

Grupos de Colisão

Transcrição

O problema que temos em mãos é que a nave inimiga é criada, aparece na cena, se move, mas logo depois desaparece.

Já consegui imaginar por quê? O que acontece é que logo após a nave aparecer, ela dispara um tiro na direção do jogador, mas ao mesmo tempo ela se move na mesma direção. Isso faz com que a nave colida com seu próprio tiro, resultando no desaparecimento de ambos.

Uma solução fácil para isso é criar agrupamentos de colisão. Um tiro disparado pela nave inimiga não pode colidir com ela mesma, assim como o jogador não pode colidir com seu próprio tiro. Essa é a ideia dos grupos.

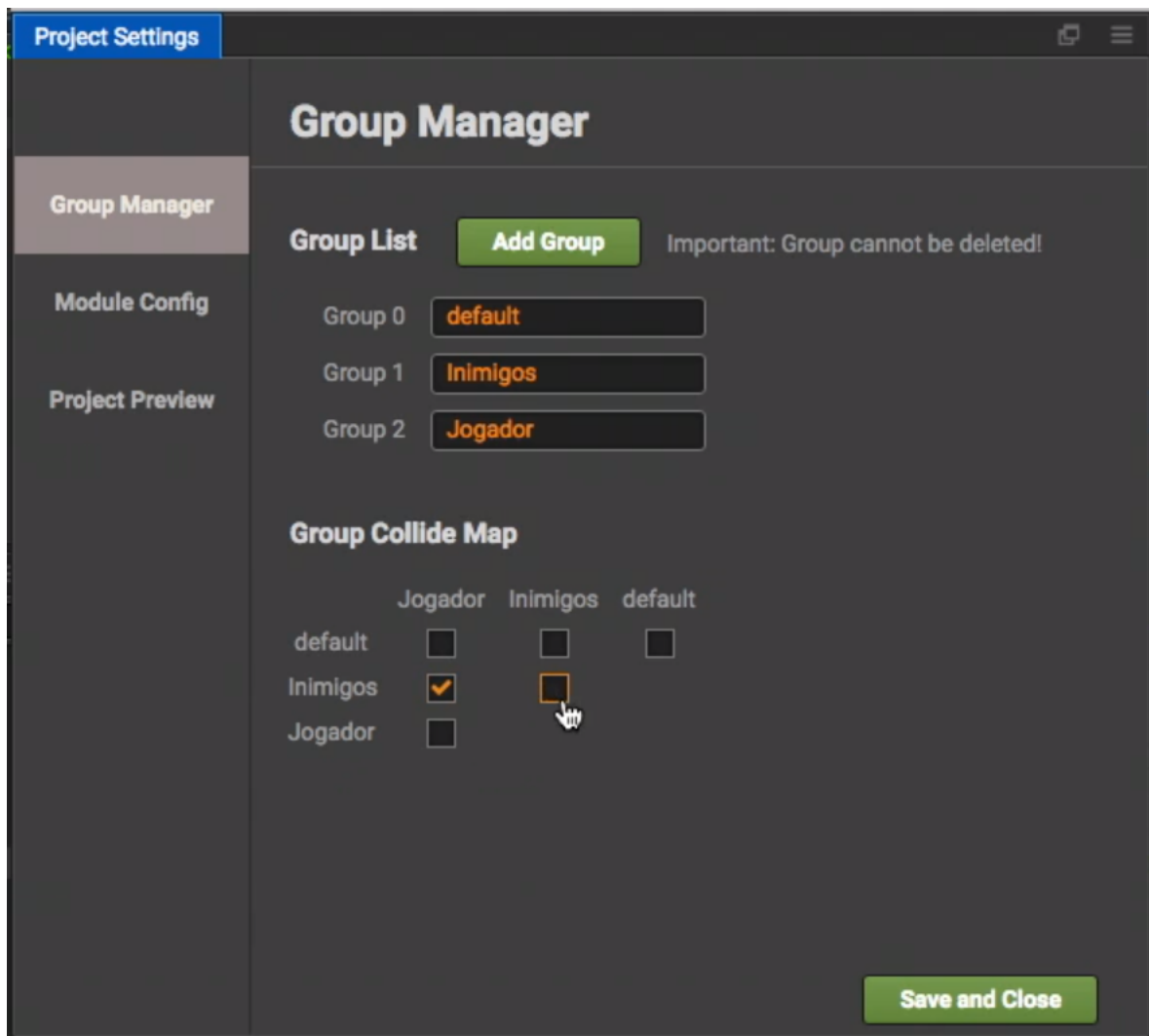
Criaremos grupos para que, dessa forma, seja possível configurarmos as regras de colisão. Com o objeto `hero` selecionado, por exemplo, podemos ir até a opção `Group` da aba `Properties` e selecionar o grupo ao qual o objeto pertence.



Os grupos `Inimigos` e `Jogador`, como visto na imagem acima ainda, não existem. Podemos facilmente criá-los clicando no botão `Edit` ao lado do botão de seleção. Será aberta uma janela de `Group Manager`.

Nesta janela podemos clicar em `Add Group` para adicionar mais grupos. Na listagem logo abaixo deste botão, podemos renomeá-los. Criaremos dois grupos chamados `Inimigos` e `Jogador` como visto na imagem anterior. Na sessão `Group Collide Map` da mesma janela, precisamos selecionar qual grupo colide com qual.

Por hora, ignoramos o grupo `default` e marcaremos apenas a linha `Inimigos` que está na coluna `Jogador`. Isso por que será apenas neste caso que haverá colisão. Depois clicamos em `Save and Close`.



Agora temos que selecionar o jogador e escolher seu grupo como `Jogador` como visto na primeira imagem. Faremos o mesmo com o inimigo, selecionando o grupo `Inimigos`. Lembrando de salvar o *prefab* no caso do inimigo.

Isso resolve o caso das naves, mas e o tiro? Há apenas um *prefab* do tiro, não podemos selecionar dois grupos. Neste caso, teremos que definir o grupo do tiro programaticamente. É um código muito simples. Só precisamos informar que o tiro pertence ao mesmo grupo de quem está atirando. No método `atirar` do *script* `Jogador.js` teremos a adição da seguinte linha:

```
tiro.group = this.node.group;
```

E no método `atirar` do *script* `Inimigo.js` adicionaremos:

```
disparo.group = this.node.group;
```

Agora sim temos este problema solucionado. Com isso, as naves inimigas aparecem e atiram normalmente contra nós!



Como podemos observar, quando o inimigo dispara contra o jogador acontece algo estranho. O tiro passa direto e não destrói o jogador. Isso passou despercebido até agora porque o inimigo não disparava antes. Para solucionar isso basta selecionarmos o objeto `hero` e na opção `Add Component` da aba `Properties`, usaremos as opções `Add Collider Component` -> `Box Collider`.

Com isso, já teremos o jogador sumindo ao ser atingido por um tiro do inimigo. E desse passo surgem outros problemas. Vejamos:

1. Após o jogador ser atingido, o jogo para e temos erros no console. Isso porque a nave inimiga não tem mais um ponto de referência para se movimentar, muito menos para atirar.
2. O tiro agora está sendo utilizado para destruir tanto o inimigo quanto o jogador, sendo que estes dois objetos possuem comportamentos diferentes. Quando o inimigo for atingido, ele deve sumir imediatamente, já o jogador não, ele deve perder uma vida que estará configurada para ele.