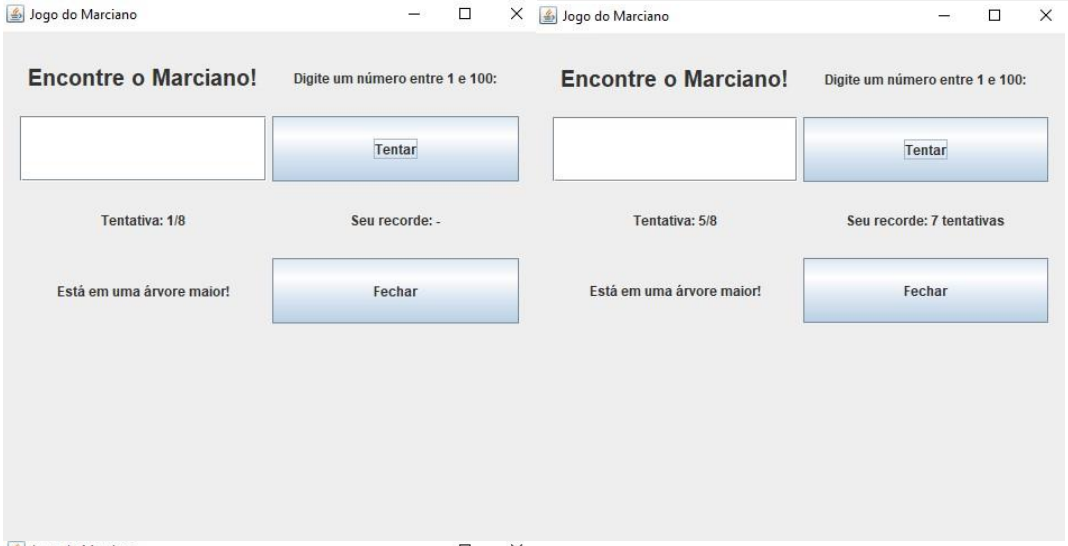
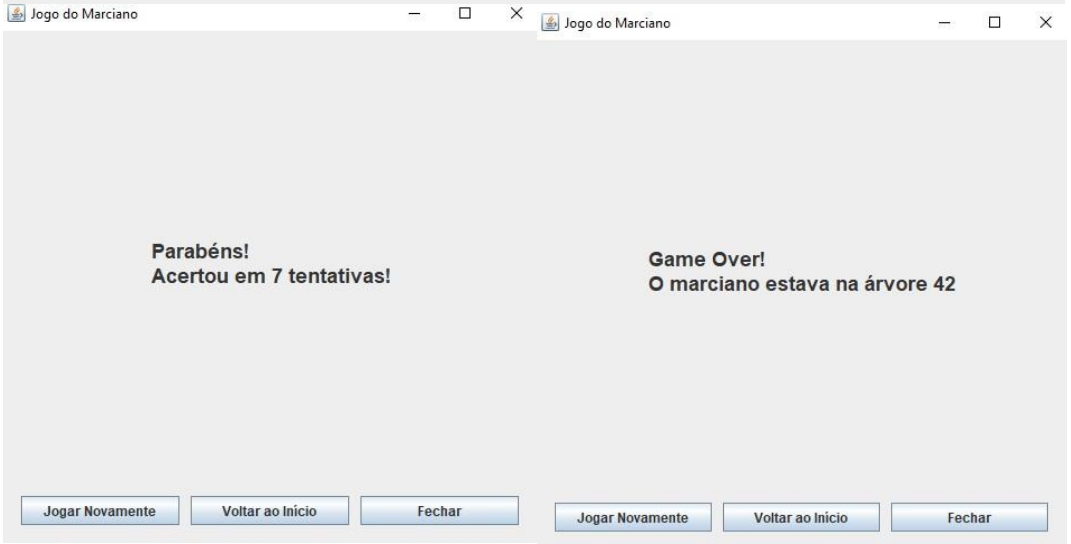
Game Studio: Tigre Produções

Eliel Lucas Trajano Neto – 01606048

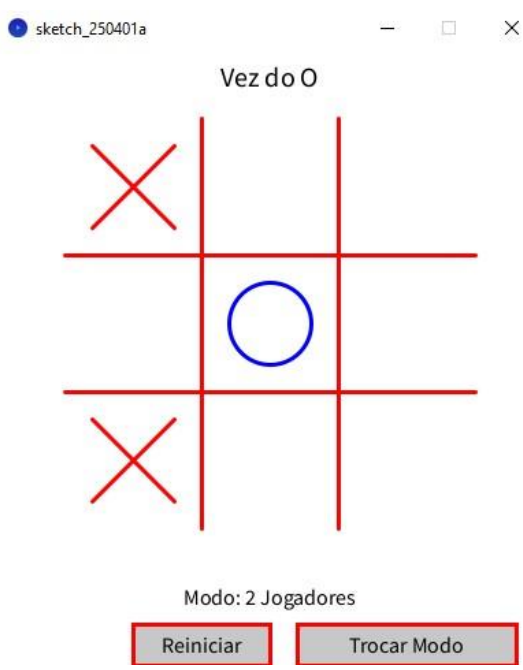
Rodrigo Andrade Cavalcanti Muniz – 01606059

