09

Gerar Zumbi numa posição não ocupada

Transcrição

[00:00] O que a gente tava fazendo aqui? A gente estava gerando uma posição aleatória, vendo se tem zumbi nessa posição, se tiver, a gente gera outra posição, ou seja, a gente está gerando uma posição, perguntando se tem, gera outra e aí essa outra eu não estou testando se tem um zumbi lá ou se não tem, ele simplesmente vai criar um zumbi.

[00:19] Aí eu teria que fazer, if, outro if, outro if, outro if aqui para eu testar várias vezes a mesma coisa, falar: olha, vai testando, vai testando até você achar uma posição que não tem zumbi.

[00:32] Não, na programação a gente pode usar o enquanto, ou seja, enquanto ele não tiver achado uma posição, eu vou continuar gerando posições. O enquanto na programação chama while. Enquanto eu ainda tiver zumbis na região, eu vou gerar outra posição.

[00:53] Gerei outra, vou testar aqui, enquanto tem zumbis na região, gera outra, e gera outra, e gera outra, e fica assim. Ele trava aqui, roda essas duas linhas, trava aqui, até ele achar uma posição que seja boa para a gente poder passar para linha debaixo, de gerar o zumbi.

[01:10] Só que o while aqui na Unity, ele tem um probleminha. Ele é bem passível da gente ficar preso aqui dentro, ou seja, se a gente nunca achar uma posição que o zumbi pode nascer, aqui no nosso caso isso não vai acontecer, porque ele vai ficar tentando até achar, se a gente não achar nenhuma posição que o nosso zumbi pode nascer, ele fica preso aqui dentro e fica rodando, nunca passa para a linha de baixo, e melhor que isso, a Unity trava ainda. Vocês querem ver?

[01:38] Olha só, eu vou criar um while true, para vocês falarem, Henrique, está de balela, a Unity nem trava. Então, olha só, o while true ele vai fazer isso aqui o tempo todo, o while vai funcionar o tempo todo. Eu vou salvar, vou salvar minha senha, garantir que está tudo salvo e vou dar play. Já travou a Unity, por que? Porque ele ficou nesse lugar aqui, nesse frame, nesse quadro, travado, tentando gerar zumbi e não está conseguindo, então ele não passa para o próximo.

[02:05] Se eu tentar tirar o play, está travado, eu tenho que vir e finalizar a tarefa e abrir a Unity novamente, ele tem esse problema. para a gente evitar esse problema, aqui no caso ele iria funcionar, mas para a gente evitar isso de acontecer sempre, eu vou mostrar uma forma para vocês de fazer o while, independente da situação, ele vai funcionar, mesmo que fique infinito lá dentro.

[02:30] Ele só não vai passar para a linha de baixo, mas o jogo não vai travar. É o seguinte, a gente vai mudar o tipo do método aqui, ao invés de void, que faz, que roda esse método pronto, eu vou mudar para IE numerator. Esse método é especial aqui na Unity porque o retorno dele é um yeld, esse retorno aqui é uma palavra que chama yeld, e nesse cara eu posso passar alguma coisa, por exemplo, posso falar: olha, faz o seguinte, retorna, o que está acontecendo aqui?

[03:06] Eu estou falando, olha, eu estou vindo aqui, rodando esse método, rodando o while, tentando rodar uma posição. Não consegui, ao invés de ficar aqui tentando, ele fala, você não conseguiu, calma então. Espera o próximo frame, ou seja, sai desse método, espera o próximo frame, sai daqui e volta de novo, testa de novo.

[03:28] Não deu? Espera o próximo frame então, o yield return null aqui, ele está esperando o próximo frame. Então, nunca vai travar naquele quadro específico que ele não conseguiu sair do ar, ele vai esperar o próximo. Aí ele vai tentar, tenta de novo, não deu? Espera o próximo, não deu, espera o próximo, isso aqui sempre vai funcionar. Essa é uma forma legal de vocês usarem o yield na Unity, só que as corrotinas, isso aqui é conhecido como co rotina, quando você usa o IE numerator.

[03:57] As co rotinas na Unity, elas são chamadas de uma forma diferente, elas não são métodos comuns. Eu tenho que dar um start, corrotine aqui, aí eu posso passar o nome do método com a chamada dentro. Então, eu vou iniciar uma corrotina, que é em gerar novo zumbi.

[04:13] Aí ele vai ficar rodando ela até ele conseguir, quando ele conseguir, ele instância um zumbi, gera um zumbi e para, pronto. De 1 em 1 segundo aqui no caso né, que é o nosso contador aqui. O padrão é 1 em 1, mas pode estar outro número lá no expector.

[04:29] Vamos salvar isso, vamos testar. Não travou, já é um bom sinal e está gerando. Vamos jogar a game aqui para baixo, não travou e está testando. Só que olha só, os zumbis estão nascendo, aqui bem no meio da minha tela. Isso é ruim. O zumbi pipoca lá do nada. Isso é ruim no jogo. Eu tenho que tentar ver, não descobrir de onde os zumbis estão vindo.

[04:58] Vamos gerar os zumbis longe do nosso jogador, além disso, a gente não sabe que região do nosso cenário que está coberta por gerador de zumbi, ou seja, eu tenho um aqui, um ali, não dá para ver tão bem.

[05:13] Eu vou mostrar um truque para vocês. Eu vou usar o método void own draw gizmos da Unity, esse método e da Unity e ele desenha basicamente um ícone no seu editor. Lembra que o sol era um gizmo aqui, que a gente pode aumentar o sol, sol da Luz. Eu posso aumentar ele, olha o gizmo aqui. O sol era um gizmo, esse ícone aqui é um gizmo. Eu vou criar o meu próprio gizmo.

