

**CENTRO UNIVERSITÁRIO MAURÍCIO DE NASSAU - UNINASSAU**  
**BACHAREL EM CIÊNCIA DA COMPUTAÇÃO**

Ingyrd Vitória de Araújo Barbosa - 01642893  
Luiz Otávio de Souza Azevedo - 01581254  
Maria Eduarda Rodrigues de Lima - 01591900  
Mateus Guerra Feitosa - 01625297  
Paulo Sérgio Barros de Souza - 01584932

**DESENVOLVIMENTO DE JOGOS DIGITAIS – GRUPO PHIBUS**  
**RELATÓRIO TÉCNICO SOBRE A CRIAÇÃO DE JOGOS COM JAVA E PROCESSING**

**RECIFE**  
**2025**

Ingryd Vitória de Araújo Barbosa - 01642893

Luiz Otávio de Souza Azevedo - 01581254

Maria Eduarda Rodrigues de Lima - 01591900

Mateus Guerra Feitosa - 01625297

Paulo Sérgio Barros de Souza - 01584932

## **DESENVOLVIMENTO DE JOGOS DIGITAIS – GRUPO PHIBUS**

### **RELATÓRIO TÉCNICO SOBRE A CRIAÇÃO DE JOGOS COM JAVA E PROCESSING**

Este trabalho tem como objetivo o desenvolvimento de jogos utilizando a linguagem Java e a ferramenta Processing, explorando conceitos fundamentais de programação e lógica aplicada ao desenvolvimento de games. Além disso, busca-se aprimorar o conhecimento dos integrantes no uso dessas tecnologias e documentar o processo por meio de um relatório detalhado, consolidando o aprendizado adquirido durante a disciplina de Desenvolvimento de Games.

Orientador: Prof. Leopoldo Rodrigues

**RECIFE**

**2025**

## RESUMO

Este trabalho, desenvolvido na disciplina de Desenvolvimento de Games, teve como objetivo a criação de jogos utilizando a linguagem Java e a ferramenta Processing. A experiência permitiu o aprofundamento em conceitos de programação, além do uso de diferentes ambientes de desenvolvimento, como Eclipse e Processing. Durante o processo, o grupo enfrentou desafios, especialmente na adaptação à sintaxe do Processing, mas conseguiu superá-los e adquirir novos conhecimentos. O projeto serviu como uma introdução prática ao desenvolvimento de jogos, proporcionando uma experiência valiosa para o portfólio acadêmico e profissional dos integrantes.

**Palavras-chave:** desenvolvimento de games, Java, Processing, programação, portfólio.

## LISTAS DE FIGURAS

Figura 1 - Tela Inicial no Jogo do Marciano.....	9
Figura 2 - Entrada do nome do jogador no Jogo do Marciano.....	9
Figura 3 - Introdução da história no Jogo do Marciano.....	10
Figura 4 - Contando a história no Jogo do Marciano.....	10
Figura 5 - Primeira tentativa do jogador no Jogo do Marciano.....	10
Figura 6 - Exibição de dicas de que o marciano está em uma árvore menor no Jogo do Marciano.....	11
Figura 7 - Exibição de dicas de que o marciano está em uma árvore maior no Jogo do Marciano.....	11
Figura 8 - Mensagem de mudança de posição do Marciano no Jogo do Marciano.....	11
Figura 9 - Tela de vitória do jogador no Jogo do Marciano.....	12
Figura 10 - Tabela de recordes no Jogo do Marciano.....	12
Figura 11 - Pergunta sobre nova partida no Jogo do Marciano.....	13
Figura 12 – Tela inicial no Jogo da Força.....	25
Figura 13 – Exibição da palavra oculta no Jogo da Força.....	26
Figura 14 – Tentativa correta do jogador no Jogo da Força.....	27
Figura 15 – Tentativa incorreta do jogador no Jogo da Força.....	28
Figura 16 – Tela de vitória no Jogo da Força.....	29
Figura 17 – Tela de derrota no Jogo da Força.....	30
Figura 18 – Opção de reinício no Jogo da Força.....	31
Figura 19 – Tela inicial do Jogo da Velha.....	48
Figura 20 – Tabuleiro vazio no início da partida no Jogo da Velha.....	50
Figura 21 – Jogada em andamento no Jogo da Velha.....	52
Figura 22 – Vitória de um jogador no Jogo da Velha.....	54
Figura 23 – Tela de empate no Jogo da Velha.....	56

Figura 24 – Opção de reinício no Jogo da Velha.....	57
---	----

## SUMÁRIO

<b>INTRODUÇÃO.....</b>	<b>8</b>
<b>2. JOGOS DESENVOLVIDOS.....</b>	<b>9</b>
2.1 JOGO DO MARCIANO.....	9
2.1.1. Descrição.....	9
2.1.2. Objetivo.....	9
2.1.3. Funcionalidades.....	9
2.1.4. Prints das Telas.....	9
2.1.4.1. Tela Inicial.....	9
2.1.4.2. Entrada do Nome do Jogador.....	10
2.1.4.3. História do Jogo.....	10
2.1.4.4. Início da Partida.....	11
2.1.4.5. Dicas do Jogo.....	11
2.1.4.6. Mudança de Posição do Marciano.....	12
2.1.4.7. Vitória do Jogador.....	13
2.1.4.8. Registro de Recordes.....	13
2.1.4.9. Pergunta sobre Nova Partida.....	14
2.1.5. Código Fonte.....	14
2.2 JOGO DA FORÇA.....	24
2.2.1. Descrição.....	24
2.2.2. Objetivo.....	25
2.2.3. Funcionalidades.....	25
2.2.4. Prints das Telas.....	25
2.2.4.1. Tela Inicia.....	25
2.2.4.2. Palavra Oculta.....	26
2.2.4.3. Tentativa Correta.....	27

2.2.4.4. <i>Tentativa Incorreta</i> .....	28
2.2.4.5. <i>Jogo Vencido</i> .....	29
2.2.4.6. <i>Jogo Perdido</i> .....	30
2.2.4.7. <i>Reinício do Jogo</i> .....	31
2.2.5. Código Fonte.....	32
2.3. JOGO DA VELHA.....	47
2.3.1. Descrição.....	47
2.3.2. Objetivo.....	47
2.3.3. Funcionalidades.....	47
2.3.4. Prints das Telas.....	48
2.3.4.1. <i>Tela Inicial</i> .....	48
2.3.4.2. <i>Tabuleiro no Início da Partida</i> .....	50
2.3.4.3. <i>Jogada em Andamento</i> .....	52
2.3.4.4. <i>Vitória de um Jogador</i> .....	54
2.3.4.5. <i>Empate no Jogo</i> .....	56
2.3.4.6. <i>Reinício do Jogo</i> .....	58
2.3.5. Código Fonte.....	58
<b>CONCLUSÃO</b> .....	<b>69</b>
<b>REFERÊNCIA</b> .....	<b>70</b>

## INTRODUÇÃO

O desenvolvimento de jogos digitais é uma área crescente dentro da tecnologia, exigindo conhecimentos em lógica de programação, ferramentas específicas e trabalho em equipe. Este trabalho, realizado na disciplina de Desenvolvimento de Games, tem como objetivo a criação de diferentes jogos utilizando a linguagem Java e a ferramenta Processing. A proposta permitiu não apenas a aplicação prática de conceitos aprendidos, mas também o desenvolvimento de habilidades essenciais, como a adaptação a novas sintaxes e o uso de ambientes de desenvolvimento integrados.

Ao longo do projeto, o grupo enfrentou desafios relacionados à curva de aprendizado do Processing, que, apesar de utilizar Java, apresenta particularidades em sua sintaxe e estrutura. No entanto, a experiência foi enriquecedora, proporcionando um melhor entendimento sobre o desenvolvimento de jogos e resultando em um material que pode ser incorporado ao portfólio dos integrantes.



## 2. JOGOS DESENVOLVIDOS

### 2.1 JOGO DO MARCIANO

#### 2.1.1. Descrição

O jogo do Marciano é um jogo de adivinhação onde o jogador precisa encontrar um pequeno marciano escondido em uma árvore numerada de 1 a 100. O jogo fornece dicas sobre se o número escolhido está acima ou abaixo do correto e permite registrar recordes.

#### 2.1.2. Objetivo

O objetivo do jogo é incentivar a lógica e o raciocínio dos jogadores ao fazerem tentativas estratégicas para encontrar o número correto no menor tempo possível e com menos tentativas.

#### 2.1.3. Funcionalidades

- Interface interativa no console.
- Registro de recordes dos melhores tempos.
- Sistema de dicas para guiar o jogador.
- Mudança ocasional da posição do marciano para aumentar a dificuldade.

#### 2.1.4. Prints das Telas

##### 2.1.4.1. Tela Inicial

Mostra o título do jogo e as opções iniciais (Iniciar ou Sair).

Figura 1 - Tela Inicial no Jogo do Marciano.

```
*****
Jogo do Marciano

Tabela de Recordes:
Nome | Tempo (s)

*****
Digite 1 para iniciar
Digite 2 para sair

Digite a sua opção:
```

Fonte: Produção própria.

#### 2.1.4.2. Entrada do Nome do Jogador

Solicitação do nome antes de iniciar a história.

Figura 2 - Entrada do nome do jogador no Jogo do Marciano

```
Digite a sua opção: 1
*****
Digite seu nome para começar:
```

Fonte: Produção própria.

#### 2.1.4.3. História do Jogo

Exibição da introdução narrativa sobre o Marciano.

Figura 3 - Introdução da história no Jogo do Marciano.

```
Digite seu nome para começar: Jogador
Bem-vindo, Jogador! Sua jornada começa agora.

O Mistério do Marciano Perdido

Pressione ENTER para continuar...
```

Fonte: Produção própria.

Figura 4 - Contando a história no Jogo do Marciano.

```
Em um planeta distante chamado Xylox, um pequeno marciano chamado Zorp se perdeu enquanto explorava a Floresta das Mil Árvores.
Ele adora brincar de esconde-esconde, mas dessa vez foi longe demais e agora ninguém consegue encontrá-lo!

Pressione ENTER para continuar...
```

Fonte: Produção própria.

#### 2.1.4.4. *Início da Partida*

Primeira tentativa do jogador de adivinhar a árvore.

Figura 5 - Primeira tentativa do jogador no Jogo do Marciano.

```
Adivinhe a Árvore do Marciano

Digite um número da árvore onde o Marciano possa estar:
```

Fonte: Produção própria.

#### 2.1.4.5. *Dicas do Jogo*

Mensagem indicando se o número está maior ou menor.

Figura 6 - Exibição de dicas de que o marciano está em uma árvore menor no Jogo do Marciano.

```
*****
Sua última tentativa: 50
Número de Tentativas: 1
O Marciano está em uma árvore menor!

Digite um número da árvore onde o Marciano possa estar:
```

Fonte: Produção própria.

Figura 7 - Exibição de dicas de que o marciano está em uma árvore maior no Jogo do Marciano.

```
*****
Sua última tentativa: 10
Número de Tentativas: 2
O Marciano está em uma árvore maior!

Digite um número da árvore onde o Marciano possa estar:
```

Fonte: Produção própria.

#### 2.1.4.6. Mudança de Posição do Marciano

Exibição da mensagem quando o Marciano troca de árvore após 10 tentativas.

Figura 8 - Mensagem de mudança de posição do Marciano no Jogo do Marciano.

```
*****
Sua última tentativa: 10
Número de Tentativas: 10
O Marciano está em uma árvore maior!
O Marciano se cansou e trocou de árvore!

Digite um número da árvore onde o Marciano possa estar:
```

Fonte: Produção própria.

#### 2.1.4.7. Vitória do Jogador

Tela que informa que o jogador encontrou Marciano.

