Conhecendo a Programação orientada a eventos

Celio Ludwig Slomp
Instituto Federal Catarinense -
Campus Blumenau
Blumenau, Santa Catarina,
Brasil
slompcelio132@gmail.com

RESUMO

A programação orientada a eventos (do inglês event-driven programming) é um paradigma da programação que baseia toda a sua funcionalidade em eventos internos e externos de sua aplicação. O principal objetivo deste artigo é o de poder conhecer mais sobre este paradigma que pouco se conhece mas que tem tantas utilizações. Este explora o paradigma por sua definição, explicando o que é o loop de eventos, os possíveis tratadores de eventos e a fila de eventos. Além disso, este também mostra que o paradigma está presente em grande parte das interfaces gráficas de linguagens de programação como nos frameworks JavaFX e Swing do Java, na biblioteca Tkinter do Python, etc. além de também estar no desenvolvimento web com o Javascript, em algumas máquinas autônomas e na maioria dos jogos. Também é mostrado diversas vantagens e desvantagens de utilizá-lo em aplicações no geral. E por fim, se conclui que por conta de sua funcionalidade única e diferenciada, é de extrema vantagem o utilizar em certos casos

PALAVRAS-CHAVES

Paradigma orientado a eventos, Programação, Javascript, Paradigmas da Programação, Interface Gráfica, GUI

1 Introdução

A programação orientada a eventos é um paradigma muito utilizado nos dias atuais por conta de suas aplicações em sensores e páginas web. Entretanto, por mais utilizado que este seja, ele não é muito conhecido no meio, normalmente o programador ou o usuário o utiliza mas não faz ideia de que está utilizando ele.

O presente artigo procura explicar a programação orientada a eventos de uma forma mais simples, utilizando exemplos e situações. Este está separado em três partes. A primeira parte abordada será a definição, explicando os conceitos mais importantes do paradigma, como o loop e o tratador de eventos juntamente com a fila de eventos. Em seguida é passado alguns exemplos de aplicações práticas utilizadas no mercado, como em interfaces gráficas, jogos, automação industrial e desenvolvimento web. Por fim, é mostrado e explicado algumas vantagens e desvantagens do uso deste paradigma no mercado.

2 Desenvolvimento

2.1 Conceitos Básicos

A programação orientada a eventos (do inglês event-driven programming) é mais um dos diversos tipos de paradigmas da programação. Este tipo de paradigma funciona de uma forma completamente diferente dos demais paradigmas de programação. Ele tem como principal diferença dos demais a espera por eventos externos ou internos do programa para poder executar algum bloco de código designado à ele.

Normalmente, as linguagens de programação, tendo uma visão geral, funcionam em um formato linear, tendo o início do código muito bem sinalizado (muitas vezes possuem até método main), passando por todo o desenvolvimento dele e assim, finalmente chegando a um fim (podendo ser um return, close(), etc.). Entretanto, a programação orientada a eventos trabalha em um formato dinâmico. Isso significa que, dependendo da forma que o usuário utilizar o programa, ele pode ir para o meio do código, e então ir para o início, voltar para o meio, e por aí vai.

Mas por que isso? É simples! Este tipo de programação funciona com base em ocorrências externas ao computador, fazendo com que, por exemplo, uma função seja executada apenas quando a tecla "X" do teclado seja pressionada, ou então mude a cor da tela quando arrastar o mouse, entre diversas outras possibilidades. E para continuar, precisa-se saber sobre o loop de evento e do tratador de eventos

2.1.1 Loop e Tratador de Eventos

O loop de eventos (do inglês event loop) ou o listener, é um programa que fica solicitando eventos ao gerenciador de dispositivos. Desta forma, assim que ele recebe um evento que foi realizado pelo usuário, ele verifica qual foi o evento realizado e o envia ao tratador de eventos (do inglês event handler) para que este execute de forma correta o bloco de código desejado. Na figura 1 um diagrama está representado o funcionamento do tratador de eventos.

Assim que o bloco de código foi executado, o loop volta ao início e fica aguardando por algum outro evento realizado pelo usuário. O loop continua até que seja reconhecido algum evento para este parar. Estes eventos podem ser o evento de end of file (EOF), botão de fechar, etc. Caso o evento não seja reconhecido, o loop apenas o ignora e espera por outro.