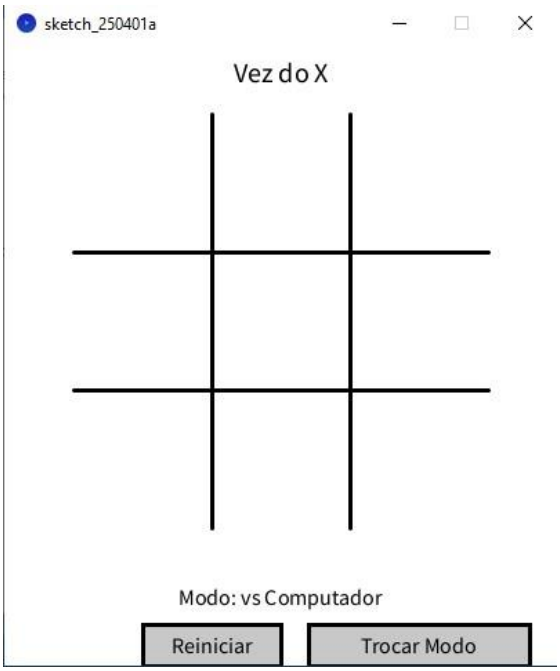
1. Jogo do Marciano

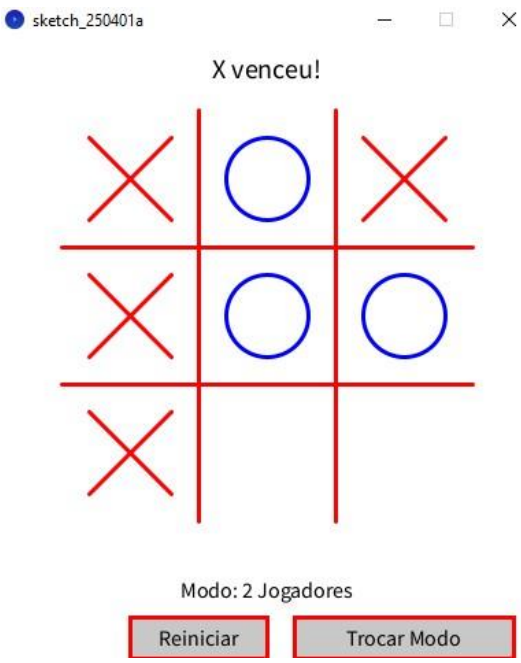


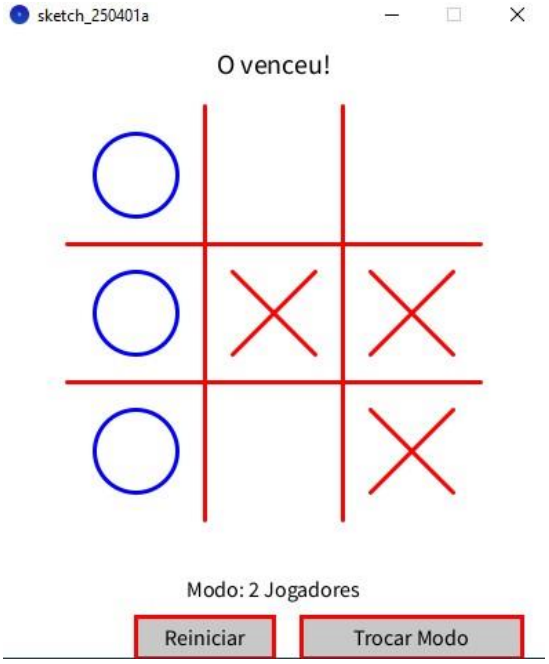


2. Jogo da velha

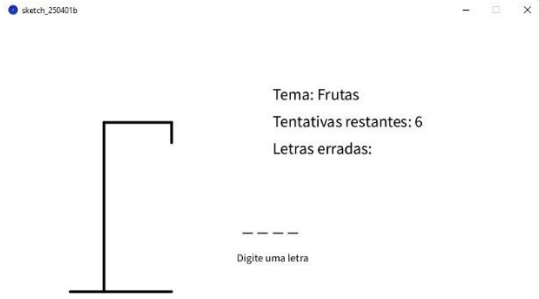


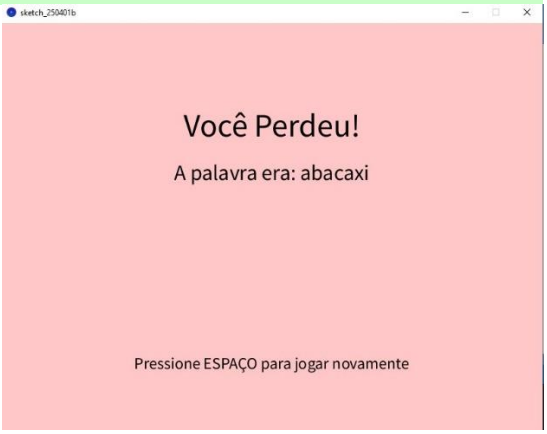




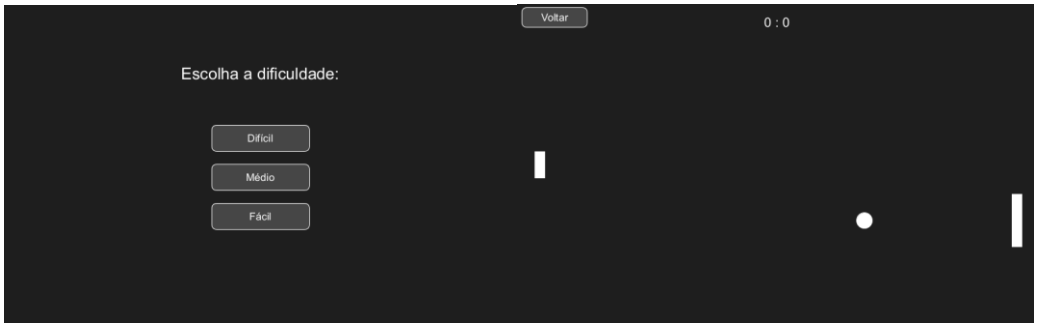


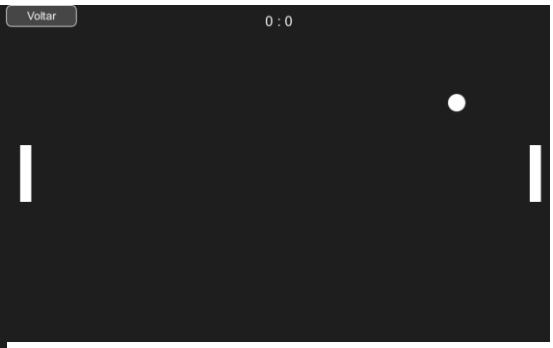
3. Jogo da forca



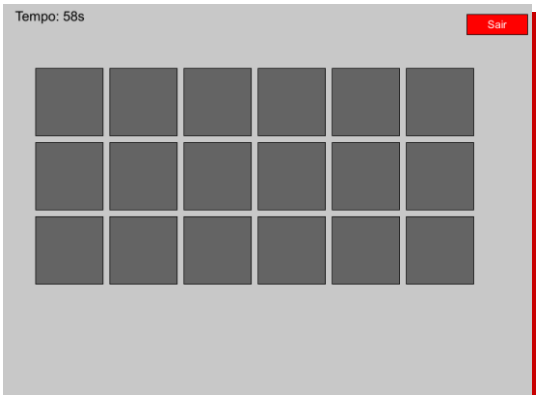


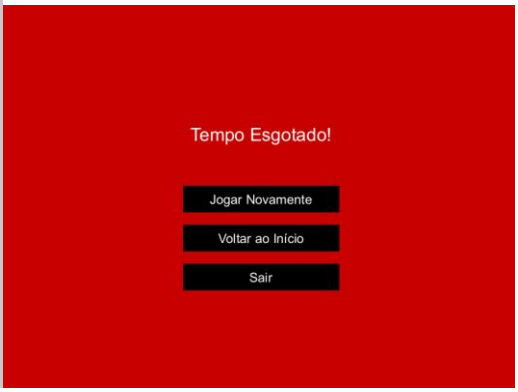
4. Jogo do Pong

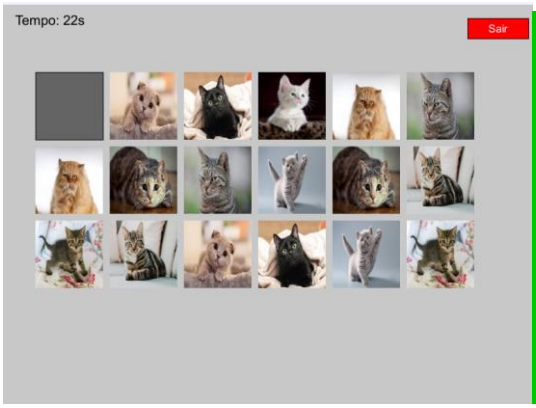




5. Jogo da Memória







1. Jogo do marciano

```
import javax.swing.*;
import java.awt.*;
import java.util.Random;
import java.util.ArrayList;
import java.util.Collections;

public class JogoMarcianoFinal extends JFrame {
    private final int LIMITE_TENTATIVAS = 8;
    private ArrayList<Integer> recordes = new ArrayList<>();

    // Componentes da tela inicial
    private JPanel initialPanel;
    private JButton iniciarButton, fecharButton1;
    private JTextArea historiaArea;
    private JLabel recordesLabel;

    // Componentes do jogo
    private JPanel gamePanel;
    private JLabel tentativaLabel;
    private JTextField entradaField;
    private JLabel mensagemLabel;
    private JLabel recordeAtualLabel;
    private JButton fecharButton2;

    // Componentes do fim de jogo
    private JPanel endPanel;
    private JButton fecharButton3;

    // Variáveis do jogo
    private int arvore;
    private int tentativas;

    public JogoMarcianoFinal() {
        initComponents();
        mostrarTelaInicial();
    }

    private void initComponents() {
        setTitle("Jogo do Marciano");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(500, 500);
        setLocationRelativeTo(null);
        setLayout(new CardLayout());

        // Tela inicial
        initialPanel = new JPanel(new BorderLayout(10, 10));
```

```
48.         initialPanel.setBorder(BorderFactory.createEmptyBorder(15,
    15, 15, 15));
49.
50.         JLabel tituloInicial = new JLabel("Jogo do Marciano",
    SwingConstants.CENTER);
51.         tituloInicial.setFont(new Font("Arial", Font.BOLD, 24));
52.
53.         // Área de história com rolagem
54.         historiaArea = new JTextArea(
55.             "História:\n" +
56.             "Num planeta distante, um marciano veio visitar a Terra
    e se escondeu\n" +
57.             "entre as árvores numeradas de 1 a 100. Sua missão é
    encontrá-lo!\n\n" +
58.             "Regras:\n" +
59.             "- Você tem " + LIMITE_TENTATIVAS + " tentativas para
    adivinhar\n" +
60.             "- A cada palpite, você receberá uma dica se o marciano
    está em\n" +
61.             "  uma árvore maior ou menor\n" +
62.             "- Tente acertar com o menor número de tentativas
    possível!"
63.         );
64.         historiaArea.setFont(new Font("Arial", Font.PLAIN, 14));
65.         historiaArea.setEditable(false);
66.         historiaArea.setLineWrap(true);
67.         historiaArea.setWrapStyleWord(true);
68.         historiaArea.setBackground(getBackground());
69.
70.         recordesLabel = new JLabel("Recordes: Nenhum recorde
    ainda", SwingConstants.CENTER);
71.         recordesLabel.setFont(new Font("Arial", Font.BOLD, 16));
72.
73.         // Painel de botões da tela inicial
74.         JPanel botoesInicioPanel = new JPanel(new GridLayout(1, 2,
    10, 10));
75.         iniciarButton = new JButton("Iniciar Jogo");
76.         iniciarButton.setFont(new Font("Arial", Font.BOLD, 16));
77.         iniciarButton.addActionListener(e -> iniciarJogo());
78.
79.         fecharButton1 = new JButton("Fechar");
80.         fecharButton1.setFont(new Font("Arial", Font.BOLD, 16));
81.         fecharButton1.addActionListener(e -> System.exit(0));
82.
83.         botoesInicioPanel.add(iniciarButton);
84.         botoesInicioPanel.add(fecharButton1);
85.
86.         JPanel centerPanel = new JPanel();
```

```
87.         centerPanel.setLayout(new BoxLayout(centerPanel,
    BoxLayout.Y_AXIS));
88.         centerPanel.add(new JScrollPane(historiaArea));
89.         centerPanel.add(Box.createRigidArea(new Dimension(0, 15)));
90.         centerPanel.add(recordesLabel);
91.         centerPanel.add(Box.createRigidArea(new Dimension(0, 15)));
92.         centerPanel.add(botoesInicioPanel);
93.
94.         initialPanel.add(tituloInicial, BorderLayout.NORTH);
95.         initialPanel.add(centerPanel, BorderLayout.CENTER);
96.
97.         // Tela do jogo
98.         gamePanel = new JPanel();
99.         gamePanel.setLayout(new GridLayout(7, 1, 5, 5)); //
    Aumentei para 7 linhas
100.            gamePanel.setBorder(BorderFactory.createEmptyBorder(1
    0, 20, 10, 20));
101.
102.            JLabel tituloJogo = new JLabel("Encontre o
    Marciano!", SwingConstants.CENTER);
103.            tituloJogo.setFont(new Font("Arial", Font.BOLD, 20));
104.
105.            JLabel instrucaoLabel = new JLabel("Digite um número
    entre 1 e 100:", SwingConstants.CENTER);
106.
107.            entradaField = new JTextField();
108.            entradaField.setHorizontalAlignment(JTextField.CENTER
    );
109.
110.            JButton tentarButton = new JButton("Tentar");
111.            tentarButton.addActionListener(e ->
    verificarPalpite());
112.
113.            tentativaLabel = new JLabel("Tentativa: 0/" +
    LIMITE_TENTATIVAS, SwingConstants.CENTER);
114.            recordeAtualLabel = new JLabel("Seu recorde: -",
    SwingConstants.CENTER);
115.            mensagemLabel = new JLabel("",
    SwingConstants.CENTER);
116.
117.            fecharButton2 = new JButton("Fechar");
118.            fecharButton2.addActionListener(e -> System.exit(0));
119.
120.            gamePanel.add(tituloJogo);