Figura 9 - Tela de vitória do jogador no Jogo do Marciano.

```
*****
🦖 Parabéns, Jogador! Você encontrou o Marciano!
*****
🕒 O Marciano estava na árvore: 41
📊 Número de Tentativas: 16
⌚ Tempo: 119 segundos
*****

🏆 Você foi o PRIMEIRO a registrar um tempo na tabela de recordes!

Gostaria de Jogar Novamente? s/n:
```

Fonte: Produção própria.

#### 2.1.4.8. Registro de Recordes

Exibição da tabela de recordes dos melhores tempos.

Figura 10 - Tabela de recordes no Jogo do Marciano.

```
*****
Jogo do Marciano

Tabela de Recordes:
Nome | Tempo (s)
Jogador | 119

*****

Digite 1 para iniciar
Digite 2 para sair

Digite a sua opção:
```

Fonte: Produção própria.

#### 2.1.4.9. Pergunta sobre Nova Partida

Tela perguntando se o jogador deseja jogar novamente.

Figura 11 - Pergunta sobre nova partida no Jogo do Marciano.

```
*****
🏆 Parabéns, Jogador! Você encontrou o Marciano!
*****
🌳 O Marciano estava na árvore: 41
📅 Número de Tentativas: 16
⌚ Tempo: 119 segundos
*****

🏆 Você foi o PRIMEIRO a registrar um tempo na tabela de recordes!
Gostaria de Jogar Novamente? s/n:
```

Fonte: Produção própria.

#### 2.1.5. Código Fonte

```
import java.util.Random;
import java.util.Scanner;
import java.util.List;
import java.util.ArrayList;
import java.util.Comparator;

public class Main {
    // Scanner para capturar a entrada do usuário via teclado
    static Scanner scanner = new Scanner(System.in);

    // Lista para armazenar os recordes dos jogadores
    static List<Record> recordes = new ArrayList<>();
```

```

// Variável global para armazenar o nome do jogador atual
static String nomeJogador;

public static void main(String[] args) {
    // Inicia o jogo chamando a tela inicial
    telaInicial();
}

static void telaInicial() {
    // Exibe o título e opções do menu inicial
    System.out.println("\n*****");
    System.out.println("Jogo do Marciano");

    // Mostra os recordes antes do jogador escolher a opção
    mostrarRecordes();

    System.out.println("Digite 1 para iniciar");
    System.out.println("Digite 2 para sair");
    System.out.print("\nDigite a sua opção: ");

    try {
        // Lê a entrada do usuário e converte para número
        int opcaoEscolhidaNumerica = Integer.parseInt(scanner.nextLine());

        // Verifica a opção escolhida pelo jogador
        switch (opcaoEscolhidaNumerica) {
            case 1:
                System.out.println("*****");
                solicitarNomeJogador(); // Solicita o nome antes da história
                contarHistoria();      // Conta a introdução do jogo
            case 2:
                System.out.println("*****");
                System.out.println("Saindo do jogo");
                break;
            default:
                System.out.println("Opção inválida");
        }
    } catch (Exception e) {
        System.out.println("Erro: " + e.getMessage());
    }
}

```

```

        gameMarciano();    // Inicia o jogo principal
        break;
    case 2:
        System.out.println("Você digitou a opção " + opcaoEscolhidaNumerica);
        System.out.println("Tchau tchau :"); // Mensagem de saída
        break;
    default:
        System.out.println("Opção inválida");
        telaInicial(); // Se a entrada for inválida, exibe o menu novamente
        break;
    }
} catch (NumberFormatException e) {
    System.out.println("Entrada inválida! Digite um número.");
    telaInicial(); // Se a entrada não for um número, exibe o menu novamente
}
}

static void solicitarNomeJogador() {
    // Solicita ao jogador que digite seu nome
    System.out.print("\nDigite seu nome para começar: ");

    // Armazena o nome digitado pelo jogador
    nomeJogador = scanner.nextLine();

    // Exibe uma mensagem de boas-vindas personalizada
    System.out.println("Bem-vindo, " + nomeJogador + "! Sua jornada começa
    agora.\n");
}

```



```
static void gameMarciano() {  
    // Inicia o jogo principal  
    game();  
  
    // Após o jogo, pergunta se o jogador quer jogar novamente  
    novaPartida();  
}
```

```
static void contarHistoria() {  
    // Exibe a introdução do jogo, apresentando a história ao jogador  
    mostrarTexto("O Mistério do Marciano Perdido");  
    esperarEnter(); // Aguarda o jogador pressionar ENTER para continuar
```

```
        mostrarTexto("Em um planeta distante chamado Xylox, um pequeno marciano  
chamado Zorp se perdeu enquanto explorava a Floresta das Mil Árvores. \nEle adora  
brincar de esconde-esconde, mas dessa vez foi longe demais e agora ninguém  
consegue encontrá-lo!");  
        esperarEnter();
```

```
        mostrarTexto("Você, " + nomeJogador + ", é um explorador intergaláctico e  
recebeu uma missão muito importante: encontrar Zorp antes que a noite caia e ele fique  
com medo no meio da floresta.");  
        esperarEnter();
```

```
        mostrarTexto("A única pista que você tem é que Zorp está escondido em uma  
árvore numerada de 1 a 100. \nSeu comunicador intergaláctico consegue detectar a  
posição aproximada de Zorp e te dirá se ele está escondido em uma árvore de número  
maior ou menor do que o que você tentou.");  
        esperarEnter();
```

```
    mostrarTexto("Será que você consegue encontrar o pequeno marciano antes que  
ele se assuste? Boa sorte, " + nomeJogador + "! 🚀👽");  
    esperarEnter();  
}
```

```
static void game() {  
    // Cria um objeto Random para gerar números aleatórios  
    Random rand = new Random();  
  
    // Define a posição inicial do Marciano em uma árvore entre 1 e 100  
    int arvoreMarciano = rand.nextInt(100) + 1;  
  
    // Variáveis auxiliares  
    int arvore = 0; // Número digitado pelo jogador  
    int numeroTentativa = 0; // Contador de tentativas  
    boolean marcianoEncontrado = false; // Indica se o jogador encontrou o Marciano  
  
    // Exibe o título do jogo  
    mostrarTexto("Adivinhe a Árvore do Marciano");  
  
    // Marca o tempo de início do jogo para medir quanto tempo o jogador leva  
    long inicioTempo = System.currentTimeMillis();  
  
    // Loop principal do jogo, continua até o jogador encontrar o Marciano  
    while (!marcianoEncontrado) {  
        System.out.print("\nDigite um número da árvore onde o Marciano possa estar: ");  
  
        // Captura a entrada do jogador  
        String opcaoEscolhida = scanner.nextLine();
```

```

try {
    // Converte a entrada do jogador para número inteiro
    arvore = Integer.parseInt(opcaoEscolhida);

    // Verifica se o número digitado está dentro do intervalo válido (1 a 100)
    if (arvore < 1 || arvore > 100) {
        System.out.println("Entrada inválida! Digite um número entre 1 e 100.");
        continue; // Volta para o início do loop
    }

    // Incrementa o número de tentativas
    numeroTentativa++;

    // Verifica se o jogador acertou a árvore correta
    if (arvore == arvoreMarciano) {
        // Calcula o tempo total que o jogador levou para encontrar o Marciano
        long tempoDecorrido = (System.currentTimeMillis() - inicioTempo) / 1000;

        // Exibe a mensagem de vitória
        System.out.println("\n*****");
        System.out.println("🎉 Parabéns, " + nomeJogador + "! Você encontrou o Marciano!");
        System.out.println("*****");
        System.out.println("🌳 O Marciano estava na árvore: " + arvoreMarciano);
        System.out.println("📊 Número de Tentativas: " + numeroTentativa);
        System.out.println("⌚ Tempo: " + tempoDecorrido + " segundos");
        System.out.println("*****\n");

        // Verifica se o jogador bateu o recorde de tempo

```

```

    Recorde melhorRecorde = obterMelhorRecorde();

    if (melhorRecorde == null) {
        // Se ainda não há recordes registrados
        System.out.println("🏆 Você foi o PRIMEIRO a registrar um tempo na
tabela de recordes!");
    } else if (tempoDecorrido < melhorRecorde.tempo) {
        // Se o tempo do jogador foi melhor que o recorde anterior
        System.out.println("🔥 NOVO RECORD MUNDIAL! Você superou " +
melhorRecorde.nome + ", que tinha " + melhorRecorde.tempo + " segundos!");
    }

    // Registra o tempo do jogador na tabela de recordes
    registrarRecorde(tempoDecorrido);

    // Define que o Marciano foi encontrado, encerrando o jogo
    marcianoEncontrado = true;
    break;
}

// Exibe as informações da tentativa
System.out.println("\n*****");
System.out.println("Sua última tentativa: " + arvore);
System.out.println("Número de Tentativas: " + numeroTentativa);

// Dá uma dica ao jogador, indicando se o Marciano está em uma árvore maior
ou menor
    if (arvore > arvoreMarciano) {
        System.out.println("O Marciano está em uma árvore menor!");
    } else {

```

```

        System.out.println("O Marciano está em uma árvore maior!");
    }

    // A cada 10 tentativas, o Marciano muda de posição
    if (numeroTentativa % 10 == 0) {
        System.out.println("O Marciano se cansou e trocou de árvore!");
        arvoreMarciano = rand.nextInt(100) + 1;
    }

} catch (NumberFormatException e) {
    // Caso o jogador digite algo que não seja um número
    System.out.println("Entrada inválida! Digite um número entre 1 e 100.");
}
}
}

```

```

// Retorna o melhor recorde registrado
static Recorde obterMelhorRecorde() {
    return recordes.isEmpty() ? null : recordes.get(0);
}

// Método para iniciar uma nova partida
static void novaPartida() {
    while (true) {
        System.out.print("\nGostaria de Jogar Novamente? s/n: ");

        // Cria um novo Scanner para capturar a entrada do usuário
        scanner = new Scanner(System.in);
    }
}

```

String opcaoEscolhida = scanner.nextLine().toLowerCase(); // Converte para minúsculas para evitar erros de comparação

```

    if (opcaoEscolhida.equals("s")) {
        System.out.println("\n*****");
        gameMarciano(); // Reinicia o jogo
    } else if (opcaoEscolhida.equals("n")) {
        telaInicial(); // Retorna ao menu inicial
        break;
    } else {
        System.out.println("Opção inválida!"); // Mensagem de erro caso a entrada
        seja inválida
    }
}
}
}

```

// Método para exibir um texto com efeito de digitação

```

static void mostrarTexto(String texto) {
    for (char letra : texto.toCharArray()) { // Percorre cada caractere do texto
        System.out.print(letra);
        try {
            Thread.sleep(30); // Pausa de 30 milissegundos entre cada caractere para
            criar o efeito
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt(); // Garante que a interrupção seja tratada
            corretamente
        }
    }
    System.out.println(); // Nova linha ao final do texto
}

```

```
// Método para pausar o jogo até o jogador pressionar ENTER
static void esperarEnter() {
    System.out.println("\nPressione ENTER para continuar...");
    scanner.nextLine(); // Aguarda o jogador pressionar ENTER
}

// Método para registrar um novo recorde
static void registrarRecorde(long tempo) {
    // Adiciona um novo recorde com o nome do jogador e o tempo gasto
    recordes.add(new Recorde(nomeJogador, tempo));

    // Ordena a lista de recordes pelo tempo (menor tempo primeiro)
    recordes.sort(Comparator.comparingLong(r -> r.tempo));

    // Mantém apenas os 10 melhores tempos na lista
    if (recordes.size() > 10) {
        recordes.remove(recordes.size() - 1);
    }
}

static void mostrarRecordes() {
    System.out.println("\nTabela de Recordes:");
    System.out.println("Nome | Tempo (s)");

    // Percorre a lista de recordes e exibe cada um
    for (Recorde recorde : recordes) {
        System.out.println(recorde.nome + " | " + recorde.tempo);
    }
}
```

```

        System.out.println("\n*****");
    }
}

class Recorde {
    String nome; // Percorre a lista de recordes e exibe cada um
    long tempo; // Tempo que ele levou para encontrar o marciano

    // Construtor da classe Recorde
    Recorde(String nome, long tempo) {
        this.nome = nome;
        this.tempo = tempo;
    }
}

```

## 2.2 JOGO DA FORCA

### 2.2.1. Descrição

O Jogo da Forca é um jogo clássico de adivinhação de palavras, onde o jogador tenta descobrir uma palavra oculta ao sugerir letras. Neste projeto, o jogo foi desenvolvido utilizando a linguagem Processing, proporcionando uma experiência visual interativa e dinâmica. O código implementa elementos gráficos e lógicos que garantem a jogabilidade, incluindo a exibição da palavra em progresso, letras já utilizadas e um sistema de penalidade para erros.



