03

Criando um Gerador de Zumbis

É bem ruim termos que ficar criando Zumbis na mão certo?

Não seria melhor criarmos os Zumbis pelo código? Então vamos criar um novo código com o nome `GeradorZumbis`. Vamos adicionar esse código em um objeto vazio, assim como criamos a ponta da arma. Indo para *Hierarquia* em **Create > Create Empty** vamos jogar o *Script*, esse objeto que vamos nomear como `Gerador de Zumbis`. Lembre-se que, ao lado do nome do objeto, você pode clicar no ícone do quadrado e definir um ícone para este objeto vazio.

Agora no *Script* temos que criar algumas variáveis. A primeira é uma que irá valer para o Zumbi que será criado, a segunda é um contador de tempo, e a terceira é de quanto em quanto tempo vamos criar Zumbis.

```
public class GeradorZumbis : MonoBehaviour {

    public GameObject Zumbi;
    private float contadorTempo = 0;
    public float TempoGerarZumbi = 1;

    void Update () {

    }
}
```

Agora vamos preencher a variável `Zumbi` com o *Prefab* do Zumbi? Não faz mais sentido ter Zumbis na *Hierarquia*, porque eles serão criados por esse gerador. Então vamos aplicar o *Prefab* e deletar todos os Zumbis. Em seguida jogue o **Prefab** na variável.

Se tiver dúvidas é o mesmo que fizemos com a **Bala**.

Agora no `Update` temos que contar o tempo no contador. Qual é a variável que o **Unity** nos fornece para fazer as coisas ficarem em segundos, não é o `Time.deltaTime`?

Então no Update iremos fazer:

```
void Update ()
{
    contadorTempo += Time.deltaTime;
}
```

Lembre-se que o operador `+=` faz a soma e já joga o resultado para ser salvo na variável. Agora estamos contando o tempo em segundos e temos que criar os Zumbis. Vamos então verificar se o tempo contado já chegou no tempo que definimos para criação de Zumbis da variável `TempoGerarZumbi`, vamos criar zumbis e vamos zerar o contador para recomeçarmos a criação.

```
void Update ()
{
    contadorTempo += Time.deltaTime;
    if(contadorTempo >= TempoGerarZumbi)
```

```
        {
            Instantiate(Zumbi, transform.position, transform.rotation);
            contadorTempo = 0;
        }
    }
```

Só que agora precisamos corrigir um problema: quando o Zumbi é criado, ele não sabe mais quem é o **Jogador**. Então, assim como fizemos com o Zumbi, temos que criar uma **Tag** com o nome `Jogador` e colocar essa Tag no nosso Jogador.

E efetuar uma mudança no código do `Start` no método `ControlaInimigo`

```
void Start () {
    Jogador = GameObject.FindWithTag("Jogador");
}
```

Assim, o Zumbi é criado procurando o **Jogador** pela Tag anexada à variável `Jogador` . Esse é um dos usos mais comuns das Tags.

Agora outro ponto interessante é que é bem chato sempre sair o mesmo tipo de Zumbi não é mesmo? Vamos fazer uma modificação para concertar isso?

No código `ControlaInimigo` do Zumbi, vamos fazer algumas modificações no `Start` . Primeiramente, quando vamos no *Prefab* do Zumbi e abrimos ele para ver os tipos, temos só um ativo, o *Prefab* que estávamos usando, certo? Vamos desativar este também. Assim nosso Zumbi não vai ficar com nenhum tipo ativo, e vamos ativar pelo código no método `Start` que acontece quando o Zumbi é criado.

```
void Start () {
    Jogador = GameObject.FindWithTag("Jogador");
    int geraTipoZumbi = Random.Range(1, 28);
    transform.GetChild(geraTipoZumbi).gameObject.SetActive(true);
}
```

Aqui estamos gerando um número inteiro(`int`), aleatório, de 1 a 27, já que o 28 (número máximo) não entra. 27 é a quantidade de tipos de Zumbis que temos. Depois, vamos no objeto do Zumbi e vamos pegar dos objetos-filho esse tipo aleatório e vamos ativar este tipo. Assim, nosso Zumbi sempre sai de uma forma diferente.