121.            gamePanel.add(instrucaoLabel);
122.            gamePanel.add(entradaField);
123.            gamePanel.add(tentarButton);
124.            gamePanel.add(tentativaLabel);
125.            gamePanel.add(recordeAtualLabel);
```

```
126.             gamePanel.add(mensagemLabel);
127.             gamePanel.add(fecharButton2);
128.
129.             // Tela de fim de jogo
130.             endPanel = new JPanel(new BorderLayout(10, 20));
131.             endPanel.setBorder(BorderFactory.createEmptyBorder(20
    , 20, 20, 20));
132.
133.             JLabel fimLabel = new JLabel("",
    SwingConstants.CENTER);
134.             fimLabel.setFont(new Font("Arial", Font.BOLD, 18));
135.             endPanel.add(fimLabel, BorderLayout.CENTER);
136.
137.             JPanel botoesFimPanel = new JPanel(new GridLayout(1,
    3, 10, 10));
138.             JButton reiniciarButton = new JButton("Jogar
    Novamente");
139.             reiniciarButton.addActionListener(e ->
    iniciarJogo());
140.
141.             JButton voltarButton = new JButton("Voltar ao
    Início");
142.             voltarButton.addActionListener(e ->
    mostrarTelaInicial());
143.
144.             fecharButton3 = new JButton("Fechar");
145.             fecharButton3.addActionListener(e -> System.exit(0));
146.
147.             botoesFimPanel.add(reiniciarButton);
148.             botoesFimPanel.add(voltarButton);
149.             botoesFimPanel.add(fecharButton3);
150.             endPanel.add(botoesFimPanel, BorderLayout.SOUTH);
151.
152.             // Adiciona todos os painéis
153.             add(initialPanel, "inicio");
154.             add(gamePanel, "jogo");
155.             add(endPanel, "fim");
156.         }
157.
158.         private void mostrarTelaInicial() {
159.             atualizarRecordesLabel();
160.             ((CardLayout)
    getContentPane().getLayout()).show(getContentPane(), "inicio");
161.         }
162.
163.         private void iniciarJogo() {
164.             Random random = new Random();
165.             arvore = random.nextInt(100) + 1;
166.             tentativas = 0;
```

```
167.
168.             tentativaLabel.setText("Tentativa: 0/" +
    LIMITE_TENTATIVAS);
169.             mensagemLabel.setText("");
170.             entradaField.setText("");
171.
172.             if (!recordes.isEmpty()) {
173.                 recordeAtualLabel.setText("Seu recorde: " +
    Collections.min(recordes) + " tentativas");
174.             } else {
175.                 recordeAtualLabel.setText("Seu recorde: -");
176.             }
177.
178.             ((CardLayout)
    getContentPane().getLayout()).show(getContentPane(), "jogo");
179.             entradaField.requestFocus();
180.         }
181.
182.         private void verificarPalpite() {
183.             try {
184.                 int palpite =
    Integer.parseInt(entradaField.getText());
185.
186.                 if (palpite < 1 || palpite > 100) {
187.                     mensagemLabel.setText("Digite um número entre
    1 e 100!");
188.                     return;
189.                 }
190.
191.                 tentativas++;
192.                 tentativaLabel.setText("Tentativa: " + tentativas
    + "/" + LIMITE_TENTATIVAS);
193.
194.                 if (palpite == arvore) {
195.                     finalizarJogo(true);
196.                     return;
197.                 }
198.
199.                 if (tentativas >= LIMITE_TENTATIVAS) {
200.                     finalizarJogo(false);
201.                     return;
202.                 }
203.
204.                 if (palpite > arvore) {
205.                     mensagemLabel.setText("Está em uma árvore
    menor!");
206.                 } else {
207.                     mensagemLabel.setText("Está em uma árvore
    maior!");
```

```
208.                 }
209.
210.                 entradaField.setText("");
211.             } catch (NumberFormatException ex) {
212.                 mensagemLabel.setText("Digite um número
    válido!");
213.             }
214.         }
215.
216.         private void finalizarJogo(boolean vitoria) {
217.             JLabel fimLabel = (JLabel) endPanel.getComponent(0);
218.
219.             if (vitoria) {
220.                 fimLabel.setText("<html>Parabéns!<br>Acertou em "
    + tentativas + " tentativas!</html>");
221.                 recordes.add(tentativas);
222.                 Collections.sort(recordes);
223.             } else {
224.                 fimLabel.setText("<html>Game Over!<br>O marciano
    estava na árvore " + arvore + "</html>");
225.             }
226.
227.             ((CardLayout)
    getContentPane().getLayout()).show(getContentPane(), "fim");
228.         }
229.
230.         private void atualizarRecordesLabel() {
231.             if (recordes.isEmpty()) {
232.                 recordesLabel.setText("Recordes: Nenhum recorde
    ainda");
233.             } else {
234.                 StringBuilder sb = new
    StringBuilder("<html>Recordes:<br>");
235.                 int max = Math.min(5, recordes.size());
236.
237.                 for (int i = 0; i < max; i++) {
238.                     sb.append(i+1).append(".
    ").append(recordes.get(i)).append(" tentativas<br>");
239.                 }
240.
241.                 sb.append("</html>");
242.                 recordesLabel.setText(sb.toString());
243.             }
244.         }
245.
246.         public static void main(String[] args) {
247.             SwingUtilities.invokeLater(() -> {
248.                 JogoMarcianoFinal jogo = new JogoMarcianoFinal();
249.                 jogo.setVisible(true);
```

```
250.             });
251.         }
252.     }
```

