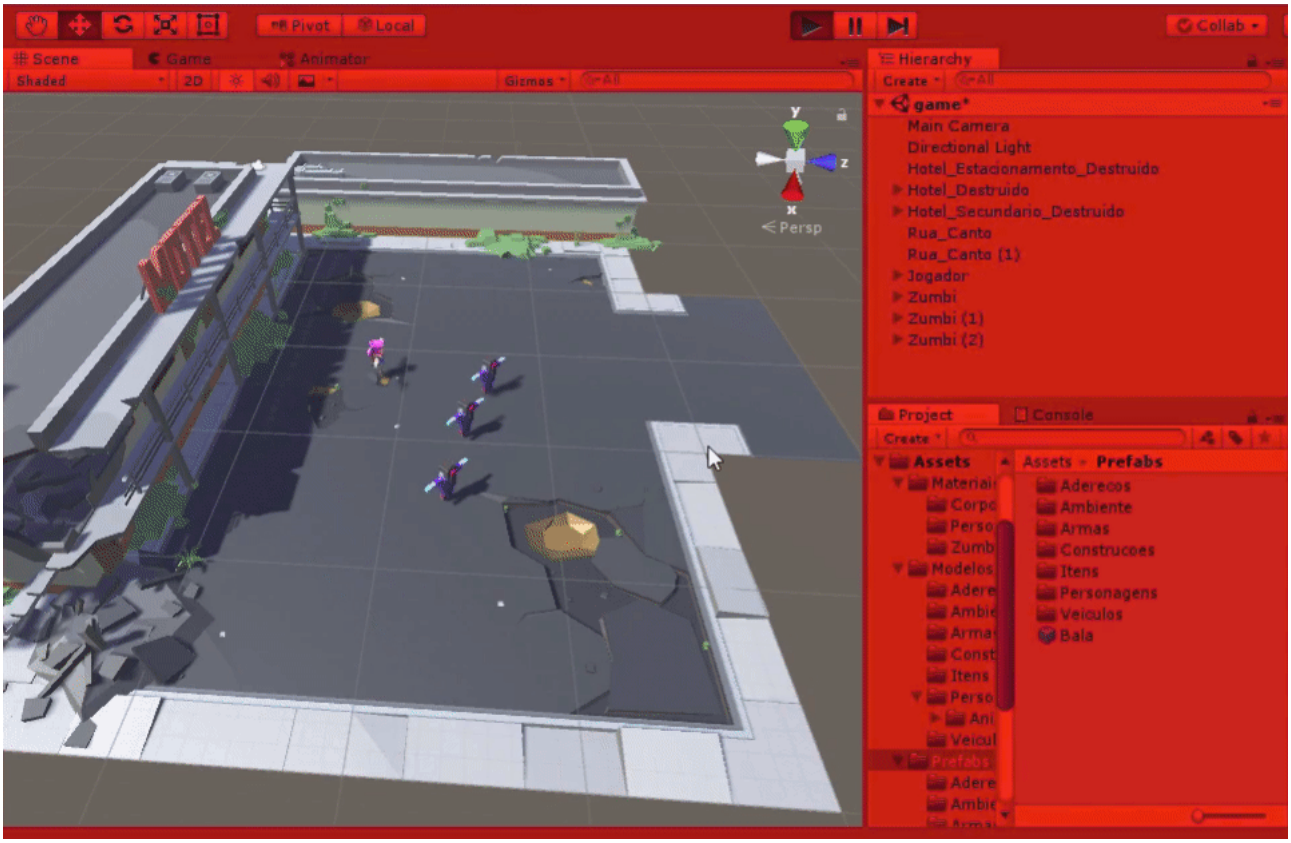
Na sequência, clicaremos em "Apply" para aplicar a etiqueta a todos os zumbis.

De volta ao *script* `Bala.cs`, em `OnTriggerEnter`, acima de `Destroy`, testaremos por meio de `if()` se o objeto com o qual a bala colidiu foi o inimigo ou não. Se for o inimigo (`tag == "Inimigo"`), o objeto será destruído por meio de `Destroy` que estará abaixo, entre chaves (`{}`). Se não, como não declaramos `else`, nada acontecerá. Salvaremos e minimizaremos o trecho de `OnTriggerEnter` da seguinte forma:

```
void OnTriggerEnter(Collider objetoDeColisao)
{
    if(objetoDeColisao.tag == "Inimigo")
    {
        Destroy(objetoDeColisao.gameObject);
    }
}
```

Como estamos fazendo uma **pergunta** por meio de `if()`, notem que utilizamos o sinal de igual duplicado (`==`) para atribuir valor a `tag`. Se estivéssemos afirmando que o valor é um número (`2`, por exemplo), poderíamos usar somente um sinal de igual (`=`). E, lembrando que todas os textos devem ser colocados entre aspas (`"`), inserimos `Inimigo` entre elas.

De volta à Unity, ativaremos "Play". A bala está destruindo o inimigo, mas ela os atravessa e destrói os que estão atrás. Se estiverem em fila, "Jogador" mata todos somente com uma bala.



Precisamos destruí-la após a colisão. Portanto, voltaremos ao *script* `Bala.cs` e, abaixo de `if`, em `OnTriggerEnter` acrescentaremos:

```
Destroy(gameObject);
```

Dessa forma, destruiremos (`Destroy`) a bala (`gameObject` , com a letra "g" minúscula para captar o objeto com esse *script*, no caso a bala). Salvaremos e minimizaremos `Bala.cs` e testaremos na Unity, ativando "Play".

Funcionou. Ao certar o cenário, somente a bala some e ao acertar um zumbi, tanto ele quanto a bala somem. "Jogador" está destruindo seus inimigos.