Na figura 2 está um pseudocódigo baseado em Python que mostra de uma forma simples a forma que o loop de eventos funciona, sendo o “while True” um laço infinito, e cada comando condicional sendo o loop enviando a entrada para o seu devido tratador de evento.

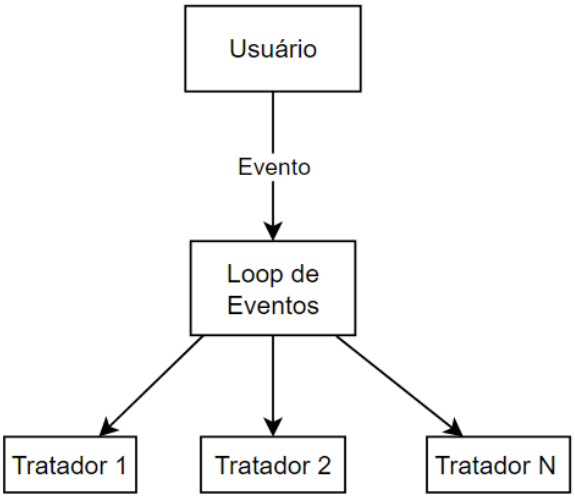

Figura 1: Diagrama do loop e tratador de eventos

2.1.2 Fila de Eventos

Um outro caso pode acontecer é do sistema possuir uma fila de eventos (do inglês event queue), nela as entradas ficam armazenadas em uma estrutura de fila. Esta fila vai liberando as entradas de forma sequencial para o loop de eventos enviar para cada entrada para o seu devido tratador. Este processo pode demorar caso tenham acontecido muitas entradas. Na figura 3 está um diagrama representando esta fila de eventos.

Um exemplo deste caso: Imagine um banco que tem dois clientes, o cliente X e o cliente Y. O cliente X faz uma transação bancária para o cliente Y no exato momento em que o cliente Y faz uma transação para o cliente X. Com a

```
while True:
    entrada = "Alguma opção"
    if entrada == "Final de arquivo - EOF":
        break
    elif entrada == "Botao de fechar pressionado":
        break
    elif entrada == "Botao de trocar cor pressionado"
        troca_cor() # chama o tratador de evento
        correto
    .
    .
    .
    else:
        print("Aguardando outra opção")
```

Figura 2: Pseudocódigo de um loop de eventos

Um exemplo deste caso: Imagine um banco que tem dois clientes, o cliente X e o cliente Y. O cliente X faz uma transação bancária para o cliente Y no exato momento em que o cliente Y faz uma transação para o cliente X. Com a fila de eventos, a transação de X para Y será executada e, durante o momento em que ela está sendo executada, a transação de Y para X fica em espera até que a transação de X para Y seja finalizada.

2.2 Aplicações Práticas

Este paradigma está muito presente dentro do mercado da programação por conta de sua enorme importância. O seu jeito de funcionar é único, o que faz com que diversos programas necessitem deste. A seguir foram listados alguns exemplos de aplicações deste paradigma dentro do mercado de trabalho.

2.2.1 Interfaces Gráficas de Usuário

As interfaces gráficas de usuário ou GUI (do inglês Graphic User Interface) é uma das principais áreas onde este paradigma é utilizado. A interface gráfica consiste na comunicação do usuário com a aplicação de uma forma não sequencial (do início ao fim), mas sim da forma que o usuário deseja. Um bom exemplo é um formulário web que possui os campos de nome, email e cpf, o usuário não pre

2.2.2 Desenvolvimento Web

Normalmente se utiliza linguagens de marcação para a produção de páginas web, tais linguagens como o HTML (abreviação de HyperText Markup Language). Como o HTML não é uma linguagem de programação, ele precisa utilizar uma linguagem de programação para poder realizar ações. Estas ações que a linguagem de programação realiza neste caso, podem variar de operações

