

Ajustando Animação - Parte 2

Transcrição

[00:00] Agora que a gente já discutiu toda a teoria matemática por trás do que a gente quer fazer, vamos aplicar isso. Então, a gente queria mexer no nosso controle de animação e o que a gente estava querendo fazer é, o que a gente discutiu que a gente ia fazer, a gente só queria verificar a minha animação, o meu eixo para direita ou para esquerda, minha animação na direção x, quando o seno do ângulo do vetor direção, que é passado aqui para a gente, for maior do que o cosseno mínimo.

[00:39] Então, eu vou ter aqui um if, if, se o meu cosseno desse ângulo for maior do que o `this.cosseno` mínimo. Aí sim, eu vou querer verificar o meu eixo X, então o cara lá, a gente vai ter que pegar calcular o ângulo do vetor direção, esse vetor pode se transformar em um ângulo e a gente vai pegar o cosseno dele e verificar se ele é maior que o cosseno mínimo. Lembrando que esse cosseno é o cosseno do ângulo espelhado lá no primeiro quadrante. Porque a gente sempre quer aquele valor positivo, e para o direção.y a gente vai usar seno, então a gente vai também calcular o seno.

[01:09] Se esse seno for maior do que um seno mínimo, então `this.senomínimo`, aí sim eu vou querer verificar, saber se a minha jogadora eu tem que ir para cima ou para baixo. Então, eu tenho o seno, eu tenho o cosseno, e a gente precisa calcular esses dois valores, a gente precisa pegar o ângulo do nosso vetor e espelhar ele lá no primeiro quadrante. Então, para a gente calcular o ângulo, eu vou criar uma variável ângulo, que vai ser igual ao meu vetor direção transformado em ângulo, e como que a gente faz isso?

[01:47] A gente usa uma função da biblioteca de matemática do JavaScript, que chama `atan2`, então eu vou chamar `math.atan2` e passar os valores do nosso vetor. O primeiro parâmetro desse método é o direção.y, é o valor de y do nosso vetor. O segundo é a direção.x, então o `math.atan2` é um método da biblioteca de matemática do JavaScript, que pega um vetor e transforma ele em um ângulo, então a gente agora tem nosso ângulo. E eu precisava calcular o cosseno e o seno dele.

[02:25] Então, eu tenho uma variável de cosseno que vai ser igual ao `math.cos`, então é o jeito de a gente calcular o cosseno de um ângulo dentro do JavaScript, `math.co` do meu ângulo né, eu quero calcular o cosseno deste ângulo. Como a gente está pegando. O nosso mouse pode estar em qualquer lugar, o nosso vetor direção pode estar em qualquer lugar, apontando para qualquer direção. Eu quero espelhar o ângulo que foi retornado para gente lá no primeiro quadrante.

[02:52] Para fazer isso, a gente vai pegar o valor absoluto do cosseno e do seno, ou seja, eu ignorava o sinal. Então, eu vou falar que o meu cosseno vai ser igual a `math.abs`, ou seja, vou pegar o valor absoluto do meu cosseno, eu vou ignorar o sinal desse número. Aí sim que vai retornar um valor positivo assim. A mesma coisa para o meu seno, então `let seno = math.sin`, o jeito de calcular o seno de um ângulo. Então, `math.sin` passando o ângulo que a gente quer calcular.

[03:21] O nosso seno vai ser igual, a gente vai também tirar o valor absoluto dele, então seno vai ser igual `math.abs`, que é o jeito da gente tirar o valor absoluto, a gente vai ignorar o sinal desse desse número. Então, eu calculei meu ângulo através desse `math atan2`, eu peguei um vetor, transformei ele em ângulo, calculei o cosseno desse ângulo e calculei o seno desse ângulo, e espelhei isso lá para o número quadrante, porque eu quero um valor entre 0 e 90, um ângulo entre 0 e 90 né? Então, a gente está espelhando qualquer ângulo que a gente pegar aqui, lá para o primeiro quadrante.