2. Jogo da velha

```
// Variáveis globais
int boardSize = 300; // Tamanho do tabuleiro
int cellSize = boardSize / 3; // Tamanho de cada célula
int[][] board = new int[3][3]; // 0 = vazio, 1 = X, 2 = O
int currentPlayer = 1; // Começa com X
boolean gameOver = false;
int winner = 0; // 0 = sem vencedor, 1 = X, 2 = O, 3 = empate
boolean vsComputer = false; // Modo de jogo (true = vs computador, false = vs jogador)

void setup() {
  size(400, 450); // Largura x Altura (incluindo espaço para botões)
  resetGame();

  // Configuração do texto
  textSize(24);
  textAlign(CENTER, CENTER);
}

void draw() {
  background(255);

  // Desenha o tabuleiro
  drawBoard();

  // Desenha as marcações (X e O)
  drawMarks();
```

```
  // Desenha o status do jogo
  drawStatus();

  // Desenha os botões
  drawButtons();

  // Verifica se é a vez do computador no modo vs computador
  if (vsComputer && currentPlayer == 2 && !gameOver) {
    delay(500); // Pequeno atraso para parecer mais natural
    computerMove();
  }
}

void drawBoard() {
  strokeWeight(3);
  line(50, 150, 350, 150); // Linha horizontal superior
  line(50, 250, 350, 250); // Linha horizontal inferior
  line(150, 50, 150, 350); // Linha vertical esquerda
  line(250, 50, 250, 350); // Linha vertical direita
}

void drawMarks() {
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      float x = 50 + j * cellSize + cellSize/2;
      float y = 50 + i * cellSize + cellSize/2;

      if (board[i][j] == 1) { // X
        stroke(255, 0, 0);
        line(x - 30, y - 30, x + 30, y + 30);
        line(x + 30, y - 30, x - 30, y + 30);
```

```
    } else if (board[i][j] == 2) { // O
      stroke(0, 0, 255);
      noFill();
      ellipse(x, y, 60, 60);
    }
   }
  }
}

void drawStatus() {
  fill(0);
  textSize(20);

  if (gameOver) {
    if (winner == 3) {
      text("Empate!", width/2, 20);
    } else if (winner == 1) {
      text("X venceu!", width/2, 20);
    } else if (winner == 2) {
      text("O venceu!", width/2, 20);
    }
  } else {
    if (currentPlayer == 1) {
      text("Vez do X", width/2, 20);
    } else {
      text("Vez do O", width/2, 20);
    }
  }

  // Mostra o modo de jogo
  textSize(16);
```

```
  if (vsComputer) {
    text("Modo: vs Computador", width/2, 400);
  } else {
    text("Modo: 2 Jogadores", width/2, 400);
  }
}

void drawButtons() {
  // Botão de reiniciar
  fill(200);
  rect(100, 420, 100, 30);
  fill(0);
  textSize(16);
  text("Reiniciar", 150, 435);

  // Botão de trocar modo
  fill(200);
  rect(220, 420, 160, 30);
  fill(0);
  text("Trocar Modo", 300, 435);
}

void mousePressed() {
  // Verifica clique nos botões
  if (mouseX >= 100 && mouseX <= 200 && mouseY >= 420 && mouseY <= 450) {
    resetGame();
    return;
  }

  if (mouseX >= 220 && mouseX <= 380 && mouseY >= 420 && mouseY <= 450) {
    vsComputer = !vsComputer;
```

```
    resetGame();
    return;
  }

  // Se o jogo acabou, não processa cliques no tabuleiro
  if (gameOver) return;

  // Verifica se o clique foi dentro do tabuleiro
  if (mouseX >= 50 && mouseX <= 350 && mouseY >= 50 && mouseY <= 350) {
    int row = (mouseY - 50) / cellSize;
    int col = (mouseX - 50) / cellSize;

    // Verifica se a célula está vazia
    if (board[row][col] == 0) {
      board[row][col] = currentPlayer;

      // Verifica se houve vencedor
      checkWinner();

      // Troca o jogador se o jogo não acabou
      if (!gameOver) {
        currentPlayer = (currentPlayer == 1) ? 2 : 1;
      }
    }
  }
}

void checkWinner() {
  // Verifica linhas
  for (int i = 0; i < 3; i++) {
    if (board[i][0] != 0 && board[i][0] == board[i][1] && board[i][0] == board[i][2]) {
```

```
    gameOver = true;
    winner = board[i][0];
    return;
  }
}

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

// Verifica diagonais
if (board[0][0] != 0 && board[0][0] == board[1][1] && board[0][0] == board[2][2]) {
  gameOver = true;
  winner = board[0][0];
  return;
}

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

// Verifica empate
boolean isTie = true;
for (int i = 0; i < 3; i++) {
```

```
    for (int j = 0; j < 3; j++) {
      if (board[i][j] == 0) {
        isTie = false;
        break;
      }
    }
    if (!isTie) break;
  }

  if (isTie) {
    gameOver = true;
    winner = 3; // Empate
  }
}

void computerMove() {
  // Primeiro, verifica se pode vencer na próxima jogada
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      if (board[i][j] == 0) {
        board[i][j] = 2; // O é o computador
        checkWinner();
        if (gameOver && winner == 2) {
          return; // Computador vence
        } else {
          board[i][j] = 0; // Desfaz a jogada
          gameOver = false;
          winner = 0;
        }
      }
    }
```

```
}

// Depois, verifica se precisa bloquear o jogador humano
for (int i = 0; i < 3; i++) {
  for (int j = 0; j < 3; j++) {
    if (board[i][j] == 0) {
      board[i][j] = 1; // X é o jogador humano
      checkWinner();
      if (gameOver && winner == 1) {
        board[i][j] = 2; // Bloqueia
        gameOver = false;
        winner = 0;
        currentPlayer = 1; // Passa a vez para o jogador humano
        return;
      } else {
        board[i][j] = 0; // Desfaz a jogada
        gameOver = false;
        winner = 0;
      }
    }
  }
}

// Se não houver jogadas críticas, faz uma jogada aleatória
ArrayList<PVector> emptyCells = new ArrayList<PVector>();
for (int i = 0; i < 3; i++) {
  for (int j = 0; j < 3; j++) {
    if (board[i][j] == 0) {
      emptyCells.add(new PVector(i, j));
    }
  }
```

```
    }

    if (emptyCells.size() > 0) {
      int randomIndex = (int)random(emptyCells.size());
      PVector move = emptyCells.get(randomIndex);
      board[(int)move.x][(int)move.y] = 2;
      checkWinner();
      currentPlayer = 1; // Passa a vez para o jogador humano
    }
  }

  void resetGame() {
    for (int i = 0; i < 3; i++) {
      for (int j = 0; j < 3; j++) {
        board[i][j] = 0;
      }
    }
    currentPlayer = 1;
    gameOver = false;
    winner = 0;
  }
```

