Curso Técnico

de Programação de Jogos Digitais

Aula 03

índice

[1. Estruturas de Controle de Fluxo 3](#_Toc169072520)

[2. Instruções Condicionais 3](#_Toc169072521)

[3. Estruturas de Controle de Exceções 5](#_Toc169072522)

[4. Aplicando Estruturas de Controle de Fluxo em Unity 6](#_Toc169072523)

[5. Referencias 7](#_Toc169072524)

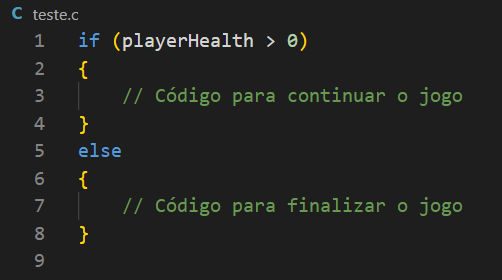
# Estruturas de Controle de Fluxo

Desenvolver jogos para dispositivos móveis utilizando a Unity requer um entendimento profundo das estruturas de controle de fluxo disponíveis na linguagem C. Essas estruturas são fundamentais para gerenciar a lógica e o comportamento do jogo, permitindo que o desenvolvedor controle o fluxo de execução do código de maneira eficiente e eficaz. Neste artigo, exploraremos as principais estruturas de controle de fluxo em C, como as instruções condicionais, loops e estruturas de controle de exceções, e como aplicá-las no desenvolvimento de jogos na Unity.

# Instruções Condicionais

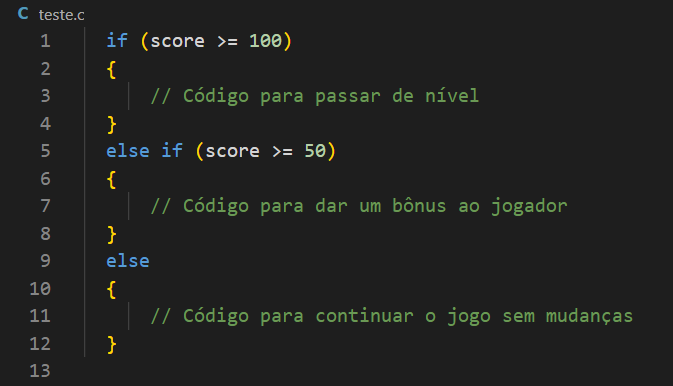
Para tomar decisões em seu jogo com base em certas condições, é essencial utilizar instruções condicionais. Em C, as principais instruções condicionais são `if`, `else if` e `else`.

A instrução `if` permite executar um bloco de código se uma condição especificada for verdadeira. O `else` fornece um caminho alternativo se a condição for falsa.



No desenvolvimento de jogos, isso pode ser usado para verificar se o jogador ainda está vivo, se um inimigo foi derrotado ou se um item foi coletado.

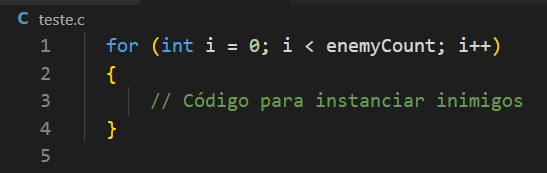
A instrução `else if` é usada para verificar múltiplas condições.



Este tipo de controle é útil para implementar diferentes níveis de dificuldade, recompensas ou eventos baseados na pontuação do jogador.

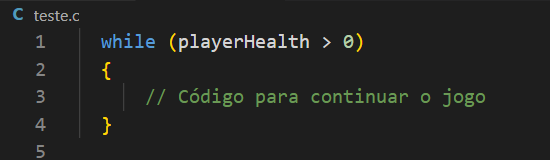
Os loops são utilizados para repetir uma sequência de instruções até que uma condição especificada seja atendida. Em C, os loops mais comuns são `for`, `while` e `do-while`.

O loop `for` é usado quando se sabe de antemão quantas vezes se quer repetir um bloco de código.



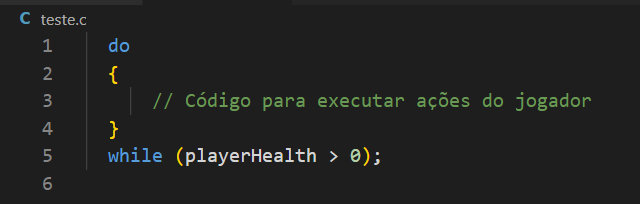
Em jogos, isso é frequentemente usado para criar múltiplos inimigos, distribuir objetos pelo cenário ou repetir ações a cada quadro.

