06

Refatorando a Jogador

Transcrição

[00:00] Vamos continuar trabalhando na nossa refatoração, só que agora no nosso jogador? Então vamos abrir o script de controlar jogador, esse script aqui já está bem grandão, e a gente tem que tirar algumas coisas aqui, vamos começar com a parte de movimentação?

[00:17] Nessa parte de movimentação aqui, é basicamente o que eu tenho no meu movimenta personagem, eu posso utilizar o método de movimentar, deixa eu fechar o do inimigo aqui né, mas olha só essa parte de baixo, essa parte de baixo aqui de rotação com o mouse é diferente do que eu tenho aqui em rotacionar.

[00:35] Essa parte de rotação com o mouse é só do nosso jogador, nenhum outro personagem no jogo vai utilizar essa parte sem ser o jogador. Então, a gente tem que criar um código separado para ele.

[00:50] Vamos primeiro extrair essa parte aqui, então vamos criar um método aqui chamado movimento jogador, aí no nosso jogador a gente vai jogar esse cara, movimento jogador, vamos aplicar o prefab né, que é sempre bom aplicar o prefab toda vez que a gente faz uma modificação, vamos abrir o movimenta jogador. Apaga tudo e vamos retirar do nosso jogador, aquela parte de rotação com o mouse, que é tudo isso aqui.

[01:27] Vou dar um Ctrl + X, que aí o Ctrl + X eu tiro e colo volta, aí fica no meu copiar para eu poder colar de volta com Ctrl + V. Então, eu dei o Ctrl + X, que eu tirei e copiei, e aqui eu vou criar um método void, tem que ser público inclusive, lembra que tem que ser público, rotação, jogador e vou colar isso aqui.

[01:55] Extrai a parte de rotação do jogador, só que o seguinte: aí você pensa, então eu tenho um script de movimento jogador e eu tenho script de movimento personagem. O jogador vai ter que ter os dois. Porque a movimentação vem de um cara e a rotação vem de outro cara, de outro script.

[02:14] Isso é meio ruim, eu ter dois scripts de movimentar. Como que a gente pode tratar isso melhor dentro da Unity? A gente pode fazer um esquema aqui que é conhecido como herança, que funciona na linguagem C #. Ao invés de eu ter o movimento personagem, eu quero que o meu movimento jogador tenha tudo que o meu movimento personagem tem, só que isso aqui de diferente.

[02:40] Aí eu poderia vir aqui, copiar tudo, colar lá para o outro, mas aí eu teria códigos iguais do outro script, então eu teria dois códigos iguais no mesmo lugar, o que eu já falei que é ruim.

[02:53] Como que funciona a herança? A herança é o seguinte, está vendo aqui no movimenta personagem que eu tenho dois pontos mono behavior? Isso aqui é uma herança, eu estou falando para o movimento personagem que ele herda de mono behavior, ou seja, um mono behavior é pai dele,e ele é filho de mono behavior.

[03:11] É por isso que a gente pode usar o awake, que a gente pode usar o start, pode usar o update. Porque os nossos scripts, o pai dele é o mono behavior, e o mono behavior que trata esses caras, que tem o awake, que tem o update, que tem o start, tudo isso está no mono behavior. Então, o que eu vou fazer aqui?

[03:29] Eu vou fazer, ao invés do movimento jogador herdar de mono behavior, eu vou fazer ele herdar de movimento personagem, que é o outro código de movimentação, ou seja, agora ele tem tudo que o movimenta personagem tem, inclusive as partes de mono behavior, porque o movimenta personagem é filho de mono behavior e o movimento jogador é filho de movimenta personagem. Eu vou criar uma cadeia aí de heranças.

[03:55] Esse cara tem tudo o que o outro tem, por exemplo eu posso chamar aqui, aqui dentro do rotação, eu posso chamar o método rotacionar, porque é como se esse método tivesse dentro desse script, aí que é um conceito bem legal dessa parte de programação, que é o conceito de herança, que é uma das premissas da orientação a objetos.

[04:18] Vamos agora corrigir os erros então desse cara, vamos corrigir o erro na máscara de chão aqui porque eu vou receber isso aqui como um parâmetro, porque essa variável está lá no script do controla jogador.

[04:30] Então, layer mask, aí eu posso dar o mesmo nome máscara chão para cancelar o erro aqui, e essa parte do rigid body jogador aqui? Eu preciso de um rigid body aqui, quem tem um rigid body? O movimenta personagem tem um rigid body e consequentemente, eu também tenho um rigid body né, é o movimenta jogador.

[04:49] Só que, eu não posso chamar esse rigid body aqui, porque ele é privado, mesmo com herança as variáveis privadas são só desse script. Mas essa parte aqui não é uma parte de rotação? Então posso utilizar o método rotacionar, que é público. Então, tira isso aqui, passa o método rotacionar, que eu vou chamar ele igual como se eu tivesse declarado esse método aqui dentro, ou seja, eu não preciso variável nenhuma, é só usar rotacionar e que direção que eu vou rotacionar?

