03

Gerar posição Aleatória

Nosso jogo está legal, mas agora que temos um cenário maior nossos zumbis parecem inteligentes demais, indo direto para o Jogador.

Vamos resolver isto? Para solucionar esse problema, nossos zumbis tem que ficar fazendo outra coisa além de perseguir o nosso personagem. Para que eles não fiquem parados, que tal a gente deixar eles vagando por aí?

Para isso temos que primeiramente definir para onde eles vão vagar e quando vão deixar de vagar e perseguir nosso personagem.

Vamos então definir que quando o zumbi estiver a mais de `15` de distância do jogador ele vai vagar, e abaixo disso ele irá nos perseguir. Esse valor é um número arbitrário que depois pode ir para uma variável para melhores testes e o código.

```
if(distancia > 15)
{
    Vagar ();
}
else if (distancia > 2.5)
{
    direcao = Jogador.transform.position - transform.position;

    movimentaInimigo.Movimentar(direcao, statusInimigo.Velocidade);

    animacaoInimigo.Atacar(false);
}
else
{
    animacaoInimigo.Atacar(true);
}
```

Agora vamos criar o método vagar:

```
void Vagar()
{
}
```

Temos que definir agora para onde nosso personagem irá quando for vagar, e para isso vamos gerar uma posição aleatória.

Vamos criar um método para isso? Que tal utilizarmos um método com retorno?

Método com retorno funcionam como os métodos normais, eles executam uma ação mas ao final retornam o que mandamos eles retornarem.

Como uma posição é um `Vector3` , temos que criar um método que retorna esse tipo de dado. Além disso, lembre-se de utilizar o método para aleatorizar uma posição que a Unity nos fornece.

Com o Random.insideUnitShere (https://docs.unity3d.com/ScriptReference/Random-insideUnitSphere.html) a Unity nos dá uma forma muito dinâmica de gerar uma posição, mas como este método só nos dá uma posição à no máximo 1 de raio de distância a partir do centro de uma esfera, nós temos que multiplicar este valor para aumentar o raio.

Depois de termos a posição, temos que fazer essa posição ser relativa ao Zumbi e cancelar a altura da nossa posição.

```
Vector3 AleatorizarPosicao ()
{
    Vector3 posicao = Random.insideUnitSphere * 10;
    posicao += transform.position;
    posicao.y = transform.position.y;

    return posicao;
}
```

Agora que temos o método da posição aleatória, falta utilizar essa posição na direção que vamos mover nosso personagem. Mas antes disso, vamos mover a linha da direção `direcao = Jogador.transform.position - transform.position;` no nosso `FixedUpdate` para dentro do `if` de perseguir a direção, já que agora essa linha faz parte da movimentação de perseguir o Jogador.

```
else if (distancia > 2.5)
{
    direcao = Jogador.transform.position - transform.position;

    movimentaInimigo.Movimentar(direcao, statusInimigo.Velocidade);

    animacaoInimigo.Atacar(false);
}
```

Agora vamos utilizar o método com retorno que criamos, e à partir dele vamos gerar uma nova direção de movimento.

```
Vagar()
{
    posicaoAleatoria = AleatorizarPosicao();
    direcao = posicaoAleatoria - transform.position;
    movimentaInimigo.Movimentar(direcao, statusInimigo.Velocidade);
}
```