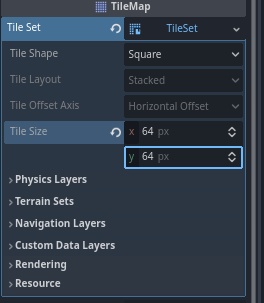
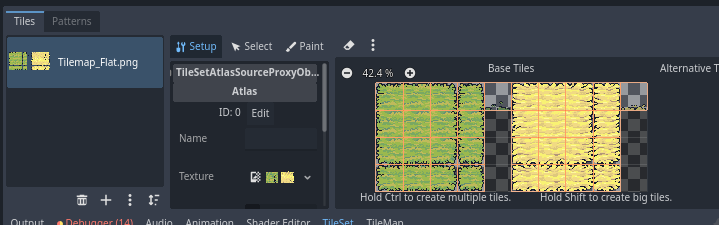
**Criando o Cenário:**

Criamos uma cena chamada main! Ela vai ser nossa cena principal. Juntamos tudo inicialmente a cena (player, mobspawner, interface)

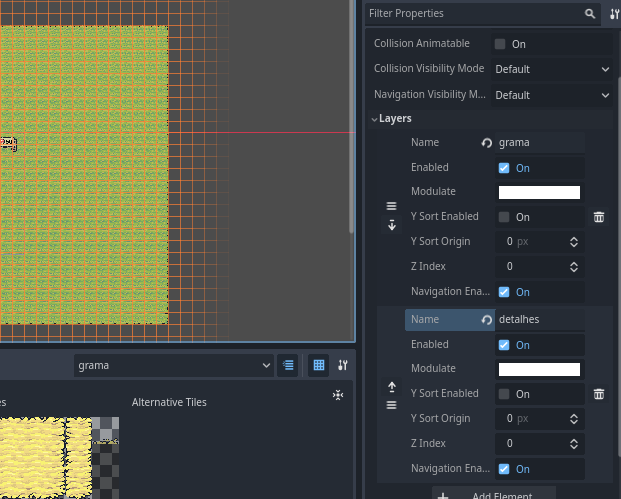
Para criar o cenário a gente adiciona um **tilemap,** nele criamos um novo tile set e ajustamos o tamanho para o do nosso cenário (64)



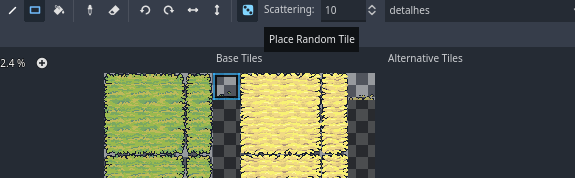


Depois de pintar o mapa, podemos usar camadas para fazer alguns efeitos.

Mexemos em layers



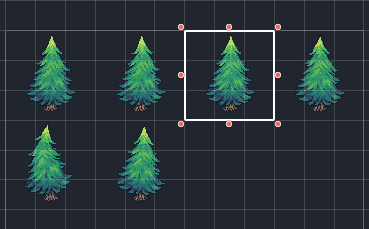
Se a gente marcar os dados e definir 10, existe 10% de chance dele pintar a área, assim fazemos:

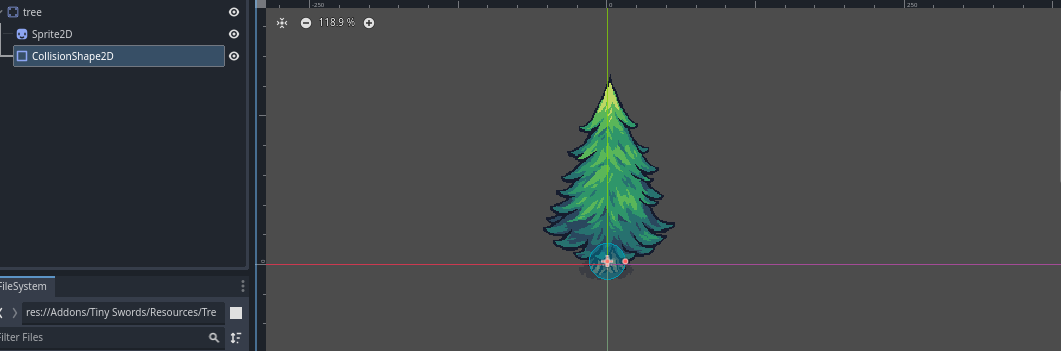


**Criando arvore:**

Eu criei uma nova cena, no qual é um staticbody2d, nele criei um collisionshape e um Sprite2d para receber a imagem.