3.Jogo da forca

```
// Variáveis globais
int boardSize = 300; // Tamanho do tabuleiro
int cellSize = boardSize / 3; // Tamanho de cada célula
int[][] board = new int[3][3]; // 0 = vazio, 1 = X, 2 = O
int currentPlayer = 1; // Começa com X
boolean gameOver = false;
int winner = 0; // 0 = sem vencedor, 1 = X, 2 = O, 3 = empate
boolean vsComputer = false; // Modo de jogo (true = vs computador, false = vs jogador)
```

```
void setup() {
  size(400, 450); // Largura x Altura (incluindo espaço para botões)
  resetGame();

  // Configuração do texto
  textSize(24);
  textAlign(CENTER, CENTER);
}

void draw() {
  background(255);

  // Desenha o tabuleiro
  drawBoard();

  // Desenha as marcações (X e O)
  drawMarks();

  // Desenha o status do jogo
  drawStatus();

  // Desenha os botões
  drawButtons();

  // Verifica se é a vez do computador no modo vs computador
  if (vsComputer && currentPlayer == 2 && !gameOver) {
    delay(500); // Pequeno atraso para parecer mais natural
    computerMove();
  }
}
```

```
void drawBoard() {
  strokeWeight(3);
  line(50, 150, 350, 150); // Linha horizontal superior
  line(50, 250, 350, 250); // Linha horizontal inferior
  line(150, 50, 150, 350); // Linha vertical esquerda
  line(250, 50, 250, 350); // Linha vertical direita
}

void drawMarks() {
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      float x = 50 + j * cellSize + cellSize/2;
      float y = 50 + i * cellSize + cellSize/2;

      if (board[i][j] == 1) { // X
        stroke(255, 0, 0);
        line(x - 30, y - 30, x + 30, y + 30);
        line(x + 30, y - 30, x - 30, y + 30);
      } else if (board[i][j] == 2) { // O
        stroke(0, 0, 255);
        noFill();
        ellipse(x, y, 60, 60);
      }
    }
  }
}

void drawStatus() {
  fill(0);
  textSize(20);
```

```
  if (gameOver) {
    if (winner == 3) {
      text("Empate!", width/2, 20);
    } else if (winner == 1) {
      text("X venceu!", width/2, 20);
    } else if (winner == 2) {
      text("O venceu!", width/2, 20);
    }
  } else {
    if (currentPlayer == 1) {
      text("Vez do X", width/2, 20);
    } else {
      text("Vez do O", width/2, 20);
    }
  }

  // Mostra o modo de jogo
  textSize(16);
  if (vsComputer) {
    text("Modo: vs Computador", width/2, 400);
  } else {
    text("Modo: 2 Jogadores", width/2, 400);
  }
}

void drawButtons() {
  // Botão de reiniciar
  fill(200);
  rect(100, 420, 100, 30);
  fill(0);
```

```
  textSize(16);
  text("Reiniciar", 150, 435);

  // Botão de trocar modo
  fill(200);
  rect(220, 420, 160, 30);
  fill(0);
  text("Trocar Modo", 300, 435);
}

void mousePressed() {
  // Verifica clique nos botões
  if (mouseX >= 100 && mouseX <= 200 && mouseY >= 420 && mouseY <= 450) {
    resetGame();
    return;
  }

  if (mouseX >= 220 && mouseX <= 380 && mouseY >= 420 && mouseY <= 450) {
    vsComputer = !vsComputer;
    resetGame();
    return;
  }

  // Se o jogo acabou, não processa cliques no tabuleiro
  if (gameOver) return;

  // Verifica se o clique foi dentro do tabuleiro
  if (mouseX >= 50 && mouseX <= 350 && mouseY >= 50 && mouseY <= 350) {
    int row = (mouseY - 50) / cellSize;
    int col = (mouseX - 50) / cellSize;
```

```
    // Verifica se a célula está vazia
    if (board[row][col] == 0) {
      board[row][col] = currentPlayer;

      // Verifica se houve vencedor
      checkWinner();

      // Troca o jogador se o jogo não acabou
      if (!gameOver) {
        currentPlayer = (currentPlayer == 1) ? 2 : 1;
      }
    }
  }
}

void checkWinner() {
  // Verifica linhas
  for (int i = 0; i < 3; i++) {
    if (board[i][0] != 0 && board[i][0] == board[i][1] && board[i][0] == board[i][2]) {
      gameOver = true;
      winner = board[i][0];
      return;
    }
  }

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

