Centro Universitário Maurício de Nassau

Game Studio: Força da Amizade

Márcio da Costa Ferreira Junior - 01596976

Pedro Henrique de Souza Cavalcanti - 01595774

Thaynã Queiroz Mota - 01582087

Yuri Gabriel Nogueira De Lima - 01600961

Trabalhos 1, 2 e 3

(Unidade 1)

Trabalho apresentado ao curso de Ciências da Computação para obtenção de nota da disciplina de Desenvolvimento de Games.
Orientador: Leopoldo Rodrigues

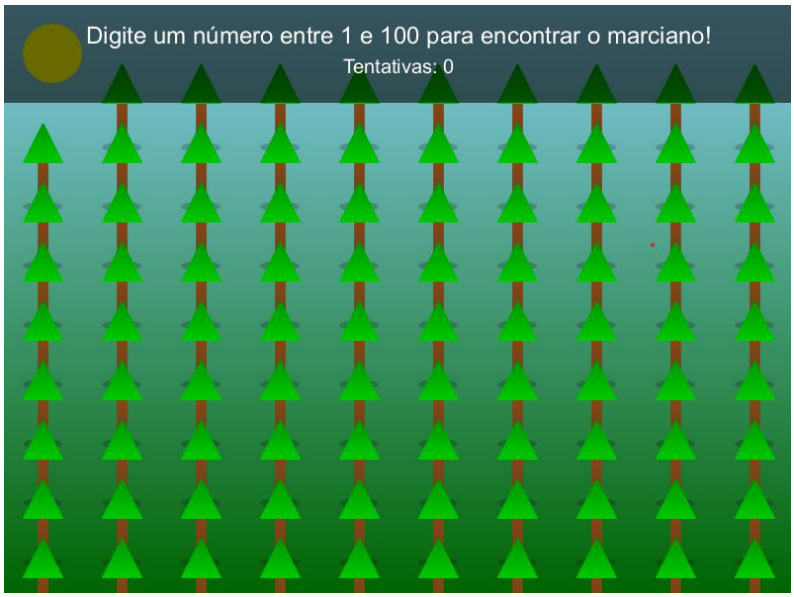
Recife

2025

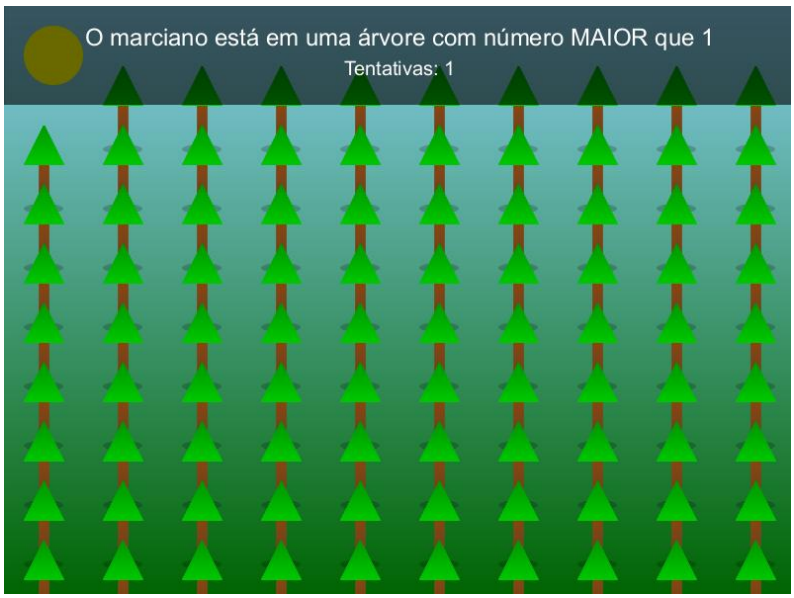
Jogo 1: Jogo do Marciano

Capturas de Tela

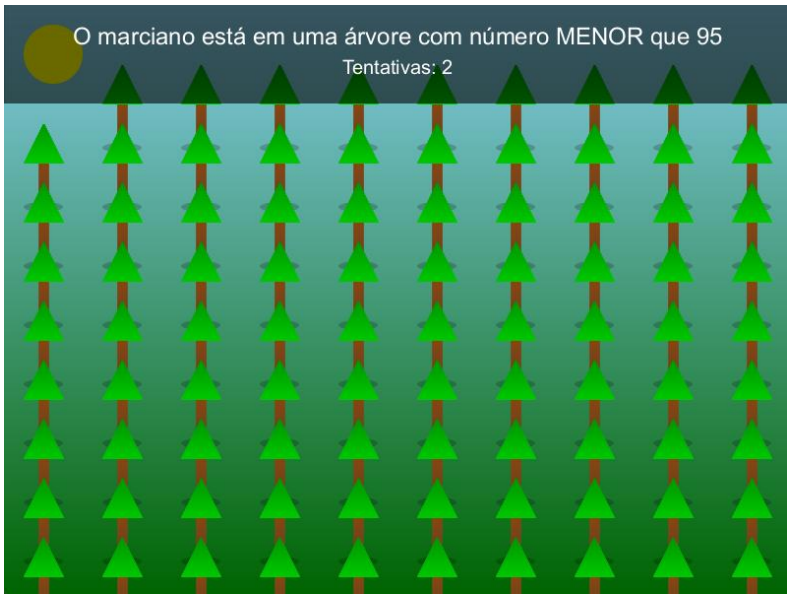
Tela inicial do Jogo do Marciano Tela inicial com as 100 árvores numeradas!



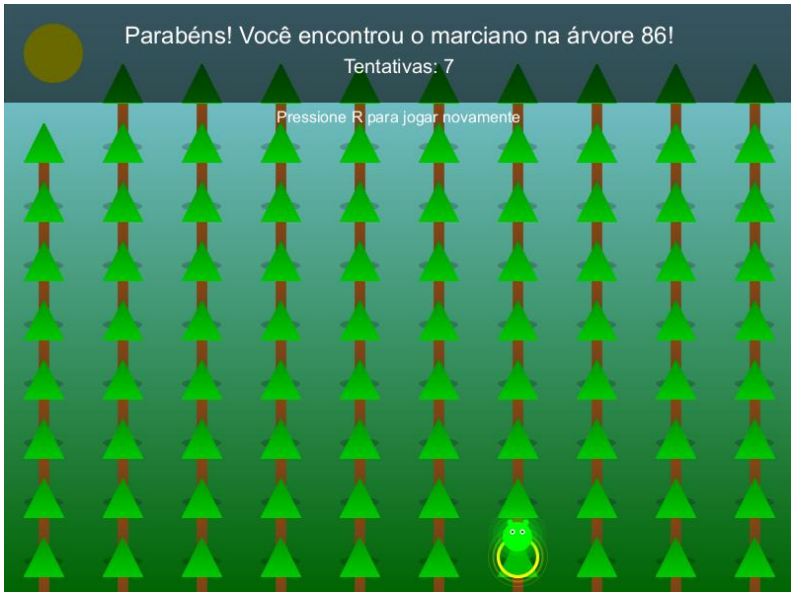
Dica "maior" no Jogo do Marciano Feedback quando o número digitado é menor que a posição do marciano!