Ai no sprite2d em region> habilitei e editei e selecionei a arvore que eu quero:

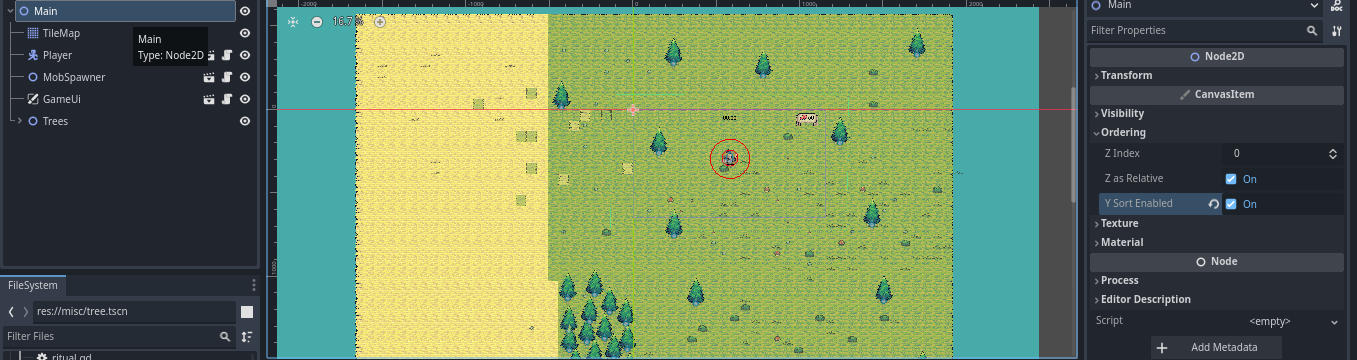




Para criar varias arvore eu crio um node2d na cena principal, arrasto a cena da arvore e coloco dentro desse node, ai só duplicar

**Ordem de renderização:**

Vai no main> ordering> e marca y sort enable



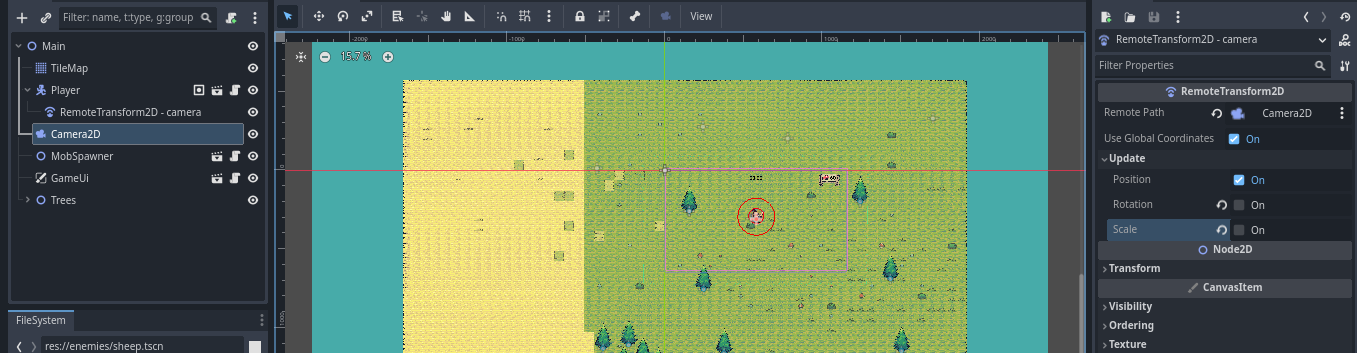
E marcar isso nas arvores.

**Movendo a câmera com o jogador:**

Criamos um node chamado camera2d no qual ela define o que vai ser mostrado na tela.

E arrastra ela para dentro do player (mas isso pode ser feito de outra forma)

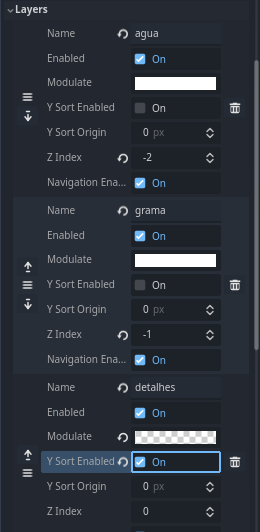
Dentro do player criamos um child node que se chama: RemoteTransform2d – ele pega tudo do jogador e empurra para outro node.