### 2.2.2. Objetivo

O objetivo do Jogo da Força é desafiar os jogadores a adivinhar uma palavra oculta antes que o número máximo de tentativas se esgote. O jogo estimula o raciocínio lógico e a memorização, além de ser uma forma divertida de aprendizado e entretenimento. O jogador deve sugerir letras e evitar erros para não completar a força e perder a partida.

### 2.2.3. Funcionalidades

- **Inicialização do Jogo:** O jogo carrega uma palavra aleatória de um conjunto pré-definido e define o número máximo de tentativas.
- **Interação do Jogador:** O jogador pode inserir letras para tentar adivinhar a palavra oculta.
- **Exibição Gráfica:** O jogo exibe a palavra parcialmente revelada, letras já utilizadas e um desenho da força que se completa a cada erro.
- **Verificação de Acertos e Erros:** O código verifica se a letra sugerida faz parte da palavra e, caso contrário, adiciona um erro ao contador.
- **Condições de Vitória e Derrota:** O jogo termina quando o jogador adivinha todas as letras corretamente ou quando atinge o número máximo de erros.
- **Reinício do Jogo:** Após o término de uma rodada, o jogo pode ser reiniciado para uma nova palavra.
- **Interface Visual:** A interface apresenta elementos visuais que tornam o jogo intuitivo e acessível para o usuário.

### 2.2.4. Prints das Telas

#### 2.2.4.1. Tela Inicia

Tela de início do Jogo da Força antes da primeira jogada.

Figura 12 – Tela inicial no Jogo da Forca.

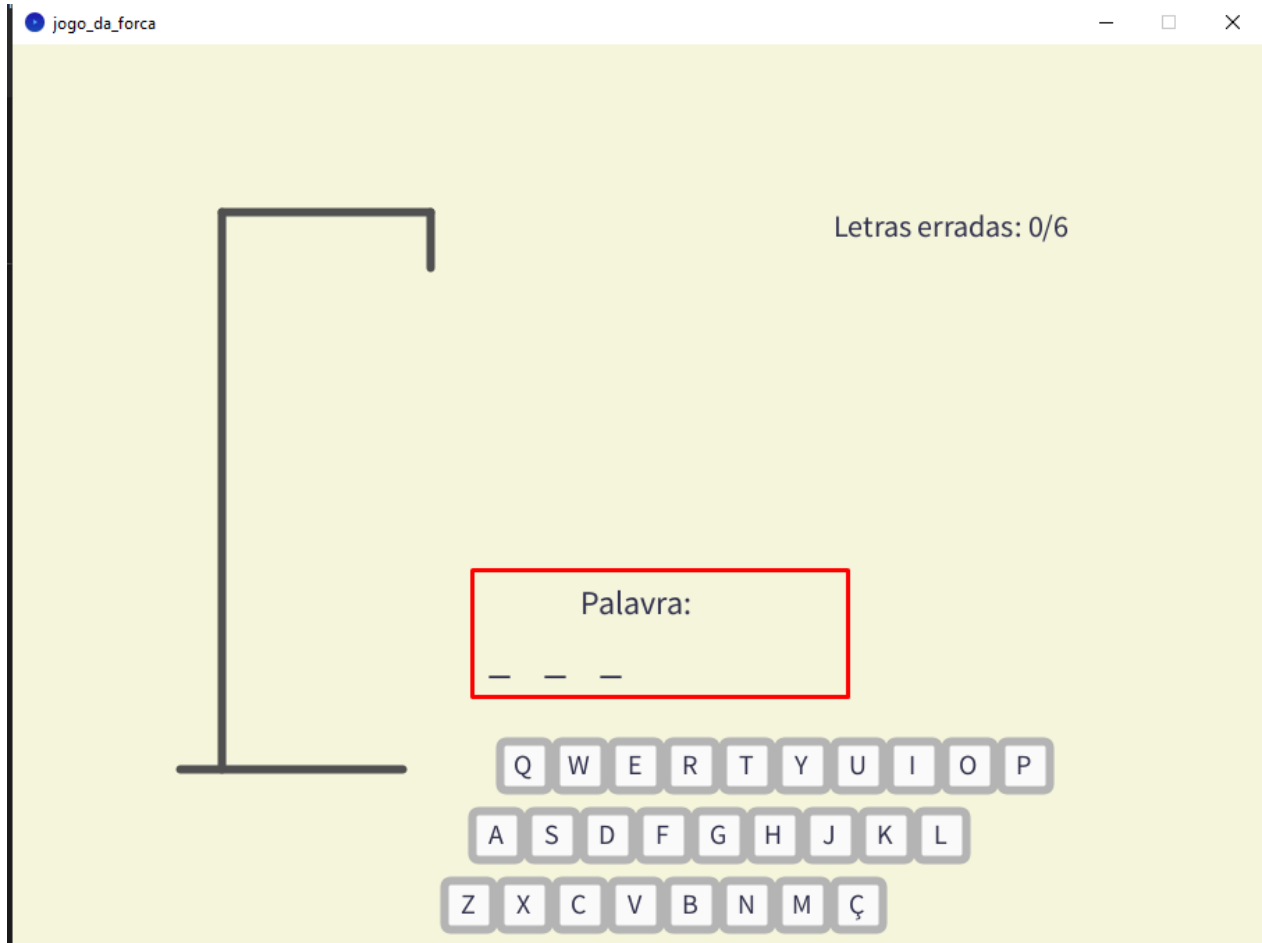


Fonte: Produção própria.

#### 2.2.4.2. Palavra Oculta

Exibição da palavra oculta no início do jogo, representada por traços.

Figura 13 – Exibição da palavra oculta no Jogo da Forca.

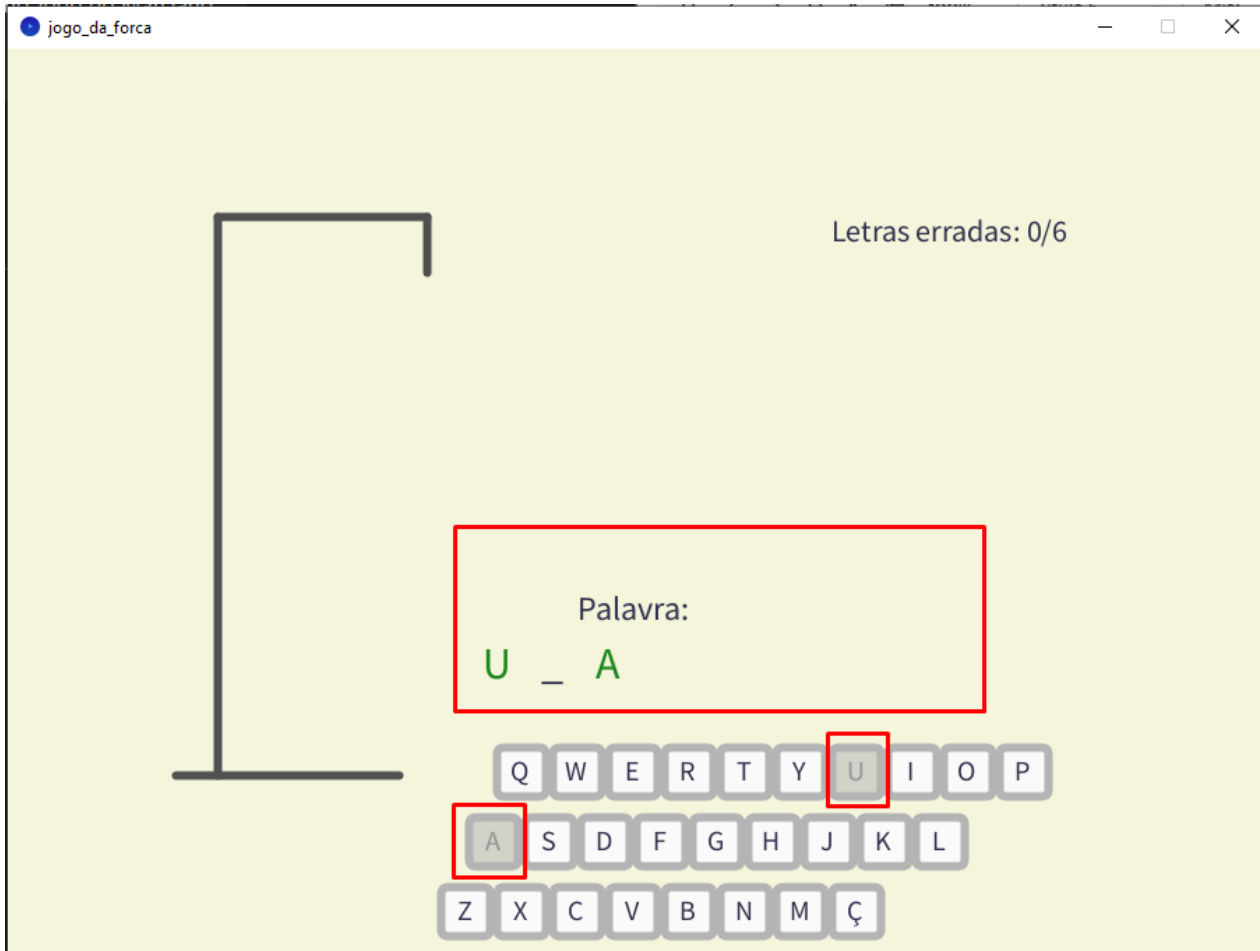


Fonte: Produção própria.

#### 2.2.4.3. Tentativa Correta

O jogador insere uma letra correta, revelando sua posição na palavra.

Figura 14 – Tentativa correta do jogador no Jogo da Forca.