[05:22] Posição da mira jogador, que é a variável que a gente criou aqui em cima. Salvo isso aqui, vamos lá no controla jogador e vamos criar uma variável que recebe o script de movimento jogador para a gente poder utilizar ele. Então, privat, movimento jogador, vou chamar de meu movimento jogador ponto e vírgula, vou receber ela aqui, meu movimento jogador igual a getcomponent movimento jogador.

[05:58] Pronto, já estou com a variável aqui, vamos olhar se o nosso jogador já tem script, já tem. Então, vamos agora no fixed update chamar aquela parte de rotação, então meu movimento jogador ponto rotação jogador, não é o rotacionar. Eu vou passar o parâmetro que eu tenho que passar, que é a máscara.

[06:19] Qual máscara que eu vou utilizar para fazer a rotação? Eu já tenho essa variável, ela chama máscara chão, ela já está nesse script. Inclusive, ela já está preenchida, que é a que a gente utilizava anteriormente. estou aqui com essa com essa variável, com esse método chamado. Vamos dar o play e testar se a rotação está funcionando tudo ok? Rotação funcionando tudo ok.

[06:41] Vamos fazer agora a mesma coisa com a movimentação, o meu movimento jogador não tem tudo que o movimento personagem tem, então eu posso chamar meu movimento jogador ponto movimentar e passar a direção e a velocidade vai ser velocidade, a variável de velocidade. [07:04] A direção é essa variável que a gente definiu aqui em cima. Estou passando a direção e a velocidade, salva, sempre é bom cada vez que você fazer uma alteração, você ir salvando e testando. Faz pequenas alterações, salva e testa.

[07:17] Porque aí você garante que quando der um erro você sabe onde é que ele tá, que foi a última coisa que você fez. Isso é bem legal na programação, de vocês aprenderem, a movimentação está funcionando. Então, a gente já refatorou a movimentação.

[07:31] Vamos refatorar a animação agora? O nosso jogador não tem na parte de animação, essa parte de ataque. Mas ele tem essa parte de movendo. O que vocês acham da gente refatorar a animação e tirar tudo isso aqui daqui de dentro, a gente jogar lá para parte de animação.

[07:50] Então o que eu posso fazer? Eu posso fazer o seguinte, ao invés de eu tratar o movendo como uma variável booleana, ou seja, que vale verdadeiro ou falso, eu posso tratar ela de outra forma. Vou mostrar para vocês o seguinte, vamos no nosso jogador e vamos no animator dele, aqui está o movendo, ele é uma variável booleana. Quando está falso, vai para ideal, quando está verdadeiro vai para correr.

[08:15] E se eu criar uma variável do tipo float aqui? Eu vou deletar o movendo agora, clicar com o botão direito aqui, delete, aí ele vai me dar um erro. Você não pode deletar o movendo porque eu estou usando ela nas seguintes transações, tudo bem, deleta de qualquer forma.

[08:34] Esse novo float, eu vou chamar ela de movendo, por que? Eu vou fazer o seguinte, ao invés de eu tratar isso aqui no código, eu vou jogar essa parte, essa lógica, para o nosso animator. Eu vou falar, olha animator, eu recebi um valor aqui, trata ele você mesmo. Eu vou falar para a Unity tratar isso, a gente não tem essa responsabilidade de saber em qual animação que a gente tá.

[09:01] A unity não tem um lugar que lida com as animações? Eu vou passar essa responsabilidade para a Unity, quando a gente fez da primeira vez, a gente tava aprendendo e queria fazer de uma forma que fosse lógico no pensamento, mas agora que a gente já entendeu, vamos jogar essa responsabilidade para a Unity tratar. De ideal para correr, eu tenho que passar para correr, quando o meu valor de movimento que agora é um valor, é um número float, quando o meu valor de movendo for maior do que 0.

[09:32] E de correr para ideal, vai ser quando o meu valor de movimento for less, menor do que 0. Só que e se for 0? Porque eu vou fazer o seguinte, o movendo vai ser um número de 0 até um valor qualquer, só que o mínimo dele é 0, então se for 0, não tem o número ser menor do que 0, porque o valor mínimo é 0. Então, teria que ser igual a zero, ou menor e igual a 0, menor do que zero, se chegasse a negativo, não vai ter como.

[10:03] para tratar isso eu fazer o seguinte, o movendo, ao invés dele ser um valor, aqui ele vai chegar a zero. Vou fazer o seguinte. Olha, ao invés de ele ser um valor que vai de 0, maior do que 0 passa para correr e menor do que 0 passa para ideal. Eu vou fazer o seguinte, se o movendo for maior do que 0.1, aí eu tenho certeza, o movendo é maior do que 0.1, então é certeza que eu quero que ele corra, passa para correr.

