

**Versão do
Professor**

Coleção
**Python para
Jogos**



JOGO DA VELHA



Universidade de Brasília

Igor Silva de Oliveira Cardoso

Python para Jogos

JOGO DA VELHA

Igor Silva de Oliveira Cardoso



Brasília, 2025

Ficha catalográfica aqui

APRESENTAÇÃO

SUMÁRIO

1. RESPONDENDO ALGUMAS PERGUNTAS

O QUE É NECESSÁRIO?	X
Instalando Pygame	X
PORQUE JOGO DA VELHA?	X

2. MATRIZES E COORDENADAS

MATRIZ	X
COORDENADA	X
EXPLORANDO MATRIZES EM PYTHON	X
Resolução e Janela	X
Trabalhando com matrizes	X
EXERCÍCIOS	

3. O JOGO DA VELHA - UMA APLICAÇÃO DE MATRIZES

TABULEIRO E CLIQUE	X
VERIFICA O FIM PARTIDA	X
BOTÕES	X
FLUXO DE CONTROLE	X
PLACAR	X
FINALIZANDO	X

4. PRÓXIMOS PASSOS

RESPONDENDO ALGUMAS PERGUNTAS

Neste capítulo, apresentaremos respostas para algumas perguntas sobre este manual, garantindo que sua jornada de aprendizado e prática seja a melhor possível. O capítulo está estruturado no formato de perguntas e respostas. No entanto, caso surjam dúvidas, sinta-se à vontade para consultar o autor por meio dos contatos disponíveis no final do livro.

O QUE É NECESSÁRIO?

O conhecimento sobre algoritmos e a linguagem Python é fundamental para aproveitar ao máximo este manual. No entanto, caso ainda não se sinta seguro em relação ao seu conhecimento, serão recomendadas matérias suplementares para garantir seu total entendimento.

Utilizaremos o Pygame¹ para a exibição gráfica e elementos visuais. No entanto, não se preocupe caso não tenha familiaridade com a biblioteca, pois ela será integrada aos módulos fornecidos por meio do repositório oficial.

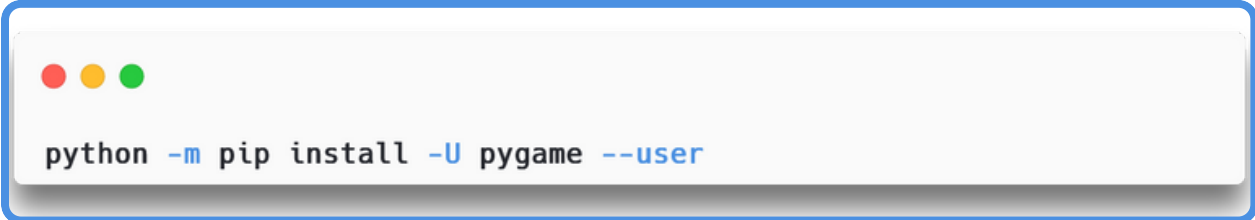
Relembre o que é uma biblioteca

Uma biblioteca é um conjunto de módulos que fornecem funcionalidades úteis, mas sem impor um fluxo de controle específico

¹ Acesse: <https://www.pygame.org/docs/ref/mouse.html>

Instalando Pygame

A maneira mais recomendada e simples de instalar o Pygame é utilizando o pip¹, o gerenciador de pacotes do Python. Para isso, abra o terminal ou prompt de comando do seu sistema operacional e execute o seguinte comando:

A terminal window with a blue border and three colored dots (red, yellow, green) in the top-left corner. The command 'python -m pip install -U pygame --user' is displayed in a monospaced font.

```
python -m pip install -U pygame --user
```

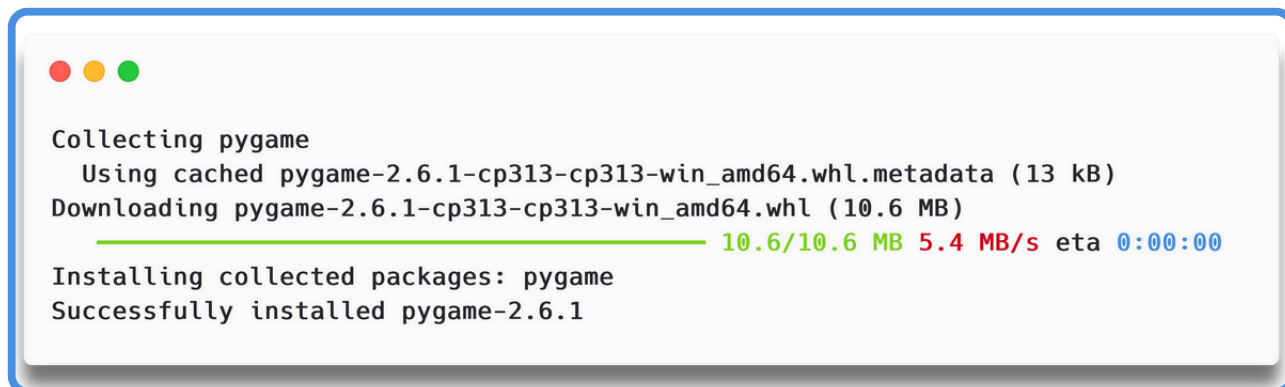
A maneira mais recomendada e simples de instalar o Pygame é utilizando o pip, o gerenciador de pacotes do Python. Para isso, abra o terminal ou prompt de comando do seu sistema operacional e execute o seguinte comando:

Relembre o que é um gerenciador de pacotes

É uma ferramenta que automatiza o processo de instalação, atualização, configuração e remoção de bibliotecas

¹ Acesse: <https://docs.python.org/pt-br/3.7/installing/index.html#basic-usage>

Após, a execução do comando o seguinte retorno é esperado:

A terminal window with a blue border and three colored window control buttons (red, yellow, green) in the top-left corner. The text inside the terminal shows the process of installing the pygame library. It starts with 'Collecting pygame', followed by 'Using cached pygame-2.6.1-cp313-cp313-win_amd64.whl.metadata (13 kB)', then 'Downloading pygame-2.6.1-cp313-cp313-win_amd64.whl (10.6 MB)'. A green progress bar is shown, followed by the text '10.6/10.6 MB 5.4 MB/s eta 0:00:00'. The final lines are 'Installing collected packages: pygame' and 'Successfully installed pygame-2.6.1'.

```
Collecting pygame
  Using cached pygame-2.6.1-cp313-cp313-win_amd64.whl.metadata (13 kB)
Downloading pygame-2.6.1-cp313-cp313-win_amd64.whl (10.6 MB)
  10.6/10.6 MB 5.4 MB/s eta 0:00:00
Installing collected packages: pygame
Successfully installed pygame-2.6.1
```

Agora estamos com o ambiente pronto!

PORQUE JOGO DA VELHA?

Você pode estar se perguntando o motivo da escolha do jogo da velha ~~como objeto de estudo para este manual~~. A resposta está na simplicidade e no poder didático desse jogo. Através dele, abordaremos conceitos fundamentais como matrizes e coordenadas, que são pilares da matemática e se conectam diretamente com a computação.

O jogo da velha nos permitirá explorar estruturas de dados, que serão utilizadas para representar o tabuleiro. Além disso, nos possibilitará exercitar temas como controle de fluxo e condições lógicas.

MATRIZES E COORDENADAS

Neste capítulo, iremos explorar alguns aspectos sobre matrizes e coordenadas, e em seguida, aplicações em Python, inclusive utilizando interface gráficas.

Ao final deste capítulo, teremos alguns exercícios práticos com a finalidade de fixar o que foi abordado e preparar para o próximo capítulo.

MATRIZES

Em Matemática, dizemos que uma matriz **$m \times n$** é uma tabela **$m . n$** números dispostos em m linhas (fila horizontais) e n colunas (filas verticais). Observe abaixo:

$$\begin{pmatrix} 8 & 1 \\ 4 & 2 \\ 9 & 5 \end{pmatrix}$$

3x2

Diagram illustrating a 3x2 matrix. The matrix is shown with its dimensions **3x2** indicated below it. An arrow labeled "m linhas" points to the matrix, indicating the number of rows. An arrow labeled "n colunas" points to the matrix, indicating the number of columns.

No entanto, para o propósito deste manual, focaremos na matriz quadrada, que é um tipo especial de matriz, onde número de linhas é igual ao número de colunas.

COORDENADA

Consideremos uma matriz A do tipo $m \times n$. Um elemento qualquer dessa matriz será representado pelo símbolo a_{ij} , no qual o índice i refere-se à linha em que se encontra tal elemento e j refere-se à coluna em que se encontra o elemento. Sendo assim, a coordenada de um elemento em uma matriz é representada pelo par (i,j) . Por exemplo, se tivermos a matriz:

$$\mathbf{A} = \begin{pmatrix} -1 & 0 & 8 \\ -2 & 5 & 1 \\ 3 & 5 & 2 \end{pmatrix}$$

- O elemento que está na linha 1, coluna 1 é $a_{11}=-1$, ou seja, sua coordenada é $(1,1)$.
- O elemento que está na linha 2, coluna 2, é $a_{22}=5$, ou seja, sua coordenada é $(2,2)$.
- O elemento que está na linha 3, coluna 3, é $a_{33}=2$, ou seja, sua coordenada é $(3,3)$.

No exemplo acima, é possível observar que o índice da linha é igual ao índice da coluna. Isso constitui a diagonal principal, que é uma das características da matriz quadrada.

EXPLORANDO MATRIZES EM PYTHON

É hora de colocar em prática! Nesta etapa, é fundamental que todas as dúvidas sobre a estrutura de uma matriz e a localização de seus elementos estejam esclarecidas, além de garantir que todos tenham acesso ao ambiente. No entanto, caso ainda haja dúvidas, recomenda-se a consulta a um material mais detalhado, como uma bibliografia de matemática de Gelson Iezzi.

