

11. Blueprint

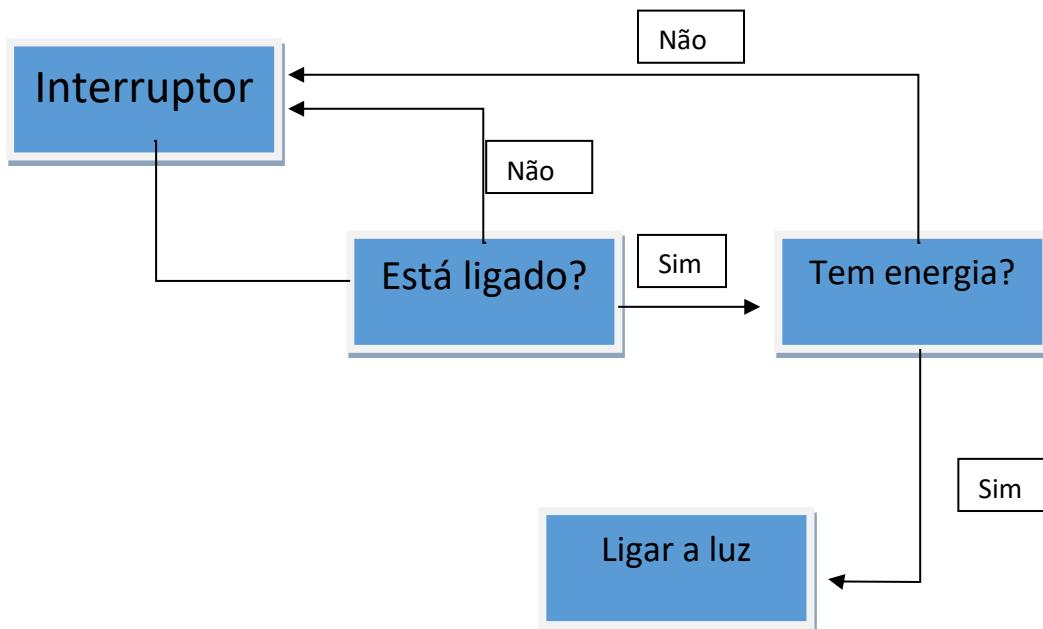
Essa é a primeira versão, em que o editor de blueprint é totalmente interativo, podendo substituir qualquer tipo de programação dentro da UE4.

11.1. Mas o que é Blueprint?

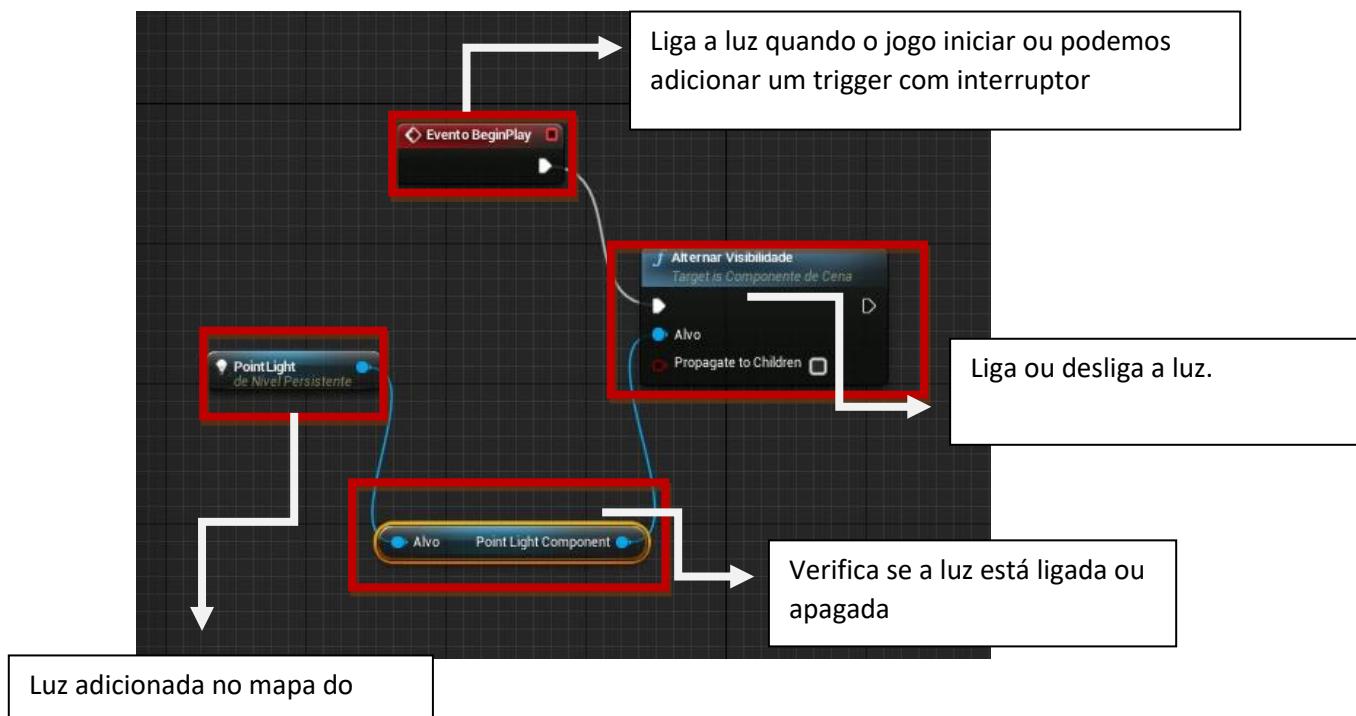
O sistema Blueprints, no Unreal Engine, é um sistema completo de scripts de jogo, baseado no conceito de usar uma interface em nó, para criar elementos de jogo a partir do Unreal Editor. Como em muitas linguagens de script (roteiros) comuns, ele é usado para definir classes orientadas a objeto (s) no mecanismo.

Esse sistema é extremamente flexível e poderoso, pois oferece aos designers a capacidade de usar praticamente toda a gama de conceitos e ferramentas, geralmente disponíveis somente aos programadores. Desse modo, não precisamos utilizar métodos complicados para realizar tarefas dentro do Game. Com o blueprint, podemos criar missões, ações para os personagens, objetivos, sistema de fome, sede, sobrevivência.

A principal inovação dessa versão do Blueprint é a utilização do sistema de modificadores para programar o mesmo sistema que vemos no editor de materiais. Veja o exemplo abaixo de como utilizá-lo.



De modo lógico é assim que utilizamos o Blueprint dentro da Unreal Engine, porém veremos como o mesmo processo fica dentro da UE4.



11.2. Como funcionam os Blueprints?

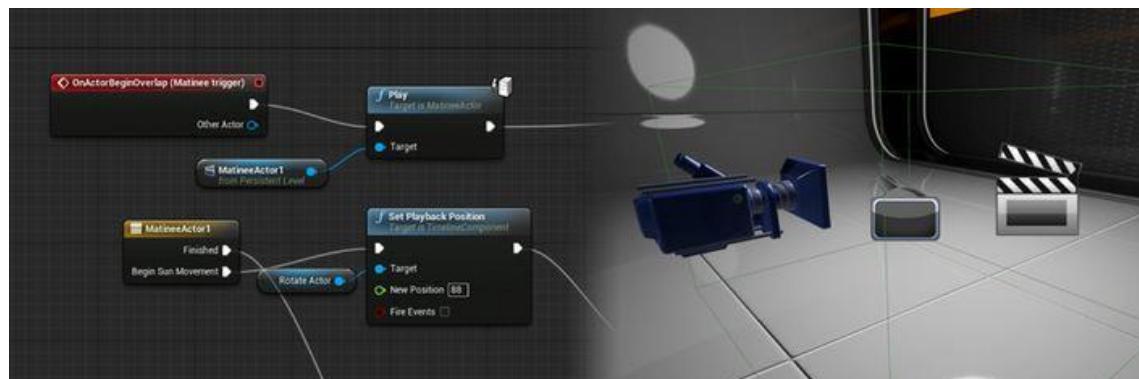
Em sua forma básica, Blueprints são visualmente “roteiros”, em que você diz onde, como e quando certa função vai acontecer. Ao conectar Nós, Eventos, Funções e Variáveis com Fios, é possível criar elementos de jogo complexos.

Os Blueprints funcionam usando gráficos de nós para várias finalidades: construção de objetos, funções individuais e eventos gerais de jogo, específicos para cada instância do Blueprint, a fim de implementar comportamento e outras funcionalidades.

11.3. Tipos de planos normalmente usados

Os tipos de Blueprint mais comuns, com os quais você estará trabalhando, são Blueprints de Nível e Classes.

11.3.1. Blueprint de Nível



Um Blueprint de Nível é um tipo especializado que atua como um gráfico de evento global, de nível global. Cada nível em seu projeto tem seu próprio plano criado por padrão, podendo ser editado dentro do Unreal Editor. No entanto, novos planos não podem ser criados através da interface do editor. Você pode controlar cinemáticas, usando agentes de Matinee e gerenciar coisas, como streaming de nível, pontos de verificação e outros sistemas relacionados ao nível. Esse plano também pode interagir com Blueprint de Classe colocados no nível, como definir qualquer variável ou acionar eventos personalizados que possam conter.

11.3.2. Blueprint Class



Blueprint Class são ideais para tornar objetos em ativos interativos, tais como portas, interruptores, itens de colecionáveis e paisagens destrutíveis. Na imagem acima, o botão e o conjunto de portas são planos separados que contêm o roteiro necessário para responder aos eventos de sobreposição de jogadores, torná-los animados, reproduzir efeitos sonoros e alterar seus materiais (o botão acende quando pressionado, por exemplo).

Nesse caso, pressionar o botão ativa um evento dentro da porta Blueprint, fazendo com que ele abra as portas, mas elas poderiam ser facilmente ativadas por outro tipo de Blueprint ou por uma sequência de Blueprint.

11.4. O que mais Blueprints podem fazer?

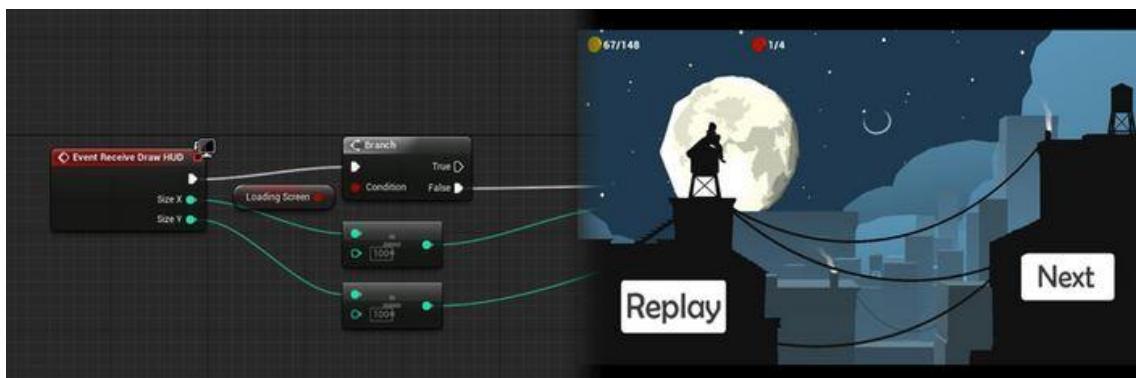
11.4.1. Criar um personagem de jogo jogável:



É possível reunir todos os elementos que você precisa para um personagem "jogável" no gráfico Blueprint. Você pode manipular o comportamento da câmera, configurar eventos de entrada para mouse, controlador e telas de toque, bem como criar um recurso Blueprint de animação para manipular animações de malha esquelética.

Quando você cria um novo Character Blueprint, ele vem com um componente de caráter que tem muito do comportamento necessário embutido, como: movimentar, saltar, nadar e cair, e o que é necessário é apenas adicionar alguns eventos de entrada de acordo com como você quer que seu personagem seja controlado.

11.4.2. Criar um HUD

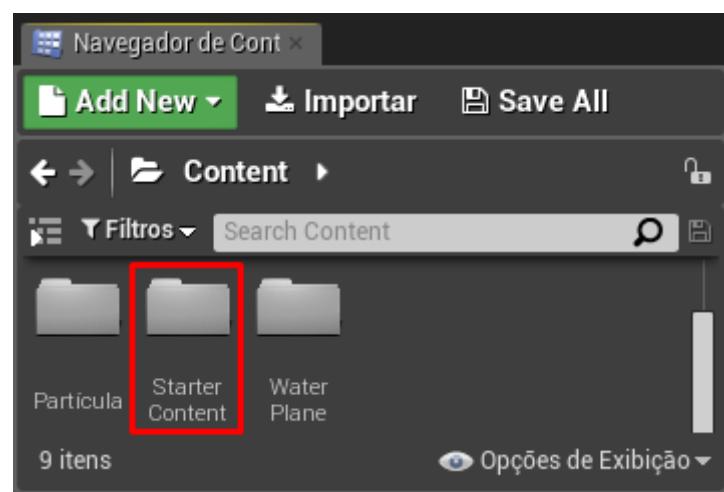


O roteiro Blueprint também pode ser usado para criar o HUD (Heads-Up Display, qualquer elemento gráfico exibido na tela para transmitir informações ao jogador) de um jogo, que é semelhante ao Blueprint Class, pois pode conter sequências de eventos e variáveis, mas é atribuído ao recurso GameMode do projeto, em vez de ser adicionado diretamente a um nível.

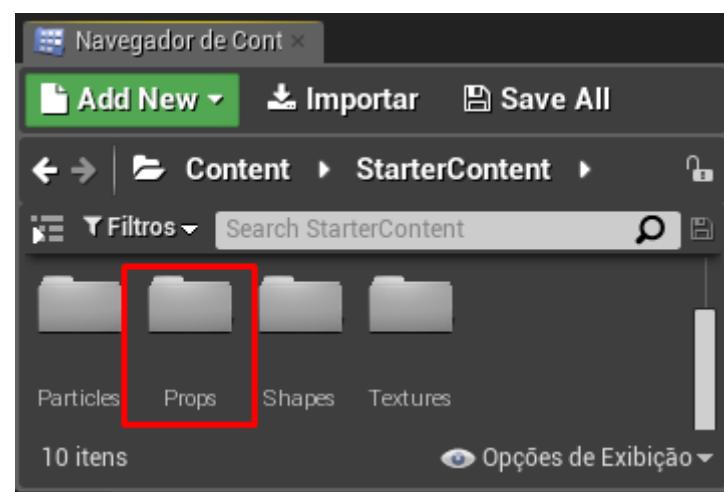
Você pode configurar um HUD para ler variáveis de outros Blueprints e usá-los para exibir uma barra de saúde, atualizar um valor de pontuação, exibir marcadores objetivos e assim por diante. Também é possível usar o HUD para adicionar caixas de acesso para elementos como botões que podem ser clicados ou, no caso de jogos para celular, podem responder à entrada por toque.