[10:29] E se o movendo for menor do que 0.1, passa para ideal. Então, agora eu garanti que se eu chegar no valor 0, 0 é menor do que 0.1, então fica na ideal. Então, de correr para ideal, menor do que 0.1, de ideal para correr, maior do que 0.1.

[10:51] Eu já falei que a gente pode deletar isso aqui, mas na verdade eu não vou deletar, eu vou copiar essa linha e vou criar um método aqui no animação personagem. Todo personagem que precisar andar agora, a gente vai usar essa mesma coisa, o valor movendo maior ou menor do que 0.1. Eu vou criar um método público aqui, que eu vou chamar de movimentar e vou passar uma variável aqui que no caso vai ser um float, que eu vou chamar aqui de valor de movimento.

[11:25] Vamos abrir e fechar chaves, declarei o movimentar. Eu vou fazer o seguinte, colei aquela parte que a gente tinha copiado, que era so set boo. O animator eu já tenho aqui, é a variável meu animator. O set boo eu vou ter que trocar, não é mais um passe valor para uma booleana, eu vou passar valor set para um float porque aquela variável é do tipo float.

[11:57] O nome dela é movendo e aí o valor que eu vou passar, vai ser o valor de movimento, beleza? Agora sim, eu posso ir lá no meu controla jogador, apagar tudo. Vamos aqui tirar essa parte de animator e do rigid body também, que a gente não precisa mais. A gente refatorou isso.

[12:18] Vamos criar uma nova variável, que vai valer o script animação de personagem, que é onde eu tenho um método de movimentar. Aí vou colocar assim, animação jogador, tirar essas duas linhas aqui que deram erro, que são as do start do update, que a gente já refatorou, que a gente já tirou daqui, a gente não vai precisar deles mais. Animação jogador vai ser igual a getcomponent animação personagem. Pronto, é isso mesmo, ponto e vírgula.

[12:53] Vamos garantir que o nosso jogador tem o script de animação personagem, se não vai dar erro, então joga ele para cá. Vamos aplicar o prefab, apliquei legal. Então, já posso utilizar essa parte de movimentar, vou vir aqui e vou fazer essa parte de movimentar. Vou falar o seguinte, olha, animação jogador roda o método movimentar, que é o

método que faz o movimento e nele eu vou passar um valor que vai ser o valor de movimento, eu vou jogar o valor da direção.

[13:28] Só que a direção é um vector 3, só que a direção que cuida, porque antes a gente falava, olha, se a direção não for 0, quer dizer que eu estou me movendo, se a direção for 0, quer dizer que eu estou parado. Então, mesmo a direção sendo um vector 3, é ela que cuida da nossa movimentação, que se ela for 0 eu estou parado.

[13:47] Só que como que eu passo uma direção, esse valor de direção que é o vector 3, sendo que lá eu tenho um float aqui e eu não tenho aqui nos meus parâmetros uma variável do tipo vector 3, eu não tenho isso. Então, o que eu vou fazer? Vou falar o seguinte, olha, eu vou passar a direção, mas eu vou passar a magnitude. O que significa a magnitude de um vector 3?

[14:13] Significa o tamanho desse vector 3, suponhamos que ele seja 1 em X, 0 em Y e 0 em z, ele tem tamanho 1, porque ele vai pegar as três casas dele e vai falar, o tamanho dele de 0 até esse valor é 1 porque ele partiu de 0 a 1, que é o valor somente em X, que é o único valor que ele tinha.

[14:36] Aí ele calcula a magnitude, então é o tamanho do meu vetor direção, é quanto que ele vale. Vai ser um número que vai de 0 até o valor que a gente vai gerar com essas variáveis aqui, mas passou de 0.1 já é para se mover.

[14:54] Vamos salvar isso, vamos esperar a gente compilar isso. Vamos dar um clear aqui, para limpar esses erros, para ter certeza que não ficou nenhum. Limpei, se os erros sumiram é porque eles não eram importantes, foi um erro só momentâneo, e aí eu limpei e ele sumiu. Se o erro ficar é porque você tem que corrigir.

[15:13] Vamos dar um play, a rotação está funcionando, a movimentação está funcionando e a animação está funcionando, então toda vez agora que eu quiser movimentar, a gente vai passar esse comportamento para o nosso animator.

[15:27] Eu falei, olha animator, eu tenho valor float aqui, é esse valor, você que vai tratar. Eu recebi o valor de movendo, você que vai pelo movendo, trocar as animações. Eu não tenho que me preocupar mais com isso no código, quem vai se preocupar é o animator da Unity, a gente passou a função para ele. Isso que é legal de ter um animator aqui.

[15:50] A gente fez exatamente isso e agora todo script que precisasse movimentar rodar em Bastante movimento, a gente pode fazer aquela mesma parte de movimento do nosso jogador e rodar método movimentado, beleza? Salva tudo aplica todos os prefeitos e salva sua semana e vamos continuar essa parte de refatoração.