Dica "menor" no Jogo do Marciano Feedback quando o número digitado é maior que a posição do marciano!



Marciano encontrado Tela mostrando o marciano encontrado após acertar o número!



Código Fonte

```
// Variáveis globais
int arvoreMarciano;  // Árvore onde o marciano está escondido
int tentativaAtual;  // Número atual que o jogador está tentando
int numeroTentativas = 0;  // Contador de tentativas
```

```
String mensagem = "Digite um número entre 1 e 100 para encontrar o marciano!";
boolean jogoTerminado = false;
PFont fonte;
color corFundo = color(0, 100, 0);
color corArvore = color(0, 200, 0);
color corTronco = color(139, 69, 19);
color corDestaque = color(255, 255, 0);
color corTexto = color(255);

void setup() {
  size(800, 600);
  iniciarJogo();
  fonte = createFont("Arial", 24);
  textFont(fonte);
}

void iniciarJogo() {
  // Gera um número aleatório entre 1 e 100
  arvoreMarciano = int(random(1, 101));
  tentativaAtual = 0;
  numeroTentativas = 0;
  mensagem = "Digite um número entre 1 e 100 para encontrar o marciano!";
  jogoTerminado = false;
}

void draw() {
  // Desenha o céu com gradiente
  for (int i = 0; i < height; i++) {
    float inter = map(i, 0, height, 0, 1);
    color c = lerpColor(color(135, 206, 235), color(0, 100, 0), inter);
    stroke(c);
    line(0, i, width, i);
  }
```

```
  // Desenha o sol
  fill(255, 255, 0);
  noStroke();
  ellipse(50, 50, 60, 60);

  // Desenha as árvores
  desenharArvores();

  // Desenha o marciano se o jogo estiver terminado
  if (jogoTerminado) {
    desenharMarciano(arvoreMarciano);
  }

  // Interface do usuário
  desenharInterface();
}

void desenharInterface() {
  // Painel de informações
  fill(0, 0, 0, 150);
  noStroke();
  rect(0, 0, width, 100);

  // Mensagem principal
  fill(corTexto);
  textAlign(CENTER);
  textSize(24);
  text(mensagem, width/2, 40);

  // Contador de tentativas
  textSize(20);
  text("Tentativas: " + numeroTentativas, width/2, 70);

  // Mostra o número atual sendo digitado
```

```
  if (tentativaAtual > 0) {
    textSize(30);
    text(tentativaAtual, width/2, 100);
  }

  // Mostra instrução para reiniciar
  if (jogoTerminado) {
    textSize(16);
    text("Pressione R para jogar novamente", width/2, 120);
  }
}

void desenharArvores() {
  for (int i = 1; i <= 100; i++) {
    float x = (i % 10) * 80 + 40;
    float y = (i / 10) * 60 + 100;

    // Desenha sombra da árvore
    fill(0, 0, 0, 50);
    noStroke();
    ellipse(x, y + 45, 40, 10);

    // Desenha o tronco com gradiente
    for (int j = 0; j < 40; j++) {
      float inter = map(j, 0, 40, 0, 1);
      color c = lerpColor(color(139, 69, 19), color(101, 67, 33), inter);
      stroke(c);
      line(x - 5, y + j, x + 5, y + j);
    }

    // Desenha a copa da árvore com gradiente
    for (int j = 0; j < 40; j++) {
      float inter = map(j, 0, 40, 0, 1);
      color c = lerpColor(color(0, 200, 0), color(0, 150, 0), inter);
```

```
      stroke(c);
      line(x - 20 + j/2, y - j, x + 20 - j/2, y - j);
    }

    // Número da árvore
    fill(255);
    textSize(12);
    textAlign(CENTER);
    text(i, x, y + 60);

    // Se o jogo estiver terminado e esta for a árvore correta, destaca-a
    if (jogoTerminado && i == arvoreMarciano) {
      stroke(corDestaque);
      strokeWeight(3);
      noFill();
      ellipse(x, y - 20, 40, 40);

      // Efeito de brilho
      for (int j = 0; j < 3; j++) {
        stroke(corDestaque, 100 - j * 30);
        strokeWeight(3 - j);
        ellipse(x, y - 20, 40 + j * 10, 40 + j * 10);
      }
    }
  }
}

void desenharMarciano(int arvore) {
  float x = (arvore % 10) * 80 + 40;
  float y = (arvore / 10) * 60 + 60;

  // Efeito de brilho ao redor do marciano
  for (int i = 0; i < 3; i++) {
    fill(0, 255, 0, 50 - i * 15);
```

```
  noStroke();
  ellipse(x, y, 40 + i * 10, 40 + i * 10);
}

// Corpo do marciano
fill(0, 255, 0);
noStroke();
ellipse(x, y, 30, 30);

// Olhos
fill(255);
ellipse(x - 5, y - 5, 5, 5);
ellipse(x + 5, y - 5, 5, 5);

// Pupilas
fill(0);
ellipse(x - 5, y - 5, 2, 2);
ellipse(x + 5, y - 5, 2, 2);

// Antenas
stroke(0, 255, 0);
strokeWeight(2);
line(x - 8, y - 15, x - 8, y - 8);
line(x + 8, y - 15, x + 8, y - 8);

// Bolinhas nas antenas
fill(0, 255, 0);
ellipse(x - 8, y - 15, 4, 4);
ellipse(x + 8, y - 15, 4, 4);

// Efeito de brilho nas antenas
for (int i = 0; i < 2; i++) {
  stroke(0, 255, 0, 100 - i * 50);
  strokeWeight(1);
```