Podemos representar uma matriz usando listas (do tipo **list**) aninhadas (listas dentro de listas). A seguir podemos ver alguns exemplos.

Relembre o que é uma lista

Uma lista em Python é uma estrutura de dados ordenada e mutável que pode armazenar diferentes tipos de elementos

```
1 matriz_A = [  
2     [1, 2, 3],  
3     [4, 5, 6],  
4     [7, 8, 9]  
5 ] # 3x3  
6  
7 matriz_B = [  
8     [1, 2, 3, 5],  
9     [4, 5, 6, 6],  
10    [7, 8, 9, 7],  
11    [4, 2, 1, 9]  
12 ] # 4x4  
13  
14 matriz_C = [  
15     [1, 0],  
16     [0, 1]  
17 ] # 2x2
```

Resolução e janela

Antes de partirmos para a visualização gráfica das matrizes, é importante explorarmos alguns aspectos relacionados à janela e resolução. Para isso, é importante garantir que todos tenham acesso ao arquivo **Janela.py**, que está disponível para download no repositório.

Uma janela é uma interface gráfica que iremos utilizar para exibir a matriz e, no próximo capítulo, as interações para o jogo. A janela possui um tamanho, denominado resolução, que afeta a clareza e a legibilidade dos conteúdos contidos na janela.

Importante

Para essa etapa, é importante verificar se o Pygame está instalado e se o arquivo Janela.py está no mesmo diretório.

¹ Arquivo Janela.py disponível em: <https://github.com/igorcardoso/jogando-com-python/blob/master/games/jogo-da-velha/Janela.py>

Vamos ao código! Crie um arquivo Python com o nome de sua preferência. Nesse arquivo, faça a importação e crie uma instância da **Janela**:

```
1 from Janela import Janela
2
3 janela = Janela(
4     resolucao_da_tela=(900, 700),
5     cor_da_janela='branco',
6     titulo_da_janela='Título da Janela'
7 )
```

Para o correto funcionamento da janela, precisaremos passar por parâmetro a largura e a altura, que estão definidas em pixels, além da cor de fundo e o nome.

O que é pixel?

Pixel é a menor unidade de informação de cor que compõe uma imagem digital

E agora é só executar? Ainda não, é precisaremos de mais duas etapas para podermos ver o funcionamento da janela.

Vamos precisar de um laço de repetição do tipo **while**, com a condição **True**, e também invocar a ação **atualizar** para garantir que a janela seja atualizada continuamente.

Porque True no while?
~~While igual a True é importante~~

```
1 from Janela import Janela
2
3 janela = Janela(
4     resolucao_da_tela=(900, 700),
5     cor_da_janela='branco',
6     titulo_da_janela='Título da Janela'
7 )
8
9 while True:
10     # Atualiza a janela
11     janela.atualizar()
```

Agora vamos executar o programa com comando:

```
> python minha_janela.py
```

É esperado que a seguinte janela abra:

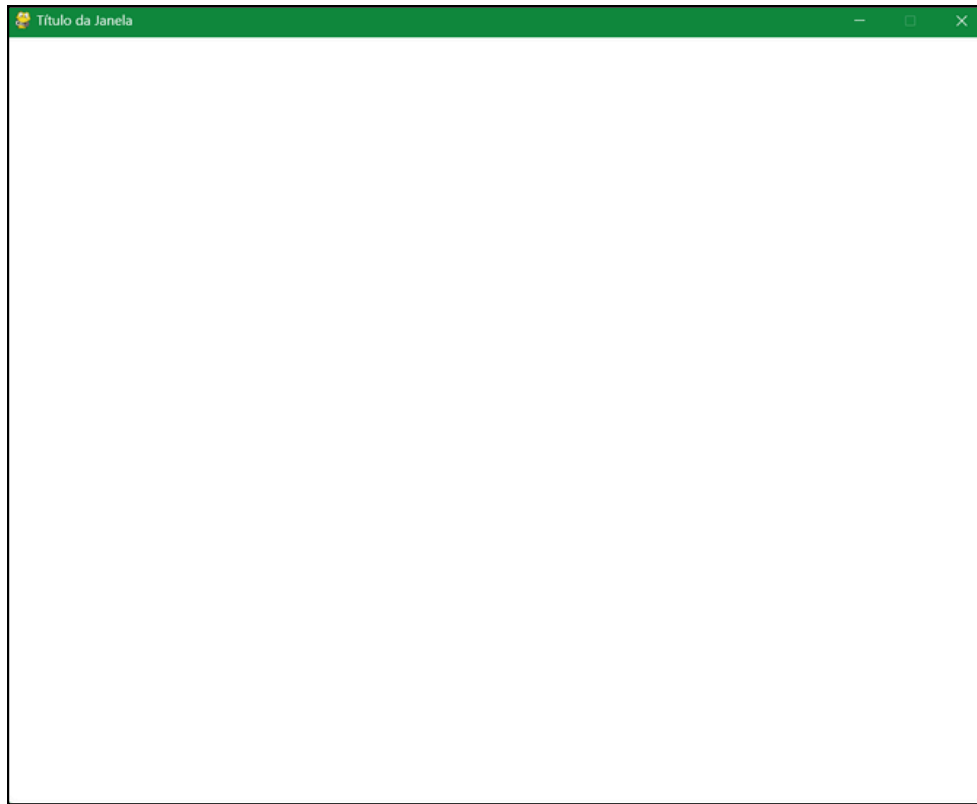


Figura X - Janela

Acabamos de criar uma janela com poucas linhas de código. O próximo passo agora é exibir nossas matrizes graficamente nessa janela.

Trabalhando com matrizes

Agora que já temos uma janela criada, e sabemos como criar uma matriz, veremos uma representação gráfica, onde será possível observar a ordem e as coordenadas de cada posição da matriz.

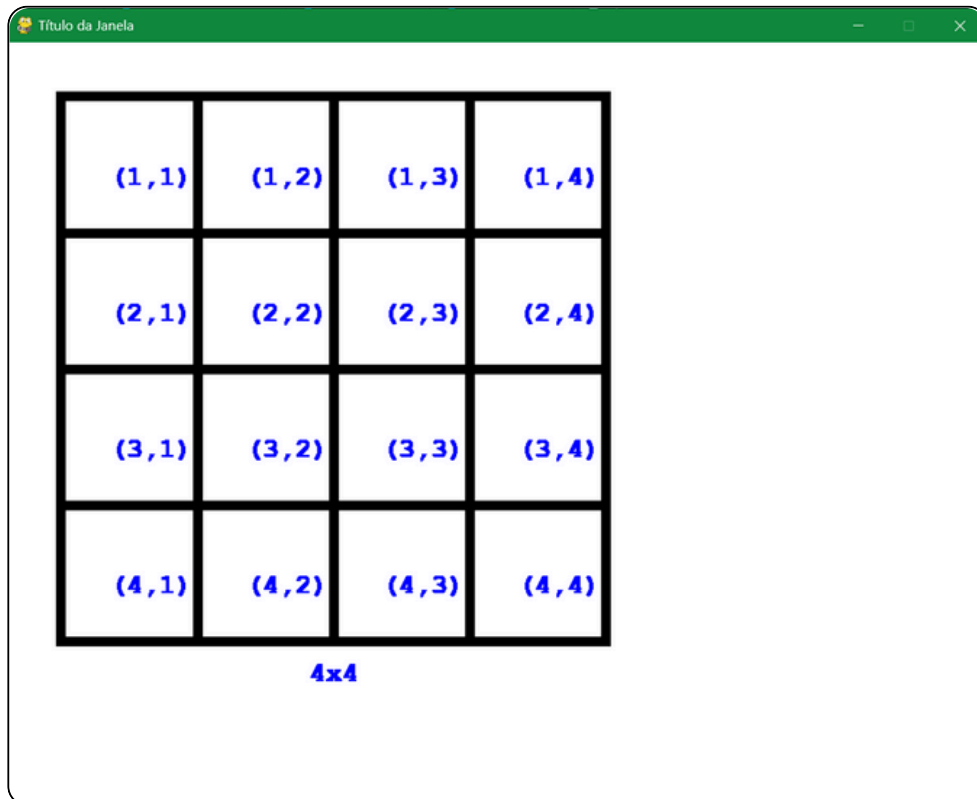
O primeiro passo é importar e instanciar `JogoDaVelha`, que está disponível para download no repositório. Por meio dela conseguiremos imprimir matrizes graficamente.

```
1 from JogoDaVelha import JogoDaVelha
2 from Janela import Janela
3
4 # Inicialização de uma janela
5 janela = Janela(
6     resolucao_da_tela=(900, 700),
7     cor_da_janela='branco',
8     titulo_da_janela='Título da Janela'
9 )
10
11 # Inicialização do jogo da velha
12 jogo_da_velha = JogoDaVelha()
13
14 while True:
15     # Atualiza a janela
16     janela.atualizar()
```

Após salvar e executar, nada de novo acontecerá, isso porque precisamos criar uma matriz, com as posições com uma string vazia. Sendo assim, iremos realizar a criação de uma matriz $n \times n$ e também passaremos, por parâmetro, a matriz e a instância da Janela criada para JogoDaVelha.

```
1 from JogoDaVelha import JogoDaVelha
2 from Janela import Janela
3
4 # Inicialização de uma janela
5 janela = Janela(
6     resolucao_da_tela=(900, 700),
7     cor_da_janela='branco',
8     titulo_da_janela='Título da Janela'
9 )
10
11 # Inicialização do jogo da velha
12 jogo_da_velha = JogoDaVelha()
13
14 while True:
15
16     # Matriz n x n
17     matriz = [
18         ['', '', '', ''],
19         ['', '', '', ''],
20         ['', '', '', ''],
21         ['', '', '', '']
22     ]
23
24     # Criar um tabuleiro n x n
25     jogo_da_velha.tabuleiro(janela, matriz)
26
27     # Atualiza a janela
28     janela.atualizar()
29
```

