Curso Técnico

de Programação de Jogos Digitais

Aula 22

índice

[1. Destruir Objetos fora de Tela 3](#_Toc169602285)

[2. Exemplo de uso 3](#_Toc169602286)

[3. Controle de objeto em nosso projeto 4](#_Toc169602287)

[4. Entendo o código da aula 6](#_Toc169602288)

[5. Referencias 8](#_Toc169602289)

# Destruir Objetos fora de Tela

Destruir objetos que saíram da tela é fundamental para manter a performance e a eficiência do jogo. Objetos que não são mais visíveis ou necessários continuam a consumir recursos do sistema, como memória e processamento, o que pode causar lentidão e quedas de desempenho. Ao remover esses objetos, o jogo pode rodar mais suavemente, especialmente em dispositivos com recursos limitados, como smartphones.

Objetos não visíveis que permanecem ativos podem levar a um consumo excessivo de memória, resultando em vazamentos de memória e, eventualmente, em travamentos ou falhas do jogo. A prática de destruir objetos desnecessários ajuda a liberar memória, permitindo que ela seja utilizada de forma mais eficiente por outros elementos do jogo.

Manter a cena do jogo limpa e organizada é essencial para o desenvolvimento e a manutenção do jogo. Objetos que saem da tela e não são removidos podem se acumular, tornando a cena desordenada e difícil de gerenciar. Remover esses objetos ajuda a manter o ambiente de desenvolvimento mais claro e menos propenso a erros.

# Exemplo de uso

Em jogos de plataforma ou jogos de tiro, inimigos e projéteis frequentemente se movem para fora da tela. Continuar a processar esses objetos pode ser desnecessário e prejudicial ao desempenho. Implementar a destruição desses objetos é uma prática comum e recomendada.

Aqui está um exemplo de como destruir objetos que saíram da tela em um jogo desenvolvido com Unity e C:

1. Definir Limites da Tela:

Definir os limites da tela para verificar quando um objeto sai dos limites visíveis.



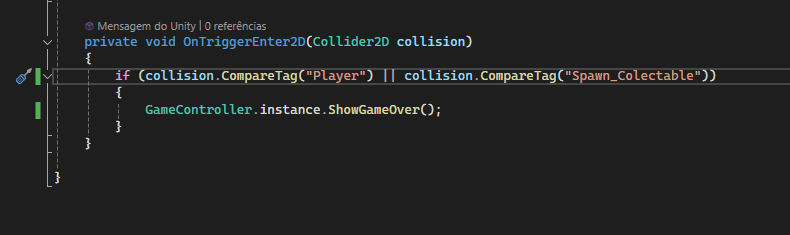
Anexar o script `DestroyOffScreen` a qualquer objeto que deva ser destruído ao sair da tela, como projéteis ou inimigos.



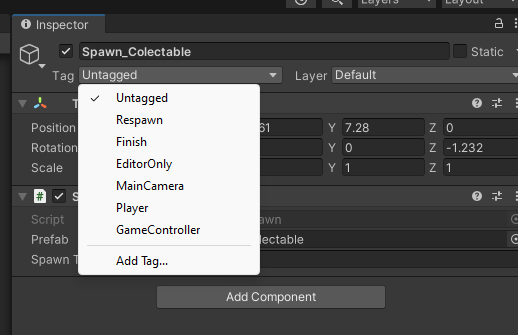
Destruir objetos que saem da tela é uma prática no desenvolvimento de jogos para garantir desempenho, eficiência e organização.

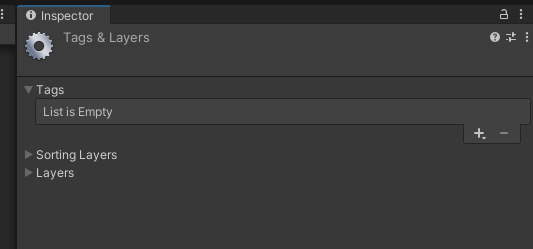
# Controle de objeto em nosso projeto

Após o objeto colecionável ser pego devemos destrui-lo e para isso vamos adicionar a seguinte linha no script obstacle

