Projetos de Desenvolvimento de Jogos

2LM Game Studio

Integrantes da Equipe "2LM":
- Diego Henrique Rodrigues - 01650828
- Lucas Oliveira Carneiro - 01636600
- Lucas Vinicius França Aires - 01627405
- Monique Rafaela Carvalho Lopes - 01633424

RELATÓRIO TÉCNICO

1. Jogo da Forca

PRINTS:

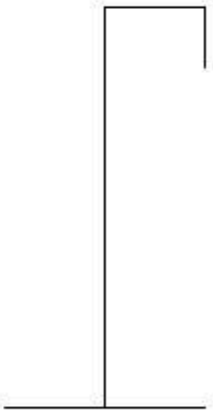
Jogo_da_forca

Categoria: Cidades

Palavra: _____

Tentativas restantes: 5

Letras erradas: []



Jogo_da_forca

Categoria: Filmes

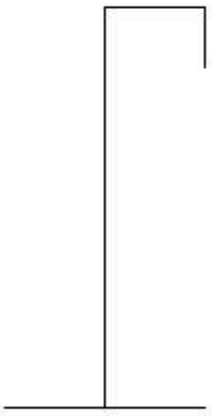
Palavra: titanic

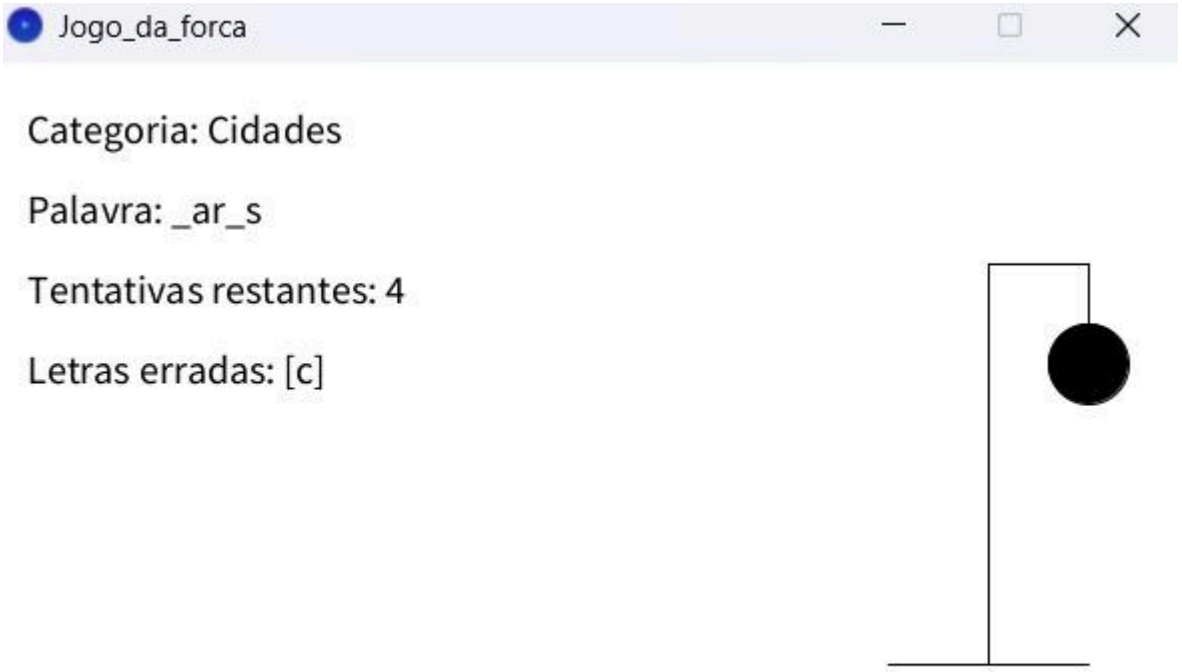
Tentativas restantes: 6

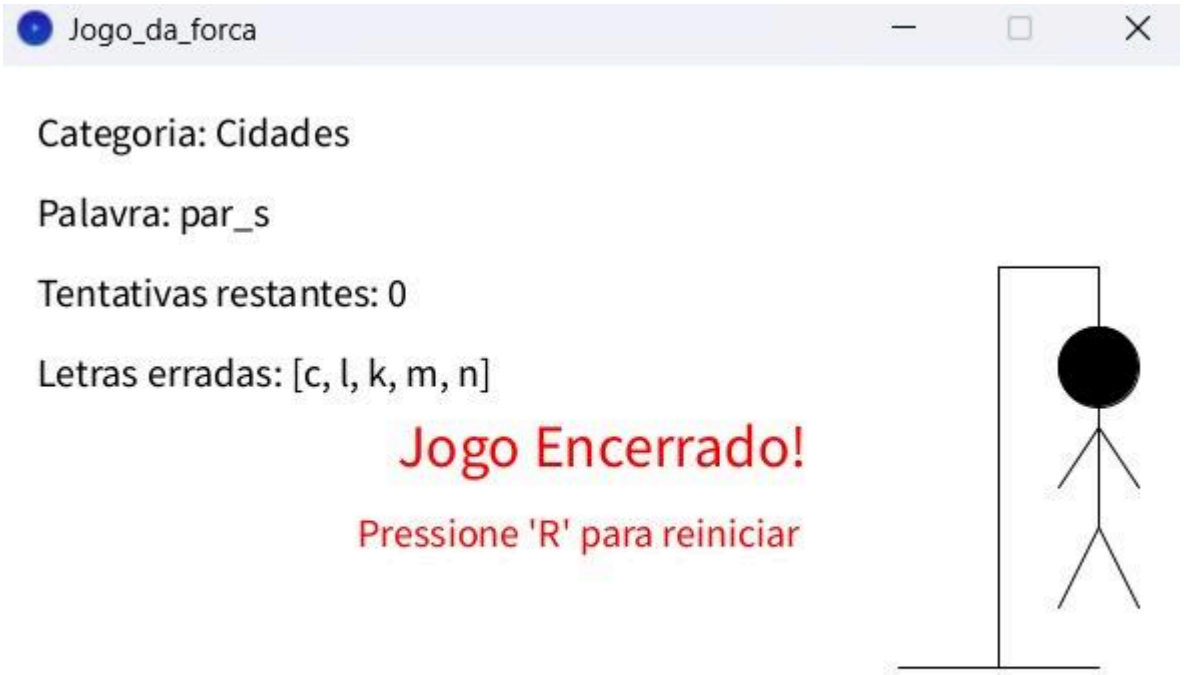
Letras erradas: []

Você venceu!