Após executar o programa com o comando **python**, o resultado esperado será o seguinte:



(1,1)	(1,2)	(1,3)	(1,4)
(2,1)	(2,2)	(2,3)	(2,4)
(3,1)	(3,2)	(3,3)	(3,4)
(4,1)	(4,2)	(4,3)	(4,4)

4x4

Figura X - Matriz 4x4 em uma janela

Essa é uma excelente oportunidade para explorar e esclarecer dúvidas sobre como localizar um elemento em uma matriz usando os recursos gráficos.

EXERCÍCIOS

É hora de colocar a mão no código por meio de alguns exercícios. Caso não consiga resolvê-los, recomenda-se a consulta a materiais suplementares, assim como às soluções de cada exercício disponíveis no repositório.

1. Calcule a soma de todos elementos da seguinte matriz $n \times n$:

```
matriz_q1 = [  
    [1, 2, 3],  
    [4, 5, 6],  
    [7, 8, 9]  
]
```

2. Calcule a soma de todos elementos iguais na seguinte matriz $n \times n$:

```
matriz_q2 = [  
    [1, 2, 1, 0],  
    [4, 1, 6, 9],  
    [1, 8, 9, 1],  
    [1, 8, 9, 2]  
]
```

3. Realize a soma de todos elementos em que os valores do índices são iguais:

```
matriz_q3 = [  
    [8, 2, 8],  
    [4, 1, 6],  
    [1, 8, 1]  
]
```

4. Realize a soma de dos elementos que estão na posições (1,4), (2,6), (5,2), (7, 7) dinamicamente:

```
matriz_q4 = [  
    [8, 2, 8, 4, 3, 7, 2, 5],  
    [4, 1, 6, 1, 8, 1, 1, 8],  
    [1, 8, 1, 8, 4, 3, 7, 2],  
    [4, 1, 6, 1, 8, 1, 1, 9],  
    [9, 1, 6, 4, 9, 3, 1, 1],  
    [8, 6, 8, 8, 8, 1, 9, 5],  
    [4, 8, 9, 1, 6, 4, 5, 5],  
    [3, 2, 0, 1, 3, 7, 3, 2],  
]
```


O JOGO DA VELHA UMA APLICAÇÃO DE MATRIZES

Neste capítulo, aplicaremos todos os conceitos desenvolvidos nas etapas anteriores para criar uma versão interativa do Jogo da Velha. Implementaremos um sistema de placar para acompanhar as partidas e adicionaremos botões com ações para controlar o fluxo do jogo.

TABULEIRO E CLIQUE

Vamos começar! Para realizar o carregamento e captação do clique no tabuleiro iremos usar os seguintes comandos:

```
1 janela.informacoes_do_mouse()  
2 jogo_da_velha.evento_de_clique(janela)
```

Neste momento, não iremos explorar a fundo a detecção de cliques. Para se aprofundar nos detalhes sobre clique e eventos, a documentação do Pygame¹ é um guia completo e sempre atualizado.

Agora é a hora de criarmos o tabuleiro do jogo da velha, mas antes disso vamos nos lembrar qual é o formato do tabuleiro do jogo da velha.