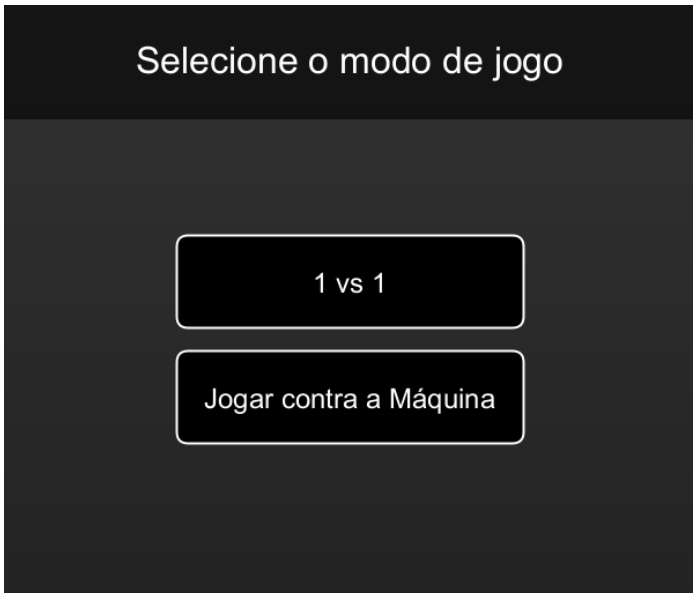
```
    line(x - 8, y - 15 - i * 2, x - 8, y - 8);
    line(x + 8, y - 15 - i * 2, x + 8, y - 8);
  }
}

void keyPressed() {
  if (key == 'r' || key == 'R') {
    iniciarJogo();
    return;
  }

  if (jogoTerminado) return;

  if (key >= '0' && key <= '9') {
    String numero = "";
    if (tentativaAtual > 0) {
      numero = str(tentativaAtual) + key;
    } else {
      numero = str(key);
    }
    tentativaAtual = int(numero);

    if (tentativaAtual > 100) {
      tentativaAtual = 100;
    }
  } else if (key == ENTER) {
    numeroTentativas++;

    if (tentativaAtual == arvoreMarciano) {
      mensagem = "Parabéns! Você encontrou o marciano na árvore " + arvoreMarciano + "!";
      jogoTerminado = true;
    } else if (tentativaAtual < arvoreMarciano) {
      mensagem = "O marciano está em uma árvore com número MAIOR que " + tentativaAtual;
    } else {
```

```
      mensagem = "O marciano está em uma árvore com número MENOR que " + tentativaAtual;
    }

    tentativaAtual = 0;
  } else if (key == BACKSPACE) {
    tentativaAtual = 0;
    mensagem = "Digite um número entre 1 e 100 para encontrar o marciano!";
  }
}
```

Jogo 2: Jogo da Velha (Tic-tac-toe)

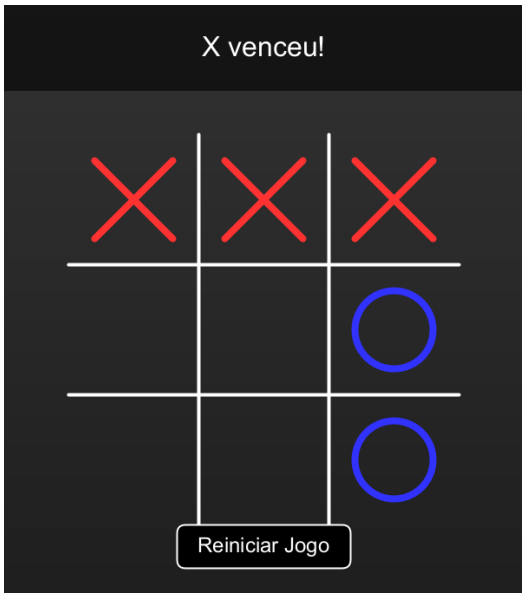
Capturas de Tela

Menu inicial do Jogo da VelhaTela do menu com opções de 1 vs 1 e Jogar contra a Máquina!

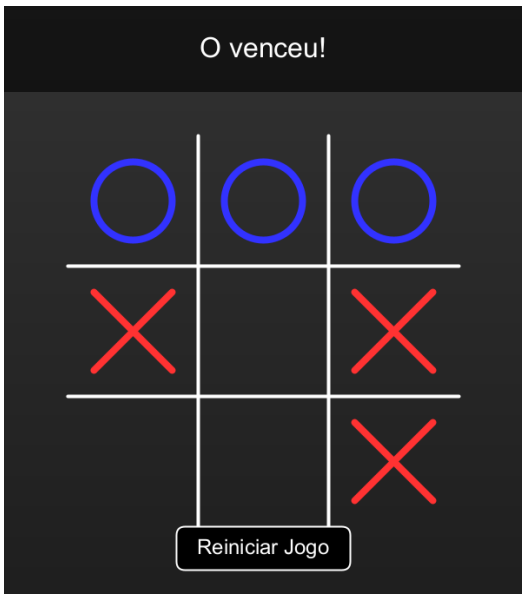


Partida em andamento / Tabuleiro com jogadas em andamento!

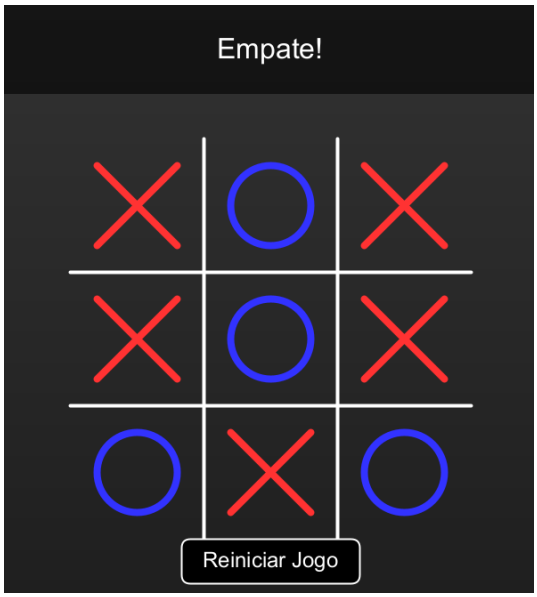
Vitória do X Tela mostrando a vitória do jogador X!



Vitória do O Tela mostrando a vitória do jogador O!



Empate Tela mostrando um empate!