Pressione 'R' para reiniciar





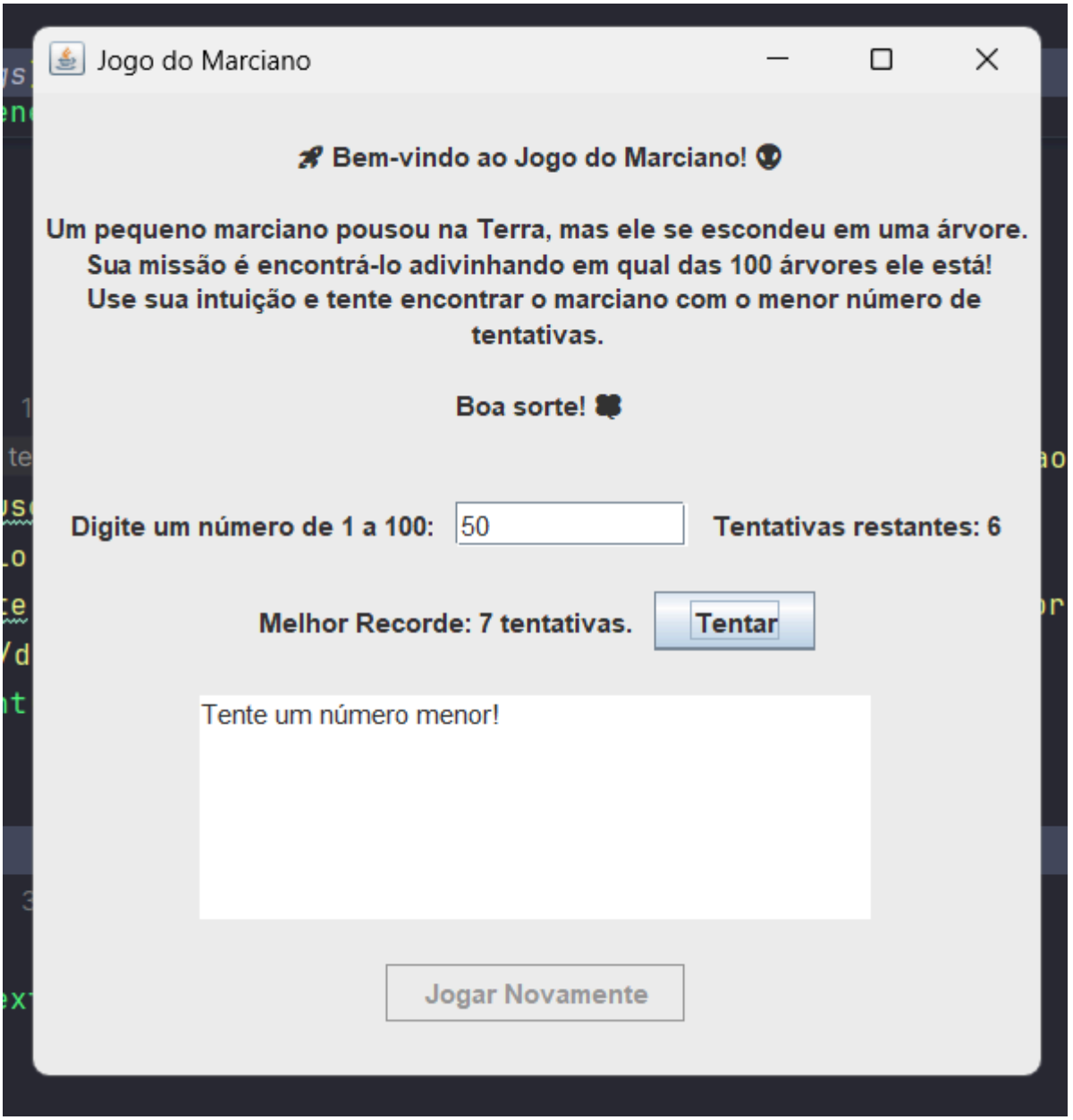


2. Jogo da Velha

PRINTS:

3. Jogo do Marciano

PRINTS:



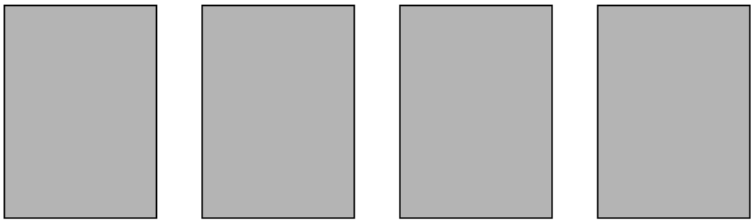
4. Jogo da Memória

PRINTS:

Tema: frutas

1: Frutas | 2: Objetos | 3: Personagens

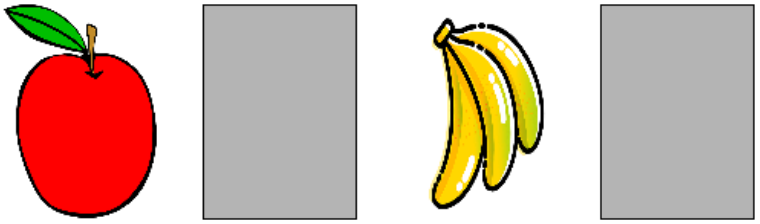
R: Reiniciar



Tema: frutas

1: Frutas | 2: Objetos | 3: Personagens

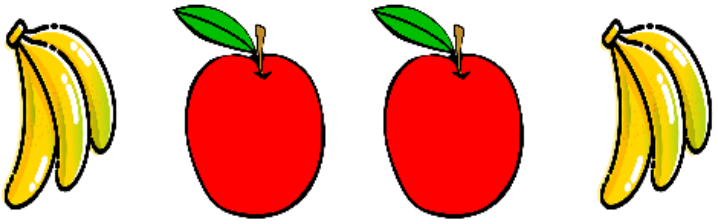
R: Reiniciar



Tema: frutas

1: Frutas | 2: Objetos | 3: Personagens

R: Reiniciar



 Você venceu!

Pressione 'R' para jogar novamente.

Tema: objetos

1: Frutas | 2: Objetos | 3: Personagens

R: Reiniciar

 Você venceu!