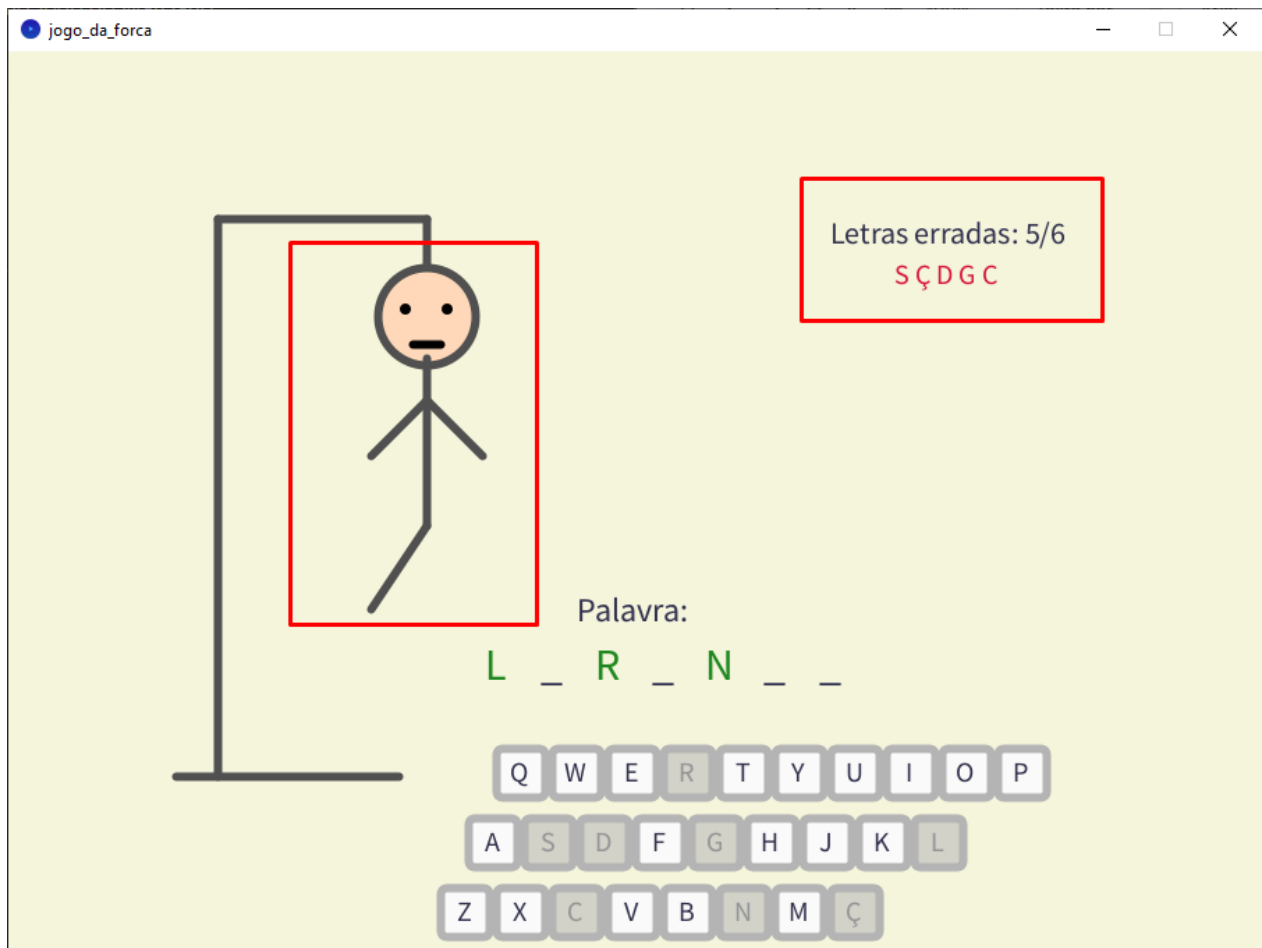


Fonte: Produção própria.

#### 2.2.4.4. Tentativa Incorreta

O jogador insere uma letra errada, resultando em uma penalidade.

Figura 15 – Tentativa incorreta do jogador no Jogo da Forca.



Fonte: Produção própria.

#### 2.2.4.5. Jogo Vencido

O jogador adivinhar todas as letras corretamente e vence a partida.

Figura 16 – Tela de vitória no Jogo da Forca.

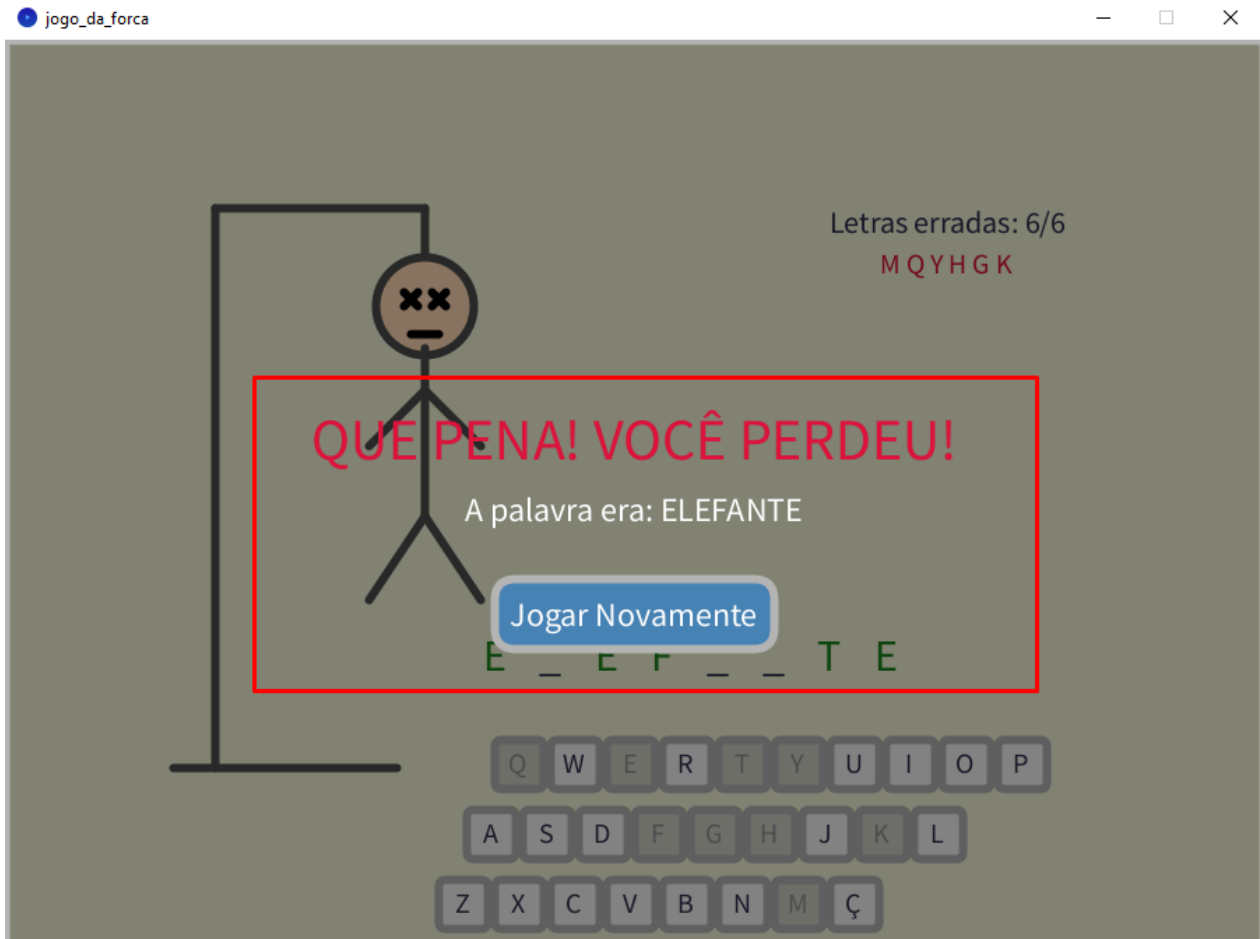


Fonte: Produção própria.

#### 2.2.4.6. Jogo Perdido

O jogador atinge o número máximo de erros, resultando em derrota.

Figura 17 – Tela de derrota no Jogo da Forca.

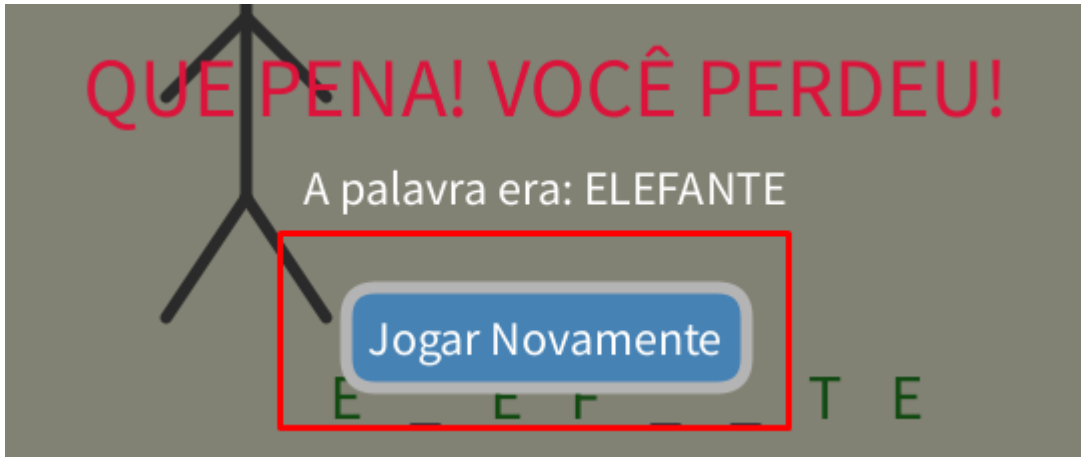


Fonte: Produção própria.

#### 2.2.4.7. Reinício do Jogo

Tela exibida após o término da rodada, permitindo reiniciar o jogo.

Figura 18 – Opção de reinício no Jogo da Forca.



Fonte: Produção própria.

#### 2.2.5. Código Fonte

// Jogo da Forca em Processing com Design Moderno

// Por: Ingryd Barbosa

```
String[] categorias = {"Frutas", "Cidades", "Animais", "Países"};
String[][] palavras = {
    {"BANANA", "MORANGO", "ABACAXI", "LARANJA", "UVA", "MELANCIA", "KIWI",
    "MELÃO", "PÊSSEGO"},
    {"SÃO PAULO", "RIO DE JANEIRO", "BRASÍLIA", "SALVADOR", "CURITIBA", "BELO
    HORIZONTE", "FORTALEZA", "RECIFE"},
    {"ELEFANTE", "GIRAFÁ", "TIGRE", "LEÃO", "ZEBRA", "MACACO", "RINOCERONTE",
    "HIPOPÓTAMO", "CROCODILO"},
    {"BRASIL", "ARGENTINA", "CANADÁ", "JAPÃO", "ALEMANHA", "ITÁLIA", "FRANÇA",
    "ESPANHA", "PORTUGAL"}
};
```



```

String palavraSecreta;
char[] letrasDescobertas;
ArrayList<Character> letrasErradas = new ArrayList<Character>();
int categoriaSelecionada = -1;
int erros = 0;
final int MAX_ERROS = 6;
boolean jogoAtivo = false;
boolean vitoria = false;

// Variáveis do cronômetro
int tempoInicial;
int tempoLimite = 120; // 2 minutos em segundos
boolean modoContraTempo = false;
boolean tempoEsgotado = false;

// Cores
color corFundo = color(245, 245, 220); // Bege claro
color corTexto = color(50, 50, 80); // Azul escuro
color corBotao = color(70, 130, 180); // Azul médio
color corBotaoHover = color(100, 150, 200); // Azul mais claro
color corBotaoTempo = color(218, 165, 32); // Dourado
color corBotaoTempoHover = color(230, 180, 50); // Dourado claro
color corErro = color(220, 20, 60); // Vermelho
color corAcerto = color(34, 139, 34); // Verde
color corTempoCritico = color(178, 34, 34); // Vermelho escuro

void setup() {
    size(900, 650);
    textAlign(CENTER, CENTER);
    smooth();
}

```

```
}
```

```
void draw() {
```

```
    background(corFundo);
```

```
    if (!jogoAtivo) {
```

```
        desenharMenu();
```

```
    } else {
```

```
        desenharJogo();
```

```
        if (erros >= MAX_ERROS || vitoria || (modoContraTempo && tempoEsgotado())) {
```

```
            desenharResultado();
```

```
        }
```

```
    }
```

```
}
```

```
// Método para remover acentos
```

```
char removerAcento(char letra) {
```

```
    switch(letra) {
```

```
        case 'Á': case 'À': case 'Â': case 'Ã': case 'Ä':
```

```
            return 'A';
```

```
        case 'á': case 'à': case 'â': case 'ã': case 'ä':
```

```
            return 'a';
```

```
        case 'É': case 'È': case 'Ê': case 'Ë':
```

```
            return 'E';
```

```
        case 'é': case 'è': case 'ê': case 'ë':
```

```
            return 'e';
```

```
        case 'Í': case 'Ì': case 'Î': case 'Ï':
```

```
            return 'I';
```

```
        case 'í': case 'ì': case 'î': case 'ï':
```

```

    return 'i';
case 'Ó': case 'Ò': case 'Ô': case 'Õ': case 'Ö':
    return 'O';
case 'ó': case 'ò': case 'ô': case 'õ': case 'ö':
    return 'o';
case 'Ú': case 'Ù': case 'Û': case 'Ü':
    return 'U';
case 'ú': case 'ù': case 'û': case 'ü':
    return 'u';
case 'Ç':
    return 'C';
case 'ç':
    return 'c';
default:
    return letra;
}
}