Acesse: <https://www.pygame.org/docs/ref/mouse.html>

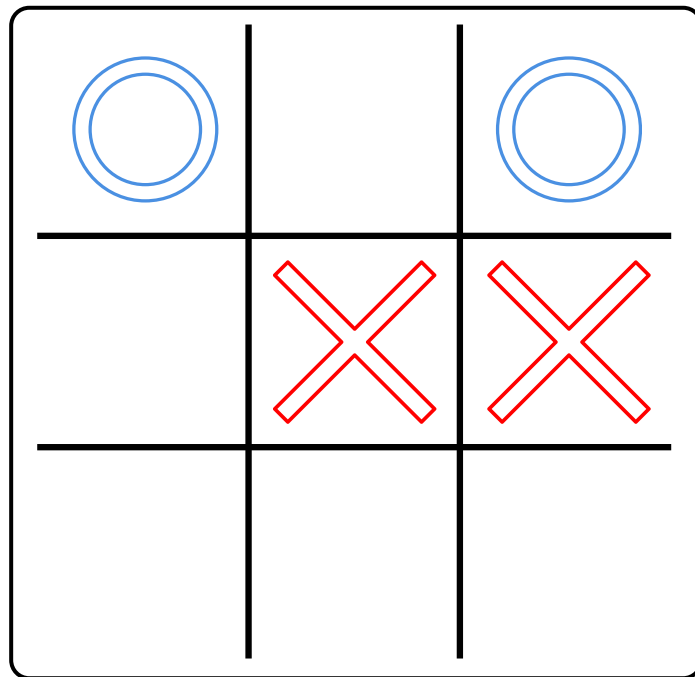


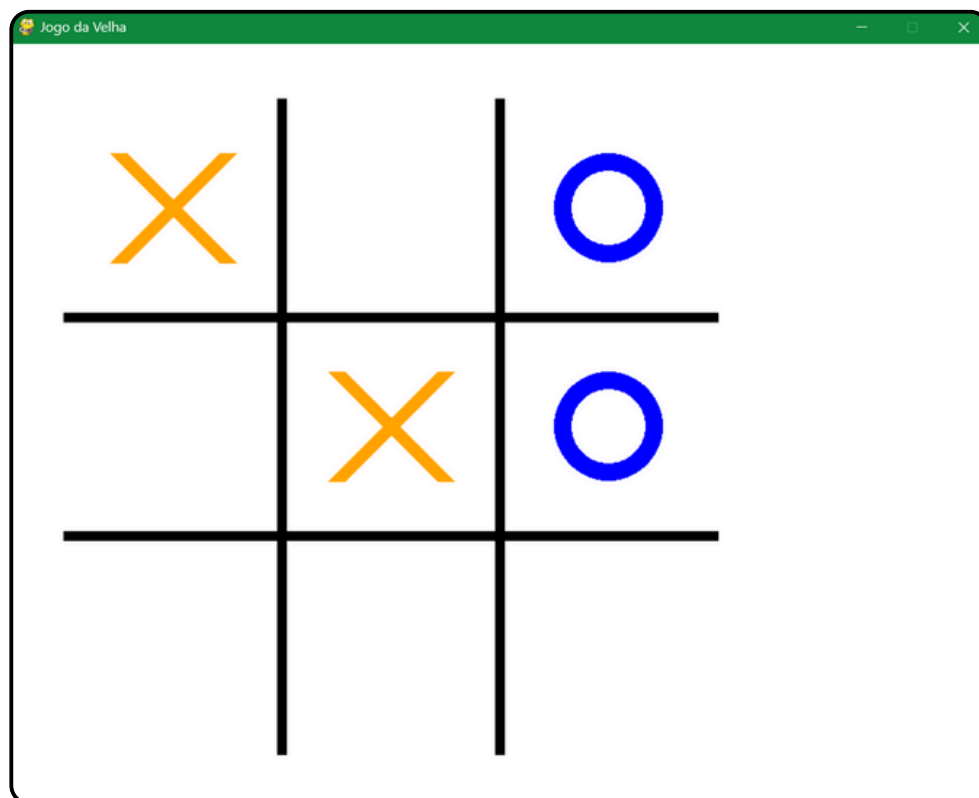
Figura x - Jogo da Velha

Sendo assim, iremos precisar uma matriz de tamanho 3x3, como é possível observar na imagem acima. Para isso, utilizando os conhecimentos obtidos no capítulo 2, e o comando para desenhar o 'x' e 'o' no tabuleiro, sendo assim, o código produzi será o seguinte:



```
1 from JogoDaVelha import JogoDaVelha
2 from Janela import Janela
3
4 # Inicialização de uma janela
5 janela = Janela(
6     resolucao_da_tela=(900, 700),
7     cor_da_janela='branco',
8     titulo_da_janela='Jogo da Velha'
9 )
10
11 # Inicialização do jogo da velha
12 jogo_da_velha = JogoDaVelha()
13
14 while True:
15     # Informações do mouse
16     janela.informacoes_do_mouse()
17
18     # Evento de clique no tabuleiro
19     jogo_da_velha.evento_de_clique(janela)
20
21     # Desenha o tabuleiro
22     matriz = [
23         ['', '', ''],
24         ['', '', ''],
25         ['', '', '']
26     ]
27
28     # Desenha o tabuleiro
29     jogo_da_velha.tabuleiro(janela, matriz)
30
31     # Desenha os X e O
32     jogo_da_velha.desenha_x_e_o(janela, 'laranja', 'azul')
33
34     # Atualiza a janela
35     janela.atualizar()
```

Executando o código acima, o resultado obtido será o seguinte:



O jogo está quase completo! Já conseguimos jogá-lo, e a validação para identificar um vencedor está funcionando. No entanto, ainda precisamos verificar se a partida realmente chegou ao fim, o que será nosso próximo tópico.

VERIFICA O FIM PARTIDA

Uma etapa importante no jogo da velha é verificar se o jogo chegou ao fim, nessa sessão iremos abordar esse tópico, onde criaremos uma função que fará a verificação se o jogo chegou ao fim.

Vamos criar a função `verifica_fim_de_jogo`, receberá uma matriz por parâmetro e realizará a verificação de fim de jogo, o resultado produzido por essa função é importante para o fluxo de controle do jogo.