[03:55] E aí eu vou verificar se o cosseno desse ângulo espelhado no primeiro quadrante, vai ser maior do que o cosseno mínimo. Se isso for verdade, eu vou verificar se a minha animação da jogadora tem que ir para esquerda ou para direita,

e aí sim eu vou usar o direção.x maior que 0 vai me retornar para direita, e direção.x menor do que 0 vai me retornar à esquerda.

[04:17] É a mesma ideia para o seno, eu tenho o seno do meu ângulo espelhado lá no primeiro quadrante, se ele for maior do que o seno do ângulo mínimo que a gente calculou, eu vou verificar a direção.Y , se ele for maior que 0, a minha animação tem que ir para cima, se for menor do que 0, minha animação tem que ir para baixo.

[04:34] Então, vamos dar um espaço aqui para a gente ter um padrão, deixar sempre um espaço do lado dos nossos operadores, e a gente precisa agora descobrir qual que é o seno e cosseno mínimo. A gente tem essas propriedades que a gente tem que criar. Então, vamos voltar lá e ver aquele último slide para lembrar como a gente tinha feito? Então, no nosso último slide a gente tinha isso, a gente tinha falado que a gente ia dar uma folga de 20 graus, então entre 0 e 20 graus, eu tinha o meu ângulo ou a minha animação correndo só no direção.x , eu só ia verificar o eixo x.

[05:08] Entre 20 e 70, a minha animação tinha que estar na diagonal, então eu ia verificar x e y, entre 70 e 90, eu ia verificar só o eixo Y. Com isso, a gente tinha visto que todos os ângulos maiores do que 20 tinham o seno maior do que o seno de 20, então a gente vai usar o seno mínimo igual ao 0, o nosso ângulo 0 mais o ângulo que a gente escolher. Porque a gente colocou o 20 aqui como exemplo, mas eu podia colocar qualquer ângulo aqui e o 90, menos 20 grau que é a forma que a gente escolheu, vai ser 70.

[05:43] Então, 70, todos os ângulos abaixo disso, que é onde a gente quer verificar o nosso direção.x , tem o cosseno maior do que o cosseno de 70. Então, a gente vai transformar isso aqui em código, a gente vai criar essas variáveis seno mínimo e cosseno mínimo, partindo desse ângulo de 20 graus que é o que a gente usou nos exemplos. Então, vamos voltar lá para o código, eu vou ter um ângulo que a gente vai usar, então o ângulo diagonal.

[06:08] Vamos criar essa propriedade aqui, então ângulo diagonal. E por que é ângulo diagonal? Porque a partir do momento que o meu mouse tiver acima ou abaixo desse ângulos, se ele tiver com essa folga, se ele passar dessa folga, a nossa animação vai ter que rodar na diagonal. Então, o meu ângulo na diagonal vai ser um float, vai ser um número. A gente também vai ter o nosso seno mínimo, que também vai ser um float, cc.float , e o cosseno mínimo, ele vai ser um float, cc.float .

[06:51] A gente tem aqui as propriedades criadas, a gente precisa atribuir elas, então quando a gente vai atribuir elas? Assim que o meu script for lido, for carregado na memória. Então, o nosso ângulo diagonal, a gente tinha falado que ia ser 20 graus, como um exemplo, então $\text{this.angulodiagonal}$ vai ser igual a 20, então a gente quer 20 graus ali de folga. O nosso seno mínimo, então o nosso this.senomínimo vai ser igual, vamos voltar no slide, o seno mínimo é o seno de 20 graus, porque é 0 mais a fórmula que eu dei.