```

```

void desenharMenu() {
    // Título
    fill(corTexto);
    textSize(42);
    text("JOGO DA FORCA", width/2, 70);
    textSize(28);
    text("Selecione uma categoria:", width/2, 120);

    // Botões de categoria
    for (int i = 0; i < categorias.length; i++) {
        boolean sobreBotao = mouseSobreBotao(width/2 - 150, 160 + i*80, 300, 50);
        fill(sobreBotao ? corBotaoHover : corBotao);
    }
}

```

```

    rect(width/2 - 150, 160 + i*80, 300, 50, 15);
    fill(255);
    text(categorias[i], width/2, 185 + i*80);
}

// Botão para modo contra o tempo
boolean sobreTempo = mouseSobreBotao(width/2 - 150, 160 + categorias.length*80 +
30, 300, 50);
fill(sobreTempo ? corBotaoTempoHover : corBotaoTempo);
rect(width/2 - 150, 160 + categorias.length*80 + 30, 300, 50, 15);
fill(255);
text(modosContraTempo ? "Modo Contra Tempo: ON" : "Modo Contra Tempo: OFF",
width/2, 185 + categorias.length*80 + 30);

// Explicação do modo contra o tempo
if (modosContraTempo) {
    fill(corTexto);
    textSize(18);
    text("Você terá " + tempoLimite + " segundos para adivinhar a palavra!", width/2, 185
+ categorias.length*80 + 85);
    textSize(28);
}

// Créditos
fill(corTexto);
textSize(14);
text("Desenvolvido por Ingryd Barbosa", width/2, height - 30);
}

void desenharJogo() {

```

```
// Desenhando a força
stroke(80);
strokeWeight(6);
noFill();

// Base da força
line(120, 520, 280, 520);
// Poste vertical
line(150, 520, 150, 120);
// Topo horizontal
line(150, 120, 300, 120);
// Corda
line(300, 120, 300, 160);

// Desenhando o boneco
if (erros > 0) { // Cabeça
  fill(255, 218, 185);
  stroke(80);
  ellipse(300, 190, 70, 70);

  if (erros >= MAX_ERROS) {
    stroke(0);
    line(285, 180, 295, 190);
    line(295, 180, 285, 190);
    line(315, 180, 305, 190);
    line(305, 180, 315, 190);
  } else {
    fill(0);
    noStroke();
    ellipse(285, 185, 8, 8);
```

```
    ellipse(315, 185, 8, 8);
  }

  noFill();
  stroke(0);
  if (vitoria) {
    arc(300, 200, 30, 20, 0, PI);
  } else {
    line(290, 210, 310, 210);
  }
}

if (erros > 1) { // Corpo
  stroke(80);
  line(300, 220, 300, 340);
}

if (erros > 2) { // Braço esquerdo
  line(300, 250, 260, 290);
}

if (erros > 3) { // Braço direito
  line(300, 250, 340, 290);
}

if (erros > 4) { // Perna esquerda
  line(300, 340, 260, 400);
}

if (erros > 5) { // Perna direita
```

```

    line(300, 340, 340, 400);
}

// Letras erradas
fill(corTexto);
textSize(22);
text("Letras erradas: " + letrasErradas.size() + "/" + MAX_ERROS, 675, 130);

fill(corErro);
textSize(20);
String erradasStr = "";
for (int i = 0; i < letrasErradas.size(); i++) {
    erradasStr += letrasErradas.get(i) + " ";
}
text(erradasStr, 675, 160);

// Palavra secreta
fill(corTexto);
textSize(24);
text("Palavra: " + (modoContraTempo ? "(Contra o tempo)" : ""), width/2, 400);

fill(0);
textSize(32);
for (int i = 0; i < letrasDescobertas.length; i++) {
    if (letrasDescobertas[i] != 0) {
        fill(corAcerto);
        text(letrasDescobertas[i], 350 + i*40, 440);
    } else {
        fill(corTexto);
        text("_", 350 + i*40, 440);
    }
}

```

```

    }
}

// Teclado
desenharTecladoCompleto();

// Cronômetro
if (modoContraTempo) {
    desenharCronometro();
}
}

void desenharTecladoCompleto() {
    String[] linhas = {"QWERTYUIOP", "ASDFGHJKL", "ZXCVBNMÇ"};

    for (int l = 0; l < linhas.length; l++) {
        for (int i = 0; i < linhas[l].length(); i++) {
            char letra = linhas[l].charAt(i);
            boolean usada = letrasErradas.contains(letra) || letraJaDescoberta(letra);
            boolean sobreLetra = mouseSobreBotao(350 + i*40 - l*20, 500 + l*50, 35, 35);

            if (usada) {
                fill(200, 200);
            } else if (sobreLetra) {
                fill(240);
            } else {
                fill(255, 220);
            }

            stroke(180);

```



```

    rect(350 + i*40 - l*20, 500 + l*50, 35, 35, 5);

    fill(usada ? 150 : corTexto);
    textSize(20);
    text(letra, 350 + i*40 - l*20 + 17, 500 + l*50 + 17);
  }
}
}

void desenharCronometro() {
  int tempoRestante = tempoLimite - (millis() - tempoInicial) / 1000;
  tempoRestante = max(0, tempoRestante);

  fill(230);
  noStroke();
  rect(width - 200, 30, 170, 50, 10);

  if (tempoRestante <= 10) {
    fill(corTempoCritico);
  } else if (tempoRestante <= 30) {
    fill(255, 165, 0);
  } else {
    fill(50, 50, 80);
  }

  textSize(28);
  text("⌚ " + nf(tempoRestante, 2) + "s", width - 115, 55);

  if (tempoRestante <= 0 && !tempoEsgotado) {
    tempoEsgotado = true;
  }
}

```

```

    }
}

```

```

boolean letraJaDescoberta(char letra) {
    for (int i = 0; i < palavraSecreta.length(); i++) {
        char letraPalavra = removerAcento(palavraSecreta.charAt(i));
        char letraDigitada = removerAcento(letra);

        if (Character.toUpperCase(letraPalavra) == Character.toUpperCase(letraDigitada)
            && letrasDescobertas[i] == palavraSecreta.charAt(i)) {
            return true;
        }
    }
    return false;
}

```

```

void desenharResultado() {
    fill(0, 0, 0, 120);
    rect(0, 0, width, height);
    fill(255);
    textSize(42);

    if (vitoria) {
        fill(corAcerto);
        text("PARABÉNS! VOCÊ GANHOU!", width/2, height/2 - 40);
    } else if (tempoEsgotado) {
        fill(corTempoCritico);
        text("TEMPO ESGOTADO!", width/2, height/2 - 40);
    } else {
        fill(corErro);
    }
}

```

```

    text("QUE PENA! VOCÊ PERDEU!", width/2, height/2 - 40);
}

fill(255);
textSize(24);
text("A palavra era: " + palavraSecreta, width/2, height/2 + 10);

// Botão "Jogar Novamente"
boolean sobreJogarNovamente = mouseSobreBotao(width/2 - 100, height/2 + 60, 200,
50);
fill(sobreJogarNovamente ? corBotaoHover : corBotao);
rect(width/2 - 100, height/2 + 60, 200, 50, 15);
fill(255);
text("Jogar Novamente", width/2, height/2 + 85);
}

boolean mouseSobreBotao(float x, float y, float w, float h) {
    return mouseX > x && mouseX < x + w && mouseY > y && mouseY < y + h;
}

void mousePressed() {
    if (!jogoAtivo) {
        for (int i = 0; i < categorias.length; i++) {
            if (mouseSobreBotao(width/2 - 150, 160 + i*80, 300, 50)) {
                iniciarJogo(i);
            }
        }
    }

    if (mouseSobreBotao(width/2 - 150, 160 + categorias.length*80 + 30, 300, 50)) {
        modoContraTempo = !modoContraTempo;
    }
}

```

```

    }
} else if (erros >= MAX_ERROS || vitoria || tempoEsgotado) {
    if (mouseSobreBotao(width/2 - 100, height/2 + 60, 200, 50)) {
        reiniciarJogo();
    }
} else {
    String[] linhas = {"QWERTYUIOP", "ASDFGHJKL", "ZXCVBNMÇ"};

    for (int l = 0; l < linhas.length; l++) {
        for (int i = 0; i < linhas[l].length(); i++) {
            char letra = linhas[l].charAt(i);
            boolean usada = letrasErradas.contains(letra) || letraJaDescoberta(letra);

            if (!usada && mouseSobreBotao(350 + i*40 - l*20, 500 + l*50, 35, 35)) {
                verificarLetra(letra);
            }
        }
    }
}
}

void iniciarJogo(int categoria) {
    categoriaSelecionada = categoria;
    palavraSecreta = palavras[categoria][(int)random(palavras[categoria].length)];
    letrasDescobertas = new char[palavraSecreta.length()];

    for (int i = 0; i < palavraSecreta.length(); i++) {
        if (palavraSecreta.charAt(i) == ' ' || palavraSecreta.charAt(i) == '-') {
            letrasDescobertas[i] = palavraSecreta.charAt(i);
        } else {

```

```
    letrasDescobertas[i] = 0;  
  }  
}
```

```
letrasErradas.clear();  
erros = 0;  
jogoAtivo = true;  
vitoria = false;  
tempoEsgotado = false;
```

```
if (modoContraTempo) {  
    tempolnicial = millis();  
}  
}
```

```
void reiniciarJogo() {  
    jogoAtivo = false;  
    vitoria = false;  
    erros = 0;  
    tempoEsgotado = false;  
    letrasErradas.clear();  
}
```

```
void verificarLetra(char letra) {  
    boolean acertou = false;
```

```
    for (int i = 0; i < palavraSecreta.length(); i++) {  
        char letraPalavra = removerAcento(palavraSecreta.charAt(i));  
        char letraDigitada = removerAcento(letra);
```

```

    if (Character.toUpperCase(letraPalavra) == Character.toUpperCase(letraDigitada)) {
        letrasDescobertas[i] = palavraSecreta.charAt(i);
        acertou = true;
    }
}

if (!acertou) {
    letrasErradas.add(letra);
    erros++;
}

vitoria = true;
for (int i = 0; i < letrasDescobertas.length; i++) {
    if (letrasDescobertas[i] == 0) {
        vitoria = false;
        break;
    }
}

boolean tempoEsgotado() {
    if (!modoContraTempo) return false;
    int tempoDecorrido = (millis() - tempolnicial) / 1000;
    return tempoDecorrido >= tempoLimite;
}

void keyPressed() {
    if (jogoAtivo && erros < MAX_ERROS && !vitoria && !tempoEsgotado) {
        if ((key >= 'a' && key <= 'z') || (key >= 'A' && key <= 'Z') || key == 'Ç' || key == 'ç') {
            verificarLetra(key);
        }
    }
}

```

```
}  
}  
}
```

## 2.3. JOGO DA VELHA

### 2.3.1. Descrição

O jogo da velha é um jogo de tabuleiro simples, jogado por dois participantes em um grid 3x3. Cada jogador escolhe um símbolo (normalmente "X" ou "O") e faz suas jogadas alternadamente, tentando formar uma linha com três símbolos iguais na horizontal, vertical ou diagonal.

### 2.3.2. Objetivo

O objetivo do jogo é formar uma linha com três símbolos iguais antes do adversário, garantindo a vitória. Caso todas as casas do tabuleiro sejam preenchidas sem que nenhum jogador complete uma linha, o jogo termina em empate.