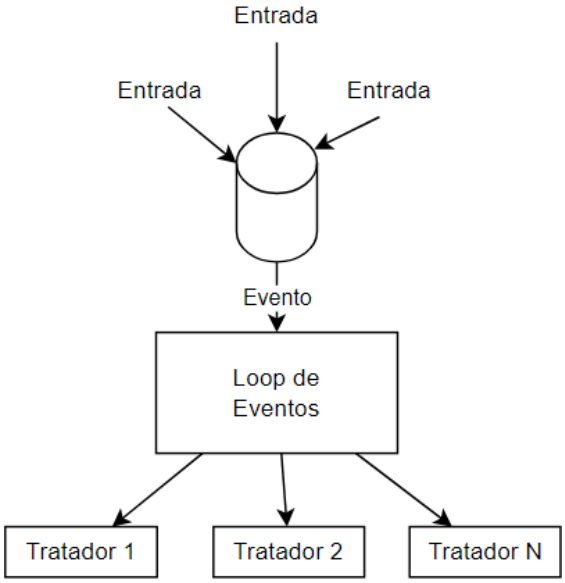

Figura 3: Diagrama de uma fila de eventos

matemáticas, verificações de dados, conexões com uma outra linguagem de programação, entre outras ações.

Na figura 4 há um exemplo utilizando a linguagem HTML para criar a interface gráfica da página. Nesta página há dois botões, o "Botao 1" e o "Botao 2". Quando esta página é aberta, ela carrega um bloco de código da linguagem de programação Javascript. Este código está entre as tags <script> e </script>. Quando a página é completamente carregada, são atribuídas às variáveis botao1 e botao2 seus devidos botões. Assim que são atribuídas, é adicionado a cada uma delas um listener, fazendo com que quando cada um dos botões seja pressionado, uma caixa de mensagem apareça na tela indicando qual botão foi pressionado.