[05:42] Aí eu vou falar que o meu gizmo aqui vai ter a cor amarela, então cor amarela. Aqui eu falei, meus gizmos vão ter a cor amarela, entrei na parte de cor e escolhi uma cor que a Unity já me deu, que é o amarelo. Ele tem amarelo, vermelho, as cores principais eles tem aqui, então você pode utilizar. Agora eu vou desenhar o gizmo, então vou pegar a parte de gizmos e vou falar, olha, desenha um gizmo para mim.

[06:12] Só que esse gizmo vai ser uma esfera, ou seja, uma bola, só que eu quero só as linhas da esfera, o wire, que seria só as linhazinhas. Aí ele vai me falar, onde você quer que eu desenhe essa esfera? No centro, na posição do nosso spawner, ou seja, no meio dele. E quanto que você quer que seja o raio da esfera? Ou seja, a distância do centro até a borda. Eu queria que fosse exatamente essa variável aqui. Por que? Porque aí eu sei até onde os meus zumbis podem nascer.

[06:44] Então eu vou criar uma variável aqui para isso, eu vou criar o privat float distância de geração ponto e vírgula. Vai começar com o valor 3, que é o que já tava lá, depois a gente pode aumentar, 3, e eu vou substituir aquela variável lá embaixo por essa variável, o 3 lá, e vou fechar a linha aqui no desenho ou esfera. Onde? Onde o meu spawner ta.

[07:14] Qual que é o raio da esfera? A distância geração que vai 3. Se eu salvar isso aqui, deixa a Unity computar. Olha lá os gizmos, eu posso selecionar agora esses cara bem mais praticamente, bem mais prático, porque eu tenho agora uma forma para eu clicar e eu posso ver, esse lugar aqui não está preenchido, então tem que preencher. Ah, esse lugar aqui também não, então tem que preencher.

[07:35] Além disso, eu posso aumentar, por exemplo, a distância de geração e ver o impacto que isso está causando. Aí agora os zumbis não pipocarem na tela, não aparecerem na tela. Vamos criar uma outra variável aqui, que vai ser um float também, vai ser distância do jogador para geração, aí eu vou colocar aqui 20, uns 20 provavelmente vai dar certo.

[08:01] Se você quiser, você pode testar isso, diminuir o valor, aumentar, e ver, notar se isso está funcionando. Pode deixar até público aqui e notar se está legal, se os zumbis ainda tão sendo criado dentro da tela. Eu espero que uns 20 já fique legal.

[08:15] Aí eu vou fazer o seguinte, eu só posso gerar zumbis, se eu tiver a 20 de distância do meu jogador, porque aí eu só posso gerar zumbis fora da câmera do meu jogo. Então, eu vou falar, olha, eu só posso gerar zumbis se a distância entre

mim e o jogador, vector 3 ponto distance, entre mim, transform ponto position e o jogador, aí eu não tenho aqui a distância do jogador. Cadê, eu tenho a variável de jogador?

[08:45] Não tem, então vamos criar. Privat, game object, jogador, vamos criar um start aqui, jogador é igual gameobject ponto find with tag, qual a tag? Jogador e pronto, pronto. Achei meu jogador. Então, a distância entre eu e o jogador ponto transform ponto position.

[09:24] Essa distância que o cálculo dentro desse parêntese, ou seja, aqui o vector 3 ponto distance, ele está calculando, aí ele está calculando dentro desse parêntese. Aí eu tenho um outro parêntese do if, ou seja, a distância desse cálculo de cá, tem que ser maior do que a distância do jogador para geração.

[09:43] Vamos dar um enter aqui nessa linha para ficar um pouco mais fácil de ver? Outro enter, pronto. Aí eu vou abrir esse if e fechar e eu só vou criar um zumbi, se a minha distância tiver ok, ou seja, se a minha distância estiver ok aqui dentro. Prontinho, vamos fazer um teste? Espera a Unity computar, vamos dar play, vamos ver se o jogo está criando zumbis aqui dentro da nossa tela. Não tá, ele só está criando zumbis lá longe. E aí note você que ele está criando muito zumbis. Então a gente diminuir essa quantidade.

[10:23] Então note você que você tem que balancear um pouco seu jogo, mas essa parte do spawner já está ok, basta você não colocar o spawner dentro do cenário também, por exemplo, colocar dentro dessa casa. Porque senão a física vai dar errado e ele vai pipocar aí para fora.

[10:40] Então, ele só testa se tem um zumbi naquela região, se tiver ele não vai criar ali. Cenário a gente não fez, então a gente pode se você quiser, colocar todas as construções numa layer de cenário e aqui no seu spawner, pode falar, olha, testa se tem um cenário e um zumbi, você selecionar duas, aqui no caso eu selecionei chão e zumbi.

[11:04] Aqui no caso ficou só zumbi. Então, você pode criar uma layer, colocar o cenário dentro dessa layer e testar os dois, porque às vezes o seu spawner está um pouquinho para dentro do nosso cenário. Quando eu digo spawner, é o nosso gerador, é porque no quesito de jogos, spawner é o nome comum do nosso gerador de zumbis beleza? Ele fica dando spaw nos zumbis.

[11:25] OK? Assim que a gente faz esse tipo de coisa e vocês viram uma coisa bem legal, que é os gizmos aqui, que é bem útil para a gente usar e ver que parte do cenário que não está preenchido.