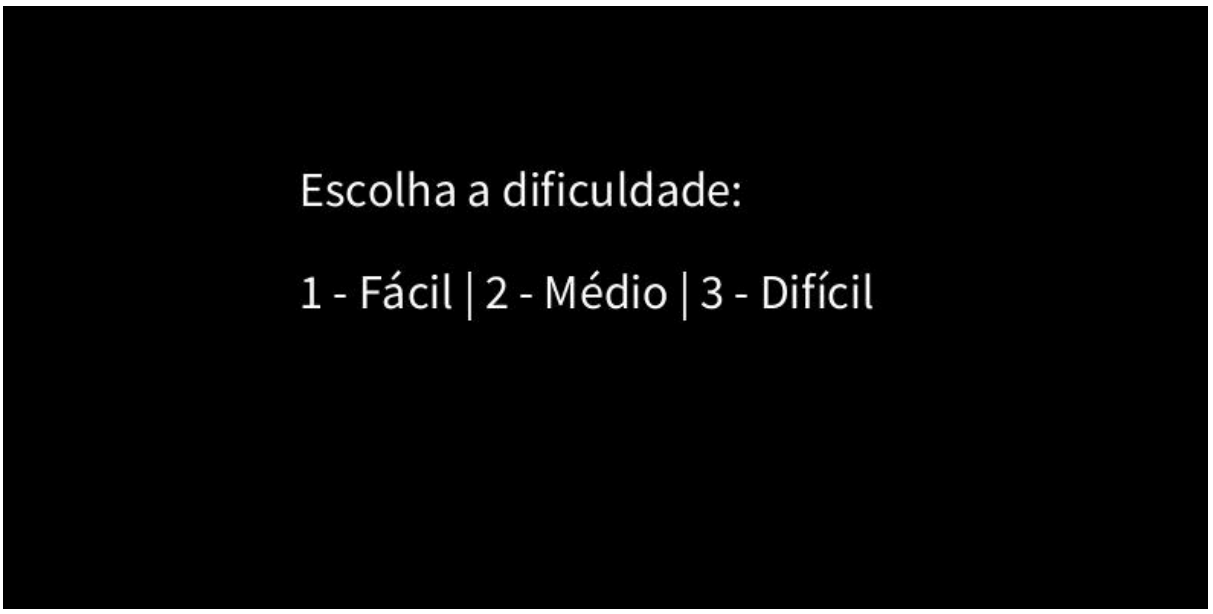
Pressione 'R' para jogar novamente.

Tema: personagens
1: Frutas | 2: Objetos | 3: Personagens
R: Reiniciar

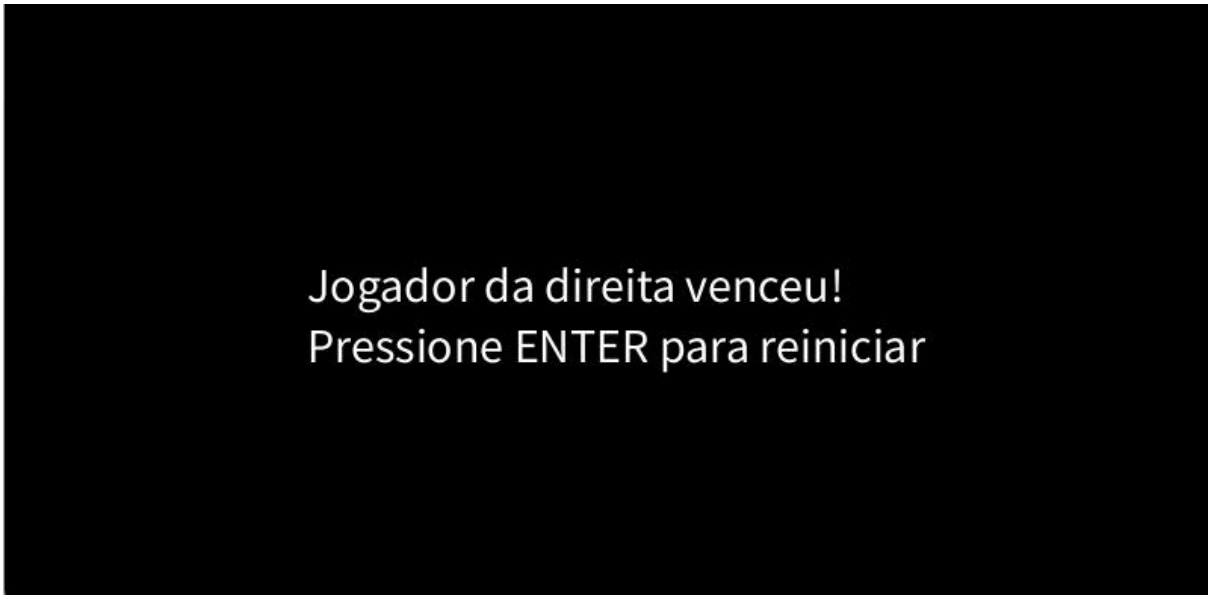
 Você venceu!
Pressione 'R' para jogar novamente.

