11

Interfaces no C#

Transcrição

[00:00] Vamos fazer mais uma refatoração e essa vai ser muito importante. Olha só, eu vou abrir aqui o script da bala. Na bala, eu tenho aqui que a bala quando ela colide com o inimigo, eu destruo o inimigo e eu toco o som de morte. Só que olha só, eu tenho na bala uma destruição ao inimigo e vai tocar o som de morte quando o inimigo morre. Não seria melhor essas coisas estarem dentro do inimigo? Então seria ideal a gente fazer a mesma coisa que a gente fez com jogador.

[00:27] Falar, olha o inimigo, você tem que tomar um dano e se esse dano for o suficiente você vai morrer, e aí sim, você vai fazer essas duas coisas. Então, fazer exatamente o que a gente fez com o jogador, só que aí eu vou ter um método tomar dano no meu jogador, vou ter um método tomar dano no inimigo, não teria uma forma da gente unir essas duas coisas, juntar isso e fazer uma convenção?

[00:47] Todo mundo que toma dano, tem que usar isso aqui, sempre tem na programação. Então, como que a gente pode fazer isso aqui no C #? Eu vou declarar uma interface, uma interface é uma forma de eu falar, olha, quem vai tomar dano tem que ter os seguintes métodos, que é por exemplo o método void tomar dano.

[01:08] Então, vamos criar um script novo aqui, C # script, toda a interface aqui no C #, o padrão ela começa com o nome I, então I maiúsculo, aí eu vou dar o nome para ela de matável, porque todo mundo que toma dano vai ser matável.

[01:22] Vou abrir essa interface, vou dar um reload aqui na solução, vamos abrir aqui, já está tudo OK. Deletar tudo e vou até tirar essa parte de mono behavior, por que? Porque isso aqui não vai ser uma classe, não vai ser um código que eu vou utilizar no meu jogo, eu vou simplesmente criar uma convenção, olha, todo mundo que for morrer, tem que ter o método tomar dano.

[01:48] Para isso eu não vou utilizar uma classe, um código normal, um script da Unity, eu vou usar uma interface. Então, declarei a interface I matável. Aí aqui dentro eu digo, olha, quem for morrer tem que ter quais métodos? O void tomar dano que recebe o inteiro, com o nome dano, ponto e vírgula aqui.

[02:11] Não precisa colocar abre e fecha chaves porque eu só estou declarando esse método, eu não vou fazer nada com ele. Estou dizendo, olha, todo mundo que for morrer tem que ter o tomar dano. Então, eu não vou utilizar ele, eu não vou criar código aqui dentro, beleza?

[02:26] E além disso eu vou criar mais um, que é para vocês verem essa implementação de uma forma né, com 2 métodos, para a gente fazer algumas vezes isso. Então, todo mundo que for matável, vai ter que ter um método tomar dano e o método morrer.

[02:40] Como que a gente diz agora que o meu jogador por exemplo, ele é matável? Da mesma forma que a gente fez a parte de herança aqui, com os dois pontos monobehavior. Só que ao invés de eu colocar aqui dois pontos I matável aqui, não, a herança já está aqui, é só colocar vírgula e dizer de quem mais que eu quero puxar, é de I matável.

[03:05] Só que herança de classe, ou seja, o mono behavior se eu botar o mouse em cima ele é uma classe. Herança de classe, ou seja, o controla jogador puxar de mono behavior, só pode ter uma, agora interfaces eu posso ter vários, eu posso vir aqui, vírgula colocar outra, vírgula colocar outra.

[03:22] Então, podia ter uma interface só para a galera que toma dano, e uma Interface para a galera só que morre, duas diferentes. Mas não, eu quis ter uma só, o I matável. Aí ele tava dando erro aqui por que?

[03:33] Porque ele fala, olha, esse código já tem o método tomar dano, mas ele não tem o método morrer. Então, eu tenho que ter também o método morrer. está vendo como é útil essa parte de interface? Eu falei, todo mundo for morrer, tem que ter o tomar dano e o método morrer lá na interface. Eu vou criar aqui o public void morrer e vou falar que quando o meu jogador morrer, eu vou fazer isso aqui, que é essa parte aqui. E toda vez que a vida for menor do que 0, eu chamo morrer.