Exercício de Conteúdo

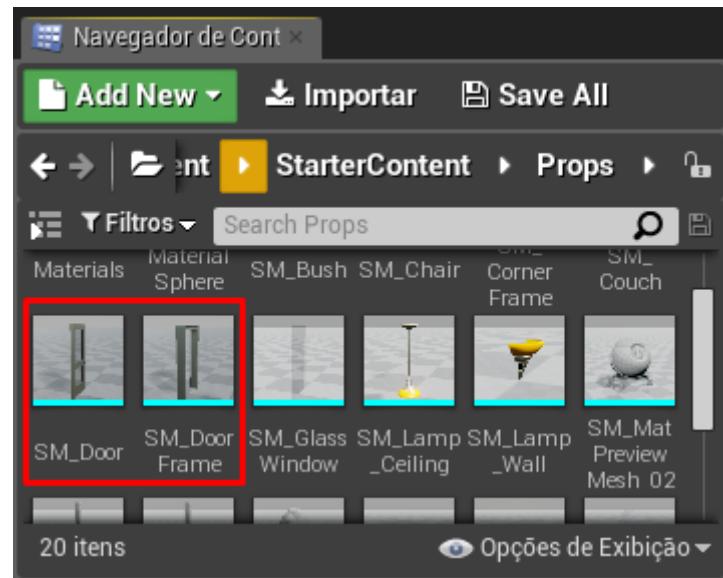
- 1- Abra a Unreal Engine 4
- 2- Selecione o projeto que você salvou com seu nome na aula 10 (sobre Sistema de Partículas), e clique em abrir.
- 3- Vamos fazer uma interação com porta, ou seja, quando você passar por ela, ele vai abrir.
- 4- Para começar então, vá em Content-> Start Content:



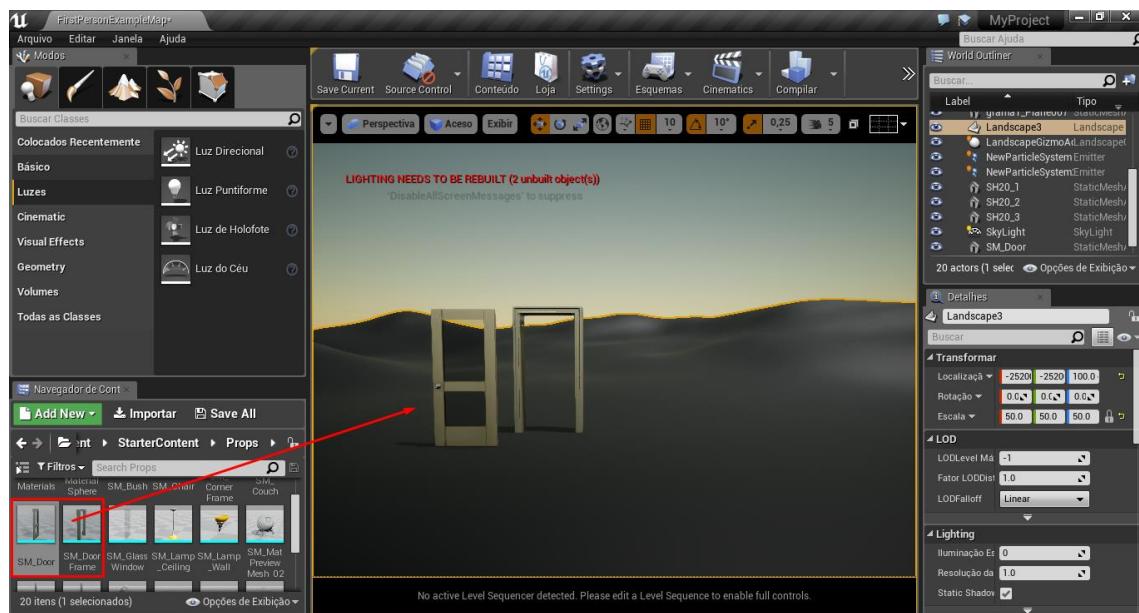
- 5- Dê um duplo clique em props:



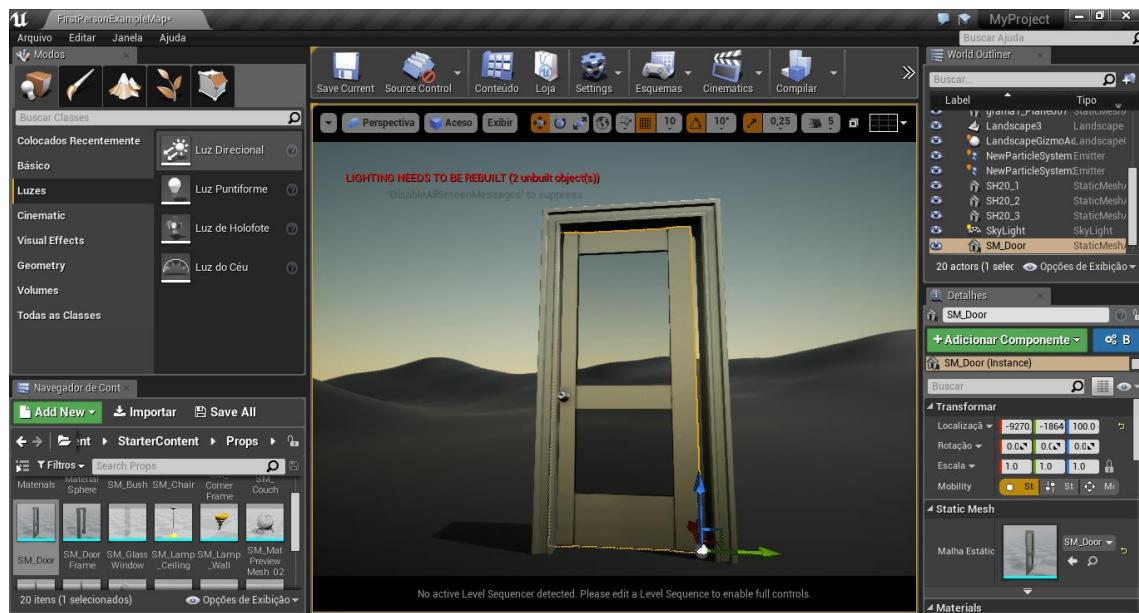
- 6- Nessa pasta, temos uns materiais próprios da Unreal. Os arquivos que procuramos são SM_Door, que é a porta, e SM_Doorframe, que seria armação da porta:



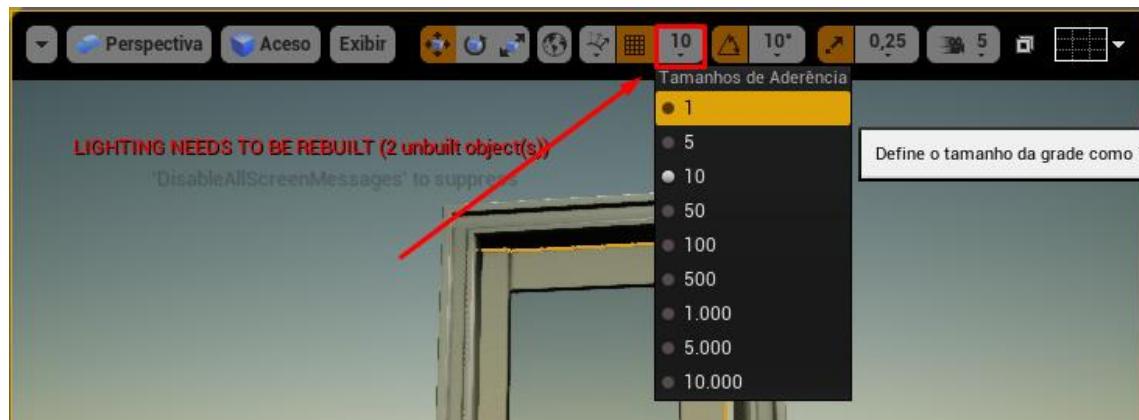
- 7- Arraste os dois para o seu level:



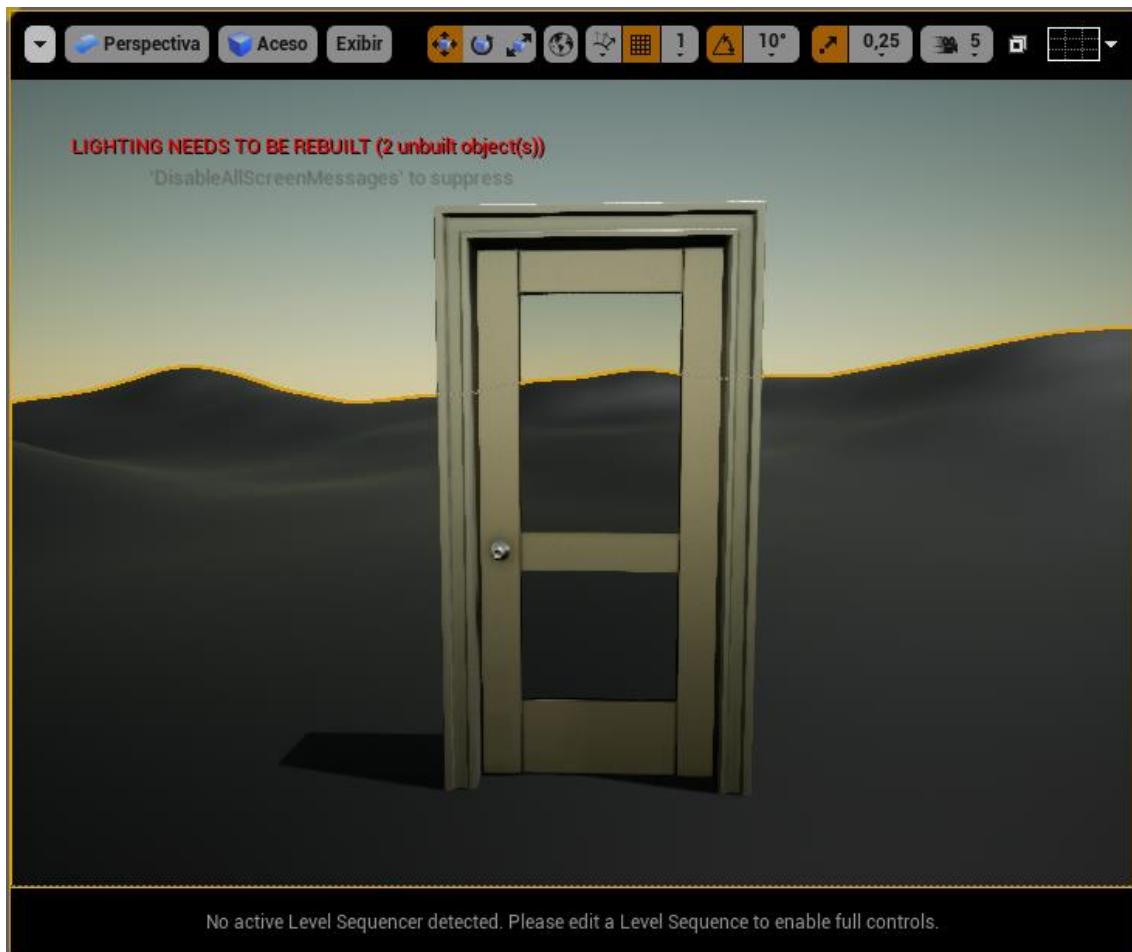
- 8- Antes de qualquer coisa, vamos ajustar a porta à sua armação. Selecione-a, usando as flechas de movimentação e tente encaixá-la na armação:



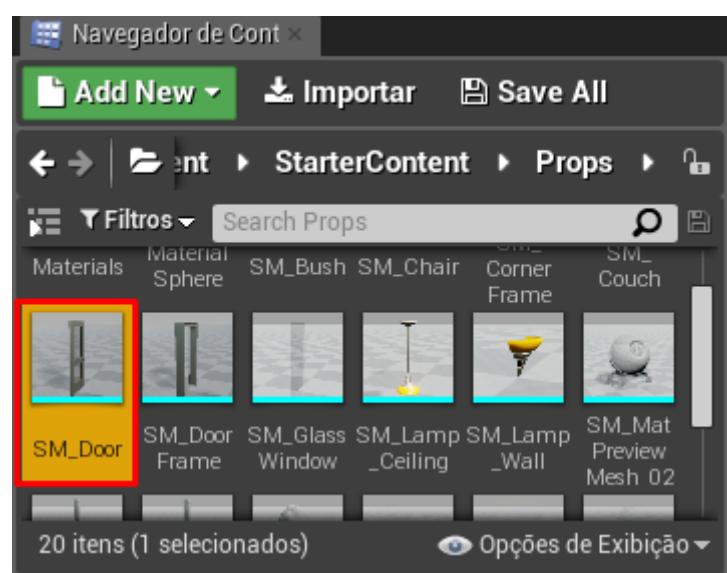
- 9- Você vai perceber que fica um vão entre as duas. Além disso, se você arrastar a porta para o outro lado tentando tapar, o vão troca de lado também. Para arrumar isso, clique no local indicado e marque a opção 1:



10- Isso vai te dar mais aderência, fazendo com que fique mais fácil movimentar o objeto:



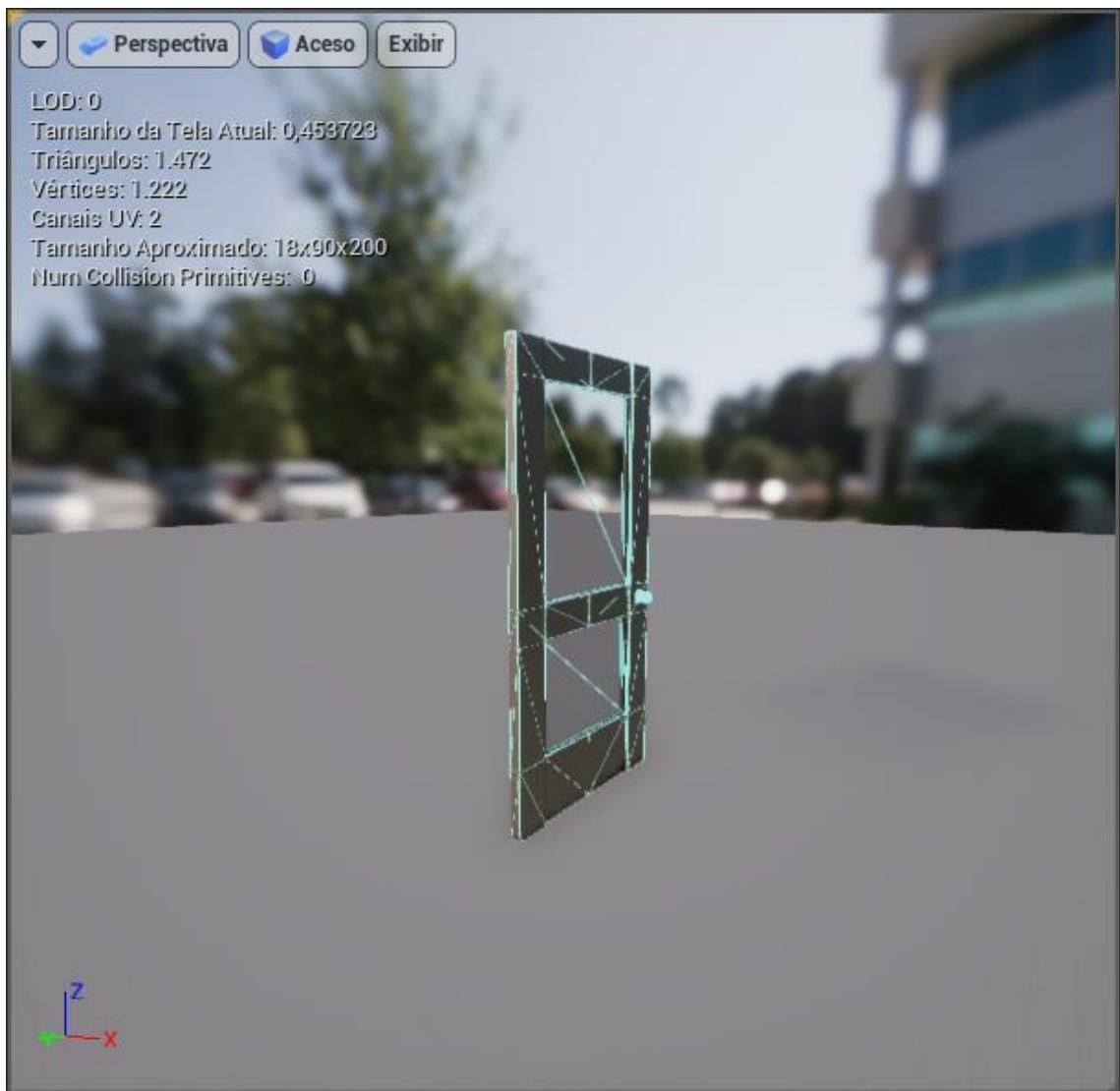
11- Clique em play e tente atravessar a porta. Você vai ver que não é possível, mas, se você atirar nas áreas que não tem madeira, vai perceber que não tem colisão. Então, vamos adicionar a colisão. No navegador de conteúdo, dê um duplo clique na malha da porta:



12- Na barra de ferramentas superior, clique em colisão:



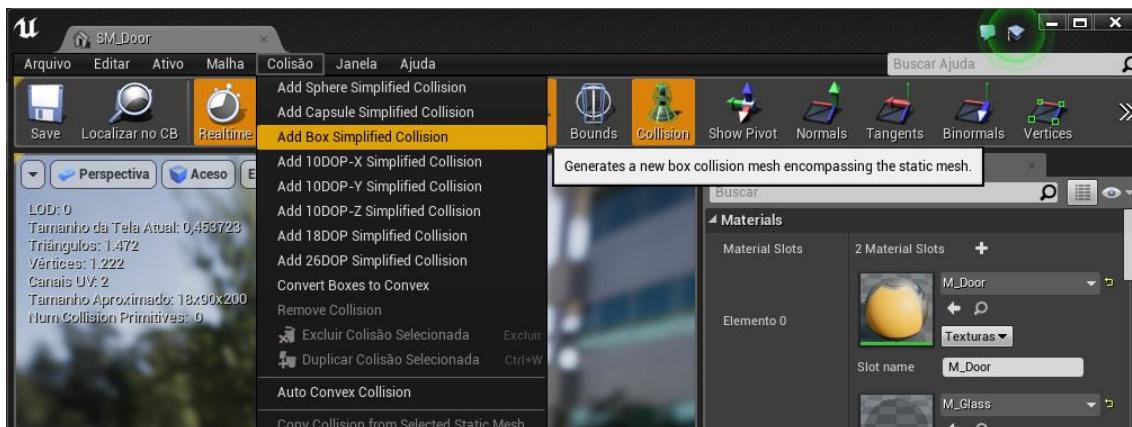
13- A porta vai ficar marcada com essa cor, mostrando que não há colisão:



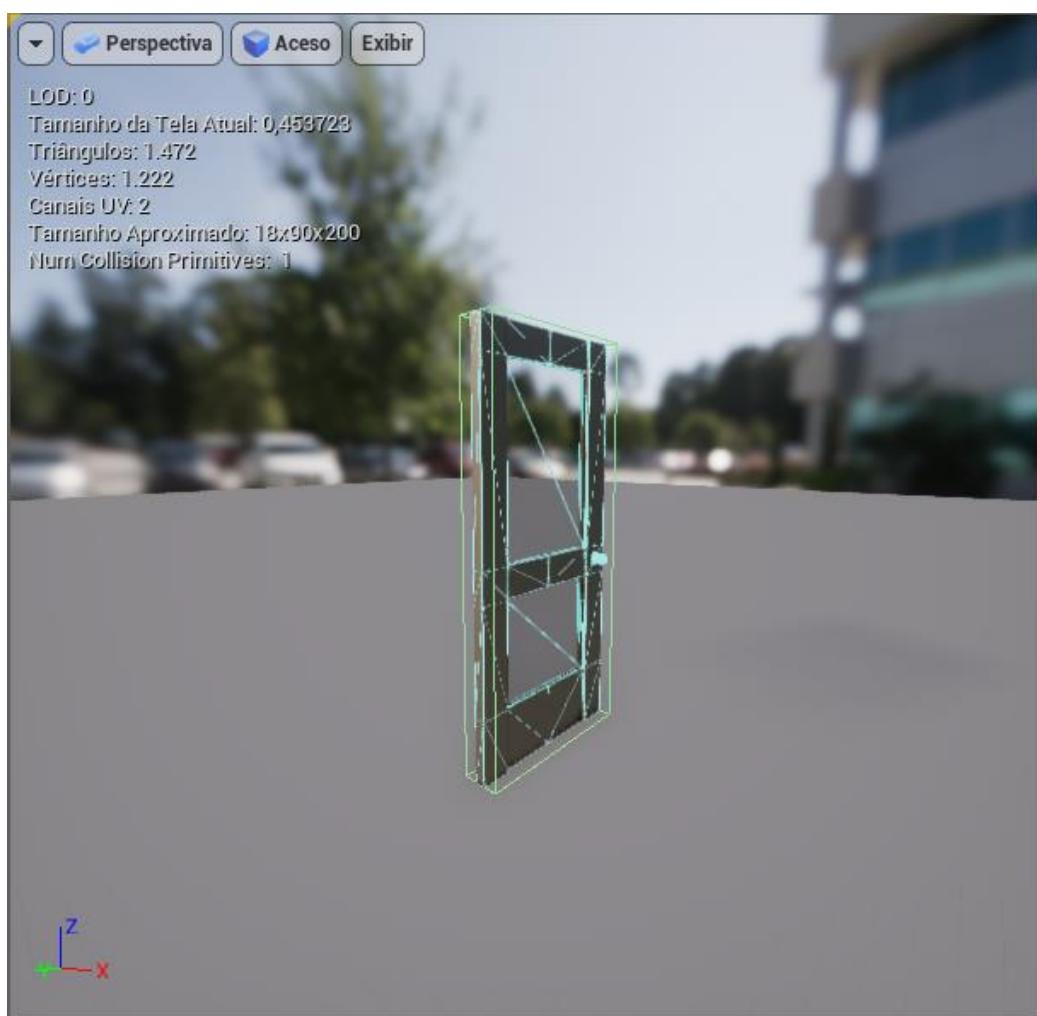
14- Então, para adicionar a colisão, clique em colisão na barra de menus supeior:



15- Clique em Add Box Simplified Collision:



16- Perceba que uma caixa verde foi adicionada a nossa porta:



17- Pronto, a colisão está adicionada. Agora, basta clicar em Save e fechar o editor de malhas:



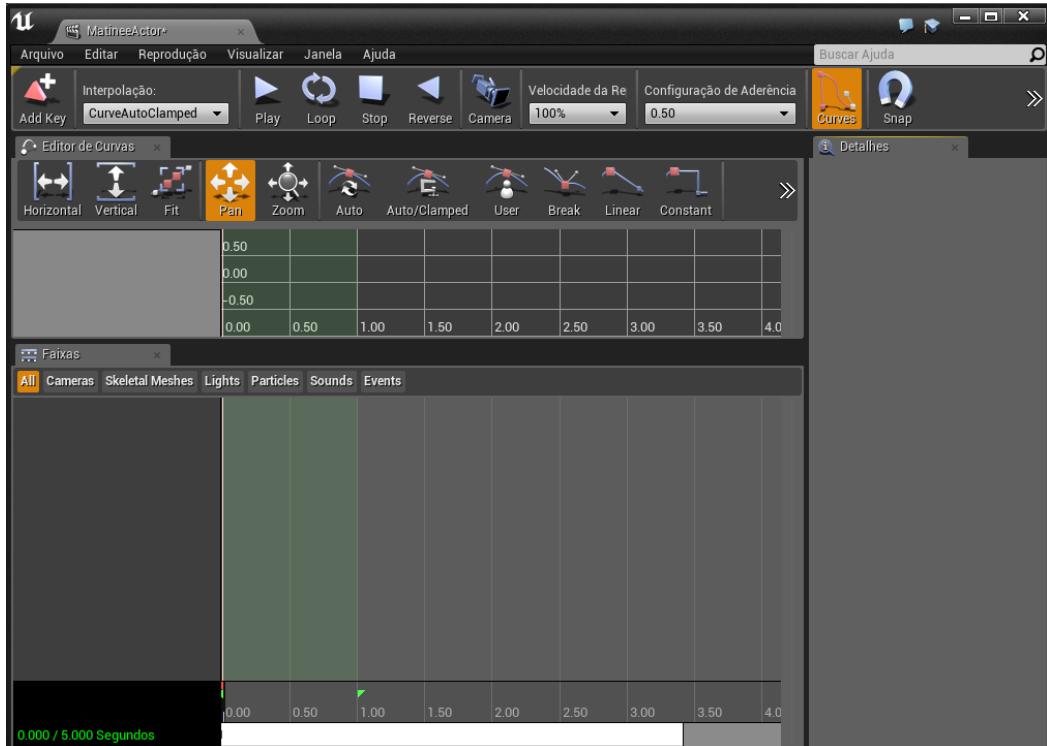
18- Agora, para fazer essa porta funcionar, ela vai precisar de uma animação de abertura, pois é um objeto estático, ou seja, imóvel. Para fazer isso, vamos usar o cinematic. Então, selecione a porta e clique em cinematic:



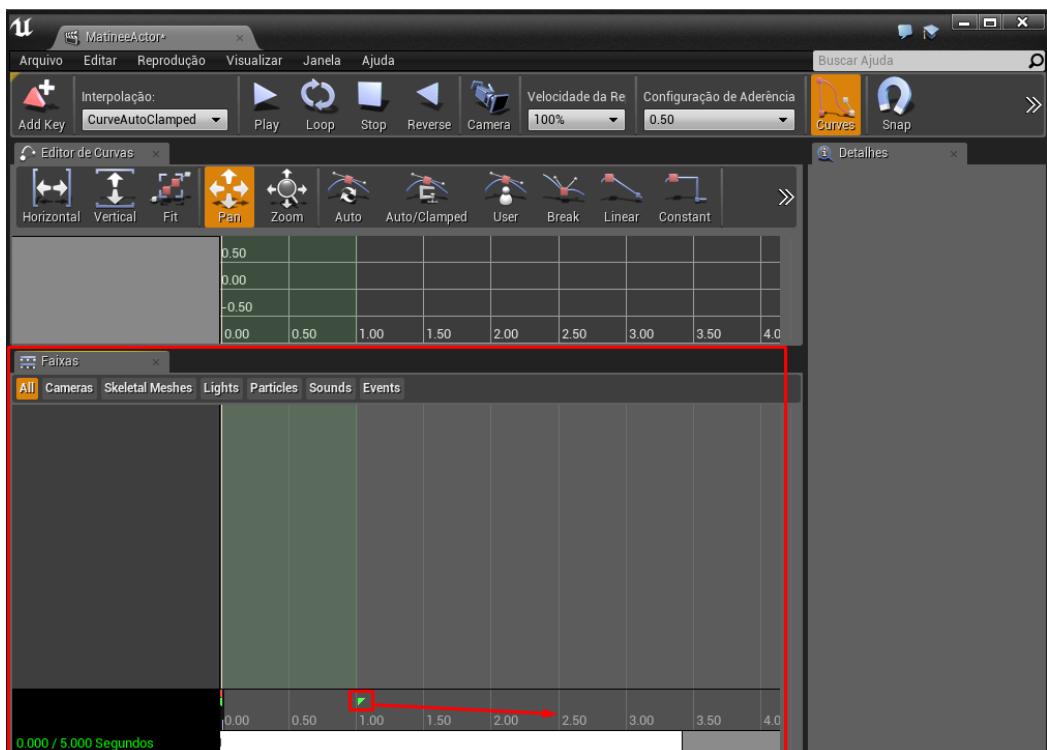
19- Clique em Add Matinee:



20- Esse Matinee que vamos adicionar é a nossa faixa de movimentação. Pois bem, após clicar em add Matinee, irá abrir a seguinte janela:



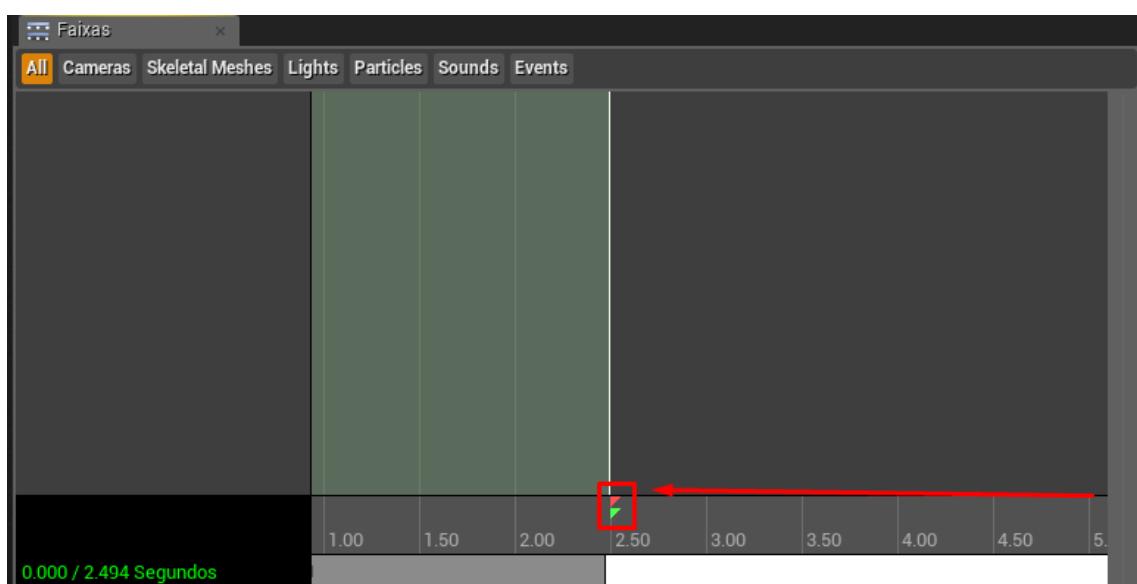
21- Na parte inferior do editor, aba Faixas. Há uma linha do tempo, onde vamos indicar quantos segundos irá demorar para nossa porta abrir. Então, clique na seta verde, indicada na imagem abaixo, e arraste até os 2,5 segundos:



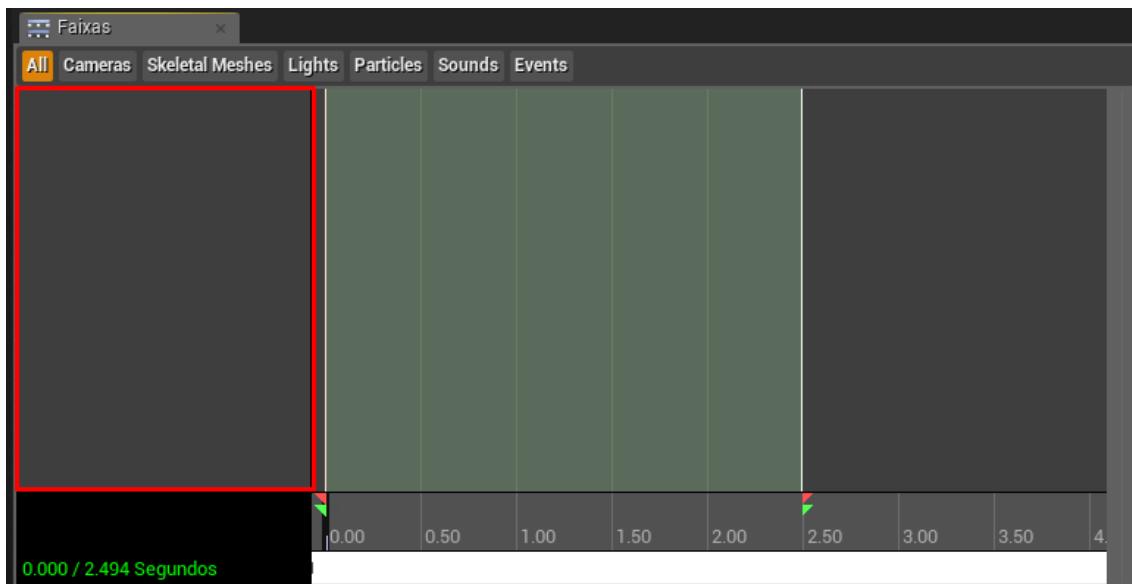
22- O próximo passo é diminuir a linha do tempo para ela durar apenas os 2,5 segundos. Para isso, vá até o fim da linha do tempo, arrastando a barra inferior e encontre o ponto vermelho que indica quando a linha do tempo acaba:



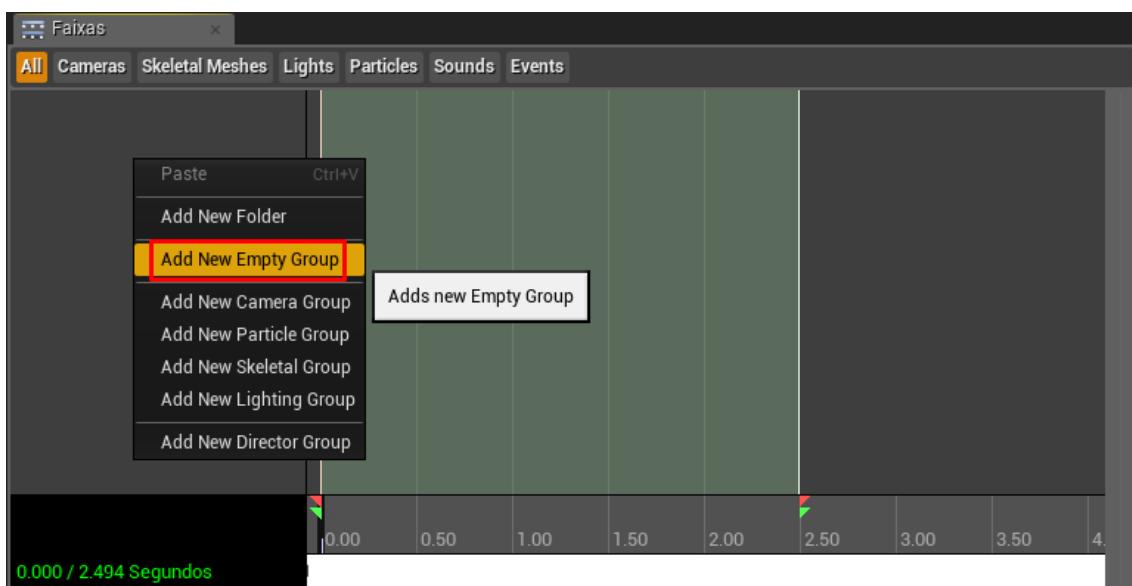
23- Arraste o ponto vermelho até os 2,5 segundos:



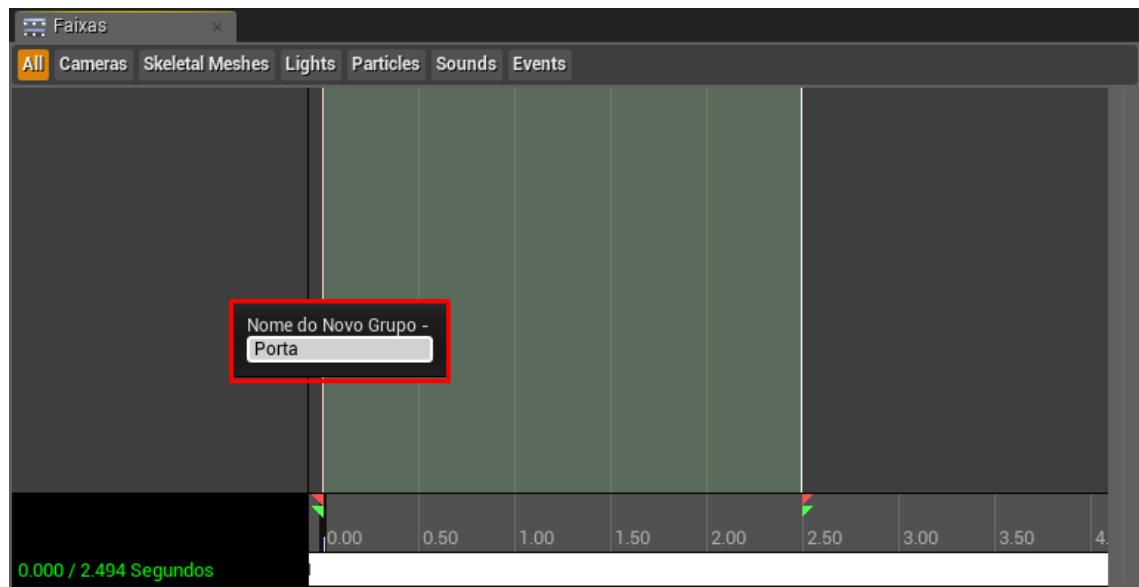
24- Arraste a barra de volta ao começo da linha do tempo. Com a porta selecionada (não esqueça de selecionar), clique no lugar indicado com o botão direito do mouse:



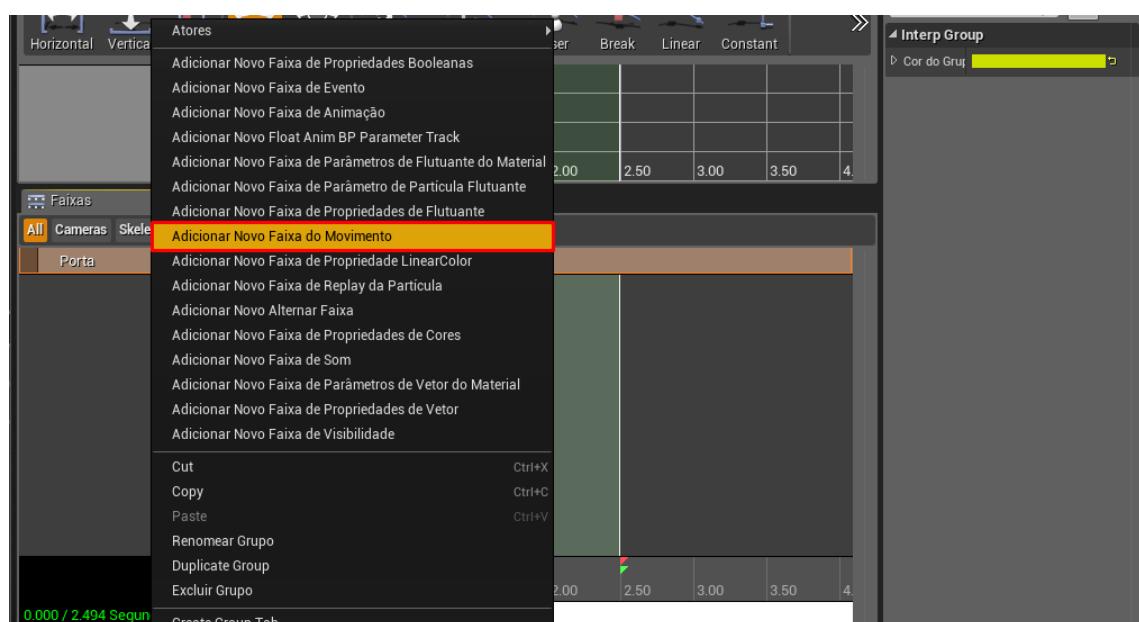
25- Vai abrir uma janelinha com algumas opções, clique em Add New Empty Group para criar um novo grupo vazio:



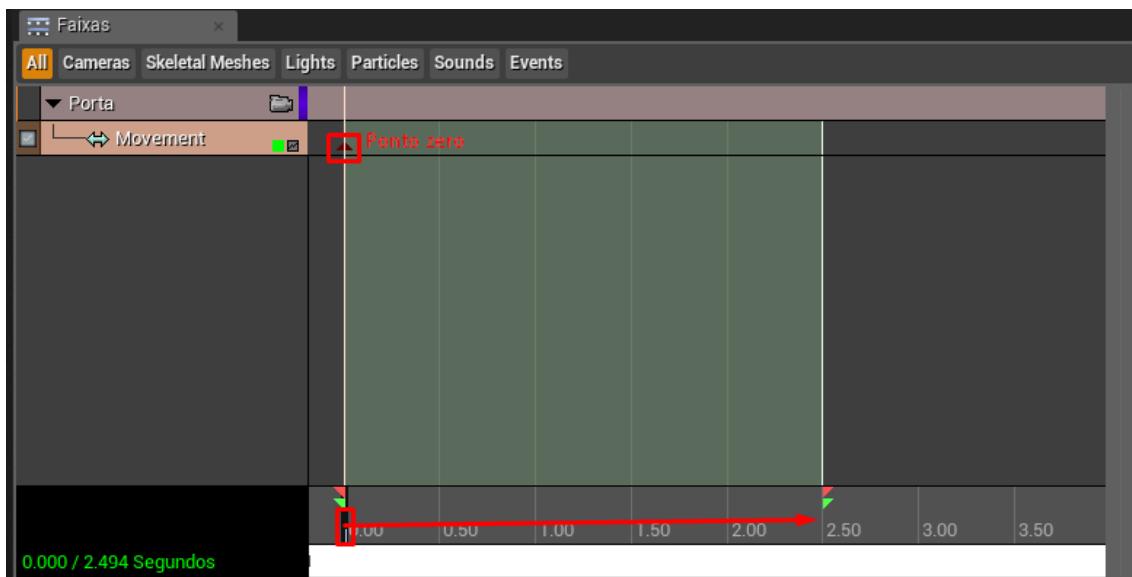
26- Em nome do novo grupo, digite Porta e clique no enter:



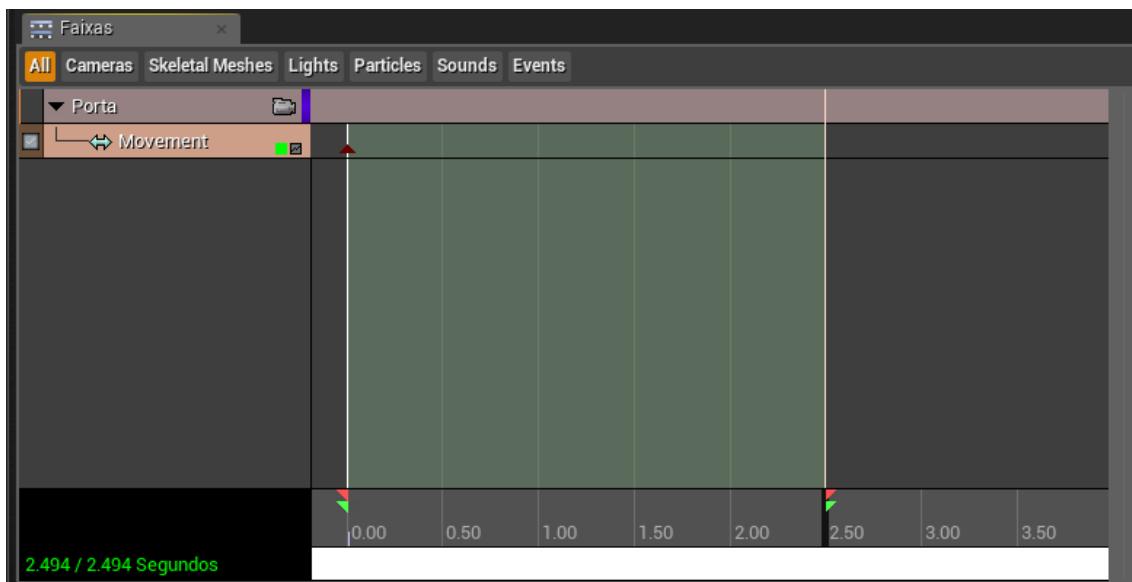
27- Agora, clique com o botão direito sobre o novo grupo criado e clique na opção Adicionar Nova Faixa de Movimento:



28- Perceba que na linha do tempo foi adicionado um ponto zero na gravação, onde a porta está no momento, no zero segundos. Para fazermos essa porta abrir, primeiro iremos movimentar o ponto preto até o fim da linha do tempo:



29- Sua linha do tempo deve estar assim:



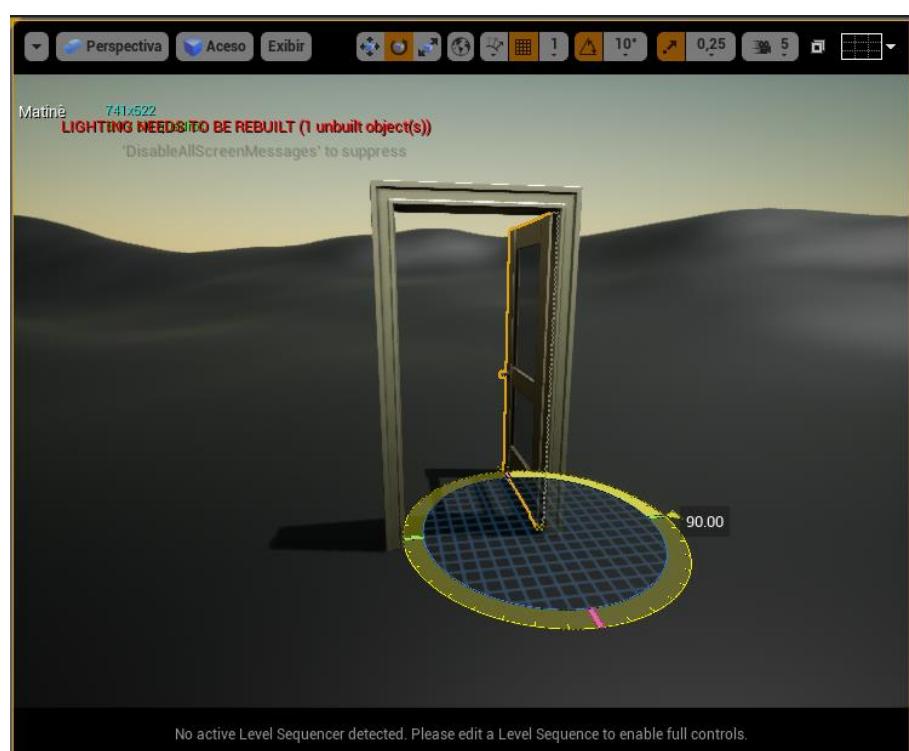
30- Agora, vamos minimizar esse editor e fazer o movimento na porta para marcarmos como ela deve ficar aos 2,5 segundos. Clique na porta:



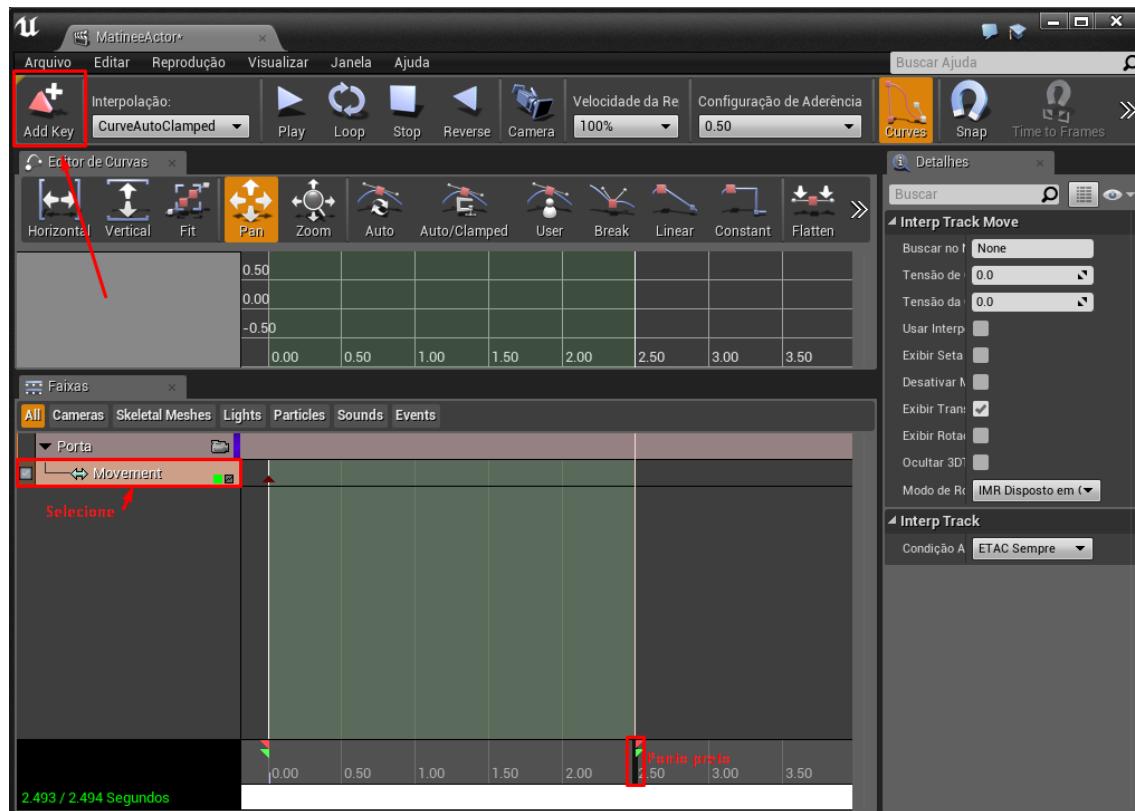
31- Clique no local indicado para mudar as setas de movimentação para as de rotação:



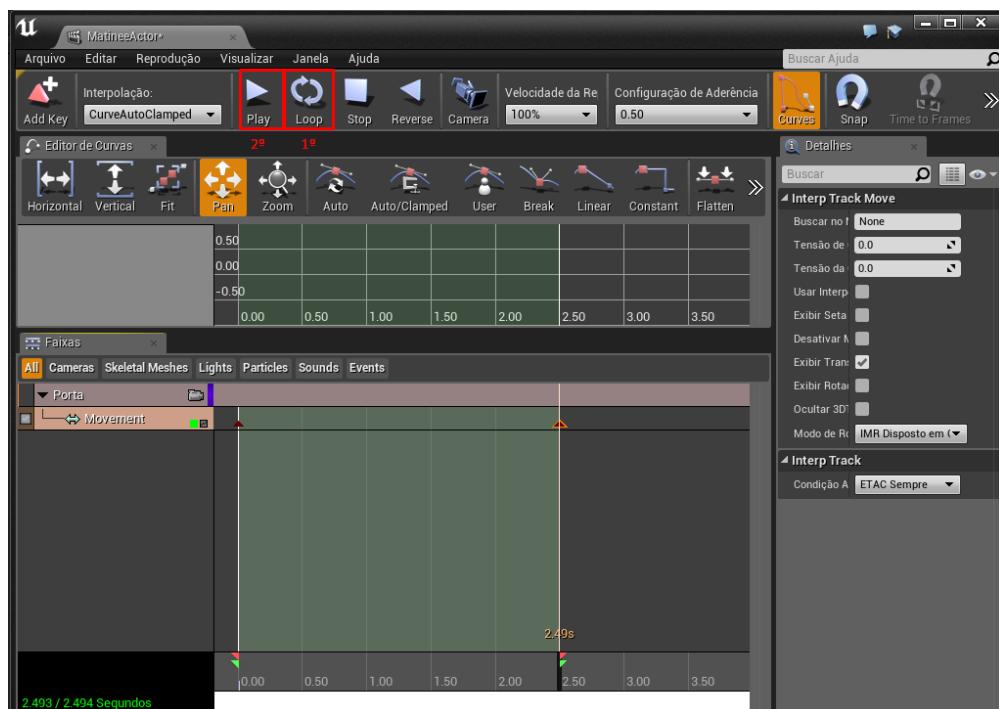
32- Usando a área em azul da ferramenta, gire a porta para o lado de fora:



33- Abra o matinee novamente. Estando o ponto preto em 2,5 segundos, selecione a faixa “Movement” e clique em Add Key para adicionar uma marcação:

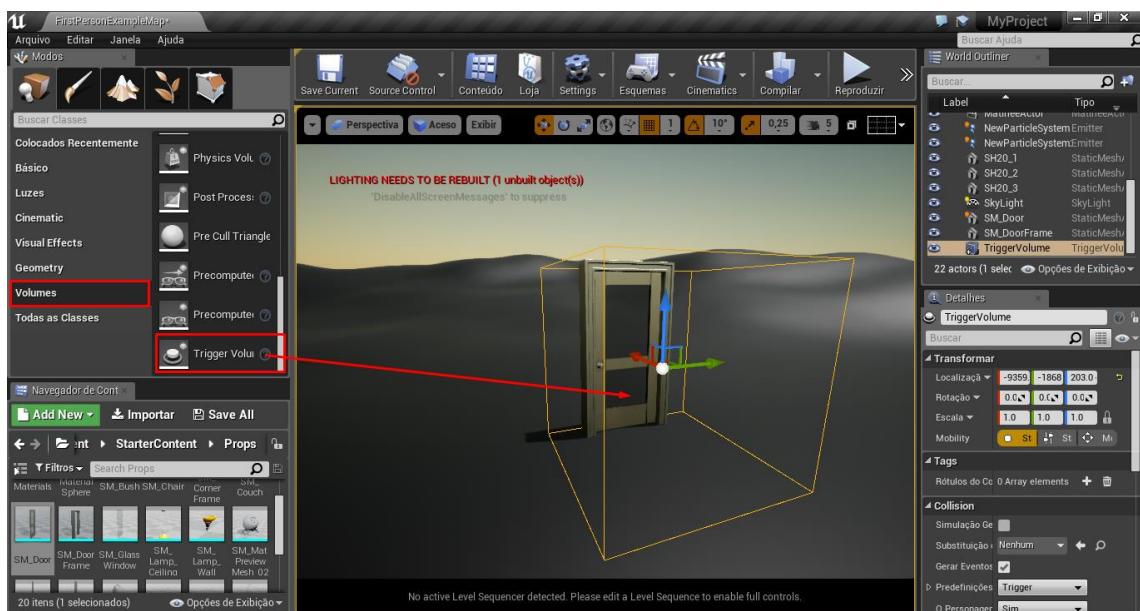


34- Agora, temos nossa porta fechada aos 0 segundos e aberta aos 2,5 segundos. Entre esse tempo, o programa vai fazer a movimentação sozinho para ficar mais bonito. Acione a opção Loop para ficar repetindo o movimento e clique em play:



35- Minimize o editor e veja se a movimentação deu certo. Com tudo ok, pode fechar a matinee.

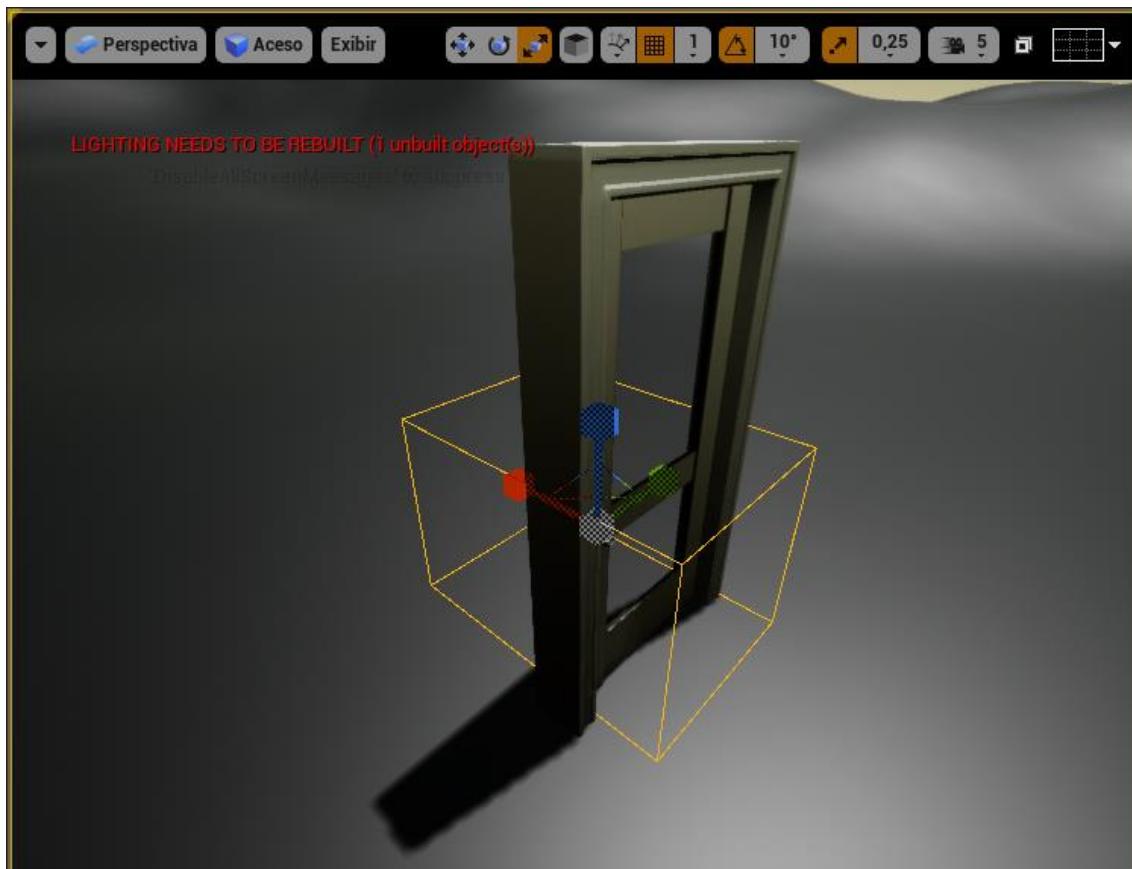
36- Agora, vamos adicionar o Trigger, que seria a área de acionamento, o espaço onde o personagem entra e a porta se abre. Clique em Volume e encontre o Trigger Volume. Logo após, arraste-o para a porta:



37- Vamos ajustar essa caixa para ficar no tamanho adequado. Para isso, clique no local indicado, para que você possa mudar as setas de movimentação para setas de modificação de tamanho:

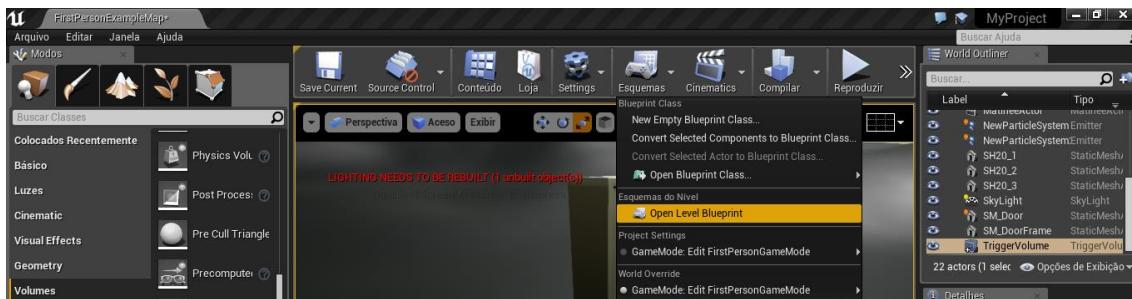


38- Posicione a câmera de lado. Assim, você poderá visualizar melhor os dois lados da porta, pois a caixa tem que ativar a porta dos dois lados. Configure-a o mais parecido possível com a imagem abaixo:

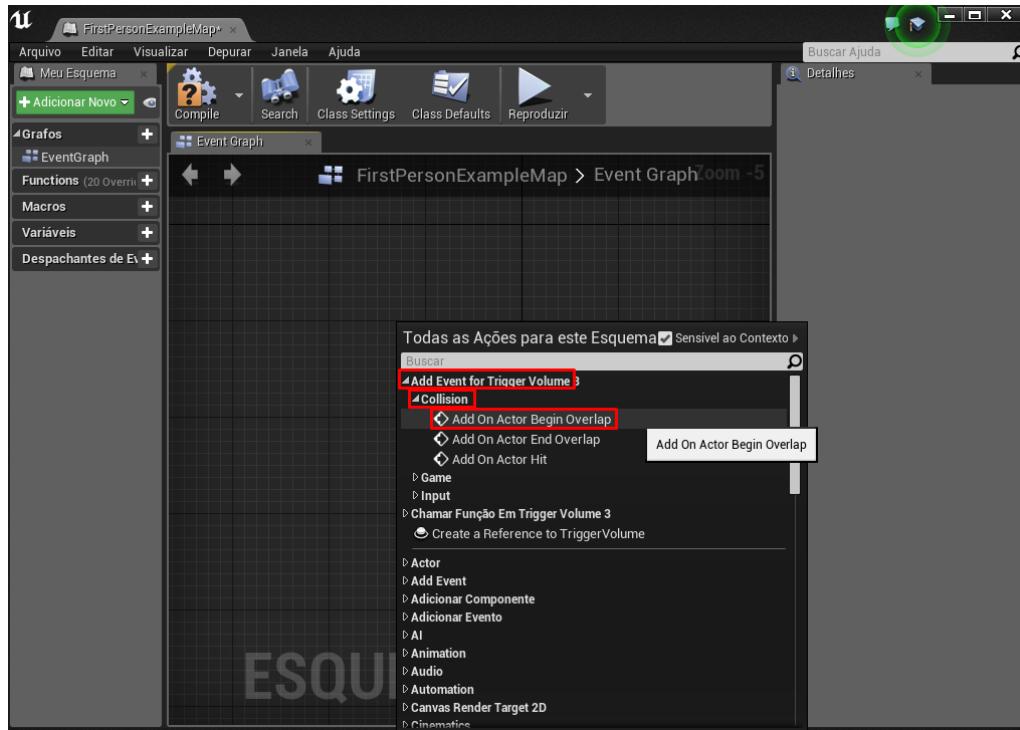


Deixe a parte da frente da caixa mais próxima da porta, para o personagem ter que encostar na porta para abri-la