[07:27] Então, o seno mínimo é igual ao meu ângulo diagonal, porque 0 mais os 20, é igual ao 0 mais o ângulo diagonal, então $\text{this.angulodiagonal}$. Só que acontece que a biblioteca de Javascript, ela trabalha sempre em ângulos em radianos, e a gente está usando ângulos em graus, então a gente precisa converter de graus para radianos. Como essa é uma operação muito comum, a cócs já tem uma função que faz isso. Então, eu posso chamar o $\text{cc.degreesToRadians}$, passando o ângulo que eu quero transformar. Então, ele vai pegar esse ângulo diagonal, no nosso caso 20 graus, e transformar em radianos, e vai falar que esse é o meu seno mínimo.

[08:19] Mas eu não quero o ângulo como sendo seno, eu quero o seno dele, então math.sin , eu vou tirar o seno desse ângulo aqui. Eu vou fazer a mesma coisa para o cosseno mínimo, então $\text{this.cossenomínimo}$ vai ser igual, o cosseno mínimo era o cosseno de 70, ou seja, 90 que é o eixo Y, menos a fórmula que a gente deu, então 90 menos a folga, $\text{this.angulo diagonal}$. 90 graus vai ser exatamente o eixo Y, se ele tiver a 90 graus eu estou exatamente olhando para cima, menos a folga que a gente quer dar. Então, 90 menos o ângulo diagonal. A gente também tem que conversar isso para radianos.

[09:03] Então, $\text{cc.degreetoradians}$ de 90 menos o meu ângulo diagonal, porque assim o ângulo diagonal eu posso alterar a qualquer hora. Só que eu também, eu quero o cosseno disso, então eu quero `Math.cos`, abrir e fechar parêntese no final. Então, o meu seno mínimo é igual ao seno do ângulo diagonal, como a gente tinha visto aqui, meu ângulo diagonal a gente está pondo que é 20 graus, então o seno de 20 vai ser igual ao meu seno mínimo, e o meu cosseno mínimo vai ser 90 menos essa folga, menos os 20 graus.

[09:45] Então, a gente criou o ângulo, calculou o seno e cosseno mínimo que a gente quer e já está usando para fazer a verificação, então quando eu preciso verificar o meu eixo X e quando eu preciso verificar o meu eixo Y. Vamos salvar, voltar na cócs, deixar ela em todos os scripts, ver se não tem nenhum erro. A gente pode abrir ela, e a gente tem o nosso jogador olhando agora nas oito direções.

[10:10] Então, a gente usou toda aquela teoria matemática e traduziu isso para código, e funciona porque agora a gente está pegando independente do ângulo que meu mouse tá, no caso da jogadora, a gente está espelhando ele lá o primeiro quadrante, e fazendo a verificação se eu preciso saber só o eixo x ou só o eixo Y, ou os dois. Então, a gente tem um código bem simples, bem tranquilo de trabalhar.

[10:32] Como a gente criou o nosso ângulo diagonal como uma variável pública, a gente pode vir aqui na jogadora, vir no script do controle de animação, ele tem um ângulo diagonal, e eu consigo colocar que o meu ângulo diagonal é 30 graus aqui direto no inspetor. Então, eu posso vir e apagar o meu ângulo diagonal aqui do código, eu não preciso atribuir ele no código, porque eu estou atribuindo ele direto no inspetor. Então, a gente vai ter a jogadora com o ângulo diagonal de 30 graus. O zumbi também usa esse mesmo script, às vezes a gente precisa entrar lá e alterar ele.

[11:12] Então, vamos pegar a nossa movimentação, animação, controle de animação e vamos colocar que o meu zumbi, ele tem lá 20 graus, então a jogadora tem uma folga maior e o zumbi tem uma folga menor. Vamos salvar o prefab, fechar e vamos testar de novo nosso jogo. Então, abre ele lá no inspetor. Então, a nossa jogadora agora tem a folga de 30 graus e o zumbi, se a gente for vir aqui, ele também, ele está andando para cima, para direita, para diagonais, ele consegue seguir a gente em todos os ângulos. A gente conseguiu consertar aí, utilizar todas as animações que o designer passou para a gente. Ele tinha passado várias animações que a gente não tinha usado e agora a gente está usando.