```
  }
}

// Verifica diagonais
if (board[0][0] != 0 && board[0][0] == board[1][1] && board[0][0] == board[2][2]) {
  gameOver = true;
  winner = board[0][0];
  return;
}

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

// Verifica empate
boolean isTie = true;
for (int i = 0; i < 3; i++) {
  for (int j = 0; j < 3; j++) {
    if (board[i][j] == 0) {
      isTie = false;
      break;
    }
  }
  if (!isTie) break;
}

if (isTie) {
  gameOver = true;
  winner = 3; // Empate
```

```
  }
}

void computerMove() {
  // Primeiro, verifica se pode vencer na próxima jogada
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      if (board[i][j] == 0) {
        board[i][j] = 2; // O é o computador
        checkWinner();
        if (gameOver && winner == 2) {
          return; // Computador vence
        } else {
          board[i][j] = 0; // Desfaz a jogada
          gameOver = false;
          winner = 0;
        }
      }
    }
  }

  // Depois, verifica se precisa bloquear o jogador humano
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      if (board[i][j] == 0) {
        board[i][j] = 1; // X é o jogador humano
        checkWinner();
        if (gameOver && winner == 1) {
          board[i][j] = 2; // Bloqueia
          gameOver = false;
          winner = 0;
```

```
        currentPlayer = 1; // Passa a vez para o jogador humano
        return;
       } else {
        board[i][j] = 0; // Desfaz a jogada
        gameOver = false;
        winner = 0;
       }
     }
    }
   }

   // Se não houver jogadas críticas, faz uma jogada aleatória
   ArrayList<PVector> emptyCells = new ArrayList<PVector>();
   for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
     if (board[i][j] == 0) {
      emptyCells.add(new PVector(i, j));
     }
    }
   }

   if (emptyCells.size() > 0) {
    int randomIndex = (int)random(emptyCells.size());
    PVector move = emptyCells.get(randomIndex);
    board[(int)move.x][(int)move.y] = 2;
    checkWinner();
    currentPlayer = 1; // Passa a vez para o jogador humano
   }
  }

  void resetGame() {
```

```
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      board[i][j] = 0;
    }
  }
  currentPlayer = 1;
  gameOver = false;
  winner = 0;
}
```

4. Jogo do Pong

```
// Estados do jogo
final int MENU = 0;
final int SELECAO = 1;
final int DIFICULDADE = 2;
final int JOGO = 3;
final int FIM = 4;
int estadoAtual = MENU;

PFont fonte;

// Botões
Botao botaoJogar, botaoSair, botaoVsPlayer, botaoVsCPU, botaoVoltar;
Botao botaoDificil, botaoMedio, botaoFacil; // Botões de dificuldade

boolean modoVsCPU = false;

// Variáveis do jogo
int pontosEsq = 0, pontosDir = 0;
Paddle paddleEsq, paddleDir;
```

```
Ball bola;

float dificuldade = 0.07; // Definindo a dificuldade inicial
float tamanhoPaddleEsq = 80; // Tamanho inicial da paleta do jogador 1
float tamanhoPaddleDir = 80; // Tamanho inicial da paleta do jogador 2
float velocidadeBola = 5; // Velocidade inicial da bola

void setup() {
  size(800, 500);
  fonte = createFont("Arial", 20);
  textFont(fonte);

  botaoJogar = new Botao("Jogar", width/2 - 75, height/2 - 40, 150, 40);
  botaoSair = new Botao("Sair", width/2 - 75, height/2 + 20, 150, 40);

  botaoVsPlayer = new Botao("2 Jogadores", width/2 - 85, height/2 - 40, 170, 40);
  botaoVsCPU = new Botao("Contra CPU", width/2 - 85, height/2 + 20, 170, 40);

  botaoVoltar = new Botao("Voltar", 10, 10, 100, 30);

  // Botões de dificuldade corrigidos
  botaoDificil = new Botao("Difícil", width/2 - 75, height/2 - 60, 150, 40); // "Fácil" será o mais difícil
  botaoMedio = new Botao("Médio", width/2 - 75, height/2, 150, 40);
  botaoFacil = new Botao("Fácil", width/2 - 75, height/2 + 60, 150, 40); // "Difícil" será o mais fácil

  iniciarJogo();
}

void draw() {
  background(30);
```

```
  switch (estadoAtual) {
    case MENU:
      mostrarMenu();
      break;
    case SELECAO:
      mostrarSelecao();
      break;
    case DIFICULDADE:
      mostrarDificuldade(); // Tela de seleção de dificuldade
      break;
    case JOGO:
      jogar();
      break;
    case FIM:
      mostrarFim();
      break;
  }
}

void mousePressed() {
  if (estadoAtual == MENU) {
    if (botaoJogar.clicado(mouseX, mouseY)) {
      estadoAtual = SELECAO;
    } else if (botaoSair.clicado(mouseX, mouseY)) {
      exit();
    }
  } else if (estadoAtual == SELECAO) {
    if (botaoVsPlayer.clicado(mouseX, mouseY)) {
      // Modo "2 Jogadores" usa a dificuldade média automaticamente
      modoVsCPU = false;
```

```
    dificuldade = 0.10; // Configuração de dificuldade média
    tamanhoPaddleEsq = 80; // Paleta normal
    tamanhoPaddleDir = 80; // Paleta normal
    velocidadeBola = 5; // Velocidade normal da bola
    iniciarJogo();
    estadoAtual = JOGO;
   } else if (botaoVsCPU.clicado(mouseX, mouseY)) {
    modoVsCPU = true;
    estadoAtual = DIFICULDADE; // Vai para a tela de seleção de dificuldade
   }
  } else if (estadoAtual == DIFICULDADE) {
   if (botaoDificil.clicado(mouseX, mouseY)) {
    dificuldade = 0.90; // "Fácil" será o modo mais difícil
    tamanhoPaddleEsq = 40; // Reduz a paleta pela metade no modo difícil
    tamanhoPaddleDir = 80; // A paleta da máquina permanece a mesma
    velocidadeBola = 10; // Aumenta a velocidade da bola no modo difícil
    iniciarJogo();
    estadoAtual = JOGO;
   } else if (botaoMedio.clicado(mouseX, mouseY)) {
    dificuldade = 0.10;
    tamanhoPaddleEsq = 80; // Paleta normal
    tamanhoPaddleDir = 80; // Paleta normal
    velocidadeBola = 5; // Velocidade normal da bola
    iniciarJogo();
    estadoAtual = JOGO;
   } else if (botaoFacil.clicado(mouseX, mouseY)) {
    dificuldade = 0.03; // "Difícil" será o modo mais fácil
    tamanhoPaddleEsq = 80; // Paleta normal
    tamanhoPaddleDir = 80; // Paleta normal
    velocidadeBola = 2; // Velocidade normal da bola
    iniciarJogo();
```