5. Jogo do Pong

PRINTS:


Escolha a dificuldade:
1 - Fácil | 2 - Médio | 3 - Difícil


Tempo: 30s
0
0


Jogador da direita venceu!
Pressione ENTER para reiniciar

CÓDIGO-FONTE - JOGO DA FORCA:

```
String[][] categorias = {
  {"maça", "banana", "uva", "laranja", "manga"},
  {"paris", "londres", "berlim", "roma", "lisboa"},
  {"titanic", "avatar", "matrix", "interestelar", "inception"},
  {"vermelho", "azul", "verde", "amarelo", "roxo"}
};


String[] nomesCategorias = {"Frutas", "Cidades", "Filmes", "Cores"};
int categoriaSelecionada;
String palavra;
char[] palavraOculta;
ArrayList<Character> letrasErradas = new ArrayList<Character>();
int tentativasRestantes = 5;
boolean jogoEncerrado = false;
boolean venceu = false;

void setup() {
  size(600, 400);
  escolherPalavra();
}

void escolherPalavra() {
  categoriaSelecionada = int(random(categorias.length));
  palavra =
categorias[categoriaSelecionada][int(random(categorias[categoriaSelecionada].length))];
  palavraOculta = new char[palavra.length()];
  for (int i = 0; i < palavraOculta.length; i++) {
    palavraOculta[i] = '_';
  }
}

void draw() {
  background(255);
```

```
  fill(0);
  textSize(20);
  text("Categoria: " + nomesCategorias[categoriaSelecionada], 20, 40);
  text("Palavra: " + new String(palavraOculta), 20, 80);
  text("Tentativas restantes: " + tentativasRestantes, 20, 120);
  text("Letras erradas: " + letrasErradas, 20, 160);
  desenharForca();

  if (jogoEncerrado) {
    textSize(30);
    if (venceu) {
      fill(0, 255, 0);
      text("Você venceu!", 220, 200);
    } else {
      fill(255, 0, 0);
      text("Jogo Encerrado!", 200, 200);
    }
    textSize(20);
    text("Pressione 'R' para reiniciar", 180, 240);
  }
}

void desenharForca() {
  stroke(0);
  line(450, 300, 550, 300);
  line(500, 300, 500, 100);
  line(500, 100, 550, 100);
  line(550, 100, 550, 130);

  if (tentativasRestantes <= 4) {
    ellipse(550, 150, 40, 40);
  }
  if (tentativasRestantes <= 3) {
    line(550, 170, 550, 230);
  }
  if (tentativasRestantes <= 2) {
    line(550, 180, 530, 210);
    line(550, 180, 570, 210);
  }
  if (tentativasRestantes <= 1) {
```

```
    line(550, 230, 530, 270);
    line(550, 230, 570, 270);
  }
}

void keyPressed() {
  if (jogoEncerrado && key == 'r') {
    reiniciarJogo();
    return;
  }

  if (tentativasRestantes > 0 && !jogoEncerrado) {
    char letra = Character.toLowerCase(key);
    boolean acertou = false;

    for (int i = 0; i < palavra.length(); i++) {
      if (palavra.charAt(i) == letra) {
        palavraOculta[i] = letra;
        acertou = true;
      }
    }

    if (!acertou && !letrasErradas.contains(letra)) {
      letrasErradas.add(letra);
      tentativasRestantes--;
    }

    if (new String(palavraOculta).equals(palavra)) {
      jogoEncerrado = true;
      venceu = true;
    } else if (tentativasRestantes == 0) {
      jogoEncerrado = true;
      venceu = false;
    }
  }
}


void reiniciarJogo() {
  tentativasRestantes = 6;
```

```
  letrasErradas.clear();
  jogoEncerrado = false;
  venceu = false;
  escolherPalavra();
}
```

