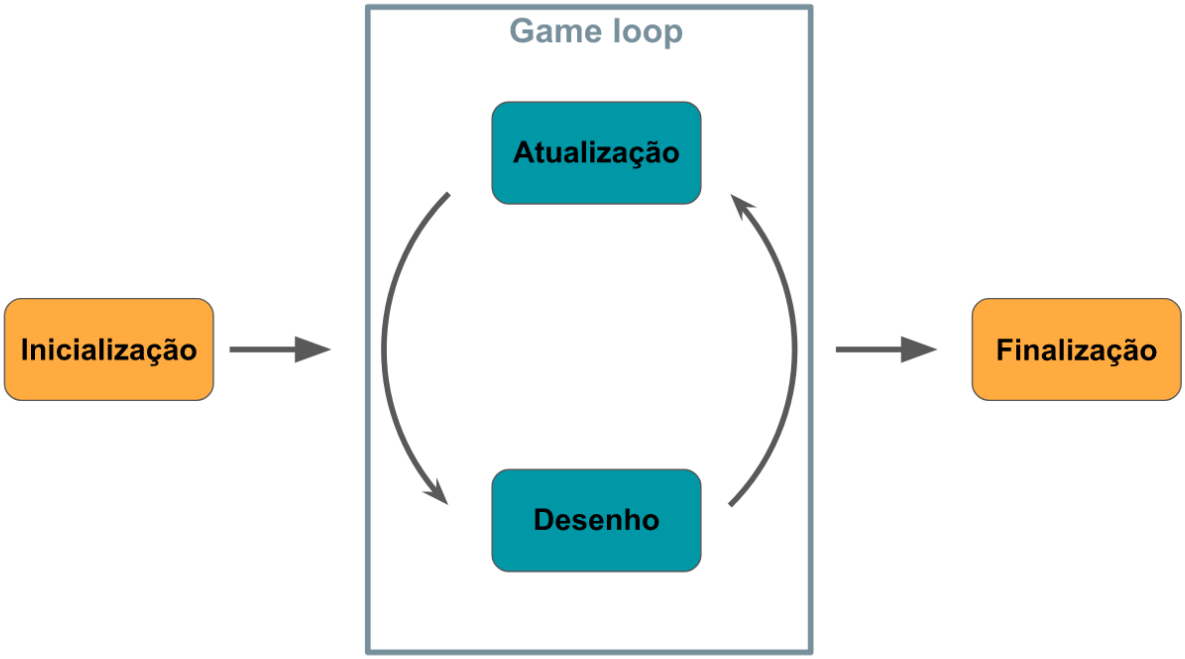
10

Update e Game Loop

Vimos que para a verificação dos *inputs* de *mouse*, precisamos utilizar o método `Update`. Esse método é chamado automaticamente pela *Unity* de tempos em tempos. Mas como isso funciona?

Todo jogo possui basicamente 3 fases a inicialização, o *game loop* e a finalização.



Durante a inicialização de do jogo, nós precisamos instanciar todos os objetos e preparar a cena onde vai acontecer o jogo. É nessa fase que vamos buscar todas as dependências que nossos objetos precisam, esse também é o momento ideal para realizarmos operações mais demoradas, isso porque o jogo ainda não começou, nós provavelmente estamos na tela de *loading* dele e o jogador não se importa de esperar um pouco nesse momento. Na *Unity* a inicialização é feita no método `Awake`.

A finalização do jogo é justamente o oposto da inicialização. Ela acontece quando fechamos o jogo e precisamos limpar a memória e liberar o processamento do computador. Essa é a etapa onde vamos destruir todos os objetos que criamos e para nossa sorte a *Unity* trata disso sozinha.

Entre a inicialização e a finalização, é onde acontece toda lógica do jogo, como os objetos se movem, atualização de interface gráfica, áudio, *inputs* dos jogadores, tudo isso acontece no que chamamos de *game loop*. Em geral essa etapa é dividida em duas partes, a atualização do estado do jogo e o desenho do jogo na tela do computador.

A atualização do estado do jogo é onde colocamos toda lógica dos nossos objetos. E é aqui que o método `Update` é executado. E como falamos esse método é executado de tempos em tempos, podemos pensar que estamos dentro de um *loop* infinito do nosso código, algo do tipo:

```
while(true)
{
```

```
        //lógica do jogo
    }
```

Só que esse tipo de `looping` infinito iria travar o computador, para isso não acontecer, dentro dos jogos, nós criamos um `looping` controlado onde, executamos o código dentro o laço de repetição, esperamos um tempo e executamos novamente.

Na *Unity*, cada componente tem seu próprio método `Update` e, durante a faze de atualização do estado do jogo, a *engine* chama esse método em cada um dos componentes. Essa separação nós ajuda a pensar de maneira mais modular no nosso código e, enquanto escrevemos um componente especifico, podemos nos focar no comportamento desejado, sem tem que nos preocupar tanto com a interferência de outros objetos.

A outra parte do *game loop*, o desenho do nosso jogo, é onde a Unity vai selecionar todos os objetos que estão dentro do quadro da câmera e exibir esses objetos para os jogadores. Existem muitos detalhes de como a seleção de objetos é feita, qual a ordem em que eles são desenhados e como a câmera funciona que a *engine* toma conta para nós, para que possamos cuidar apenas de fazer o jogo.

Se você tem interesse de saber em mais detalhes o que a Unity faz, por trás dos panos de uma olhada nessa página - (https://docs.unity3d.com/Manual/ExecutionOrder.html)https://docs.unity3d.com/Manual/ExecutionOrder.html (https://docs.unity3d.com/Manual/ExecutionOrder.html) - da documentação onde eles detalham qual a ordem de eventos.