2.2.3 Sistemas Embarcados

Este paradigma é muito utilizado no meio de sistemas de automação industrial por conta de sua enorme quantidade de botões e sensores que possuem diversas tarefas e estão presentes em determinadas máquinas. Estas tarefas podem ser um fator redundante de segurança, realizar alguma tarefa, etc. Um exemplo deste caso, visando a segurança do operário, pode ser o caso de serras de cortar madeira. Estas serras contam com um sistema onde caso algum tipo de carne as encostem, mesmo durante o corte de algum objeto, a serra para de funcionar. Este processo acontece por causa de um sensor onde caso ele detecte a condutividade da pele, envia um sinal que é captado por um outro equipamento que faz a serra parar.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta charset="UTF-8">
<meta name="viewport"
content="width=device-width,
initial-scale=1.0">
<title>Exemplo</title>
<script>
    document.addEventListener("DOMContentLoaded", function () {
        let botao1 = document.getElementById("botao1");
        let botao2 = document.getElementById("botao2");
        botao1.addEventListener("click", function () {
            alert("botao1 pressionado");
        });
        botao2.addEventListener("click", function () {
            alert("botao2 pressionado");
        });
    });
</script>
</head><body>
    <button type="button" id="botao1">Botao 1</button>
    <button type="button" id="botao2">Botao 2</button>
</body></html>
```

Figura 4: Código exemplo com HTML e Javascript

2.2.4 Jogos

Para produzir um jogo, é necessário um motor gráfico, também conhecido como engine. Este motor gráfico é como se fosse um esqueleto do jogo, ele possui as mecânicas, funcionalidades, GUI, etc. Cada jogo possui suas próprias teclas de comando, onde cada tecla tem uma tarefa específica. Por exemplo, utilizar as teclas W, A, S e D para se movimentar dentro dele, não é necessário apertar cada uma de forma sequencial, elas estão lá, mas você não precisa apertar todas elas para fazer algo, basta pressionar quando for necessário.

2.3 Vantagens e Desvantagens

O paradigma orientado a eventos possui uma enorme importância no mercado nos dias de hoje. Entratando, utilizá-lo possui diversas vantagens e desvantagens. A seguir há alguns exemplos de vantagens e desvantagens de sua utilização.

2.3.1 Vantagens

- Pode ser executado de forma assíncrona: isso faz com que o programa não precise funcionar de uma forma sequencial (início, meio e fim), ele é dividido em diversos blocos onde cada bloco tem um papel durante a execução do programa.
- O código fica mais limpo: por conta do código ser dividido em blocos, existe um tratador de eventos para cada evento que ocorre no programa, fazendo com que ele fique mais legível.
- Facilidade na manutenção: Levando em consideração um código mais limpo, a manutenção dele fica muito mais prático já que se sabe exatamente onde cada evento possível fica separado dentro do código.
- Facilidade em adição de funcionalidades: como o código fica inteiramente separado por blocos, a melhoria dele fica muito mais simples também, já que não se é preciso de um conhecimento sequencial lógico e nem de uma modificação muito grande no código atual.

2.3.2 Desvantagens

- Dificuldade em realizar o debug: por conta da execução com base em eventos, para serem realizados os testes, são necessários simular estes mesmos eventos, o que pode complicar com aplicações enormes.
- Pode ocorrer efeito cascata: às vezes pode acontecer de um evento chamar outros, e estes outros chamarem ainda mais outros, o que pode causar lentidão.
- Muitos eventos ao mesmo tempo podem causar uma sobrecarga: com muitos eventos acontecendo, o programa pode exigir muita demanda do hardware e, como no caso anterior, pode acontecer de causar lentidão.

3 Conclusão

Com isso, entendemos como funciona a programação orientada a eventos, entendendo com exemplos o funcionamento do loop e do tratador de eventos juntamente com a fila de eventos. Além também de serem apresentados exemplos práticos de uso deste, como sua imensa necessidade em desenvolvimento web utilizando HTML e Javascript, além das principais interfaces gráficas como JavaFX e Swing do Java e o Tkinter do Python, também vimos que este paradigma não está somente na computação e sim na área da automação industrial, sendo utilizada para segurança, etc. e principalmente na área de jogos, já que o pressionamento de botões não ocorre de forma sequencial.

Tendo conhecimento de todas as vantagens que este paradigma possui, como o código separado por blocos, o que permite uma execução assíncrona, uma enorme facilidade de manutenção por conta dos setores de cada bloco e também uma grande praticidade para a adição de novas funcionalidades, já que adicionando mais um bloco não irá afetar os outros. Também é preciso saber das desvantagens dele, como as dificuldades no debug, por conta de ter que simular certos eventos, além disso, pode ocorrer efeito cascata também, já que um bloco pode executar outro, que então executa outro, de uma forma muito longa.

Desta forma, concluímos que, por mais que a programação orientada a eventos não seja tão conhecida como os paradigmas da programação mais comuns, como a programação procedural, orientada a objetos, entre outros, ela é extremamente utilizada pela maioria das pessoas, não somente dentro da área da computação. Além disso, este é utilizado por muitos programadores que às vezes nem sabem que estão a utilizando e que também nem entendem como funciona.

REFERÊNCIAS

[1] [n.d.]. Event-Driven Programming. https://www.lenovo.com/in/en/glossary/event-driven-programming/

[2] OJAKIAN K. [n.d.]. Event-Driven Programming. https://fsw01.bcc.cuny.edu/kerry.ojakian/TeachingPages/Old_Courses_UP/CSI32_Fall2019/CourseMaterials/Ferg_event_driven_programming.pdf

[3] [n.d.]. Event (Computing) - Event Handler. https://en.wikipedia.org/wiki/Event_(computing)#Event_handler

[4] [n.d.]. Event Loop. https://en.wikipedia.org/wiki/Event_loop

[5] [n.d.]. Event-Driven Programming. https://en.wikipedia.org/wiki/Event-driven_programming

[6] [n.d.]. What, Why, How of Event-Driven Programming. https://quix.io/blog/what-why-how-of-event-driven-programming#:~:text=Event%2Ddriven%20programming%20(EDP),programs%2C%20sensor%20outputs%2C%20etc.

[7] [n.d.]. What is the Event-Driven Programming Paradigm? https://www.geeksforgeeks.org/what-is-the-event-driven-programming-paradigm/

[8] [n.d.]. Event-Driven Programming. https://www.studysmarter.co.uk/explanations/computer-science/computer-programming/event-driven-programming/