[04:02] Falei que toda vez que a vida for menor ou igual a zero, eu rodo o método morrer, e o método morrer faz a mesma coisa que tava lá dentro, você vai subir lá e vai ver que não está com erro nenhum , roda para a Unity aqui, não tem erro nenhum. Vamos fazer isso no nosso inimigo agora? Vamos falar que o nosso inimigo, vou descer essa chave aqui, nosso inimigo também é matável.

[04:22] Só que ao invés de eu ir lá e ter que criar os métodos na mão, olha só, eu vou clicar com botão aqui Ctrl. e aí ele fala: olha, eu posso implementar essa interface para você aqui. Aí se eu clicar nesse botão implement interface, ele mesmo já cria os dois métodos, o método tomar dano e o método morrer. Já fez isso para mim, vou tirar isso aqui que ele gerou aqui dentro, pronto, vamos subir.

[05:01] Ele também colocou essa parte de system aqui, que é o que está dando erro aqui embaixo. Então, eu posso simplesmente remover isso aqui, tirei essa parte desse system, eu não vou usar essa parte de sistema. estou usando as coleções, os genéricos e a Unity. Então, tiro system lá que tava dando erro no random aqui, porque tem um random da Unity e tem um random do System, aí não sabia qual que ele ia utilizar.

[05:29] Agora eu tenho o método tomar dano e o método morrer,eu vou fazer a mesma coisa que eu fiz com o meu jogador. Nosso zumbi tem uma vida, então vamos lá no zumbi, nosso zumbi tem uma vida, vamos lá no nosso zumbi. Só que eu vou colocar a vida inicial dele 1, porque aí ele tem 1 de vida, eu rodo qualquer dano e ele vai simplesmente morrer, a vida inicial dele 1.

[05:51] Aí eu vou falar, olha o zumbi, você vai ter o dano, então toda vez que eu pegar e chamar o método tomar dano, eu vou pegar a sua vida atual, no status do inimigo ponto vida, ela vai ser menos ou igual a dano, igualzinho o jogador.

[06:05] E se essa vida for menor ou igual a zero, eu vou rodar o método morrer. Agora é só a gente pegar essa parte da bala e jogar para cá. Então, eu vou pegar essas duas partes aqui, essas duas linhas Ctrl + C, vou jogar esse aqui para parte de morrer. O destrói, eu não preciso do objeto colisão.

[06:32] Vou só destruir este objeto, então gameobject, eu vou destruir eu mesmo. E o play one shot eu não tenho a variável de som de morte, então vou criar essa variável aqui. Vou criar um public audio clip som de morte, aí o erro vai sumir lá de baixo, mas eu tenho que vir aqui no zumbi, deixar ele compilar isso aqui e falar, quem que é o som de morte aqui, beleza? Quem que é o som de morte?

[06:59] Eu vou lá jogar o som aqui, clicando na bolinha aqui do lado, o som de morte vai ser o som morte zumbi 01. Pronto, esse som aqui é o som de morte do nosso zumbi. Então, quando o nosso zumbi tomar dando, ele vai pegar a vida e tirar o dano, se a vida for menor ou igual a zero, ele morre, se ele morre ele é destruído e toca o som de morte.

[07:20] Em teoria está igual, certo? Só que agora na bala, eu tenho que fazer ele tomar dano. Então, vou vir aqui e vou falar, olha, o objeto de colisão, que no caso dentro desse if aqui, é o nosso inimigo, o nosso zumbi, eu vou pegar o componente dele que é o controla inimigo e vou rodar o método tomar dano.

[07:40] Passando um valor qualquer aqui, porque se ele tomar 1 de dano ele já morre, então eu vou passar 1, ele vai tomar 1 de dano. Tomei 1 de dano. Vamos testar isso aqui para ver se está idêntico? Vamos lá, salva. Pronto, nosso zumbi já está morrendo, só que dessa forma a gente tem muito mais flexibilidade.

[08:00] Porque se eu colocar a vida do zumbi inicial 3 aqui, a bala da 1 de dano, o zumbi vai morrer com três tiros. Eu tenho muito mais flexibilidade dessa forma e é muito mais rápido de implementar os métodos que eu preciso, quando eu preciso matar um personagem, bem bacana essa parte de interface.