Bug de quando o personagem vai para cima e some:

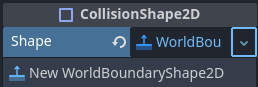
Marca o ysortenabel no mapa e também mexer nas camadas no layer- zindex



**Definindo limites do mundo:**

Existe varias formas de fazer isso.

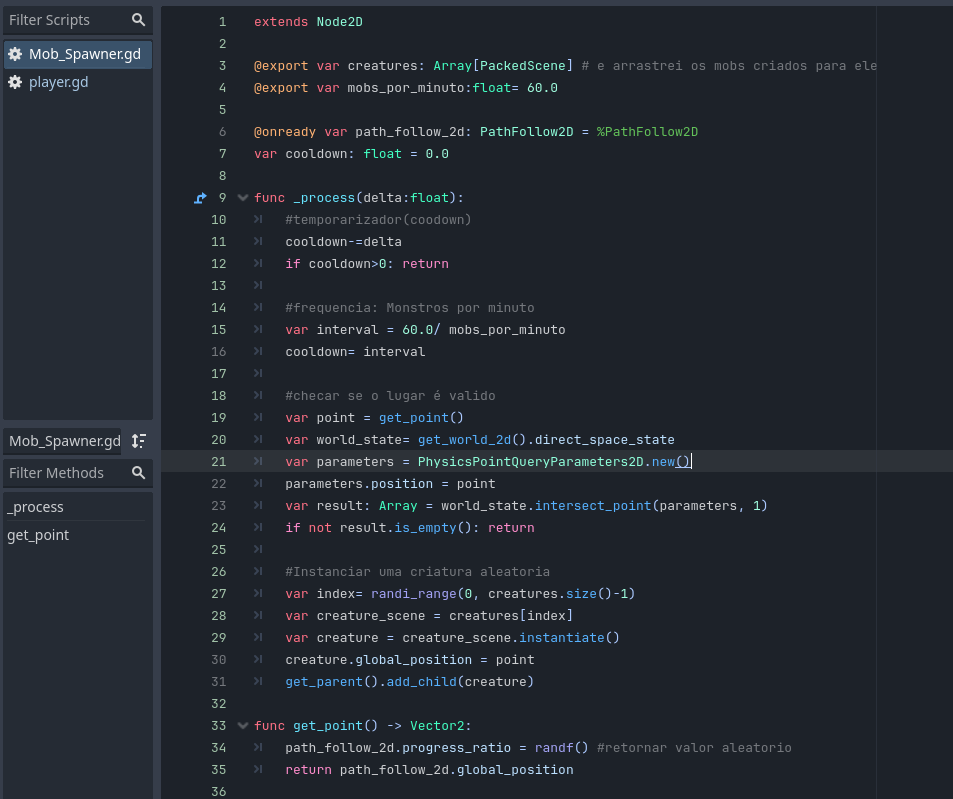
O que vamos fazer de inicio é criar um novo staticbody2d com collisionshape, e usar o worldboundaryshape.



Ou podemos usar um retângulo mesmo e quando fazer marca “make unique”  para poder duplicar.

Só que com isso o mobspawner pode nascer depois do limite do mapa.

Então vamos mexer no mobspawner:



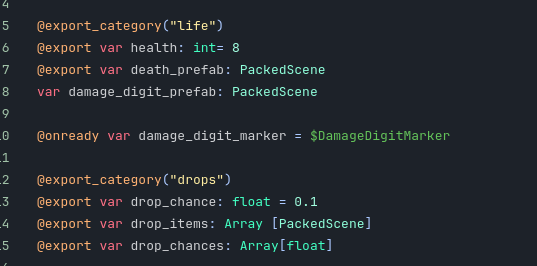
Só que desse jeito se for nascer perto de uma arvore, ele também ignora. Então colocamos o world\_limit na camada 4 e perguntamos para o código.