```
      estadoAtual = JOGO;
    }
  } else if (estadoAtual == JOGO) {
    if (botaoVoltar.clicado(mouseX, mouseY)) {
      estadoAtual = MENU;
    }
  } else if (estadoAtual == FIM) {
    if (botaoVoltar.clicado(mouseX, mouseY)) {
      estadoAtual = MENU;
    }
  }
}

void keyPressed() {
  if (estadoAtual == JOGO) {
    if (key == 'w') paddleEsq.mover(-1);
    if (key == 's') paddleEsq.mover(1);
    if (!modoVsCPU) {
      if (keyCode == UP) paddleDir.mover(-1);
      if (keyCode == DOWN) paddleDir.mover(1);
    }
  }
}

void keyReleased() {
  if (estadoAtual == JOGO) {
    paddleEsq.parar();
    paddleDir.parar();
  }
}
```

```
void mostrarMenu() {
  fill(255);
  textAlign(CENTER);
  textSize(30);
  text("PONG", width/2, 120);

  botaoJogar.desenhar();
  botaoSair.desenhar();
}

void mostrarSelecao() {
  fill(255);
  textAlign(CENTER);
  textSize(25);
  text("Escolha o modo de jogo:", width/2, 120);

  botaoVsPlayer.desenhar();
  botaoVsCPU.desenhar();
}

void mostrarDificuldade() { // Tela de seleção de dificuldade
  fill(255);
  textAlign(CENTER);
  textSize(25);
  text("Escolha a dificuldade:", width/2, 120);

  botaoDificil.desenhar(); // "Fácil" é o mais difícil
  botaoMedio.desenhar(); // Média dificuldade
  botaoFacil.desenhar(); // "Difícil" é o mais fácil
}
```

```
void mostrarFim() {
  fill(255);
  textAlign(CENTER);
  textSize(28);
  String vencedor = pontosEsq == 7 ? "Jogador 1 venceu!" : "Jogador 2 venceu!";
  if (modoVsCPU && pontosDir == 7) vencedor = "CPU venceu!";
  text(vencedor, width/2, height/2 - 20);

  botaoVoltar.desenhar();
}

void iniciarJogo() {
  pontosEsq = 0;
  pontosDir = 0;
  paddleEsq = new Paddle(30, tamanhoPaddleEsq);
  paddleDir = new Paddle(width - 40, tamanhoPaddleDir);
  bola = new Ball();
}

void jogar() {
  paddleEsq.atualizar();
  paddleDir.atualizar();
  bola.atualizar();

  if (modoVsCPU) {
    paddleDir.seguirBola(bola);
  }

  paddleEsq.desenhar();
  paddleDir.desenhar();
  bola.desenhar();
```

```
  fill(255);
  textSize(20);
  textAlign(CENTER);
  text(pontosEsq + " : " + pontosDir, width/2, 40);

  botaoVoltar.desenhar();

  if (pontosEsq >= 7 || pontosDir >= 7) {
    estadoAtual = FIM;
  }
}

class Botao {
  String texto;
  float x, y, w, h;

  Botao(String texto, float x, float y, float w, float h) {
    this.texto = texto;
    this.x = x;
    this.y = y;
    this.w = w;
    this.h = h;
  }

  void desenhar() {
    fill(70);
    stroke(255);
    rect(x, y, w, h, 8);
    fill(255);
    textAlign(CENTER, CENTER);
```

```
    textSize(16);
    text(texto, x + w/2, y + h/2);
  }

  boolean clicado(float mx, float my) {
    return mx > x && mx < x + w && my > y && my < y + h;
  }
}

class Paddle {
  float x, y, w, h;
  float velocidade = 5;
  float dy = 0;

  Paddle(float x, float h) {
    this.x = x;
    this.h = h;
    this.w = 15;
    this.y = height/2 - h/2;
  }

  void atualizar() {
    y += dy;
    y = constrain(y, 0, height - h);
  }

  void desenhar() {
    fill(255);
    rect(x, y, w, h);
  }
```

```
  void mover(int dir) {
    dy = dir * velocidade;
  }

  void parar() {
    dy = 0;
  }

  void seguirBola(Ball b) {
    float centro = y + h/2;
    float erro = b.y - centro;

    dy = erro * dificuldade;

    float maxVelocidade = 4;
    dy = constrain(dy, -maxVelocidade, maxVelocidade);
  }
}

class Ball {
  float x, y, r = 12;
  float vx, vy;
  float velocidade = 5;

  Ball() {
    resetar();
  }

  void resetar() {
    x = width/2;
    y = height/2;
```

```
  vx = random(1) > 0.5 ? velocidade : -velocidade;
  vy = random(-3, 3);
}

void atualizar() {
  x += vx;
  y += vy;

  if (y < 0 || y > height) vy *= -1;

  if (colidiu(paddleEsq) || colidiu(paddleDir)) {
    vx *= -1.05;
    vy = random(-4, 4);
  }

  if (x < 0) {
    pontosDir++;
    resetar();
  }

  if (x > width) {
    pontosEsq++;
    resetar();
  }
}

void desenhar() {
  fill(255);
  ellipse(x, y, r*2, r*2);
}
```

```
  boolean colidiu(Paddle p) {
    return x - r < p.x + p.w && x + r > p.x && y > p.y && y < p.y + p.h;
  }
}
```

5. Jogo da Memória

```
// === Jogo da Memória com Animais ===

final int TOTAL_PARES = 9;
final int TOTAL_CARTAS = TOTAL_PARES * 2;
final int COLUNAS = 6;
final int LINHAS = 3;
final int LARGURA_CARTA = 100;
final int ALTURA_CARTA = 100;

PImage[][] temas = new PImage[3][TOTAL_PARES]; // 0: gatos, 1: cachorros, 2: passaros
String[] nomesTemas = {"Gatos", "Cachorros", "Passaros"};
int temaSelecionado = -1;

int[] embaralhamento = new int[TOTAL_CARTAS];
boolean[] cartaVirada = new boolean[TOTAL_CARTAS];
boolean[] cartaFixa = new boolean[TOTAL_CARTAS];

int primeiraCarta = -1;
int segundaCarta = -1;
boolean aguardando = false;
int tempoVirada;

PFont fonte;
String estado = "inicio";
```

```
int tempoInicio;
int tempoLimite = 60;

void setup() {
  size(800, 600);
  fonte = createFont("Arial", 20, true);

  for (int t = 0; t < 3; t++) {
    for (int i = 0; i < TOTAL_PARES; i++) {
      String nomeBase = "";
      if (t == 0) nomeBase = "gato";
      if (t == 1) nomeBase = "cachorro";
      if (t == 2) nomeBase = "passaro";
      temas[t][i] = loadImage(nomeBase + i + ".jpg");
      temas[t][i].resize(LARGURA_CARTA, ALTURA_CARTA);
    }
  }

  embaralharCartas();
}

void draw() {
  background(200);

  if (estado.equals("inicio")) {
    telaInicio();
  } else if (estado.equals("jogando")) {
    telaJogo();
    mostrarTempo();
    verificarVitoria();
  } else if (estado.equals("fim")) {
```