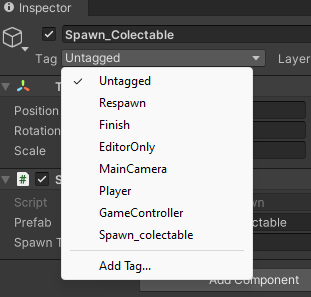


Selecione o objeto spawn\_Colectable e adicione uma tag





Escolha a tag criada

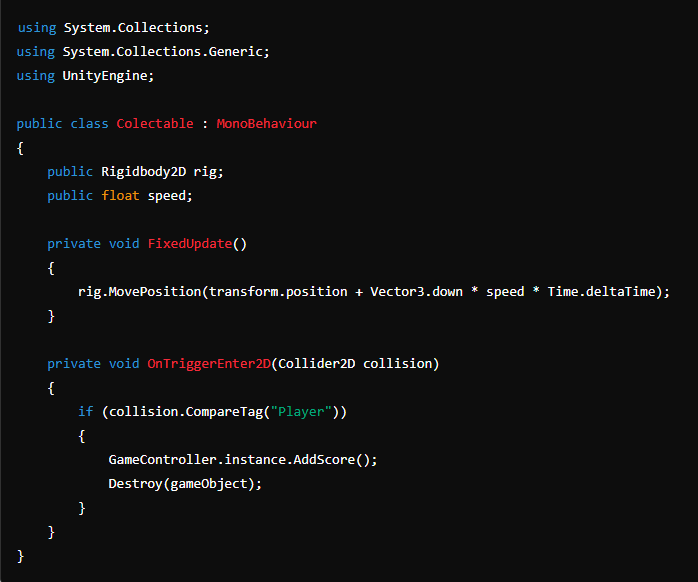


Ainda não iremos exportar para um apk pois ainda devemos aprender mais alguns conceitos

# Entendo o código da aula

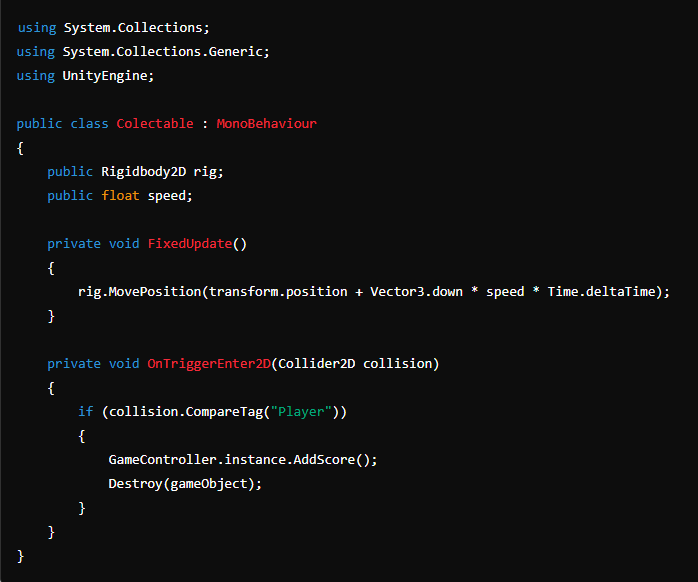
Vamos analisar e explicar detalhadamente o código fornecido, linha por linha:

Importação de Namespaces:



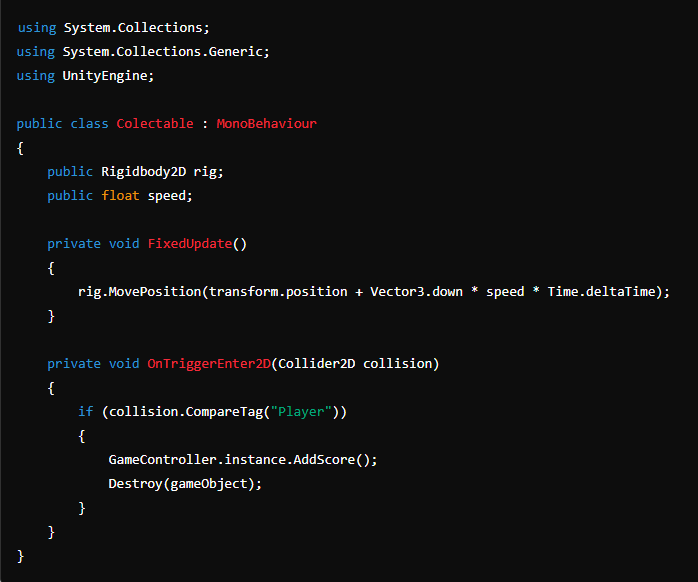
* Estes `using` statements importam namespaces necessários. `System.Collections` e `System.Collections.Generic` são usados para trabalhar com coleções de dados, como listas e arrays. `UnityEngine` fornece acesso às funcionalidades do Unity.