**Criando sistema de dropping dos inimigos:**

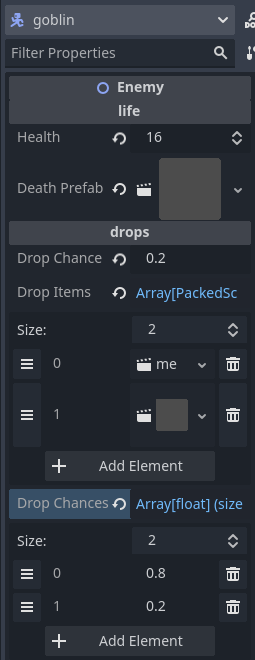
Vamos colocar para quando matar um monstro ele dropar carne.

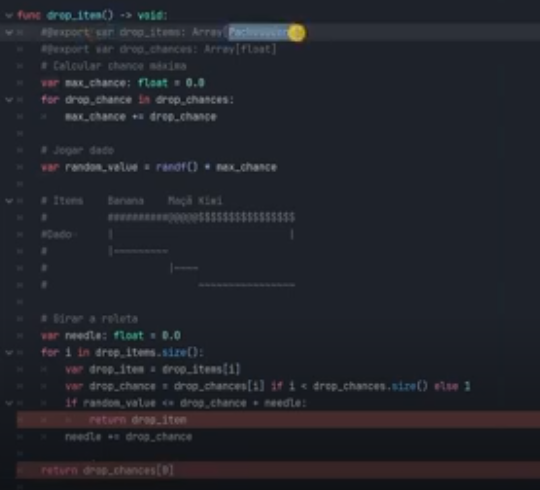
No script do inimigo:

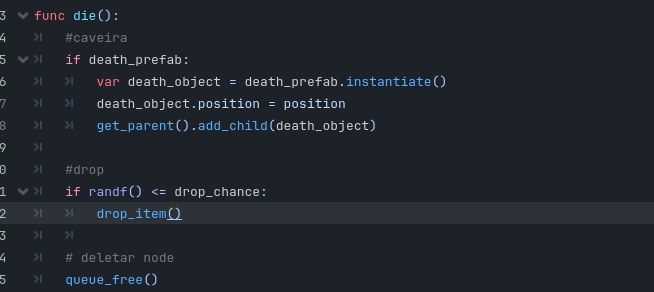


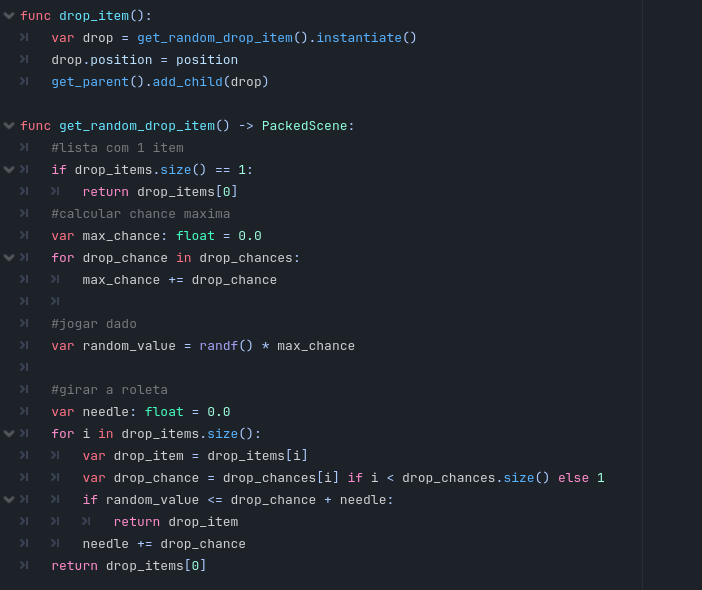
Em cada inimigo eu arrasto a carne para o drop\_items e também coloco a % de chance de drop. Além disso, coloquei um drop\_chances para caso tenha 2 items eu fazer a % de dropar cada item.

Ex: globin, coloquei uma carne dourada que recupera 100% de vida.