Vamos o que é uma função

~~Pixel é a menor unidade de informação de cor que compõe uma imagem digital~~

A lógica de verificação será implementada iterando a matriz e verificando se cada posição está preenchida. Ao final, será verificada a quantidade de posições ocupadas. Se esse número for maior ou igual a 9, a função retornará `True`; caso contrário,

retornará False. Para essa verificação, utilizaremos um operador ternário.

Relembre o que é o operador ternário

Pixel é a menor unidade de informação de cor que compõe uma imagem digital

```
1 def verifica_fim_de_jogo(matriz_atual):
2     campos_marcados = 0
3     for y in range(3):
4         for x in range(3):
5             if matriz_atual[y][x] != '':
6                 campos_marcados += 1
7
8     return True if campos_marcados == 9 else False
```

Após a implementação da função acima, é preciso recuperar o estado atual da matriz do tabuleiro e, em seguida, chamar a função passando essa matriz como argumento.

```
1 # Matriz atual
2 matriz_atual = jogo_da_velha.matriz_do_tabuleiro
3
4 # Verifica se o jogo acabou
5 jogo_da_velha.fim_do_jogo = verifica_fim_de_jogo(matriz_atual)
```

BOTÕES

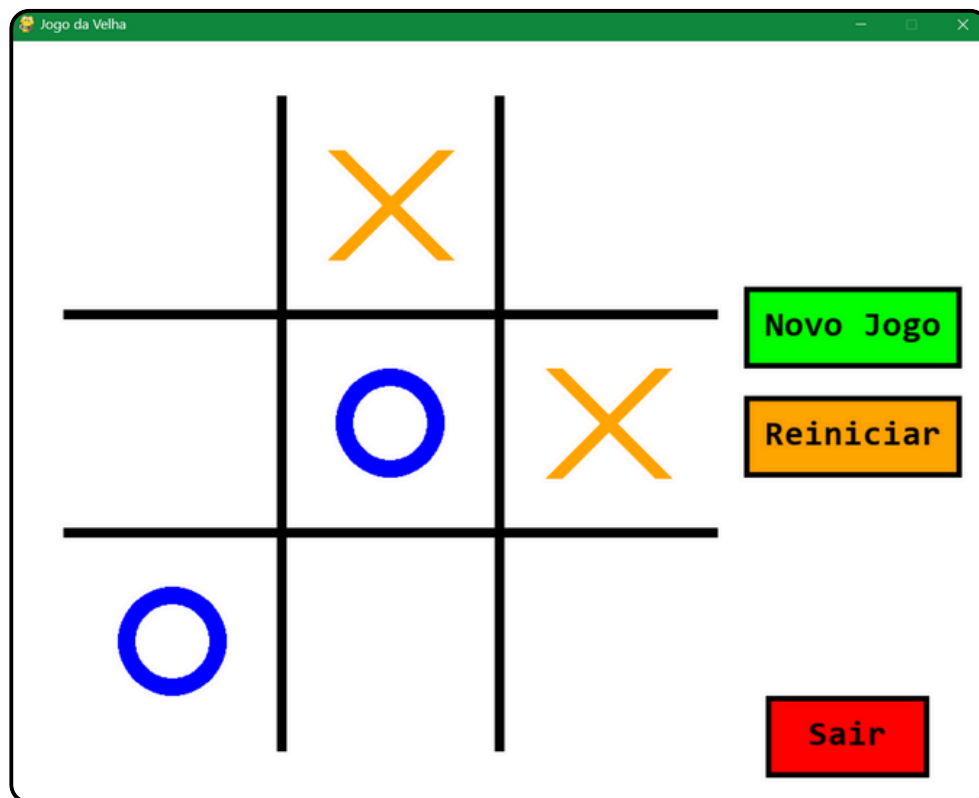
Vamos incluir os botões necessários para controlar o fluxo do jogo. Para isso, serão criados três botões: um para iniciar uma nova partida, um para reiniciar o jogo e outro para sair.

É importante ressaltar que o botão faz parte do módulo Janela, onde é necessário definir algumas configurações. A primeira é a posição do botão na janela, seguida pelo seu tamanho, em pixels, além da cor e da descrição do botão.



```
1 botao_de_novo_jogo = janela.botao((675, 225), (200, 75), 'verde', 'Novo Jogo')  
2 botao_de_reiniciar = janela.botao((675, 325), (200, 75), 'laranja', 'Reiniciar')  
3 botao_de_sair = janela.botao((695, 600), (150, 75), 'vermelho', 'Sair')
```

Após a inclusão do código acima em tudo o que produzimos, ao executar o programa, o resultado será o seguinte:



Nosso jogo está quase pronto! Já temos o Jogo da Velha funcionando e os botões na tela. O próximo passo é implementar as ações dos botões, e é isso que veremos na próxima seção.

FLUXO DE CONTROLE

Com os botões já na tela, vamos implementar as ações, o que nos permitirá controlar o fluxo do jogo.