Código Fonte

```
// Variáveis globais
int[][] tabuleiro = new int[3][3];  // 0 = vazio, 1 = X, 2 = O
boolean vezDoX = true;  // true = vez do X, false = vez do O
boolean jogoTerminado = false;
String mensagem = "Selecione o modo de jogo";
PFont fonte;
color corX = color(255, 50, 50);    // Vermelho
color corO = color(50, 50, 255);    // Azul
color corLinha = color(255, 255, 255);  // Branco
color corFundo = color(30, 30, 30);     // Cinza escuro
boolean modoComputador = false;  // true = modo contra computador, false = modo 2 jogadores
boolean menuAtivo = true;  // true = menu de seleção, false = jogo em andamento

void setup() {
  size(600, 700);
  fonte = createFont("Arial", 32);
  textFont(fonte);
  iniciarJogo();
```

```
}

void iniciarJogo() {
  // Limpa o tabuleiro
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      tabuleiro[i][j] = 0;
    }
  }
  vezDoX = true;
  jogoTerminado = false;
  menuAtivo = true;
  mensagem = "Selecione o modo de jogo";
}

void draw() {
  // Desenha o fundo com gradiente
  for (int i = 0; i < height; i++) {
    float inter = map(i, 0, height, 0, 1);
    color c = lerpColor(color(50, 50, 50), color(30, 30, 30), inter);
    stroke(c);
    line(0, i, width, i);
  }

  // Desenha o painel de informações
  fill(0, 0, 0, 150);
  noStroke();
  rect(0, 0, width, 100);

  // Desenha a mensagem
  fill(255);
  textAlign(CENTER);
  textSize(32);
  text(mensagem, width/2, 60);
```

```
  if (menuAtivo) {
    desenharMenu();
  } else {
    // Desenha o tabuleiro
    desenharTabuleiro();

    // Desenha o botão de reiniciar
    desenharBotaoReiniciar();

    // Se for modo computador e for a vez do O, faz a jogada
    if (modoComputador && !vezDoX && !jogoTerminado) {
      fazerJogadaComputador();
    }
  }
}

void desenharMenu() {
  float x = width/2 - 150;
  float y = 200;
  float largura = 300;
  float altura = 80;

  // Botão Modo 2 Jogadores
  fill(0);  // Fundo preto
  stroke(255);
  strokeWeight(2);
  rect(x, y, largura, altura, 10);

  fill(255);  // Texto branco
  textAlign(CENTER);
  textSize(24);
  text("1 vs 1", width/2, y + 50);
```

```
  // Botão Modo Computador
  y += 100;
  fill(0);  // Fundo preto
  rect(x, y, largura, altura, 10);
  fill(255);  // Texto branco
  text("Jogar contra a Máquina", width/2, y + 50);
}

void desenharTabuleiro() {
  float tamanho = 150;  // Tamanho de cada célula
  float inicioX = (width - tamanho * 3) / 2;
  float inicioY = 150;

  // Desenha as linhas do tabuleiro
  stroke(corLinha);
  strokeWeight(4);

  // Linhas horizontais
  for (int i = 1; i < 3; i++) {
    line(inicioX, inicioY + i * tamanho, inicioX + 3 * tamanho, inicioY + i * tamanho);
  }

  // Linhas verticais
  for (int i = 1; i < 3; i++) {
    line(inicioX + i * tamanho, inicioY, inicioX + i * tamanho, inicioY + 3 * tamanho);
  }

  // Desenha os X's e O's
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      float x = inicioX + j * tamanho;
      float y = inicioY + i * tamanho;

      if (tabuleiro[i][j] == 1) {
```

```
        desenharX(x, y, tamanho);
      } else if (tabuleiro[i][j] == 2) {
        desenharO(x, y, tamanho);
      }
    }
  }
}

void desenharX(float x, float y, float tamanho) {
  stroke(corX);
  strokeWeight(8);
  float margem = tamanho * 0.2;
  line(x + margem, y + margem, x + tamanho - margem, y + tamanho - margem);
  line(x + margem, y + tamanho - margem, x + tamanho - margem, y + margem);
}

void desenharO(float x, float y, float tamanho) {
  stroke(corO);
  strokeWeight(8);
  noFill();
  float margem = tamanho * 0.2;
  ellipse(x + tamanho/2, y + tamanho/2, tamanho - 2*margem, tamanho - 2*margem);
}

void desenharBotaoReiniciar() {
  float x = width/2 - 100;
  float y = height - 100;
  float largura = 200;
  float altura = 50;

  // Desenha o botão
  fill(0);  // Fundo preto
  stroke(255);
  strokeWeight(2);
```

```
  rect(x, y, largura, altura, 10);

  // Desenha o texto
  fill(255);  // Texto branco
  textAlign(CENTER);
  textSize(24);
  text("Reiniciar Jogo", width/2, y + 32);
}

void mousePressed() {
  if (menuAtivo) {
    float x = width/2 - 150;
    float y = 200;
    float largura = 300;
    float altura = 80;

    // Verifica clique no botão de 2 jogadores
    if (mouseX >= x && mouseX <= x + largura &&
        mouseY >= y && mouseY <= y + altura) {
      modoComputador = false;
      menuAtivo = false;
      mensagem = "Vez do jogador X";
    }

    // Verifica clique no botão de modo computador
    y += 100;
    if (mouseX >= x && mouseX <= x + largura &&