### 2.3.3. Funcionalidades

- Exibição de um tabuleiro 3x3 interativo.
- Alternância automática entre os jogadores.
- Verificação automática de vitória ou empate.
- Indicação do jogador vencedor ou mensagem de empate quando aplicável.
- Reinício da partida após o término do jogo.

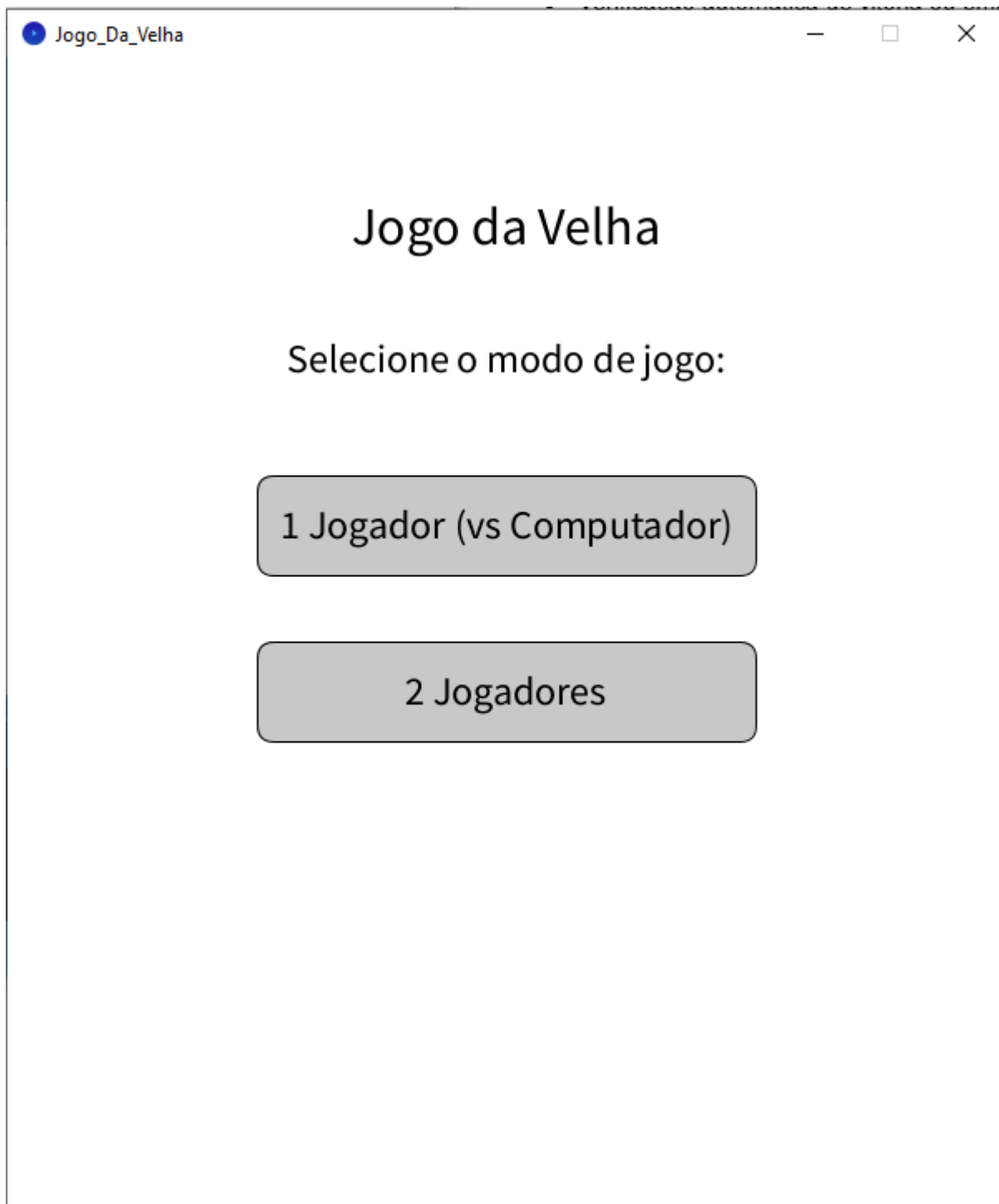
#### 2.3.4. Prints das Telas

##### *2.3.4.1. Tela Inicial*

Apresentação da interface inicial do jogo, exibindo o título e o botão para iniciar a partida.



Figura 19 – Tela inicial do Jogo da Velha.

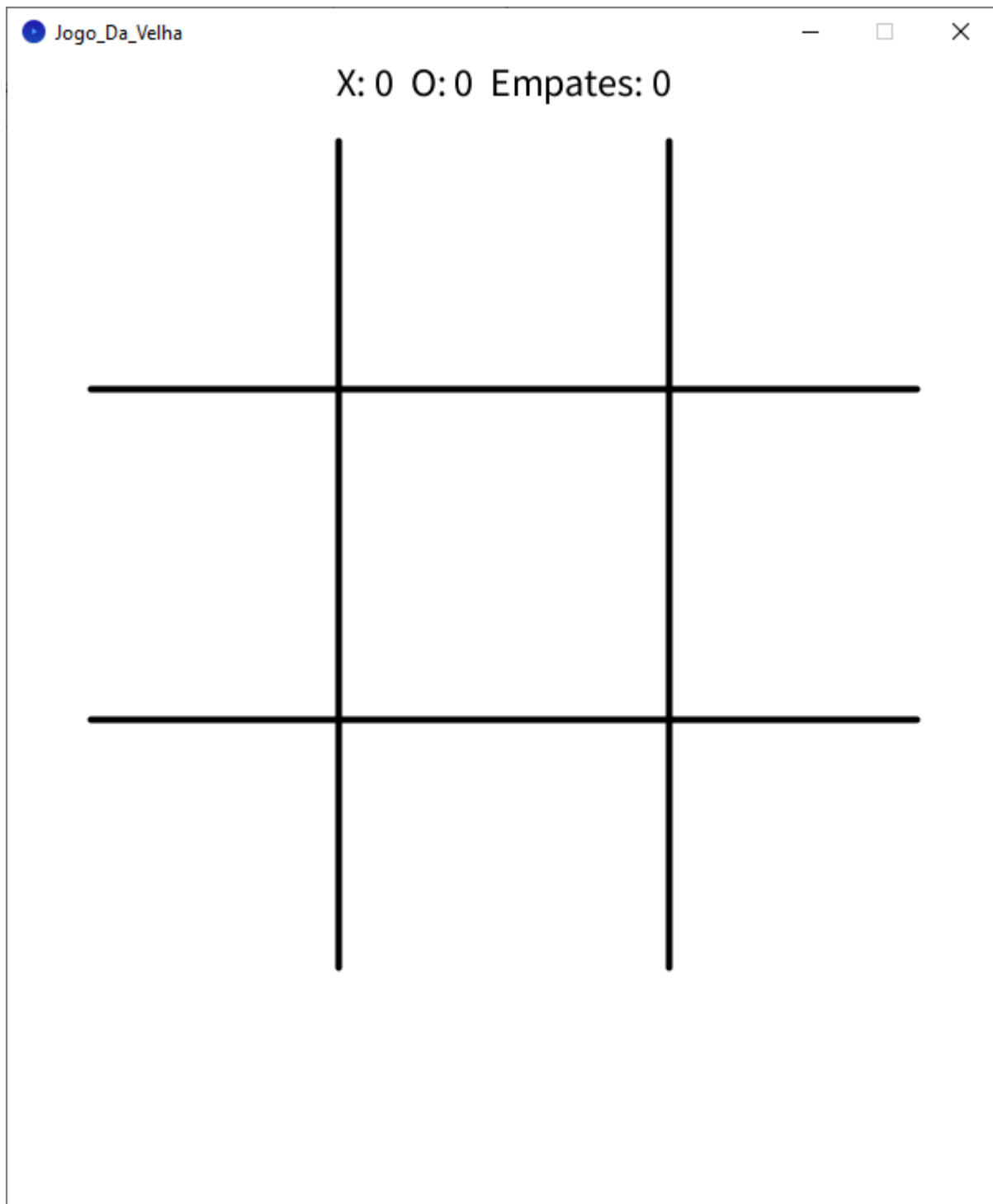


Fonte: Produção própria.

#### *2.3.4.2. Tabuleiro no Início da Partida*

Exibição do tabuleiro vazio no início do jogo, aguardando a primeira jogada.

Figura 20 – Tabuleiro vazio no início da partida no Jogo da Velha.

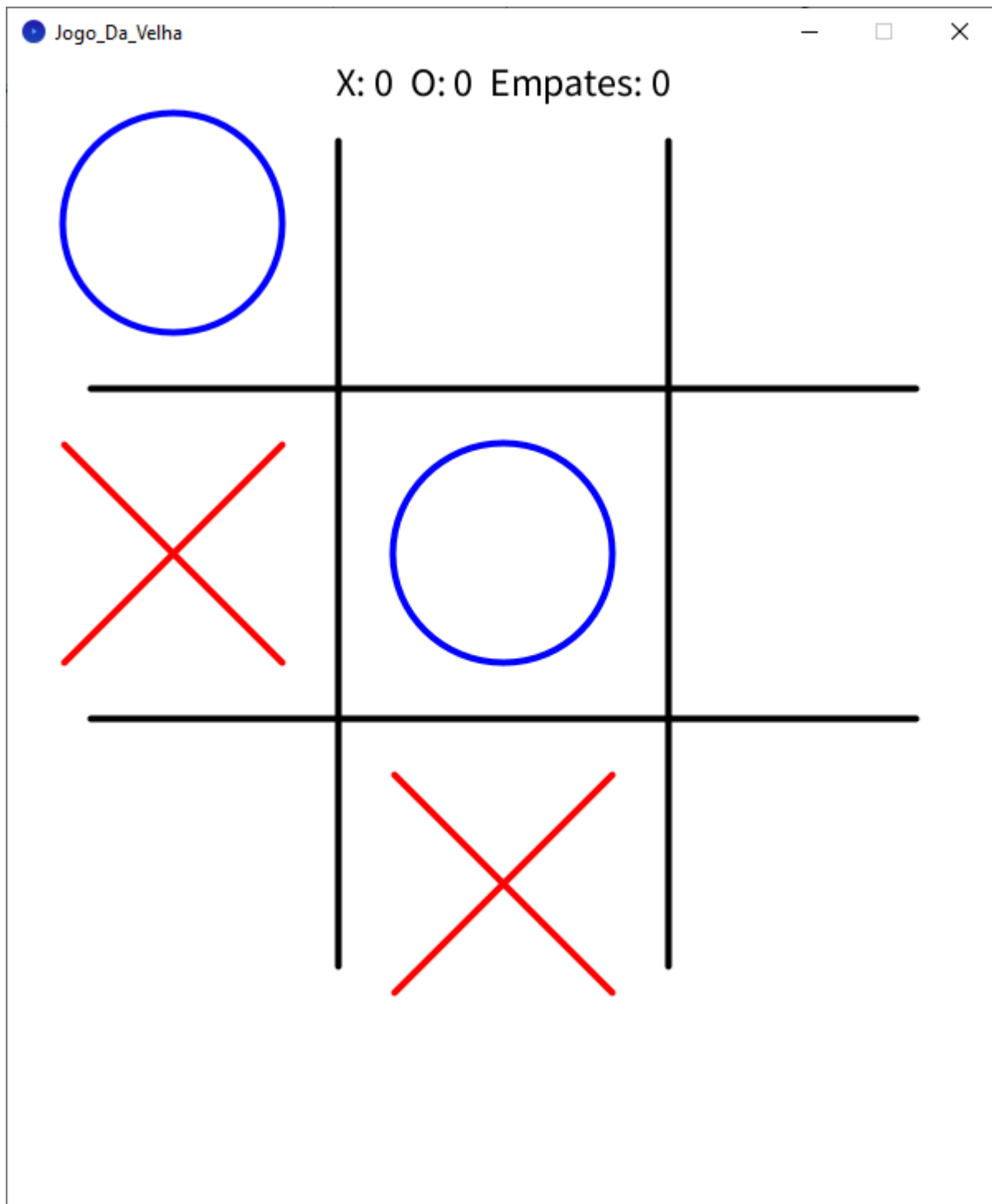


Fonte: Produção própria.