O loop `while` repete um bloco de código enquanto uma condição for verdadeira.



Esse loop é útil para manter o jogo em execução enquanto certas condições são verdadeiras, como manter o jogador vivo ou o jogo ativo.

O loop `do-while` é semelhante ao `while`, mas garante que o bloco de código seja executado pelo menos uma vez.

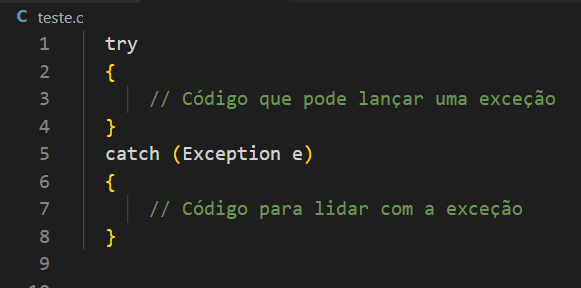


Este loop pode ser útil para operações que precisam ocorrer pelo menos uma vez antes de verificar a condição, como um menu inicial ou uma configuração de jogo.

# Estruturas de Controle de Exceções

Para lidar com erros e situações inesperadas de maneira graciosa, é essencial usar estruturas de controle de exceções. Em C, isso é feito utilizando `try`, `catch`, `finally` e `throw`.

O bloco `try` contém o código que pode gerar uma exceção, enquanto o bloco `catch` contém o código para lidar com essa exceção.

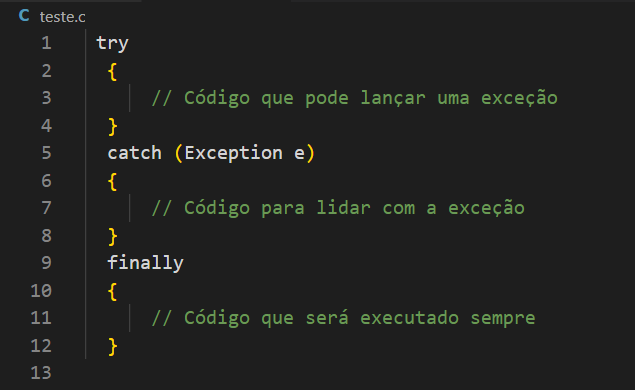


Em jogos, isso pode ser utilizado para gerenciar erros de leitura/escrita de arquivos, conexões de rede ou operações que dependem de recursos externos.



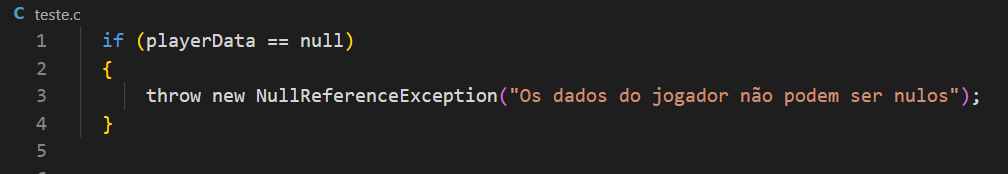
Em um jogo, a estrutura try-catch pode ser usada para capturar exceções ao carregar recursos, como arquivos de áudio ou texturas, permitindo que o jogo continue executando de forma estável mesmo se ocorrerem erros.

O bloco `finally` é utilizado para executar um código que deve ocorrer independentemente de uma exceção ter sido lançada ou não.



Isso é útil para liberar recursos, fechar conexões ou realizar qualquer limpeza necessária após a execução do bloco `try`.

A instrução `throw` é usada para lançar uma exceção explicitamente.



Isso é útil para criar exceções personalizadas ou para sinalizar condições de erro específicas no jogo.

# Aplicando Estruturas de Controle de Fluxo em Unity

No desenvolvimento de jogos com Unity, as estruturas de controle de fluxo são usadas extensivamente para definir a lógica do jogo, comportamento dos personagens, gerenciar eventos e estados do jogo. Por exemplo, ao programar um jogo de plataforma, pode-se usar instruções condicionais para verificar se o jogador está no chão antes de permitir um salto, loops para mover plataformas continuamente e exceções para lidar com erros de carregamento de níveis.

Implementar essas estruturas de controle de fluxo com eficiência melhora a jogabilidade, torna o código mais organizado e facilita a manutenção e a expansão do jogo. Assim, dominar essas estruturas é um passo essencial para qualquer desenvolvedor de jogos que deseje criar experiências envolventes e profissionais.

# 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.