        mouseY >= y && mouseY <= y + altura) {
      modoComputador = true;
      menuAtivo = false;
      mensagem = "Vez do jogador X";
    }
    return;
  }
```

```
  if (jogoTerminado) {
    // Verifica se clicou no botão de reiniciar
    float x = width/2 - 100;
    float y = height - 100;
    if (mouseX >= x && mouseX <= x + 200 && mouseY >= y && mouseY <= y + 50) {
      iniciarJogo();
    }
    return;
  }

  // Se for modo computador e for a vez do O, não permite jogada
  if (modoComputador && !vezDoX) {
    return;
  }

  // Calcula a posição do clique no tabuleiro
  float tamanho = 150;
  float inicioX = (width - tamanho * 3) / 2;
  float inicioY = 150;

  int coluna = int((mouseX - inicioX) / tamanho);
  int linha = int((mouseY - inicioY) / tamanho);

  // Verifica se o clique foi dentro do tabuleiro
  if (linha >= 0 && linha < 3 && coluna >= 0 && coluna < 3) {
    // Verifica se a posição está vazia
    if (tabuleiro[linha][coluna] == 0) {
      fazerJogada(linha, coluna);
    }
  }
}

void fazerJogada(int linha, int coluna) {
```

```
  // Faz a jogada
  tabuleiro[linha][coluna] = vezDoX ? 1 : 2;

  // Verifica se o jogo terminou
  if (verificarVitoria()) {
    jogoTerminado = true;
    mensagem = (vezDoX ? "X" : "O") + " venceu!";
  } else if (verificarEmpate()) {
    jogoTerminado = true;
    mensagem = "Empate!";
  } else {
    // Passa a vez
    vezDoX = !vezDoX;
    mensagem = "Vez do jogador " + (vezDoX ? "X" : "O");
  }
}

void fazerJogadaComputador() {
  // Procura uma jogada vencedora
  int[] jogada = encontrarJogadaVencedora(2);
  if (jogada != null) {
    fazerJogada(jogada[0], jogada[1]);
    return;
  }

  // Bloqueia jogada vencedora do oponente
  jogada = encontrarJogadaVencedora(1);
  if (jogada != null) {
    fazerJogada(jogada[0], jogada[1]);
    return;
  }

  // Tenta jogar no centro
  if (tabuleiro[1][1] == 0) {
```

```
    fazerJogada(1, 1);
    return;
  }

  // Joga em uma posição aleatória
  ArrayList<int[]> posicoesVazias = new ArrayList<int[]>();
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      if (tabuleiro[i][j] == 0) {
        posicoesVazias.add(new int[]{i, j});
      }
    }
  }

  if (!posicoesVazias.isEmpty()) {
    int[] posicao = posicoesVazias.get(int(random(posicoesVazias.size())));
    fazerJogada(posicao[0], posicao[1]);
  }
}

int[] encontrarJogadaVencedora(int jogador) {
  // Verifica linhas
  for (int i = 0; i < 3; i++) {
    int[] jogada = verificarLinha(i, jogador);
    if (jogada != null) return jogada;
  }

  // Verifica colunas
  for (int j = 0; j < 3; j++) {
    int[] jogada = verificarColuna(j, jogador);
    if (jogada != null) return jogada;
  }

  // Verifica diagonais
```

```
  int[] jogada = verificarDiagonal(jogador);
  if (jogada != null) return jogada;

  return null;
}

int[] verificarLinha(int linha, int jogador) {
  int vazios = 0;
  int[] posicaoVazia = null;

  for (int j = 0; j < 3; j++) {
    if (tabuleiro[linha][j] == jogador) {
      vazios++;
    } else if (tabuleiro[linha][j] == 0) {
      posicaoVazia = new int[]{linha, j};
    }
  }

  if (vazios == 2 && posicaoVazia != null) {
    return posicaoVazia;
  }

  return null;
}

int[] verificarColuna(int coluna, int jogador) {
  int vazios = 0;
  int[] posicaoVazia = null;

  for (int i = 0; i < 3; i++) {
    if (tabuleiro[i][coluna] == jogador) {
      vazios++;
    } else if (tabuleiro[i][coluna] == 0) {
      posicaoVazia = new int[]{i, coluna};
```

```
    }
  }

  if (vazios == 2 && posicaoVazia != null) {
    return posicaoVazia;
  }

  return null;
}

int[] verificarDiagonal(int jogador) {
  // Diagonal principal
  int vazios = 0;
  int[] posicaoVazia = null;

  for (int i = 0; i < 3; i++) {
    if (tabuleiro[i][i] == jogador) {
      vazios++;
    } else if (tabuleiro[i][i] == 0) {
      posicaoVazia = new int[]{i, i};
    }
  }

  if (vazios == 2 && posicaoVazia != null) {
    return posicaoVazia;
  }

  // Diagonal secundária
  vazios = 0;
  posicaoVazia = null;

  for (int i = 0; i < 3; i++) {
    if (tabuleiro[i][2-i] == jogador) {
      vazios++;
```

```
    } else if (tabuleiro[i][2-i] == 0) {
      posicaoVazia = new int[]{i, 2-i};
    }
  }

  if (vazios == 2 && posicaoVazia != null) {
    return posicaoVazia;
  }

  return null;
}

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

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

  // Verifica diagonais
  if (tabuleiro[0][0] != 0 &&
      tabuleiro[0][0] == tabuleiro[1][1] &&
```