#### *2.3.4.3. Jogada em Andamento*

Tabuleiro com algumas jogadas feitas, mostrando a alternância entre os jogadores.

Figura 21 – Jogada em andamento no Jogo da Velha.

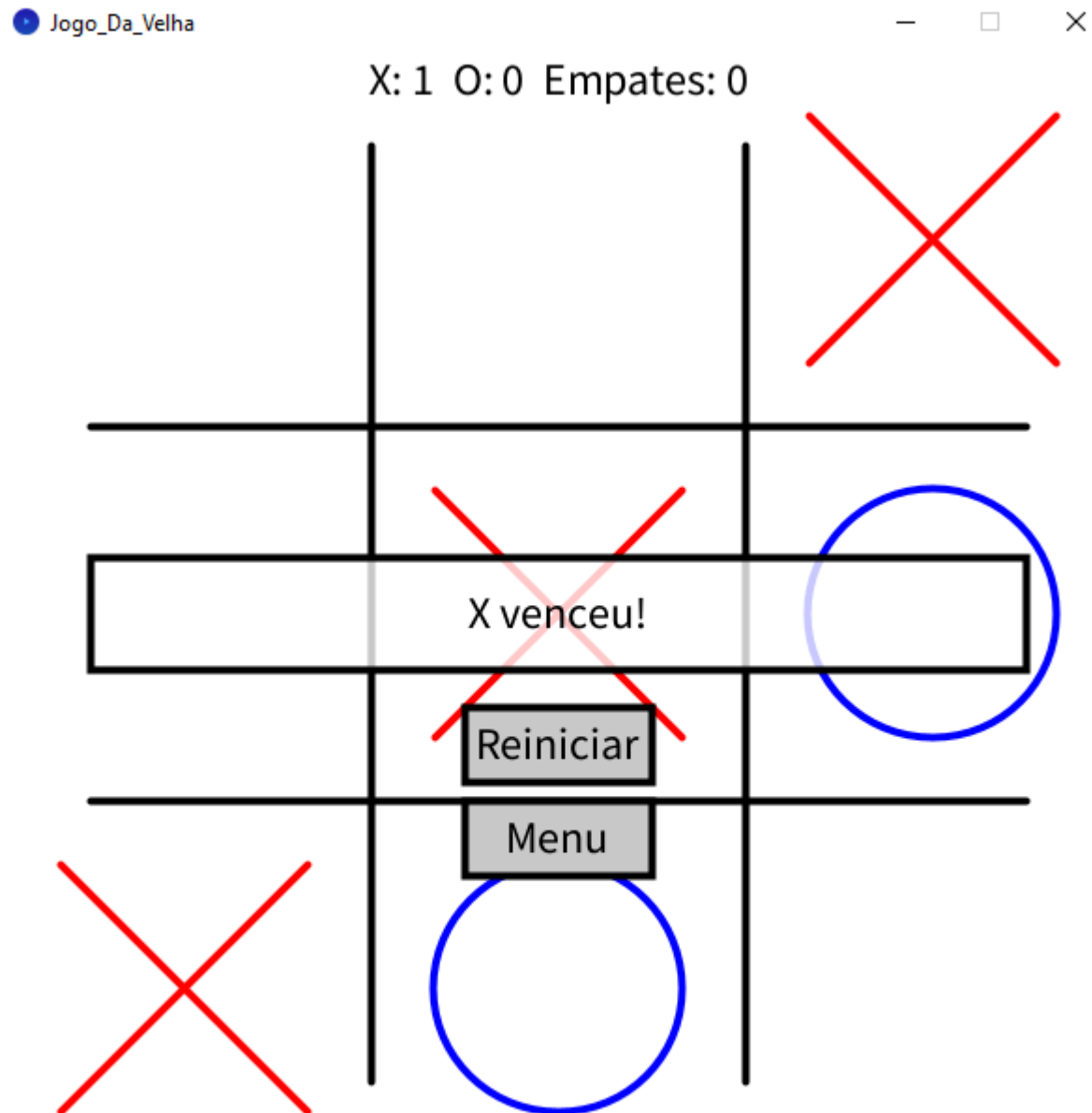


Fonte: Produção própria.

#### *2.3.4.4. Vitória de um Jogador*

Tela indicando a vitória de um dos jogadores ao formar uma linha de três símbolos iguais.

Figura 22 – Vitória de um jogador no Jogo da Velha.



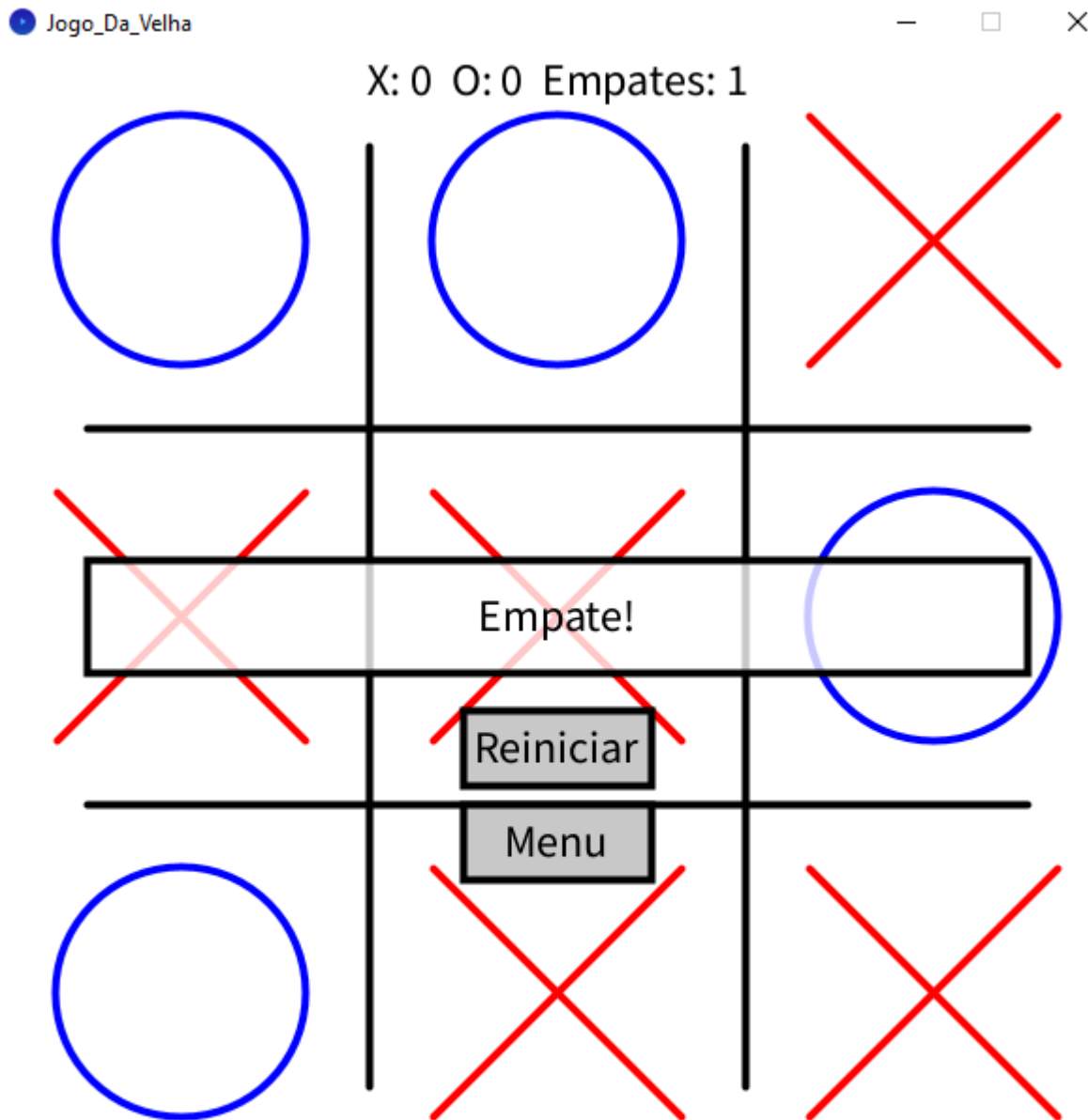
Fonte: Produção própria.

#### 2.3.4.5. *Empate no Jogo*

Exibição do tabuleiro preenchido sem um vencedor, indicando o empate.



Figura 23 – Tela de empate no Jogo da Velha.

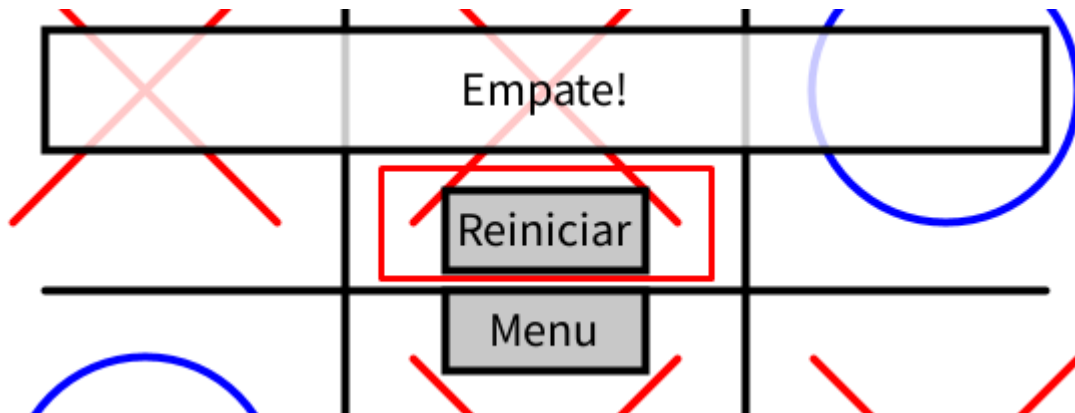


Fonte: Produção própria.

#### 2.3.4.6. Reinício do Jogo

Tela exibindo a opção para reiniciar a partida após um jogo encerrado.

Figura 24 – Opção de reinício no Jogo da Velha.



Fonte: Produção própria.

