Curso Técnico

de Programação de Jogos Digitais

Aula 15

índice

[1. Adição de Bibliotecas em Scripts na Unity 3](#_Toc169514554)

[2. Por que Usar Namespaces? 3](#_Toc169514555)

[3. Ciclos de Vida em Unity: Start e Update 3](#_Toc169514556)

[4. Limites de tela 4](#_Toc169514557)

[5. Referencias 6](#_Toc169514558)

# Adição de Bibliotecas em Scripts na Unity

Em scripts de C Sharp na Unity, a adição de bibliotecas é essencial para ampliar a funcionalidade e simplificar o desenvolvimento de jogos. Isso é feito através da declaração de namespaces usando a palavra-chave `using` no início do script.

Namespaces são coleções organizadas de classes, interfaces, structs e outros tipos que permitem que os desenvolvedores acessem funcionalidades específicas sem precisar escrever código complexo do zero. Eles ajudam a evitar conflitos de nomes e tornam o código mais legível e modular.

# Por que Usar Namespaces?

1. Organização: Ajuda a organizar o código em grupos lógicos.

2. Legibilidade: Facilita a leitura e manutenção do código.

3. Modularidade: Permite usar bibliotecas específicas conforme necessário, sem sobrecarregar o script com funcionalidades desnecessárias.

4. Prevenção de Conflitos: Evita conflitos de nomes ao permitir a coexistência de classes com o mesmo nome em diferentes namespaces.

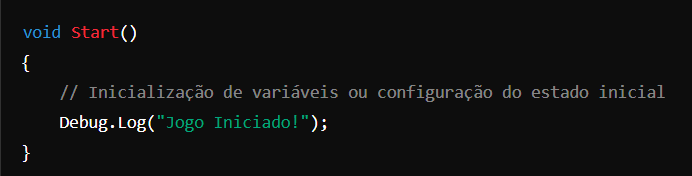


A adição de namespaces em scripts na Unity é uma prática fundamental que permite aos desenvolvedores utilizar coleções de classes e funcionalidades predefinidas, tornando o desenvolvimento de jogos mais eficiente e organizado.

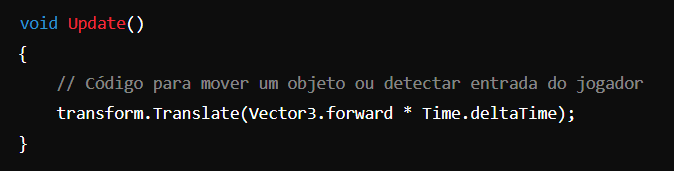
# Ciclos de Vida em Unity: Start e Update

Em scripts de C Sharp usados na Unity, os métodos `Start` e `Update` desempenham papéis fundamentais no ciclo de vida dos objetos de jogo.

O método `Start` é chamado uma vez ao iniciar o script. É utilizado para inicializações que precisam ocorrer antes do primeiro quadro do jogo. Aqui, você pode configurar variáveis, estados iniciais dos objetos, ou qualquer configuração que precise estar pronta quando o jogo começar a rodar.



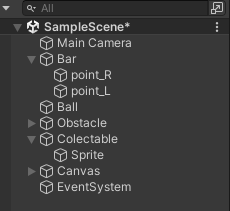
O método `Update` é chamado uma vez por quadro, tornando-o ideal para implementar a lógica do jogo que precisa ser verificada ou atualizada continuamente. Este método é usado para detectar entradas do jogador, mover objetos, verificar condições e atualizar a interface do usuário.