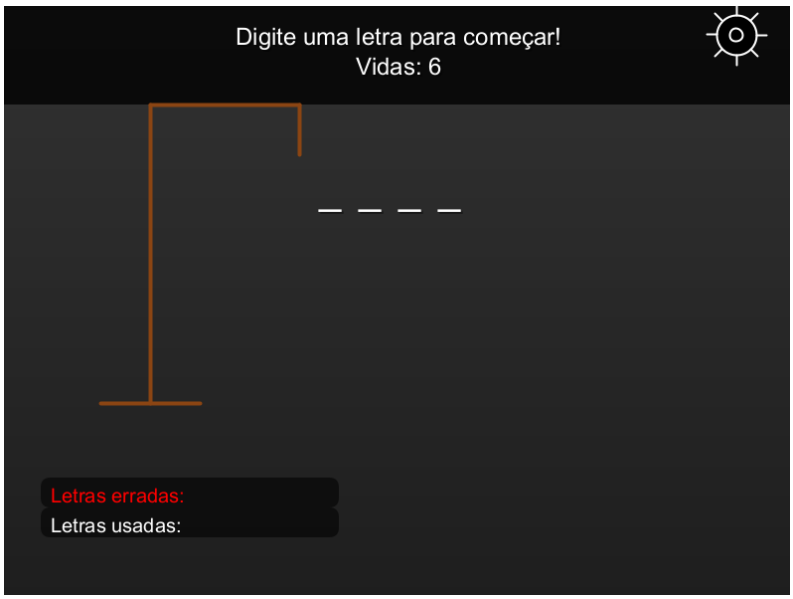
```
      tabuleiro[1][1] == tabuleiro[2][2]) {
    return true;
  }

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

  return false;
}

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

Jogo 3: Jogo da Forca

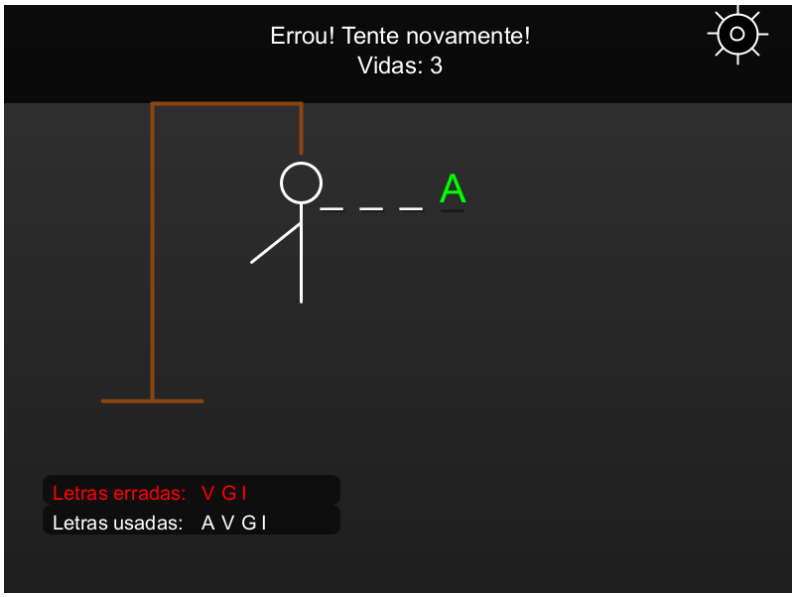
Capturas de Tela

Menu de níveis do Jogo da Forca Tela de seleção de nível (Fácil, Médio, Difícil)!

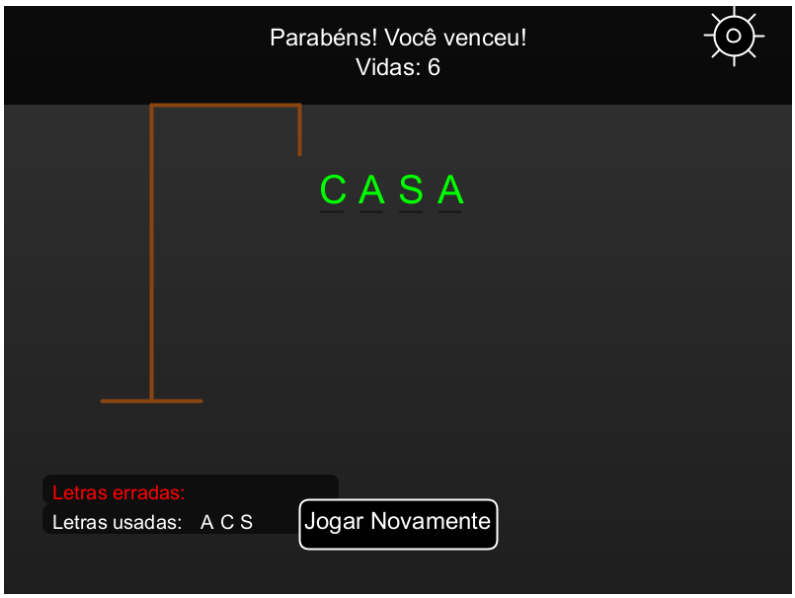
Jogo da Forca – Início Tela inicial com a palavra oculta e forca vazia!



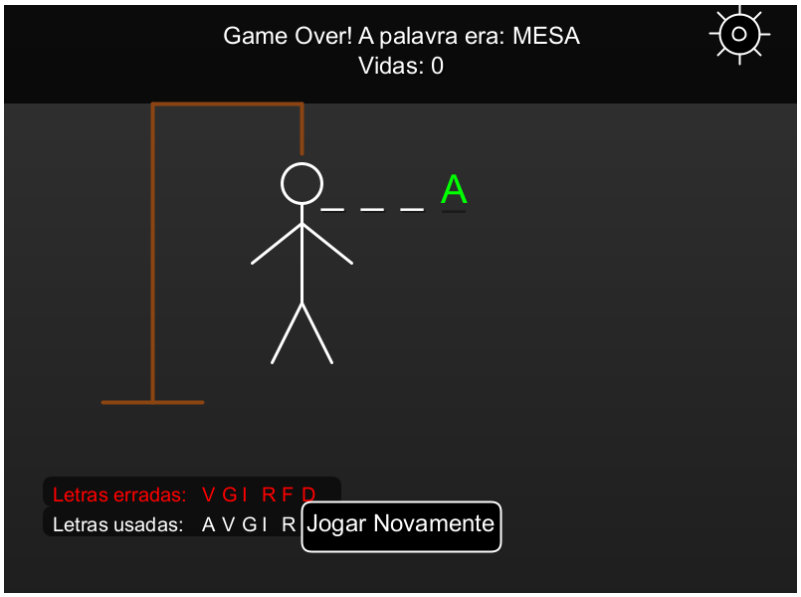
Jogo em andamento Tela com algumas letras descobertas e boneco parcialmente desenhado!



Vitória no Jogo da Forca Tela de vitória com a palavra completa!



Derrota no Jogo da Forca Tela de derrota com a palavra revelada e boneco completo!



Código Fonte