#### 2.3.5. Código Fonte

// Jogo da Velha (Tic Tac Toe) em Processing

// Versão final com reset de contadores ao voltar ao menu

```
final int TAMANHO = 600;
```

```
final int TAM_CELULA = TAMANHO / 3;
```

```
final int MARGEM = 50;
```

```
char[][] tabuleiro = new char[3][3];
```

```
boolean vezX = true;
```

```
boolean jogoAtivo = false;
```

```
boolean vsComputador;
```

```
boolean telaInicial = true;
```

```
boolean resultadoProcessado = false;
```

```
int vitoriasX = 0;
```

```
int vitoriasO = 0;
```

```
int empates = 0;

void setup() {
  size(600, 700);
  textSize(20);
  textAlign(CENTER, CENTER);
  reiniciarJogo();
}

void draw() {
  background(255);

  if (telainicial) {
    desenharTelainicial();
  } else {
    desenharJogo();
  }
}

void desenharTelainicial() {
  fill(0);
  textSize(32);
  text("Jogo da Velha", width/2, 100);
  textSize(24);
  text("Selecione o modo de jogo:", width/2, 180);

  fill(200);
  rect(width/2 - 150, 250, 300, 60, 10);
  fill(0);
  text("1 Jogador (vs Computador)", width/2, 280);
```

```

fill(200);
rect(width/2 - 150, 350, 300, 60, 10);
fill(0);
text("2 Jogadores", width/2, 380);
}

```

```

void desenharJogo() {
    // Desenha o placar
    fill(0);
    text("X: " + vitoriasX + " O: " + vitoriasO + " Empates: " + empates, width/2, 15);

    // Desenha o tabuleiro
    strokeWeight(4);
    line(TAM_CELULA, MARGEM, TAM_CELULA, TAMANHO - MARGEM);
    line(TAM_CELULA * 2, MARGEM, TAM_CELULA * 2, TAMANHO - MARGEM);
    line(MARGEM, TAM_CELULA, TAMANHO - MARGEM, TAM_CELULA);
    line(MARGEM, TAM_CELULA * 2, TAMANHO - MARGEM, TAM_CELULA * 2);

    // Desenha os X e O no tabuleiro
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            float x = j * TAM_CELULA + TAM_CELULA / 2;
            float y = i * TAM_CELULA + TAM_CELULA / 2;

            if (tabuleiro[i][j] == 'X') {
                drawX(x, y);
            } else if (tabuleiro[i][j] == 'O') {
                drawO(x, y);
            }
        }
    }
}

```

```

    }
}

```

```

// Verifica se há um vencedor ou empate

```

```

char vencedor = verificarVencedor();

```

```

boolean jogoTerminou = (vencedor != ' ' || tabuleiroCompleto());

```

```

if (jogoTerminou && jogoAtivo) {

```

```

    jogoAtivo = false;

```

```

    if (!resultadoProcessado) {

```

```

        if (vencedor == 'X') {

```

```

            vitoriasX++;

```

```

        } else if (vencedor == 'O') {

```

```

            vitoriasO++;

```

```

        } else {

```

```

            empates++;

```

```

        }

```

```

        resultadoProcessado = true;

```

```

    }

```

```

}

```

```

// Exibe mensagem de resultado se o jogo terminou

```

```

if (!jogoAtivo) {

```

```

    fill(255, 200);

```

```

    rect(MARGEM, TAMANHO/2 - 30, TAMANHO - 2*MARGEM, 60);

```

```

    fill(0);

```

```

    String mensagem = "";

```

```

    if (vencedor == 'X') {

```

```

        mensagem = "X venceu!";

```

```

    } else if (vencedor == 'O') {
        mensagem = "O venceu!";
    } else {
        mensagem = "Empate!";
    }
    text(mensagem, TAMANHO/2, TAMANHO/2);

    // Botão para reiniciar
    fill(200);
    rect(TAMANHO/2 - 50, TAMANHO/2 + 50, 100, 40);
    fill(0);
    text("Reiniciar", TAMANHO/2, TAMANHO/2 + 70);

    // Botão para voltar ao menu
    fill(200);
    rect(TAMANHO/2 - 50, TAMANHO/2 + 100, 100, 40);
    fill(0);
    text("Menu", TAMANHO/2, TAMANHO/2 + 120);
}

// Se for vez do computador e o jogo está ativo
if (vsComputador && !vezX && jogoAtivo) {
    jogadaComputador();
    vezX = true;
}
}

void mousePressed() {
    if (telaInicial) {
        // Verifica clique na tela inicial

```

```

if (mouseX >= width/2 - 150 && mouseX <= width/2 + 150) {
    if (mouseY >= 250 && mouseY <= 310) {
        // Modo 1 jogador
        vsComputador = true;
        telaInicial = false;
        jogoAtivo = true;
        reiniciarJogo();
    } else if (mouseY >= 350 && mouseY <= 410) {
        // Modo 2 jogadores
        vsComputador = false;
        telaInicial = false;
        jogoAtivo = true;
        reiniciarJogo();
    }
}
} else {
    // Verifica clique no botão de reiniciar
    if (!jogoAtivo && mouseX >= TAMANHO/2 - 50 && mouseX <= TAMANHO/2 + 50) {
        if (mouseY >= TAMANHO/2 + 50 && mouseY <= TAMANHO/2 + 90) {
            reiniciarJogo();
            return;
        } else if (mouseY >= TAMANHO/2 + 100 && mouseY <= TAMANHO/2 + 140) {
            // Voltar ao menu - Zera os contadores
            resetarContadores();
            telaInicial = true;
            return;
        }
    }
}

// Se o jogo não está ativo, não processa cliques no tabuleiro

```

```

if (!jogoAtivo) return;

// Verifica clique no tabuleiro
if (mouseX >= MARGEM && mouseX < TAMANHO - MARGEM &&
    mouseY >= MARGEM && mouseY < TAMANHO - MARGEM) {

    int coluna = (mouseX - MARGEM) / TAM_CELULA;
    int linha = (mouseY - MARGEM) / TAM_CELULA;

    // Verifica se a célula está vazia
    if (tabuleiro[linha][coluna] == ' ') {
        tabuleiro[linha][coluna] = vezX ? 'X' : 'O';
        vezX = !vezX;
    }
}

// Função para resetar os contadores
void resetarContadores() {
    vitoriasX = 0;
    vitoriasO = 0;
    empates = 0;
}

void drawX(float x, float y) {
    stroke(255, 0, 0);
    line(x - TAM_CELULA/3, y - TAM_CELULA/3, x + TAM_CELULA/3, y +
TAM_CELULA/3);

```



```

    line(x + TAM_CELULA/3, y - TAM_CELULA/3, x - TAM_CELULA/3, y +
TAM_CELULA/3);
    stroke(0);
}

```

```

void drawO(float x, float y) {
    stroke(0, 0, 255);
    noFill();
    ellipse(x, y, TAM_CELULA/1.5, TAM_CELULA/1.5);
    stroke(0);
}

```

```

char verificarVencedor() {
    // Verifica linhas
    for (int i = 0; i < 3; i++) {
        if (tabuleiro[i][0] != ' ' && tabuleiro[i][0] == tabuleiro[i][1] && tabuleiro[i][1] ==
tabuleiro[i][2]) {
            return tabuleiro[i][0];
        }
    }
}

```

```

// Verifica colunas
for (int j = 0; j < 3; j++) {
    if (tabuleiro[0][j] != ' ' && tabuleiro[0][j] == tabuleiro[1][j] && tabuleiro[1][j] ==
tabuleiro[2][j]) {
        return tabuleiro[0][j];
    }
}
}

```

```

// Verifica diagonais

```

```

        if (tabuleiro[0][0] != ' ' && tabuleiro[0][0] == tabuleiro[1][1] && tabuleiro[1][1] ==
tabuleiro[2][2]) {
            return tabuleiro[0][0];
        }
        if (tabuleiro[0][2] != ' ' && tabuleiro[0][2] == tabuleiro[1][1] && tabuleiro[1][1] ==
tabuleiro[2][0]) {
            return tabuleiro[0][2];
        }

        return ' '; // Sem vencedor
    }

```

```

boolean tabuleiroCompleto() {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (tabuleiro[i][j] == ' ') {
                return false;
            }
        }
    }
    return true;
}

```

```

void reiniciarJogo() {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            tabuleiro[i][j] = ' ';
        }
    }
    vezX = true;
}

```

```
jogoAtivo = true;
resultadoProcessado = false;
}
```

```
void jogadaComputador() {
    // Primeiro verifica se pode vencer
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (tabuleiro[i][j] == ' ') {
                tabuleiro[i][j] = 'O';
                if (verificarVencedor() == 'O') {
                    return;
                }
                tabuleiro[i][j] = ' ';
            }
        }
    }

    // Depois verifica se pode bloquear o jogador
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (tabuleiro[i][j] == ' ') {
                tabuleiro[i][j] = 'X';
                if (verificarVencedor() == 'X') {
                    tabuleiro[i][j] = 'O';
                    return;
                }
                tabuleiro[i][j] = ' ';
            }
        }
    }
}
```

```
}

// Tenta jogar no centro
if (tabuleiro[1][1] == ' ') {
    tabuleiro[1][1] = 'O';
    return;
}

// Tenta jogar em um canto vazio
int[][] cantos = {{0, 0}, {0, 2}, {2, 0}, {2, 2}};
for (int[] canto : cantos) {
    if (tabuleiro[canto[0]][canto[1]] == ' ') {
        tabuleiro[canto[0]][canto[1]] = 'O';
        return;
    }
}

// Joga em qualquer posição vazia
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
        if (tabuleiro[i][j] == ' ') {
            tabuleiro[i][j] = 'O';
            return;
        }
    }
}
}
```

## **CONCLUSÃO**

A realização deste trabalho foi uma oportunidade valiosa para aprofundar os conhecimentos em programação e desenvolvimento de jogos. A experiência de trabalhar em equipe e explorar ferramentas como Eclipse e Processing trouxe aprendizados importantes, especialmente no que diz respeito à adaptação a novas tecnologias e à implementação de jogos interativos.

Apesar do tempo limitado para adicionar diferenciais aos jogos desenvolvidos, o projeto cumpriu seu propósito de proporcionar uma experiência prática no desenvolvimento de games. Além disso, serviu como um primeiro contato mais aprofundado com a lógica envolvida na criação de jogos, podendo ser utilizado como base para projetos futuros. O conhecimento adquirido certamente contribuirá para desafios acadêmicos e profissionais na área de tecnologia.

## REFERÊNCIA.

PROCESSING. *Reference*. Disponível em: <https://processing.org/reference>. Acesso em: 26 mar. 2025.

ORACLE. *Java Platform, Standard Edition Documentation*. Disponível em: <https://docs.oracle.com/en/java/javase>. Acesso em: 26 mar. 2025.

TECNO FÁCIL. **COMO FAZER A CAPA ABNT! FÁCIL E RÁPIDO! 2025!** [Vídeo]. YouTube, 5 set. 2022. Disponível em: <https://www.youtube.com/watch?v=DGlnPa8SV2s&list=PLjaie-bhmklKzs07mmdQmsSsb3wljX8DU&index=2>. Acesso em: 24 mar. 2025.

TECNO FÁCIL. **COMO FAZER A FOLHA DE ROSTO ABNT! FÁCIL E RÁPIDO! 2025!** [Vídeo]. YouTube, 11 set. 2022. Disponível em: <https://www.youtube.com/watch?v=EAzYX71KxQU&list=PLjaie-bhmklKzs07mmdQmsSsb3wljX8DU&index=3>. Acesso em: 24 mar. 2025.

TECNO FÁCIL. **COMO FAZER O RESUMO ABNT! FÁCIL E RÁPIDO! 2025!** [Vídeo]. YouTube, 17 set. 2022. Disponível em: <https://www.youtube.com/watch?v=fygY7PtCjIA&list=PLjaie-bhmklKzs07mmdQmsSsb3wljX8DU&index=4>. Acesso em: 24 mar. 2025.

TECNO FÁCIL. **COMO FAZER O SUMÁRIO ABNT! FÁCIL E RÁPIDO! 2025!** [Vídeo]. YouTube, 27 out. 2022. Disponível em: <https://www.youtube.com/watch?v=zkr-Q73s4AI&list=PLjaie-bhmklKzs07mmdQmsSsb3wljX8DU&index=5>. Acesso em: 24 mar. 2025.

TECNO FÁCIL. **COMO FAZER A INTRODUÇÃO ABNT! FÁCIL E RÁPIDO! 2025!** [Vídeo]. YouTube, 7 nov. 2022. Disponível

em: [https://www.youtube.com/watch?v=\\_BZW2bRCodw&list=PLjaie-bhmklKzs07mmdQmsSsb3wljX8DU&index=7](https://www.youtube.com/watch?v=_BZW2bRCodw&list=PLjaie-bhmklKzs07mmdQmsSsb3wljX8DU&index=7). Acesso em: 24 mar. 2025.

TECNO FÁCIL. **COMO FAZER AS REFERÊNCIAS ABNT! FÁCIL E RÁPIDO! 2025!**  
[Vídeo]. YouTube, 15 nov. 2022. Disponível  
em: <https://www.youtube.com/watch?v=43m777TUcEg&list=PLjaie-bhmklKzs07mmdQmsSsb3wljX8DU&index=12>. Acesso em: 24 mar. 2025.