Declaração da Classe Colectable:



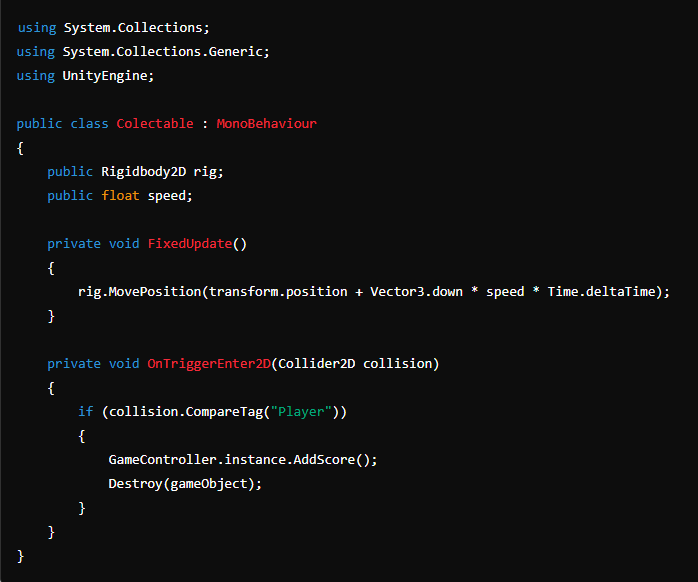
* Declara a classe `Colectable`, que herda de `MonoBehaviour`, tornando-a um componente que pode ser anexado a objetos no Unity.

Declaração de Variáveis Públicas:



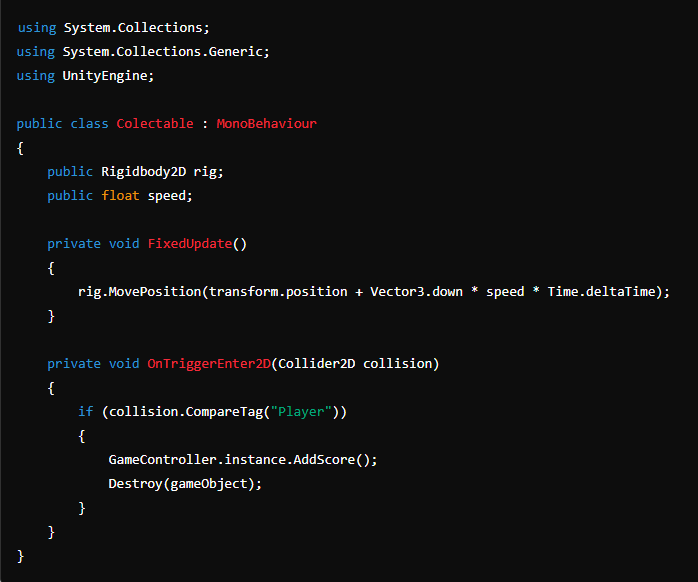
* `rig` é uma variável pública do tipo `Rigidbody2D`, que será usada para manipular a física do objeto.
* `speed` é uma variável pública do tipo `float` que controla a velocidade do movimento do objeto.

Método FixedUpdate:



* `FixedUpdate` é chamado em intervalos fixos, ideal para atualizações relacionadas à física.
* O método move o `Rigidbody2D` para baixo ao longo do eixo Y a uma velocidade constante. A posição é ajustada com base na variável `speed` e no tempo decorrido (`Time.deltaTime`) desde o último frame, garantindo um movimento suave e consistente.

Método OnTriggerEnter2D:



* `OnTriggerEnter2D` é chamado quando outro `Collider2D` entra no trigger collider anexado ao objeto do script.
* Verifica se o objeto que colidiu possui a tag "Player".
* Se a tag do objeto colidido for "Player", o método `AddScore` do `GameController` é chamado para adicionar pontos ao jogador.
* Após isso, o próprio objeto coletável é destruído usando `Destroy(gameObject)`, removendo-o da cena.

O script `Colectable` controla um objeto coletável no jogo, fazendo com que ele se mova para baixo constantemente. Quando o objeto coletável colide com um objeto marcado como "Player", ele adiciona pontos ao jogador e depois se destrói. Este comportamento é típico em jogos onde o jogador coleta itens para ganhar pontos ou habilidades.

Os códigos usados em nossas aulas estão disponíveis em nosso repositório no GitHub.



https://github.com/rildexter/pjd/tree/main/djm

# Referencias

1. Unity Technologies. (2020). **Unity User Manual.** Unity Technologies.
2. [C#, 2020] **Visual C# Developer Center**, Microsoft Docs.

Microsoft Corporation. (2020).

1. Geig, M. (2018). **Unity 2018 Game Development in 24 Hours**, Sams Teach Yourself. Sams Publishing.
2. Hocking, J. (2015). **Unity in Action: Multiplatform Game Development in C#.** Manning Publications.