39- Depois de posicionado, vamos criar o nosso blueprint. Para isso, clique em esquemas e depois em Open Level Blueprint:

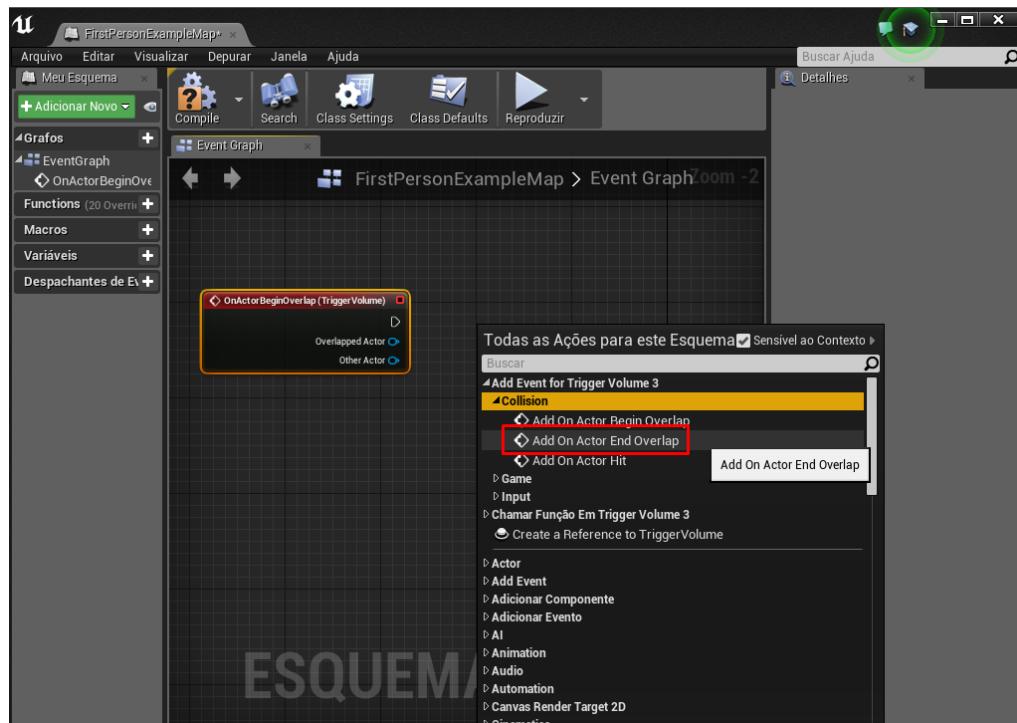


40- Vai abrir a área de programação do blueprint. Para começar, com o Trigger selecionado, clique com o botão direito do mouse na parte vazia do editor e clique em Add event for trigger Volume-> collision-> Add on Actor begin Overlap:

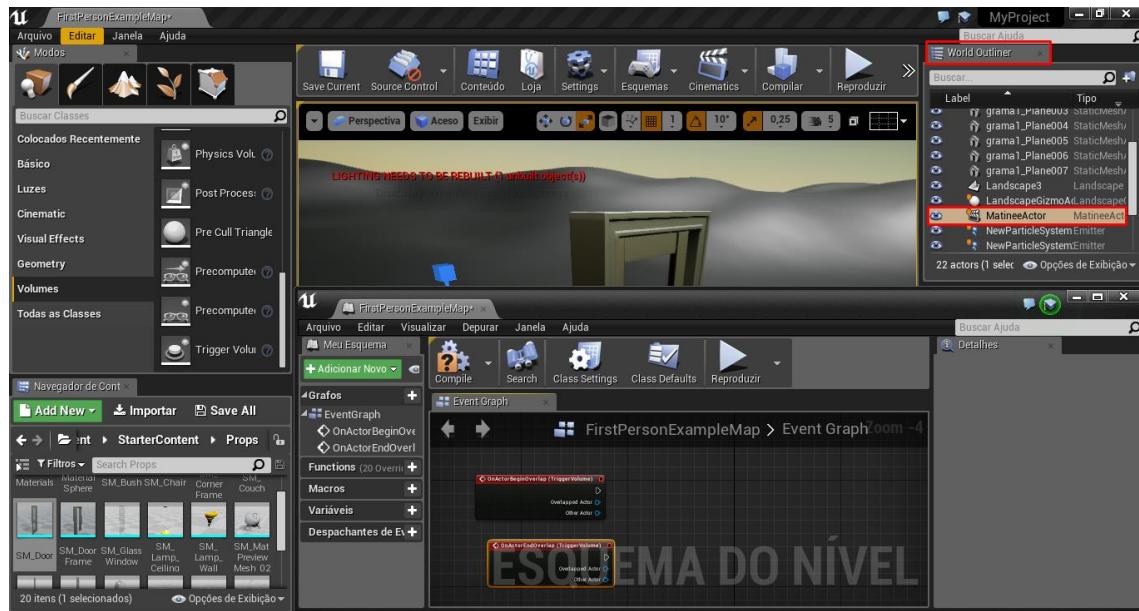


Escolhemos colisão por primeiro pois esse modificador vai ser ativado quando encostarmos na porta, ou seja, quando tiver uma colisão. O Begin é para quando se entra no trigger e o End quando sai.

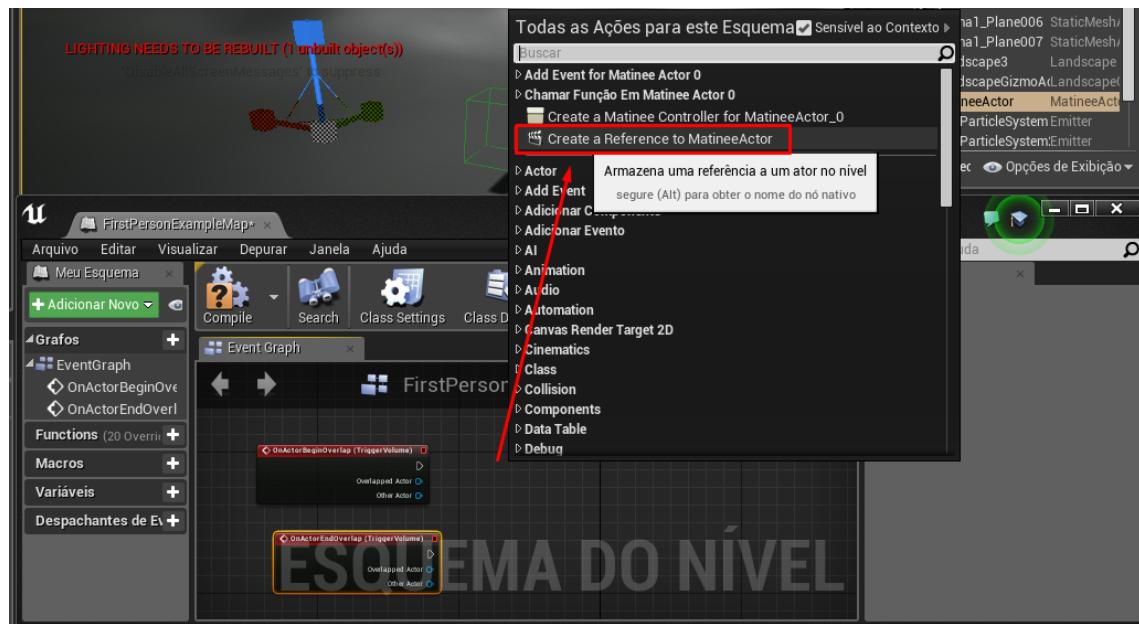
41- Agora, adicione o Actor End Overlap:



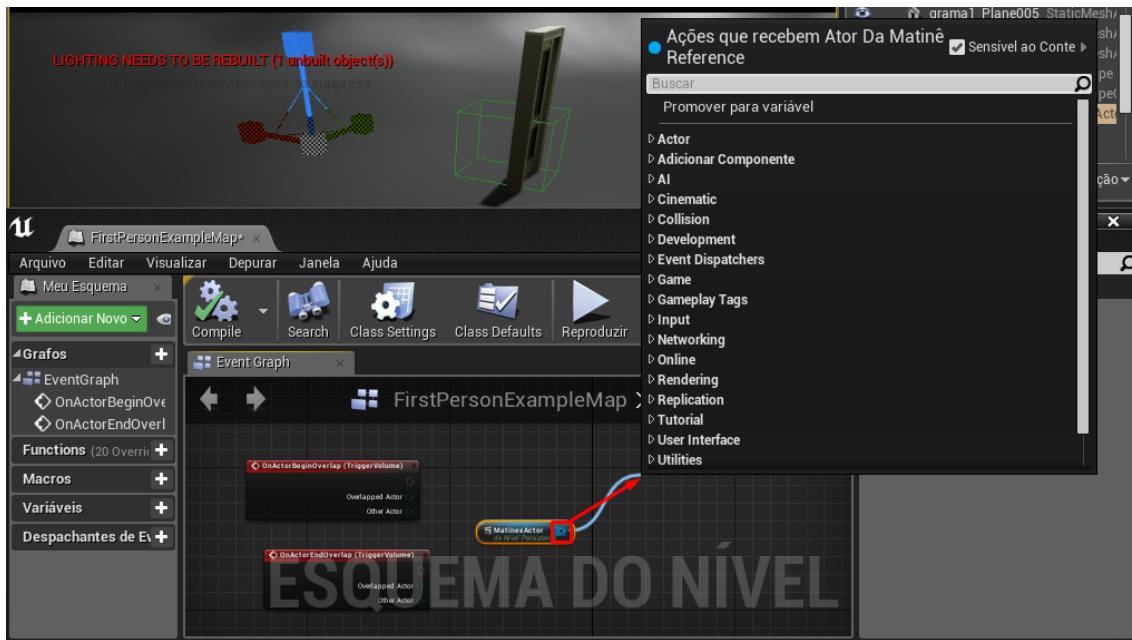
42- Minimize o Editor de Blueprint, visualize o World Outliner e encontre a matinee, que é a nossa animação de movimento da porta. Se você quiser, pode renomeá-la para se achar mais fácil. Quando encontrá-la, clique para selecionar:



43- Voltando ao Blueprint, clique novamente com o botão direito no espaço vazio e clique em criar uma referência para a matinee:



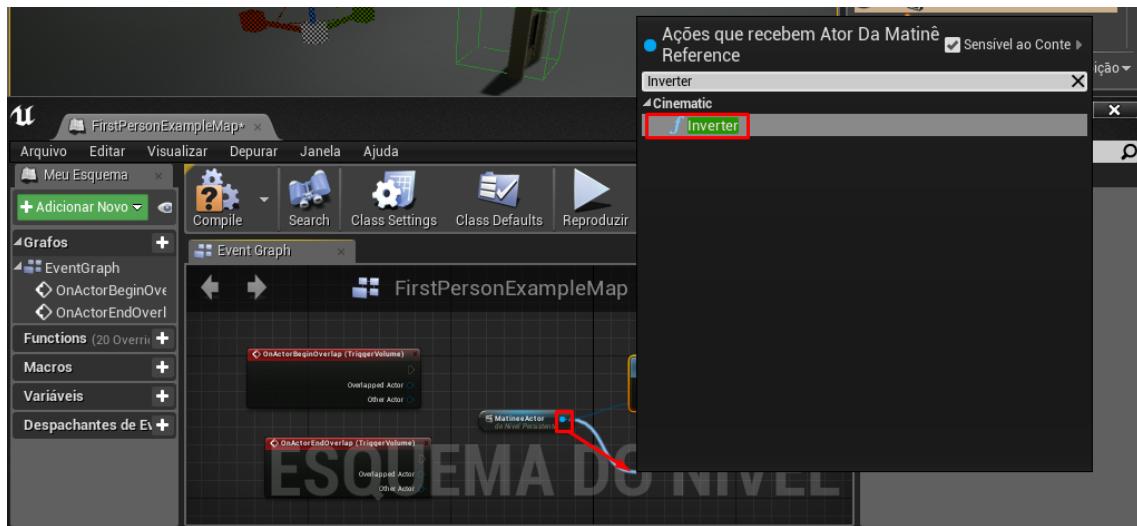
44- Essa referência representa nossa animação, e ela necessita de comandos que serão adicionados agora. Para isso, clique no canal azul da referência e arraste para abrir a janela de ações:



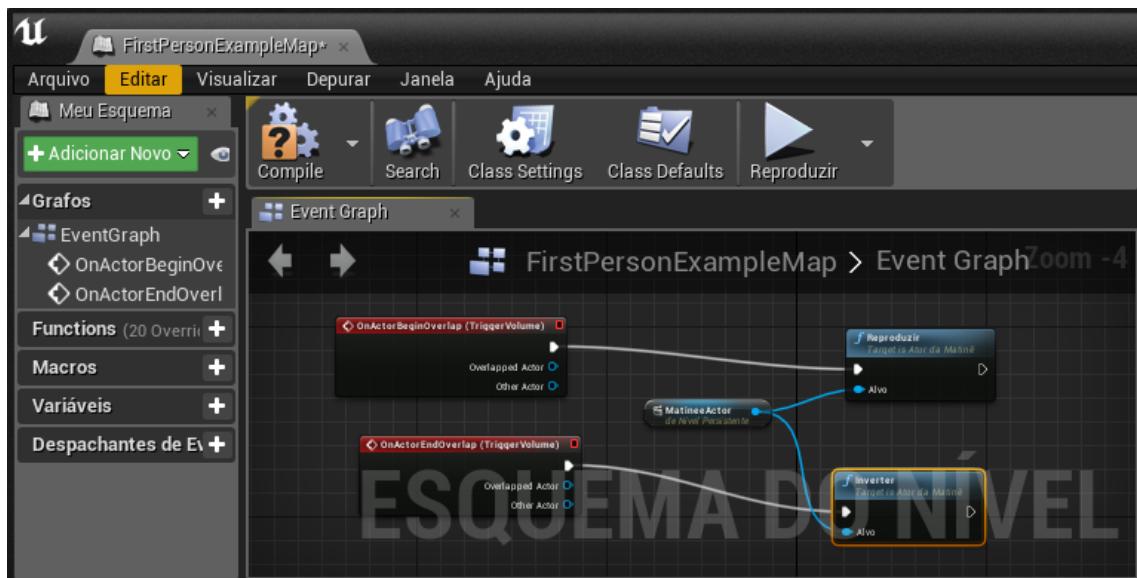
45- Digite Reproduzir e clique na ação que aparecer para adicioná-la:



46- Clique no canal azul novamente. Arraste para baixo para abrir a janela de ações mais uma vez e digite inverter:



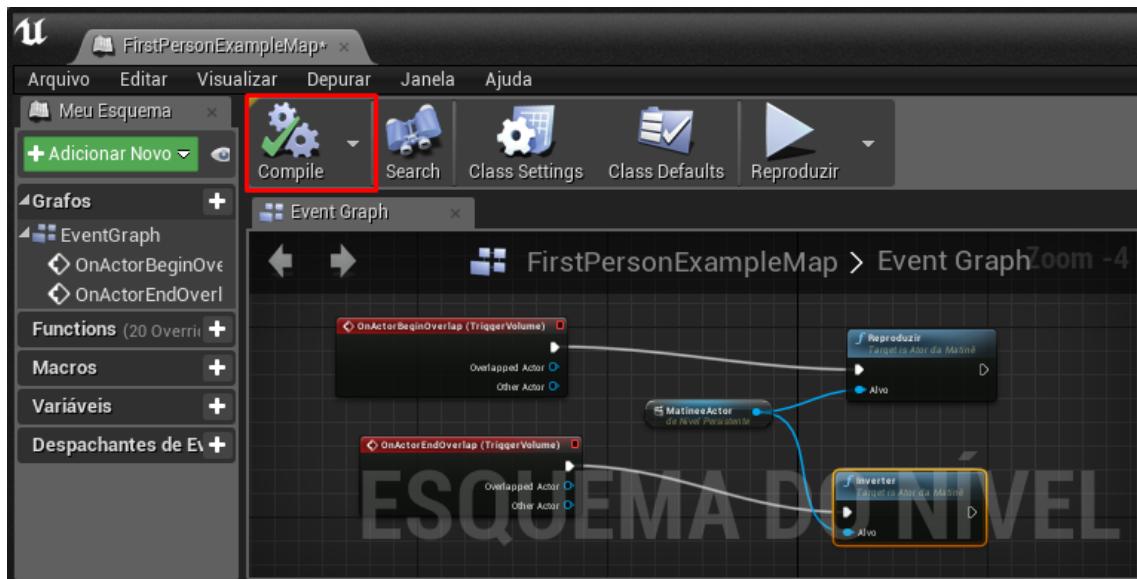
47- Clique no Inverter para adicionar. Essas ações que adicionamos são, respectivamente, o reproduzir, para quando a porta abrir, e o inverter, para a porta fechar. E quando ele vai reproduzir ou inverter? Ao entrar naquela área do Trigger. Para ativar isso, ligue os canais, como mostra a imagem abaixo:



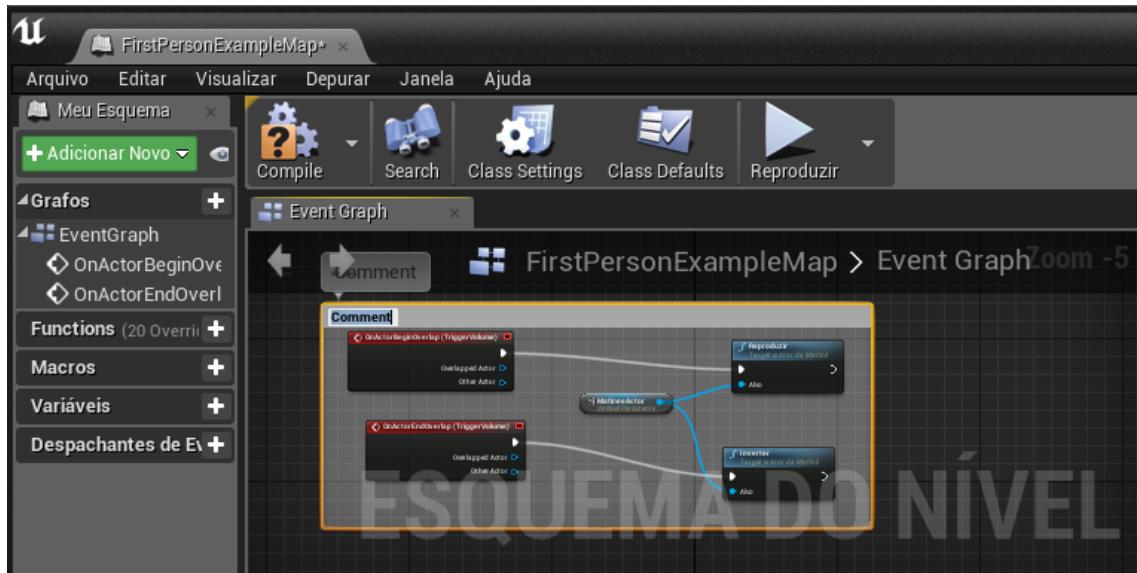
48- Para saber se está tudo certo, clique em compile, para o programa dizer se está correto:



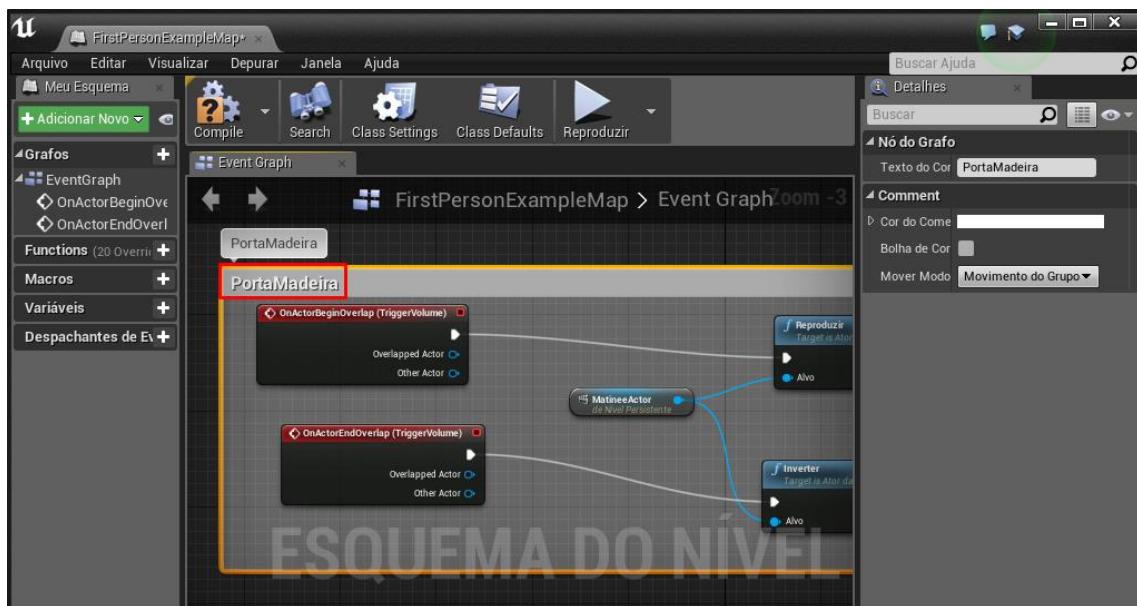
49- Aparecendo o certinho verde, quer dizer que está tudo ok:



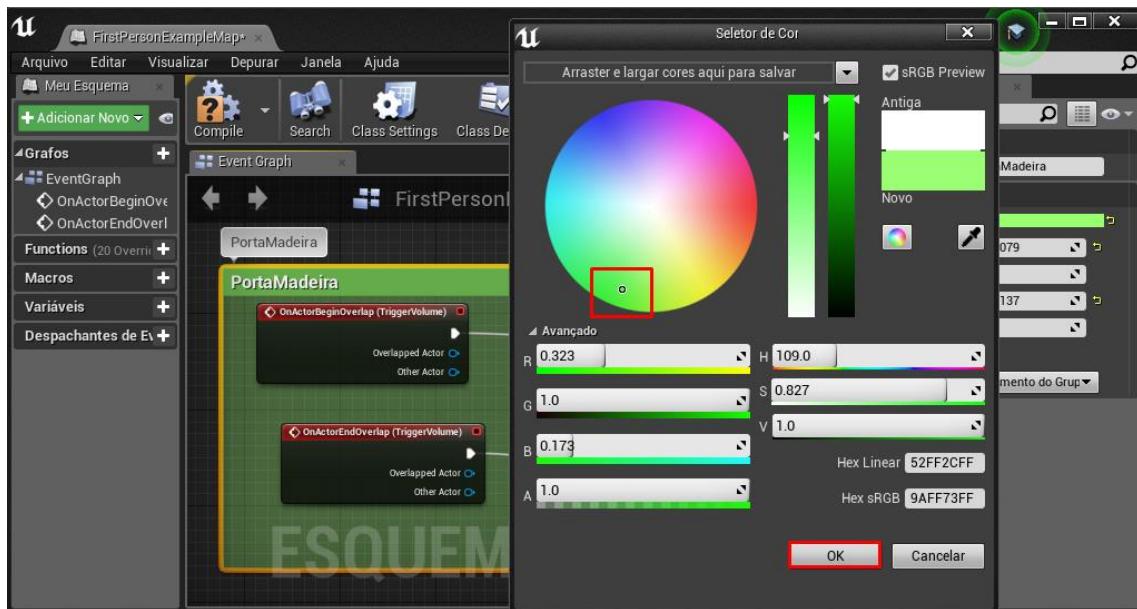
50- Agora, vamos mostrar uma maneira de deixar essas ações organizadas, para que não haja confusão e você se perca caso o seu game apresente várias ações. Então, selecione todos os objetos e depois aperte C no seu teclado:



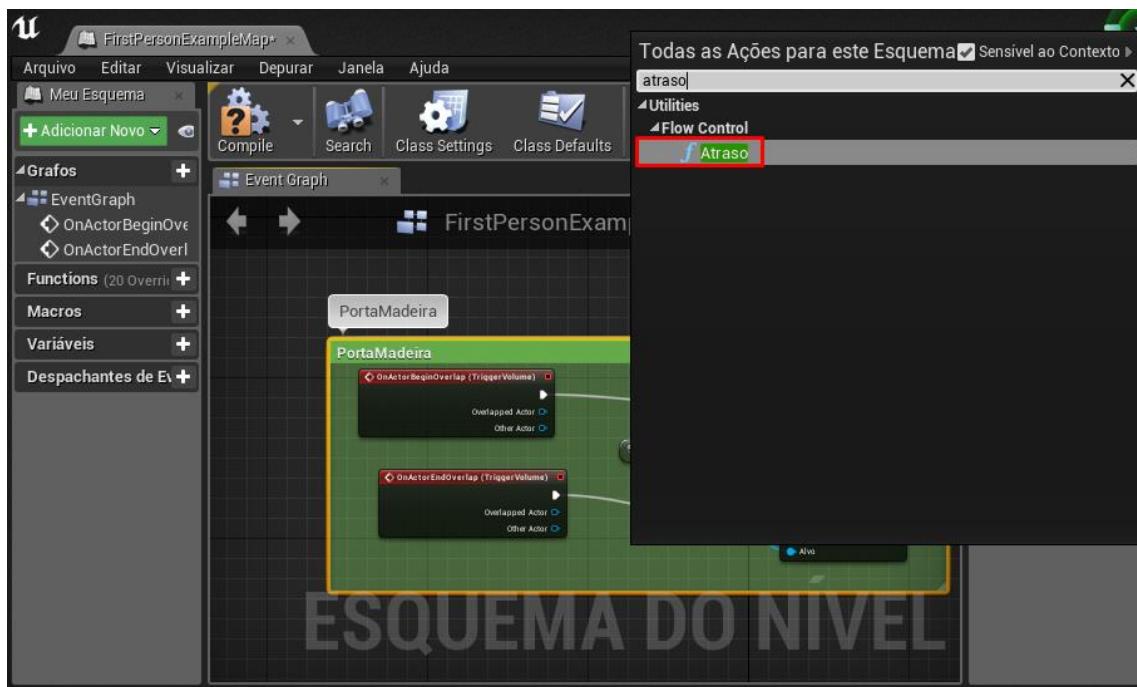
51- Vai abrir essa janela de comentário. Onde diz Comment, vamos renomear para PortaMadeira:

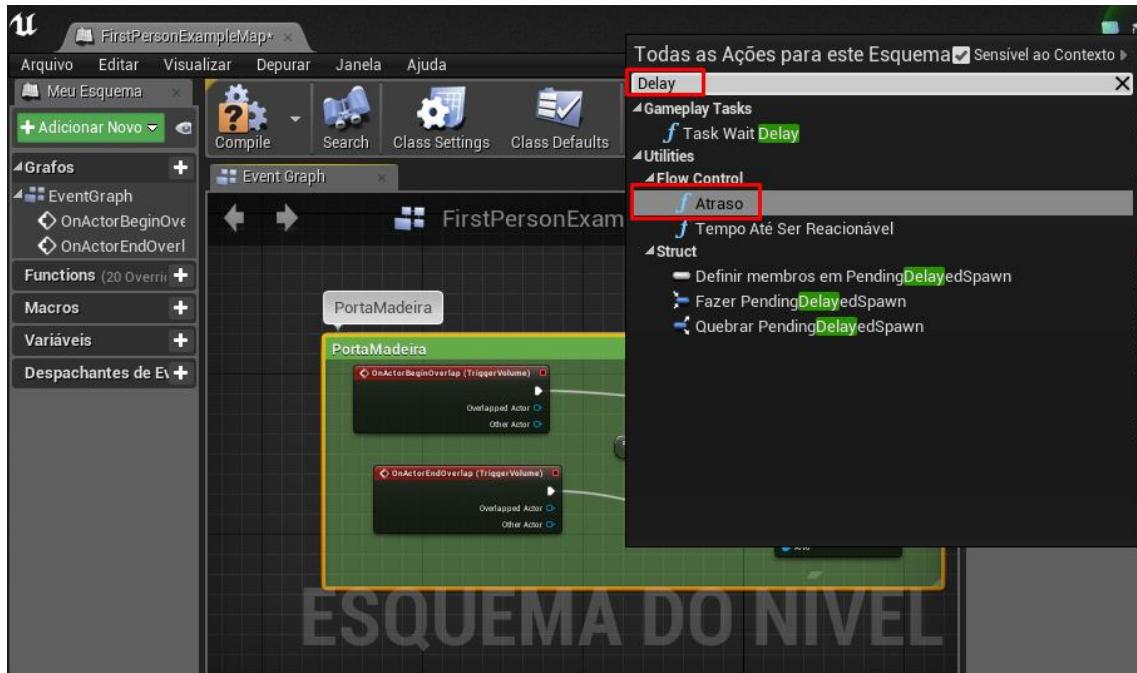


52- Em detalhes, dê um duplo clique em Cor do Comentário e escolha a cor abaixo, clicando em ok:

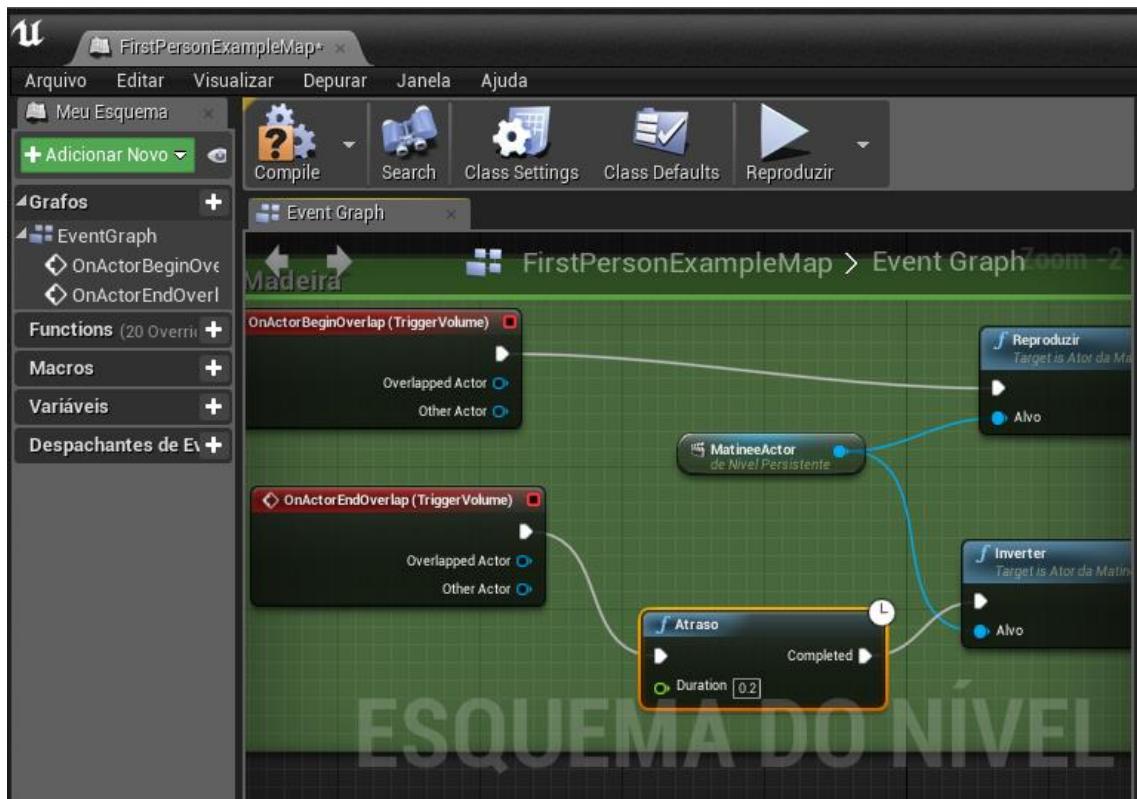


53- Está quase tudo pronto, só falta adicionar um atraso na porta, para quando você passar ela não fechar direto. Para isso, clique com botão direito no lugar vazio e digite Atraso ou Delay, ambas vão achar a ação:

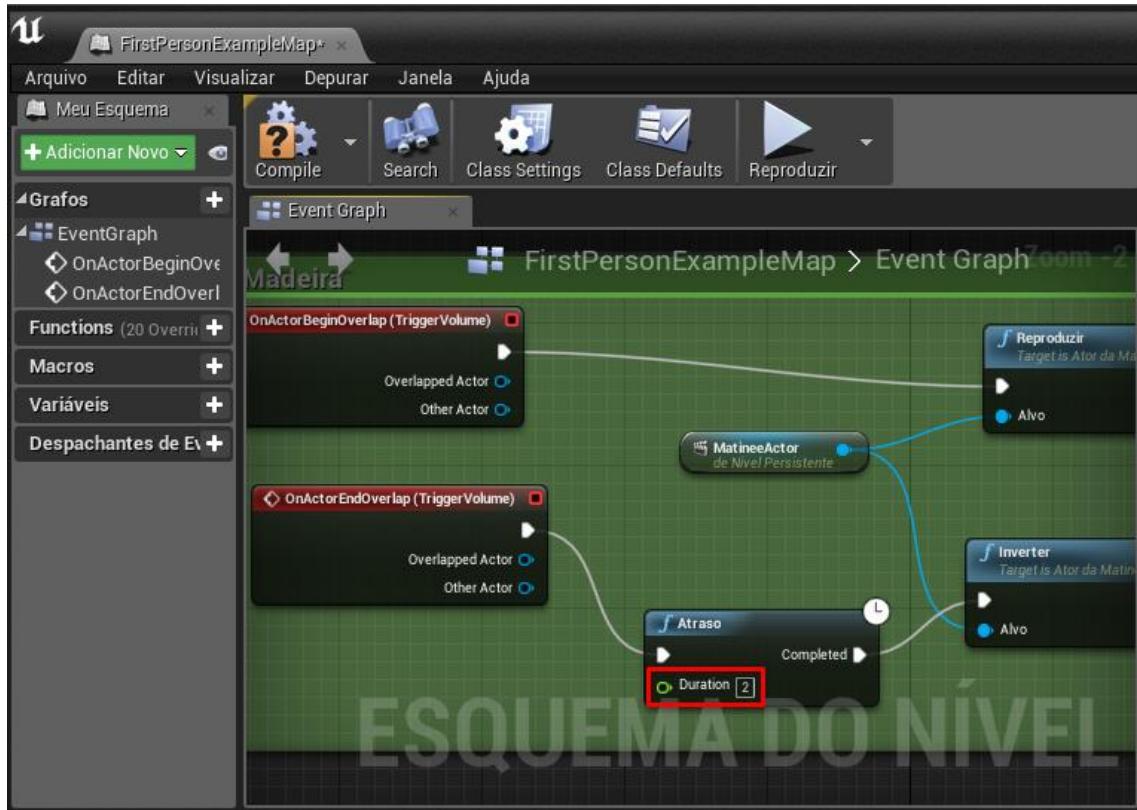




54- Clique na ação atraso para adicionar e conecte os canais de acordo com a imagem abaixo:



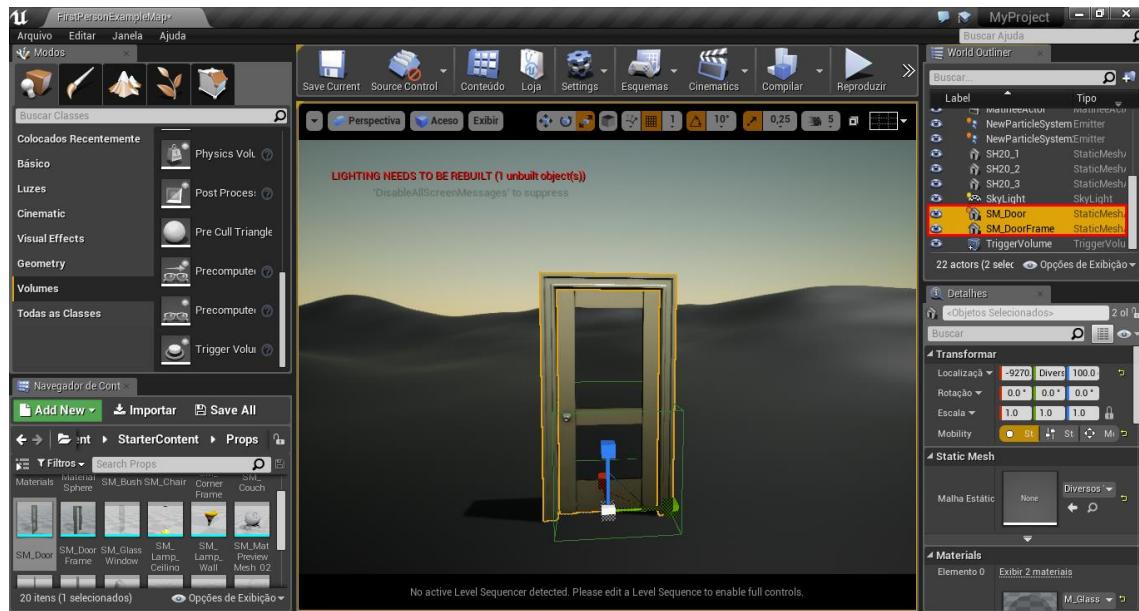
55- Perceba que o atraso foi conectado na ação end e inverter. Isso quer dizer que o atraso só vai ocorrer quando ela for fechar. Em duration, digite 2, que seria 2 segundos:



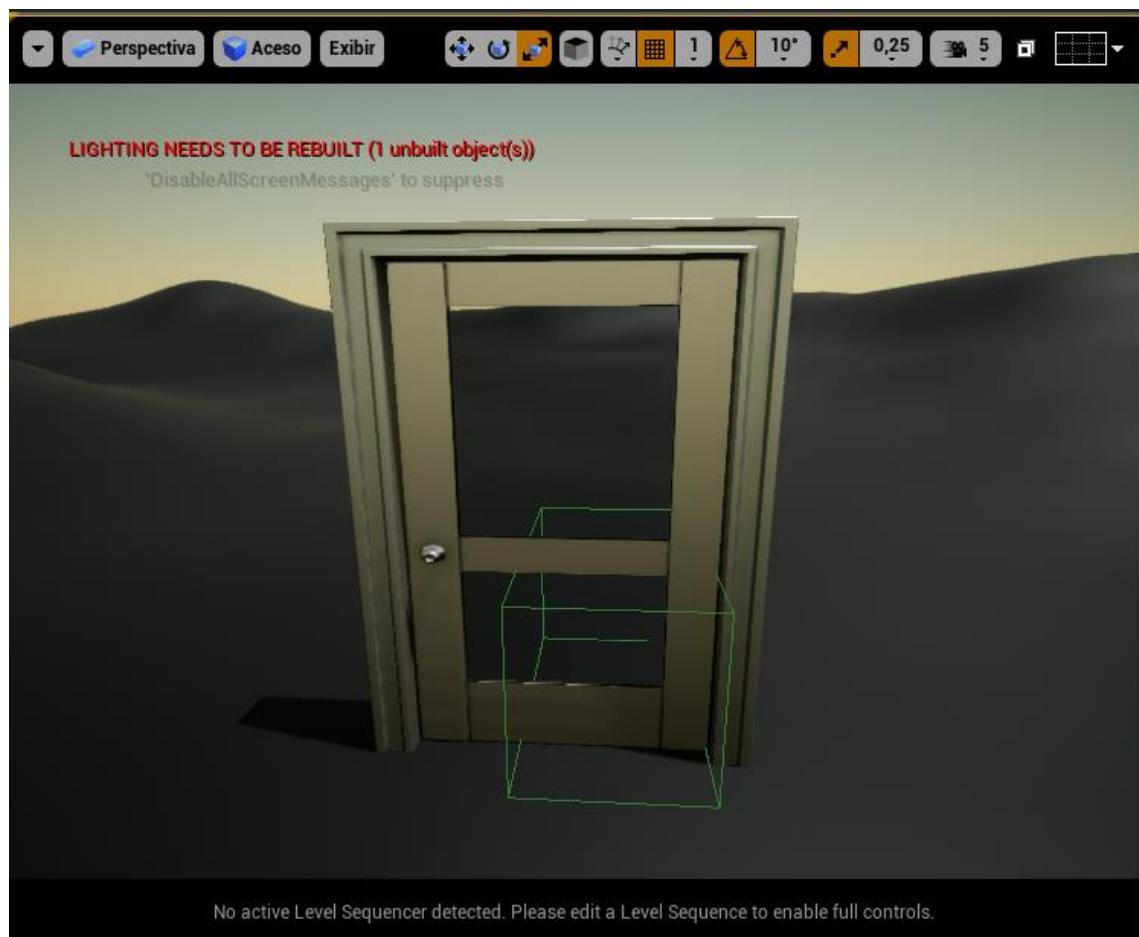
56- Clique em compile novamente e feche o editor de Blueprint:



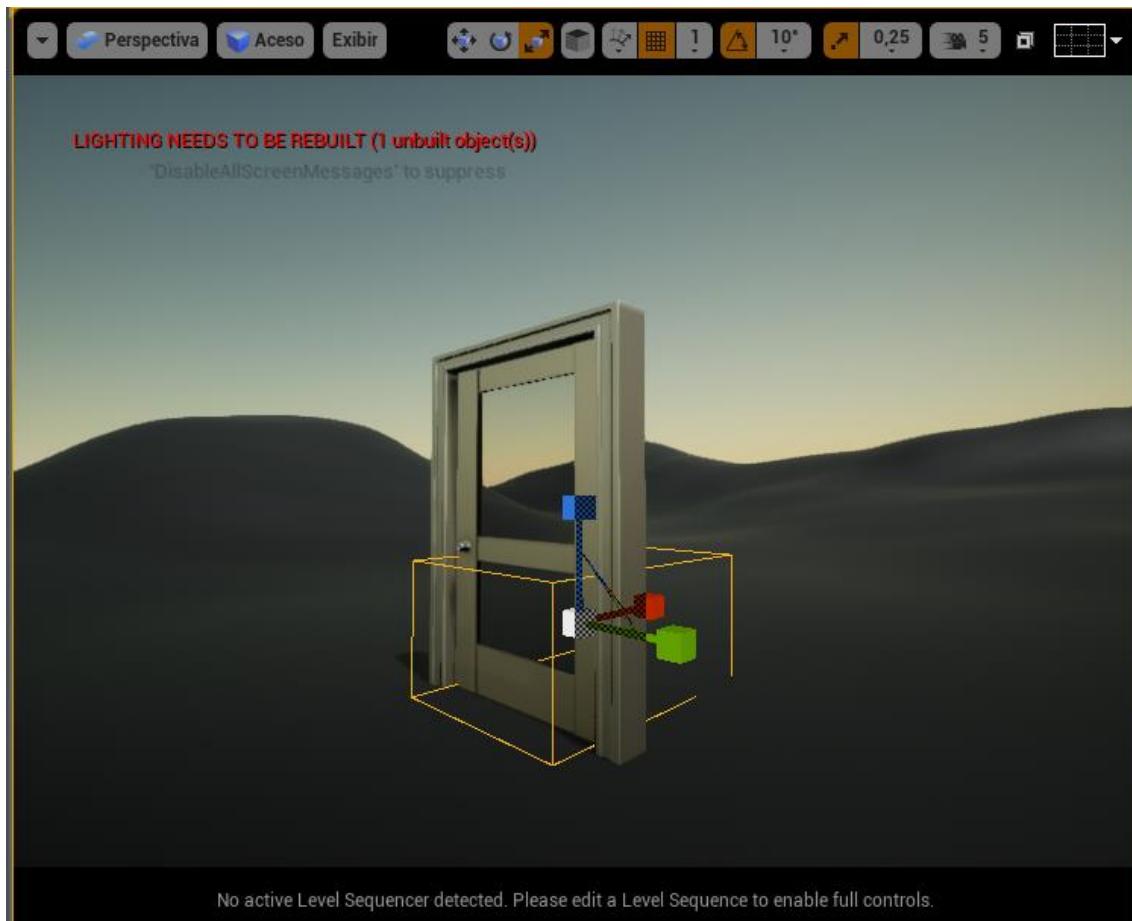
57- Se você clicar em reproduzir, verá que ao chegar perto da porta ela abre, mas você não conseguirá passar, pois está pequena. Para resolver isso, basta aumentar seu tamanho. Selecione a porta e a armação:



58- Usando as setas de modificação de tamanho, aumente o tamanho da porta:



59- Aumente também o tamanho do Trigger:



A parte de trás deve ser maior, pelo fato de que, como a porta abre para trás, se você tivesse que encostar nela para abrir, ela bateria em você e fecharia, por isso deixe um espaço maior para abrir por trás.

60- Caso a porta fique com algum vânio ou algum bug, basta arrumar recriando a animação cinematic, isso pode ocorrer pelo fato de que tivemos que aumentar a porta. Muito bem, salve o projeto e feche a engine.