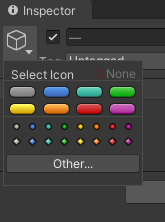
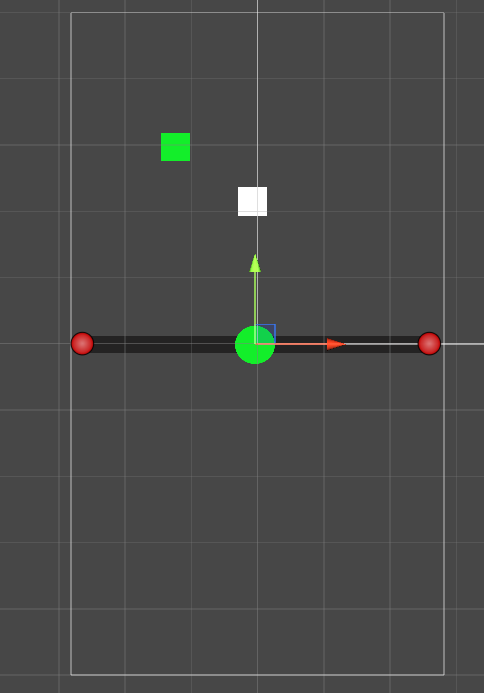
Update: Usado para lógica que deve ser verificada e atualizada continuamente a cada quadro. Ele é chamado repetidamente durante todo o ciclo de vida do jogo.

# Limites de tela

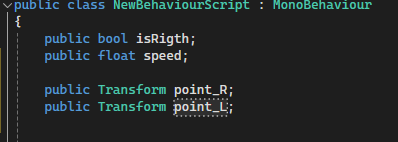
Em nossas últimas atualizações do projeto de aula podemos perceber que a bolinha do nosso jogo sai da tela então, devemos acrescentar um limite de tela.

Vamos então renomear nosso objeto barra e criar dois objetos vazios em seus extremos. A bolinha ao se aproximar a estes objetos deverão inverter o movimento da bolinha, da mesma forma que é feito através do clique em tela.

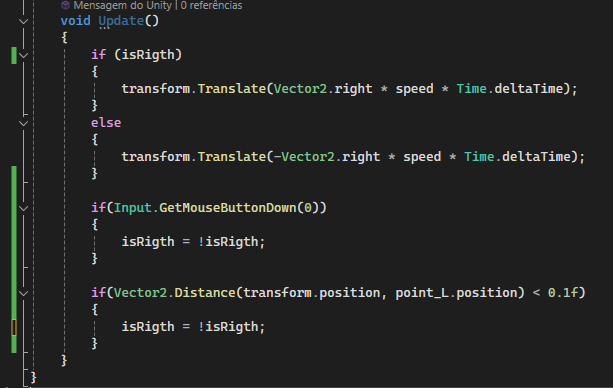
Para visualizar (temporariamente) os objetos vamos gerar um ícone para eles

Vamos para nosso script ball e então referenciar nossos dois objetos.

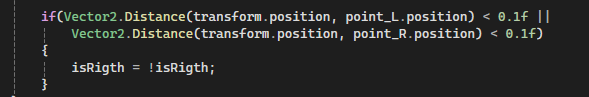


Primeiro passo é adicionar o reconhecimento dos dois ícones criados na Unity e então colocar a captura das distancias dos objetos.

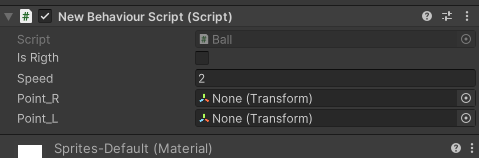


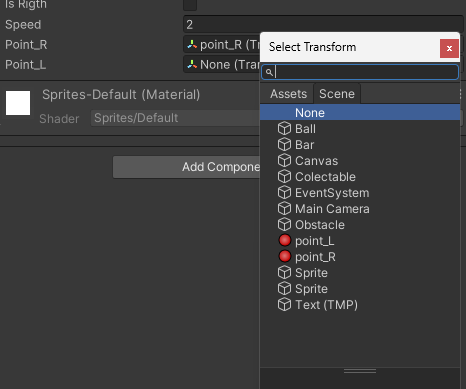
Então adicione a lógica para o outro lado :

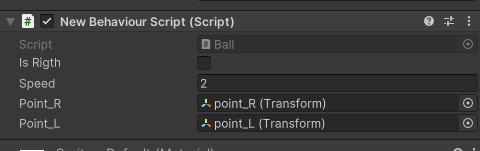
< 0.1f || Vector2.Distance(transform.position, point\_R.position) < 0.1f)



Retorne para a Unity e referencie os objetos point para a Ball. No inspector do objeto ball temos o Point\_R e Point\_L sem estarem apontados. Clique em cada um deles e escolha o ícone de cada um deles.







Faça novamente o teste de seu projeto.

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.