```
    telaFinal();
  }

  if (aguardando) {
    if (embaralhamento[primeiraCarta] == embaralhamento[segundaCarta]) {
      cartaFixa[primeiraCarta] = true;
      cartaFixa[segundaCarta] = true;
      primeiraCarta = -1;
      segundaCarta = -1;
      aguardando = false;
    } else if (millis() - tempoVirada > 1000) {
      cartaVirada[primeiraCarta] = false;
      cartaVirada[segundaCarta] = false;
      primeiraCarta = -1;
      segundaCarta = -1;
      aguardando = false;
    }
  }
}

void telaInicio() {
  textFont(fonte);
  textAlign(CENTER);
  fill(0);
  textSize(32);
  text("Jogo da Memória", width / 2, 80);

  textSize(20);
  text("Escolha um tema:", width / 2, 140);

  for (int i = 0; i < nomesTemas.length; i++) {
```

```
    int x = width / 2 - 110;
    int y = 180 + i * 70;
    fill(temaSelecionado == i ? color(100, 200, 100) : 0);
    rect(x, y, 220, 40);
    fill(255);
    text(nomesTemas[i], width / 2, y + 27);
  }

  if (temaSelecionado != -1) {
    fill(0);
    rect(width / 2 - 100, 450, 200, 40);
    fill(255);
    text("Iniciar Jogo", width / 2, 477);
  }

  fill(255, 0, 0);
  rect(width - 110, 20, 90, 30);
  fill(255);
  textSize(16);
  textAlign(CENTER, CENTER);
  text("Sair", width - 65, 35);
}

void telaJogo() {
  for (int i = 0; i < TOTAL_CARTAS; i++) {
    int x = (i % COLUNAS) * (LARGURA_CARTA + 10) + 50;
    int y = (i / COLUNAS) * (ALTURA_CARTA + 10) + 100;

    if (cartaFixa[i] || cartaVirada[i]) {
      image(temas[temaSelecionado][embaralhamento[i]], x, y);
    } else {
```

```
      fill(100);
      rect(x, y, LARGURA_CARTA, ALTURA_CARTA);
    }
  }

  fill(255, 0, 0);
  rect(width - 110, 20, 90, 30);
  fill(255);
  textSize(16);
  textAlign(CENTER, CENTER);
  text("Sair", width - 65, 35);
}

void telaFinal() {
  background(estadoVitoria() ? color(0, 200, 0) : color(200, 0, 0));
  fill(255);
  textFont(fonte);
  textSize(28);
  textAlign(CENTER, CENTER);
  text(estadoVitoria() ? "Você Ganhou!" : "Tempo Esgotado!", width / 2, height / 2 - 100);

  textSize(20);
  fill(0);
  rect(width / 2 - 120, height / 2 - 20, 240, 40);
  fill(255);
  text("Jogar Novamente", width / 2, height / 2);

  fill(0);
  rect(width / 2 - 120, height / 2 + 40, 240, 40);
  fill(255);
  text("Voltar ao Início", width / 2, height / 2 + 60);
```

```
  fill(0);
  rect(width / 2 - 120, height / 2 + 100, 240, 40);
  fill(255);
  text("Sair", width / 2, height / 2 + 120);
}

void mousePressed() {
  if (estado.equals("inicio")) {
    for (int i = 0; i < nomesTemas.length; i++) {
      int y = 180 + i * 70;
      if (mouseX > width / 2 - 110 && mouseX < width / 2 + 110 && mouseY > y && mouseY < y + 40) {
        temaSelecionado = i;
      }
    }

    if (temaSelecionado != -1 && mouseX > width / 2 - 100 && mouseX < width / 2 + 100 && mouseY > 450 && mouseY < 490) {
      reiniciarJogo(); // <- IMPORTANTE!
      estado = "jogando";
    }

    if (mouseX > width - 110 && mouseX < width - 20 && mouseY > 20 && mouseY < 50) {
      exit();
    }
  } else if (estado.equals("jogando")) {
    if (mouseX > width - 110 && mouseX < width - 20 && mouseY > 20 && mouseY < 50) {
      exit();
    }

    if (!aguardando) {
```

```
    for (int i = 0; i < TOTAL_CARTAS; i++) {
      int x = (i % COLUNAS) * (LARGURA_CARTA + 10) + 50;
      int y = (i / COLUNAS) * (ALTURA_CARTA + 10) + 100;

      if (!cartaFixa[i] && !cartaVirada[i] && mouseX > x && mouseX < x + LARGURA_CARTA &&
mouseY > y && mouseY < y + ALTURA_CARTA) {
        cartaVirada[i] = true;
        if (primeiraCarta == -1) {
          primeiraCarta = i;
        } else if (segundaCarta == -1 && i != primeiraCarta) {
          segundaCarta = i;
          aguardando = true;
          tempoVirada = millis();
        }
        break;
      }
    }
  }
 } else if (estado.equals("fim")) {
  if (mouseX > width / 2 - 120 && mouseX < width / 2 + 120) {
    if (mouseY > height / 2 - 20 && mouseY < height / 2 + 20) {
      reiniciarJogo();
      estado = "jogando";
    } else if (mouseY > height / 2 + 40 && mouseY < height / 2 + 80) {
      estado = "inicio";
      temaSelecionado = -1;
    } else if (mouseY > height / 2 + 100 && mouseY < height / 2 + 140) {
      exit();
    }
  }
 }
```

```
}

void mostrarTempo() {
  int tempoRestante = tempoLimite - (millis() - tempoInicio) / 1000;
  fill(0);
  textSize(20);
  textAlign(LEFT);
  text("Tempo: " + tempoRestante + "s", 20, 30);
  if (tempoRestante <= 0) estado = "fim";
}

void verificarVitoria() {
  for (boolean fixa : cartaFixa) {
    if (!fixa) return;
  }
  estado = "fim";
}

boolean estadoVitoria() {
  for (boolean fixa : cartaFixa) {
    if (!fixa) return false;
  }
  return true;
}

void embaralharCartas() {
  int[] pares = new int[TOTAL_CARTAS];
  for (int i = 0; i < TOTAL_PARES; i++) {
    pares[i * 2] = i;
    pares[i * 2 + 1] = i;
  }
```

```
  for (int i = 0; i < TOTAL_CARTAS; i++) {
    int r = int(random(i, TOTAL_CARTAS));
    int temp = pares[i];
    pares[i] = pares[r];
    pares[r] = temp;
  }
  for (int i = 0; i < TOTAL_CARTAS; i++) {
    embaralhamento[i] = pares[i];
    cartaVirada[i] = false;
    cartaFixa[i] = false;
  }
}

void reiniciarJogo() {
  embaralharCartas();
  primeiraCarta = -1;
  segundaCarta = -1;
  aguardando = false;
  tempoInicio = millis();
}
```