CÓDIGO-FONTE - JOGO DA VELHA:

```
int[][] board = new int[3][3];
boolean playerTurn = true;
boolean gameOver = false;
int winner = 0;
boolean againstComputer = false;
boolean modeSelected = false;

void setup() {
  size(300, 300);
}

void draw() {
  background(245, 235, 220);
  if (!modeSelected) {
    drawModeSelection();
  } else {
    drawBoard();
    if (gameOver) {
      displayWinner();
    }
  }
}

void mousePressed() {
  if (!modeSelected) {
    if (mouseY > height / 4 && mouseY < height / 4 + 100) {
      if (mouseX < width / 2) {
        againstComputer = false;
      } else {
        againstComputer = true;
      }
      modeSelected = true;
      resetBoard();
    }
    return;
  }
```

```
  if (!gameOver) {
    int row = mouseY / 100;
    int col = mouseX / 100;
    if (row >= 0 && row < 3 && board[row][col] == 0) {
      board[row][col] = playerTurn ? 1 : 2;
      playerTurn = !playerTurn;
      checkWinner();
      if (againstComputer && !playerTurn && !gameOver) {
        computerMove();
      }
    }
  } else {
    resetBoard();
  }
}

void drawBoard() {
  stroke(139, 69, 19);
  strokeWeight(6);
  for (int i = 1; i < 3; i++) {
    line(i * 100, 0, i * 100, 300);
    line(0, i * 100, 300, i * 100);
  }
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      if (board[i][j] == 1) {
        drawX(j, i);
      } else if (board[i][j] == 2) {
        drawO(j, i);
      }
    }
  }
}

void drawX(int x, int y) {
  stroke(139, 69, 19);
  strokeWeight(8);
  line(x * 100 + 20, y * 100 + 20, x * 100 + 80, y * 100 + 80);
  line(x * 100 + 80, y * 100 + 20, x * 100 + 20, y * 100 + 80);
}

void drawO(int x, int y) {
  stroke(139, 69, 19);
  strokeWeight(8);
  noFill();
  ellipse(x * 100 + 50, y * 100 + 50, 60, 60);
}
```

```
void checkWinner() {
  for (int i = 0; i < 3; i++) {
    if (board[i][0] != 0 && board[i][0] == board[i][1] && board[i][1] == board[i][2]) {
      winner = board[i][0];
      gameOver = true;
      return;
    }
    if (board[0][i] != 0 && board[0][i] == board[1][i] && board[1][i] == board[2][i]) {
      winner = board[0][i];
      gameOver = true;
      return;
    }
  }
  if (board[0][0] != 0 && board[0][0] == board[1][1] && board[1][1] == board[2][2]) {
    winner = board[0][0];
    gameOver = true;
    return;
  }
  if (board[0][2] != 0 && board[0][2] == board[1][1] && board[1][1] == board[2][0]) {
    winner = board[0][2];
    gameOver = true;
    return;
  }
  boolean draw = true;
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      if (board[i][j] == 0) {
        draw = false;
      }
    }
  }
  if (draw) {
    gameOver = true;
    winner = 0;
  }
}

void displayWinner() {
  fill(139, 69, 19);
  rectMode(CENTER);
  rect(width / 2, height / 2, 220, 50);

  fill(245, 235, 220);
  textSize(24);
  textAlign(CENTER, CENTER);

  if (winner == 0) {
```

```
    text("Empate!", width / 2, height / 2);
  } else {
    text("Jogador " + (winner == 1 ? "X" : "O") + " venceu!", width / 2, height / 2);
  }
}


void resetBoard() {
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      board[i][j] = 0;
    }
  }
  gameOver = false;
  winner = 0;
  playerTurn = true;
}

void computerMove() {
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      if (board[i][j] == 0) {
        board[i][j] = 2;
        checkWinner();
        playerTurn = true;
        return;
      }
    }
  }
}

void drawModeSelection() {
  fill(139, 69, 19);
  rect(0, height / 4, width / 2, 100);
  rect(width / 2, height / 4, width / 2, 100);
  fill(245, 235, 220);
  textSize(22);
  textAlign(CENTER, CENTER);
  text("2 Jogadores", width / 4, height / 4 + 50);
  text("Contra PC", 3 * width / 4, height / 4 + 50);
}
```