```
// Variáveis globais
String[][] palavras = {
  // Nível Fácil (palavras curtas e comuns)
  {
    "CASA", "BOLA", "GATO", "CACHORRO", "LIVRO",
    "MESA", "CADEIRA", "JANELA", "PORTA", "ÁRVORE",
    "FLOR", "SOL", "LUA", "MAR", "RIO"
  },
  // Nível Médio (palavras médias e um pouco mais complexas)
  {
    "COMPUTADOR", "INTERNET", "PROGRAMACAO", "ALGORITMO",
"DESENVOLVIMENTO",
    "APLICACAO", "SOFTWARE", "HARDWARE", "SISTEMA", "REDE",
    "DADOS", "INFORMATICA", "TECNOLOGIA", "PROCESSAMENTO",
"INFORMACAO"
  },
  // Nível Difícil (palavras longas e mais complexas)
  {
```

```
    "DESENVOLVIMENTO", "PROGRAMACAO", "APLICACAO", "COMPUTADOR",
"ALGORITMO",
    "INTERFACE", "SOFTWARE", "HARDWARE", "SISTEMA", "REDE",
    "DADOS", "INFORMATICA", "TECNOLOGIA", "PROCESSAMENTO",
"INFORMACAO"
  }
};

String palavra;  // Palavra a ser adivinhada
String palavraAtual = "";      // Palavra com as letras descobertas
ArrayList<Character> letrasUsadas = new ArrayList<Character>();
ArrayList<Character> letrasErradas = new ArrayList<Character>();
int vidas = 6;                 // Número de vidas
boolean jogoTerminado = false;
String mensagem = "Escolha um nível para começar!";
PFont fonte;
color corForca = color(139, 69, 19);    // Marrom
color corLetra = color(255, 255, 255);  // Branco
color corLetraErrada = color(255, 0, 0); // Vermelho
color corLetraCerta = color(0, 255, 0);  // Verde
int nivelAtual = -1;  // -1 significa que nenhum nível foi escolhido

void setup() {
  size(800, 600);
  fonte = createFont("Arial", 32);
  textFont(fonte);
}

void iniciarJogo() {
  // Escolhe uma palavra aleatória do nível atual
  palavra = palavras[nivelAtual][int(random(palavras[nivelAtual].length))];

  // Inicializa a palavra atual com underscores
  palavraAtual = "";
  for (int i = 0; i < palavra.length(); i++) {
```

```
    palavraAtual += "_";
  }

  // Limpa as listas de letras
  letrasUsadas.clear();
  letrasErradas.clear();

  // Reseta as vidas e o estado do jogo
  vidas = 6;
  jogoTerminado = false;
  mensagem = "Digite uma letra para começar!";
}

void draw() {
  // Desenha o fundo com gradiente
  for (int i = 0; i < height; i++) {
    float inter = map(i, 0, height, 0, 1);
    color c = lerpColor(color(50, 50, 50), color(30, 30, 30), inter);
    stroke(c);
    line(0, i, width, i);
  }

  // Desenha o painel de informações com efeito de sombra
  fill(0, 0, 0, 200);
  noStroke();
  rect(0, 0, width, 100);

  // Desenha a mensagem com sombra
  fill(0, 0, 0, 100);
  textAlign(CENTER);
  textSize(24);
  text(mensagem, width/2 + 2, 42);

  fill(255);
```

```
  text(mensagem, width/2, 40);

  // Desenha as vidas restantes com sombra
  fill(0, 0, 0, 100);
  text("Vidas: " + vidas, width/2 + 2, 72);

  fill(255);
  text("Vidas: " + vidas, width/2, 70);

  // Desenha o botão de engrenagem (sempre visível durante o jogo)
  if (nivelAtual != -1) {
    desenharBotaoEngrenagem();
  }

  // Se nenhum nível foi escolhido, desenha os botões de nível
  if (nivelAtual == -1) {
    desenharBotoesNivel();
  } else {
    // Desenha a forca
    desenharForca();

    // Desenha a palavra
    desenharPalavra();

    // Desenha as letras usadas
    desenharLetrasUsadas();

    // Se o jogo terminou, desenha o botão de reiniciar
    if (jogoTerminado) {
      desenharBotaoReiniciar();
    }
  }
}
```

```
void desenharBotoesNivel() {
  float y = height/2 - 100;
  float largura = 200;
  float altura = 50;
  float espaco = 20;

  // Botão Fácil
  float x = width/2 - largura/2;
  if (mouseX >= x && mouseX <= x + largura && mouseY >= y && mouseY <= y + altura) {
    fill(50);
  } else {
    fill(0);
  }
  stroke(255);
  strokeWeight(2);
  rect(x, y, largura, altura, 10);

  // Botão Médio
  if (mouseX >= x && mouseX <= x + largura && mouseY >= y + altura + espaco && mouseY
<= y + altura * 2 + espaco) {
    fill(50);
  } else {
    fill(0);
  }
  rect(x, y + altura + espaco, largura, altura, 10);

  // Botão Difícil
  if (mouseX >= x && mouseX <= x + largura && mouseY >= y + (altura + espaco) * 2 &&
mouseY <= y + (altura + espaco) * 3) {
    fill(50);
  } else {
    fill(0);
  }
  rect(x, y + (altura + espaco) * 2, largura, altura, 10);
```

```
  // Textos dos botões
  fill(255);
  textAlign(CENTER);
  textSize(24);
  text("Nível Fácil", width/2, y + 32);
  text("Nível Médio", width/2, y + altura + espaco + 32);
  text("Nível Difícil", width/2, y + (altura + espaco) * 2 + 32);
}

void desenharForca() {
  float x = 150;  // Movido mais para a esquerda
  float y = 400;

  stroke(corForca);
  strokeWeight(4);

  // Base da forca
  line(x - 50, y, x + 50, y);

  // Poste vertical
  line(x, y, x, y - 300);

  // Topo da forca
  line(x, y - 300, x + 150, y - 300);

  // Corda
  line(x + 150, y - 300, x + 150, y - 250);

  // Desenha o boneco se houver erros
  if (letrasErradas.size() > 0) {
    // Cabeça
    noFill();
    stroke(corLetra);
    strokeWeight(3);
```

```
    ellipse(x + 150, y - 220, 40, 40);

    if (letrasErradas.size() > 1) {
      // Corpo
      line(x + 150, y - 200, x + 150, y - 100);
    }

    if (letrasErradas.size() > 2) {
      // Braço esquerdo
      line(x + 150, y - 180, x + 100, y - 140);
    }

    if (letrasErradas.size() > 3) {
      // Braço direito
      line(x + 150, y - 180, x + 200, y - 140);
    }

    if (letrasErradas.size() > 4) {
      // Perna esquerda
      line(x + 150, y - 100, x + 120, y - 40);
    }

    if (letrasErradas.size() > 5) {
      // Perna direita
      line(x + 150, y - 100, x + 180, y - 40);
    }
  }
}

void desenharPalavra() {
  float x = width/2 - (palavra.length() * 20);
  float y = 200;

  textAlign(LEFT);
```

```
  textSize(40);

  // Adiciona sombra ao texto
  fill(0, 0, 0, 100);
  for (int i = 0; i < palavraAtual.length(); i++) {
    text("_", x + i * 40 + 2, y + 2);
  }

  for (int i = 0; i < palavraAtual.length(); i++) {
    char letra = palavraAtual.charAt(i);
    if (letra == '_') {
      fill(255);
      text("_", x + i * 40, y);
    } else {
      fill(corLetraCerta);
      text(letra, x + i * 40, y);
    }
  }
}

void desenharLetrasUsadas() {
  float x = 50;
  float y = 500;

  textAlign(LEFT);
  textSize(20);