Os botões serão identificados pela descrição que foi atribuída a cada um deles. Para isso, utilizaremos condicionais e adicionaremos a ação correspondente à condição. Para o botão 'Sair', executaremos a função `quit` do Python; para a reinicialização, será chamado o método `reiniciar_jogo` da classe `JogoDaVelha`; e, por fim, para a opção de 'Novo Jogo', será executado o método `novo_jogo` da classe `JogoDaVelha`.

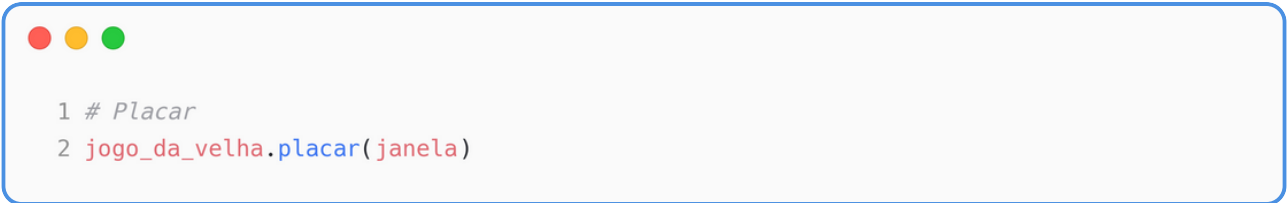


```
1 if botao_de_sair == 'Sair':  
2     quit()  
3 elif botao_de_reiniciar == 'Reiniciar':  
4     jogo_da_velha.reiniciar_jogo()  
5 elif botao_de_novo_jogo == 'Novo Jogo':  
6     jogo_da_velha.novo_jogo()
```

Executando o programa novamente, nenhuma mudança visual será observada. No entanto, os botões já estarão funcionando.

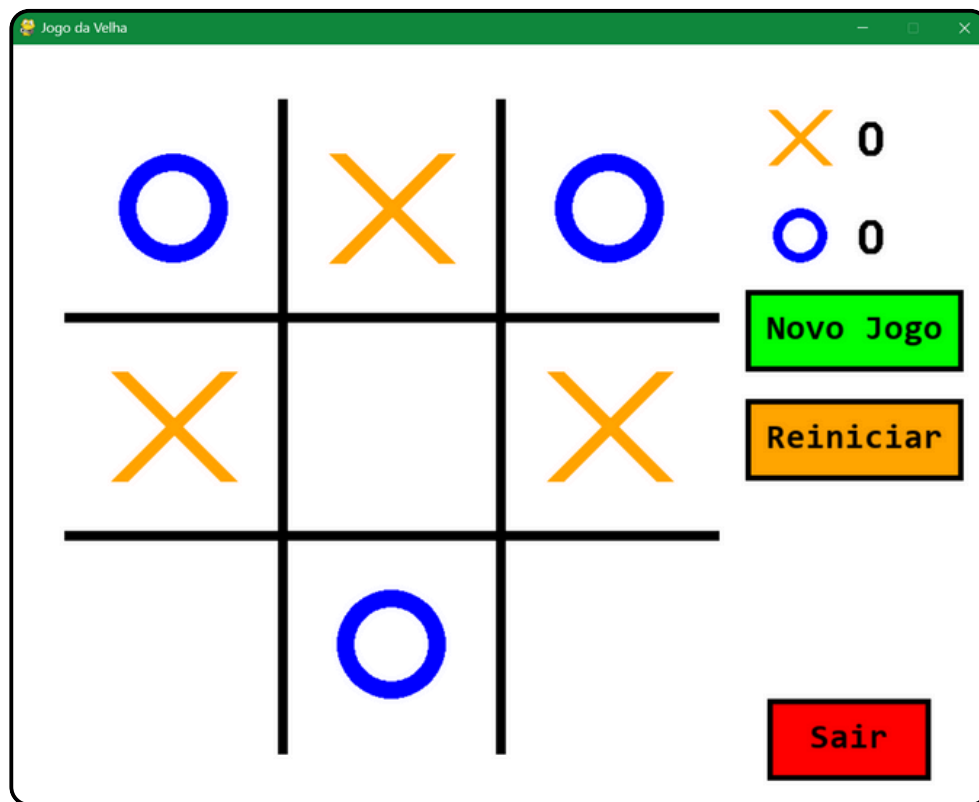
PLACAR

Nesta seção, vamos incluir o placar da partida na janela. Para isso, precisaremos invocar o método `placar` da classe `JogoDaVelha`, passando a instância da janela.



```
1 # Placar
2 jogo_da_velha.placar(janela)
```

Executando o programa o seguinte resultado será observado:



FINALIZANDO

O jogo está finalizado! Tudo está pronto. Implementamos o tabuleiro com base em nossos conhecimentos de matrizes, criamos os botões e ações para o controle do fluxo e, além disso, inserimos o placar e o contador de vitórias.