CÓDIGO-FONTE - JOGO DO MARCIANO:

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

public class JogoDoMarcianoGUI {

    private static final int maxTentativas = 7;
    private static int recorde = 7;
    private static int tentativas;
    private static int numeroMarciano;

    public static void main(String[] args) {
        JFrame frame = new JFrame("Jogo do Marciano");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(500, 500);
        frame.setLayout(new BorderLayout());

        JPanel painel = new JPanel();
        painel.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 20));

        JLabel textoIntro = getJLabel();
        painel.add(textoIntro);

        JTextField entradaNumero = new JTextField();
        entradaNumero.setColumns(10);
        painel.add(new JLabel("Digite um número de 1 a 100:"));
        painel.add(entradaNumero);

        JLabel labelTentativas = new JLabel("Tentativas restantes: " + maxTentativas);
        JLabel labelRecorde = new JLabel("Melhor Recorde: " + recorde + " tentativas.");
        painel.add(labelTentativas);
        painel.add(labelRecorde);

        JButton btnTentar = new JButton("Tentar");
        painel.add(btnTentar);

        JTextArea resultado = new JTextArea(3, 30);
        resultado.setEditable(false);
        resultado.setLineWrap(true);
        resultado.setWrapStyleWord(true);
        resultado.setPreferredSize(new Dimension(300, 100));
```

```
        painel.add(resultado);

        JButton btnJogarNovamente = new JButton("Jogar Novamente");
        btnJogarNovamente.setEnabled(false);
        painel.add(btnJogarNovamente);

        frame.add(painel, BorderLayout.CENTER);

        frame.setVisible(true);

        btnTentar.addActionListener(e -> {
            if (tentativas == 0) {
                resetarJogo();
            }

            if (tentativas == 0) {
                numeroMarciano = new Random().nextInt(100) + 1;
            }

            String textoEntrada = entradaNumero.getText();
            try {
                int tentativa = Integer.parseInt(textoEntrada);
                if (tentativa < 1 || tentativa > 100) {
                    resultado.setText("Por favor, insira um número entre 1 e 100.");
                    return;
                }
                tentativas++;

                if (tentativa < numeroMarciano) {
                    resultado.setText("Tente um número maior!");
                } else if (tentativa > numeroMarciano) {
                    resultado.setText("Tente um número menor!");
                } else {
                    resultado.setText("🎉 Parabéns! Você encontrou o marciano na árvore " +
numeroMarciano + " em " + tentativas + " tentativas!");
                    if (tentativas < recorde) {
                        recorde = tentativas;
                        labelRecorde.setText("Melhor Recorde: " + recorde + " tentativas.");
                        resultado.append("\n🏆 Novo recorde!");
                    }
                    btnTentar.setEnabled(false);
```

```
                btnJogarNovamente.setEnabled(true);
            }

            labelTentativas.setText("Tentativas restantes: " + (maxTentativas - tentativas));

            if (tentativas >= maxTentativas) {
                resultado.append("\n😢 Você atingiu o número máximo de tentativas!");
                btnTentar.setEnabled(false);
                btnJogarNovamente.setEnabled(true);
            }

        } catch (NumberFormatException ex) {
            resultado.setText("Por favor, insira um número válido.");
        }
    });

    btnJogarNovamente.addActionListener(e -> {
        resetarJogo();
        resultado.setText("Boa sorte na próxima tentativa! 🍀");
        labelTentativas.setText("Tentativas restantes: " + maxTentativas);
        btnTentar.setEnabled(true);
        btnJogarNovamente.setEnabled(false);
    });

    resetarJogo();
}

private static JLabel getJLabel() {
    JLabel textoIntro = new JLabel("<html><div style='text-align: center;'>🚀 Bem-vindo ao
Jogo do Marciano! 👽<br><br>" +
            "Um pequeno marciano pousou na Terra, mas ele se escondeu em uma
árvore.<br>" +
            "Sua missão é encontrá-lo adivinhando em qual das 100 árvores ele está!<br>" +
            "Use sua intuição e tente encontrar o marciano com o menor número de
tentativas.<br><br>" +
            "Boa sorte! 🍀<br><br></div></html>");
    textoIntro.setHorizontalAlignment(SwingConstants.CENTER);
    return textoIntro;
}