  // Letras erradas com fundo
  fill(0, 0, 0, 150);
  noStroke();
  rect(x - 10, y - 25, 300, 30, 10);

  fill(corLetraErrada);
  text("Letras erradas: ", x, y);
```

```
  for (int i = 0; i < letrasErradas.size(); i++) {
    text(letrasErradas.get(i), x + 150 + i * 20, y);
  }

  // Letras usadas com fundo
  fill(0, 0, 0, 150);
  noStroke();
  rect(x - 10, y + 5, 300, 30, 10);

  fill(255);
  text("Letras usadas: ", x, y + 30);
  for (int i = 0; i < letrasUsadas.size(); i++) {
    text(letrasUsadas.get(i), x + 150 + i * 20, y + 30);
  }
}

void desenharBotaoEngrenagem() {
  float engrenagemX = width - 60;
  float engrenagemY = 30;
  float engrenagemTamanho = 40;

  // Efeito de hover na engrenagem
  if (mouseX >= engrenagemX - engrenagemTamanho/2 &&
      mouseX <= engrenagemX + engrenagemTamanho/2 &&
      mouseY >= engrenagemY - engrenagemTamanho/2 &&
      mouseY <= engrenagemY + engrenagemTamanho/2) {
    fill(50);
  } else {
    fill(0);
  }

  stroke(255);
  strokeWeight(2);
  ellipse(engrenagemX, engrenagemY, engrenagemTamanho, engrenagemTamanho);
```

```
  // Desenha os dentes da engrenagem
  for (int i = 0; i < 8; i++) {
    float angulo = i * PI / 4;
    float x1 = engrenagemX + cos(angulo) * engrenagemTamanho/2;
    float y1 = engrenagemY + sin(angulo) * engrenagemTamanho/2;
    float x2 = engrenagemX + cos(angulo) * (engrenagemTamanho/2 + 10);
    float y2 = engrenagemY + sin(angulo) * (engrenagemTamanho/2 + 10);
    line(x1, y1, x2, y2);
  }

  // Desenha o centro da engrenagem
  fill(0);
  ellipse(engrenagemX, engrenagemY, engrenagemTamanho/3, engrenagemTamanho/3);
}

void desenharBotaoReiniciar() {
  float x = width/2 - 100;
  float y = height - 100;
  float largura = 200;
  float altura = 50;

  // Efeito de hover
  if (mouseX >= x && mouseX <= x + largura && mouseY >= y && mouseY <= y + altura) {
    fill(50);
  } else {
    fill(0);
  }

  stroke(255);
  strokeWeight(2);
  rect(x, y, largura, altura, 10);

  // Desenha o texto com sombra
```

```
  fill(0, 0, 0, 100);
  textAlign(CENTER);
  textSize(24);
  text("Jogar Novamente", width/2 + 2, y + 32);

  fill(255);
  text("Jogar Novamente", width/2, y + 30);
}

void mousePressed() {
  if (nivelAtual == -1) {
    float y = height/2 - 100;
    float largura = 200;
    float altura = 50;
    float espaco = 20;
    float x = width/2 - largura/2;

    // Verifica clique no botão Fácil
    if (mouseX >= x && mouseX <= x + largura && mouseY >= y && mouseY <= y + altura) {
      nivelAtual = 0;
      iniciarJogo();
    }
    // Verifica clique no botão Médio
    else if (mouseX >= x && mouseX <= x + largura && mouseY >= y + altura + espaco &&
mouseY <= y + altura * 2 + espaco) {
      nivelAtual = 1;
      iniciarJogo();
    }
    // Verifica clique no botão Difícil
    else if (mouseX >= x && mouseX <= x + largura && mouseY >= y + (altura + espaco) * 2
&& mouseY <= y + (altura + espaco) * 3) {
      nivelAtual = 2;
      iniciarJogo();
    }
  } else {
```

```
    // Verifica clique no botão de engrenagem (sempre visível durante o jogo)
    float engrenagemX = width - 60;
    float engrenagemY = 30;
    float engrenagemTamanho = 40;

    if (mouseX >= engrenagemX - engrenagemTamanho/2 &&
        mouseX <= engrenagemX + engrenagemTamanho/2 &&
        mouseY >= engrenagemY - engrenagemTamanho/2 &&
        mouseY <= engrenagemY + engrenagemTamanho/2) {
      nivelAtual = -1;  // Volta para a seleção de nível
      mensagem = "Escolha um nível para começar!";
    }
    // Verifica clique no botão Jogar Novamente (apenas quando o jogo terminou)
    else if (jogoTerminado) {
      float x = width/2 - 100;
      float y = height - 100;
      if (mouseX >= x && mouseX <= x + 200 && mouseY >= y && mouseY <= y + 50) {
        iniciarJogo();
      }
    }
  }
}

void keyPressed() {
  if (jogoTerminado) {
    return;
  }

  // Converte a tecla pressionada para maiúscula
  char letra = Character.toUpperCase(key);

  // Verifica se a tecla pressionada é uma letra
  if (letra >= 'A' && letra <= 'Z') {
    // Verifica se a letra já foi usada
```

```
if (letrasUsadas.contains(letra)) {
  mensagem = "Esta letra já foi usada!";
  return;
}

// Adiciona a letra à lista de letras usadas
letrasUsadas.add(letra);

// Verifica se a letra está na palavra
boolean acertou = false;
String novaPalavra = "";

for (int i = 0; i < palavra.length(); i++) {
  if (palavra.charAt(i) == letra) {
    novaPalavra += letra;
    acertou = true;
  } else {
    novaPalavra += palavraAtual.charAt(i);
  }
}

if (acertou) {
  palavraAtual = novaPalavra;
  mensagem = "Acertou! Continue tentando!";

  // Verifica se o jogador venceu
  if (palavraAtual.equals(palavra)) {
    jogoTerminado = true;
    mensagem = "Parabéns! Você venceu!";

  }
} else {
  letrasErradas.add(letra);
  vidas--;
```

```
      mensagem = "Errou! Tente novamente!";

      // Verifica se o jogador perdeu
      if (vidas <= 0) {
        jogoTerminado = true;
        mensagem = "Game Over! A palavra era: " + palavra;
      }
    }
  } else {
    // Se não for uma letra, mostra mensagem de ajuda
    mensagem = "Digite uma letra (A-Z) para jogar!";
  }
}
```