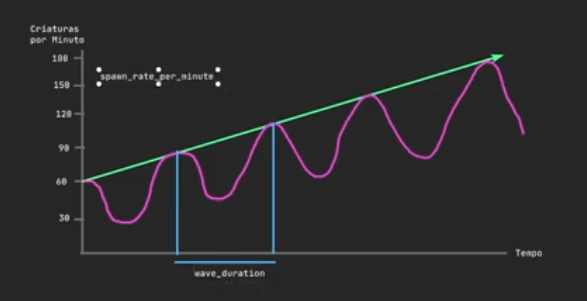




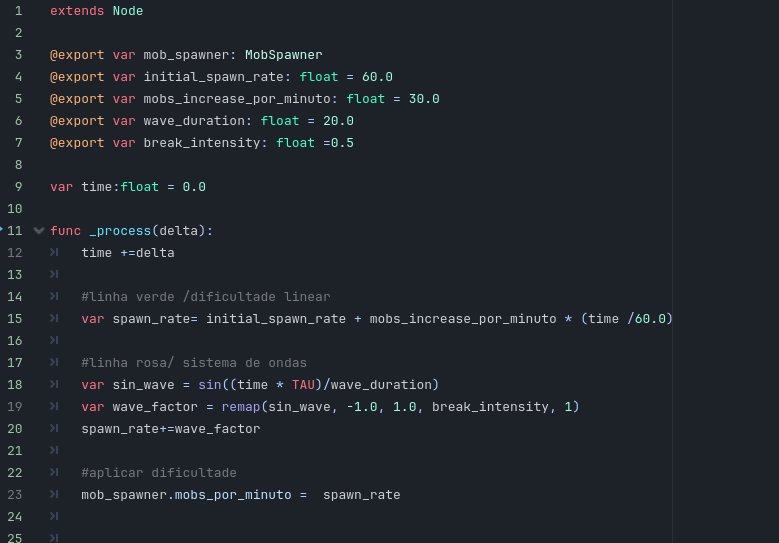




**Onda de dificuldade:**



Criei um novo script na pasta systems:



**Interface do game over:**

Vai ser parecido com a GameUi, a gente cria um canvaslayer com ancora pegando toda a tela. Ai coloca um coloreact preto com um panel que iremos personalizar a textura utilizando dos assets.





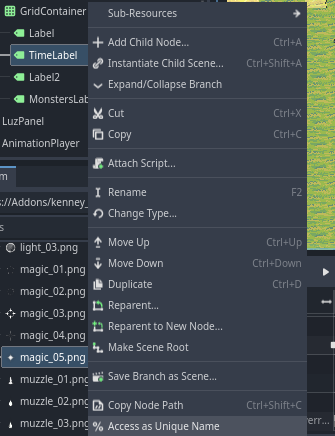
Vamos criar um novo node que ainda não vimos que se chama: GridContainer – container que ajusta o tamanho automaticamente.



Vamos criar agora um animationplayer nela para animar o blackground e os painéis.



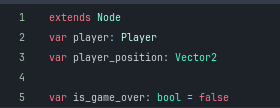
No timelab e no monsterlab vou da acesso para usar no script



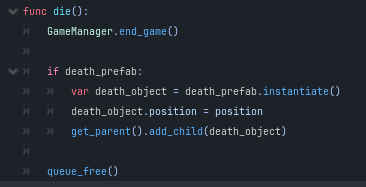
E transformei numa cena.

**Programando para dizer que o jogo acabou:**

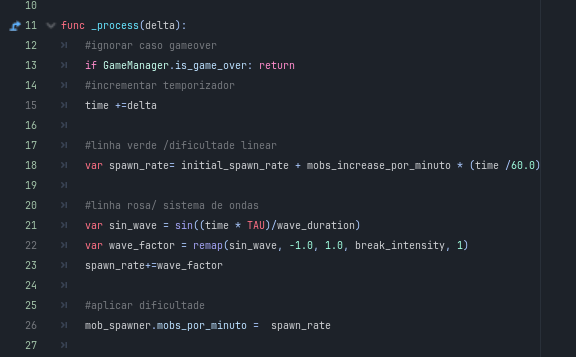
No gamemanager:



No player:

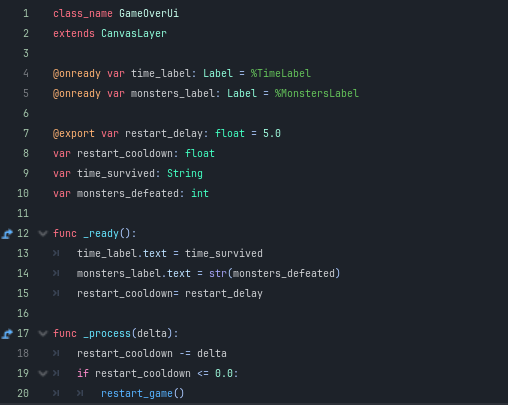


No dificultysytem:



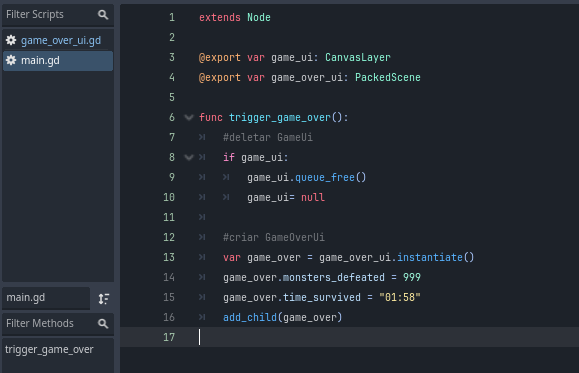
E colocar essa primeira linha em todos os \_process importante para dizer que o jogo acabou (como mobspawner, followplayer).

Agora no script do gameover:

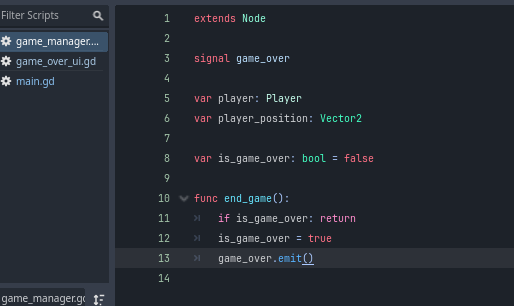


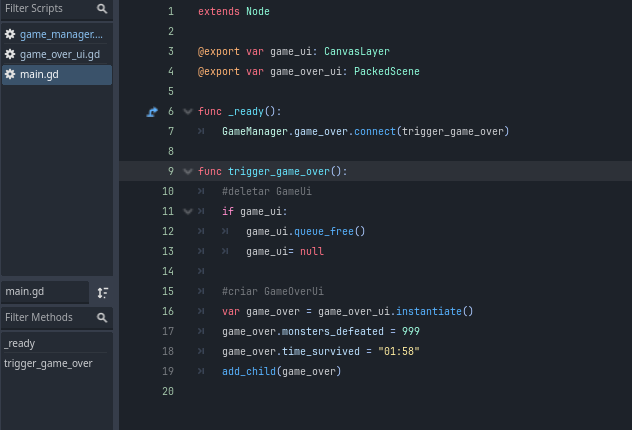
**Agora como podemos chamar essa cena de morte?**

Criamos um script na cena do jogo:

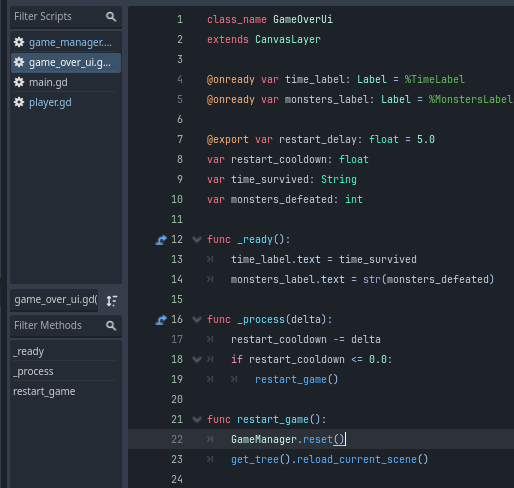


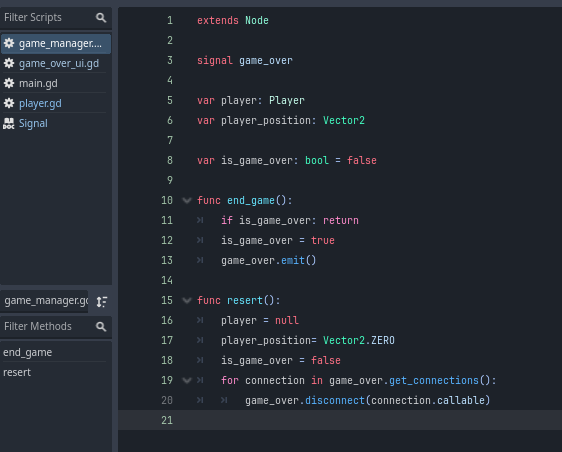
Emitindo o sinal para todos saberem que morreu:





E para reiniciar o jogo?





Detalhes finais:

No game ui que esta contando o tempo, contudo será melhor se isso for transferido para o game manager