private static void resetarJogo() {
```

```
        tentativas = 0;
        numeroMarciano = new Random().nextInt(100) + 1;
    }
}
```

CÓDIGO-FONTE - JOGO DA MEMÓRIA:

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;

final int NUM_PARES = 4;
Carta[] cartas;
String temaAtual = "frutas";
boolean podeVirar = true;
Carta primeiraCarta = null;
Carta segundaCarta = null;
int tempoEspera = 0;
PFont fonte;
boolean venceu = false;

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

void draw() {
  background(255);

  // Interface
  fill(0);
  textAlign(LEFT, TOP);
  text("Tema: " + temaAtual, 10, 10);
  text("1: Frutas | 2: Objetos | 3: Personagens", 10, 40);
  text("R: Reiniciar", 10, 70);
```

```
  // Mostrar cartas
  for (Carta c : cartas) {
    c.mostrar();
  }

  // Verifica tempo de espera para virar de volta
  if (!podeVirar && millis() > tempoEspera) {
    if (!primeiraCarta.combinaCom(segundaCarta)) {
      primeiraCarta.virada = false;
      segundaCarta.virada = false;
    }
    primeiraCarta = null;
    segundaCarta = null;
    podeVirar = true;
  }

  // Vitória
  if (venceu) {
    fill(0, 180, 0);
    textAlign(CENTER, CENTER);
    textSize(30);
    text("🎉 Você venceu! 🎉", width / 2, height - 100);
    textSize(20);
    text("Pressione 'R' para jogar novamente.", width / 2, height - 60);
  }
}

void mousePressed() {
  if (!podeVirar || venceu) return;

  for (Carta c : cartas) {
    if (c.foiClicada(mouseX, mouseY) && !c.virada) {
      c.virada = true;

      if (primeiraCarta == null) {
        primeiraCarta = c;
      } else if (segundaCarta == null && c != primeiraCarta) {
        segundaCarta = c;
        podeVirar = false;
```

```
        tempoEspera = millis() + 1000;

        if (primeiraCarta.combinaCom(segundaCarta)) {
          primeiraCarta = null;
          segundaCarta = null;
          podeVirar = true;

          if (jogoVencido()) {
            venceu = true;
          }
        }
      }

      break;
    }
  }
}

void keyPressed() {
  if (key == '1') temaAtual = "frutas";
  else if (key == '2') temaAtual = "objetos";
  else if (key == '3') temaAtual = "personagens";
  else if (key == 'r' || key == 'R') {
    iniciarJogo();
    return;
  }
  iniciarJogo();
}

void iniciarJogo() {
  venceu = false;
  primeiraCarta = null;
  segundaCarta = null;
  podeVirar = true;

  String[] nomes = listImageNames(temaAtual);

  // Agrupar imagens por base (ex: maca → [maca1.png, maca2.png])
  HashMap<String, ArrayList<String>> paresPorBase = new HashMap<String,
ArrayList<String>>();
```

```
for (String nome : nomes) {
  String base = nome.substring(0, nome.length() - 5); // remove "1.png" ou "2.png"
  if (!paresPorBase.containsKey(base)) {
    paresPorBase.put(base, new ArrayList<String>());
  }
  paresPorBase.get(base).add(nome);
}

// Selecionar somente os pares válidos
ArrayList<String> basesValidas = new ArrayList<String>();
for (String base : paresPorBase.keySet()) {
  if (paresPorBase.get(base).size() == 2) {
    basesValidas.add(base);
  }
}

// Embaralhar e criar cartas
Collections.shuffle(basesValidas);
ArrayList<Carta> baralho = new ArrayList<Carta>();

for (int i = 0; i < NUM_PARES && i < basesValidas.size(); i++) {
  String base = basesValidas.get(i);
  ArrayList<String> imagens = paresPorBase.get(base);
  for (String nomeImg : imagens) {
    PImage img = loadImage("temas/" + temaAtual + "/" + nomeImg);
    baralho.add(new Carta(img, base));
  }
}

// Embaralhar cartas
Collections.shuffle(baralho);
cartas = baralho.toArray(new Carta[0]);

// Posicionar em grid
for (int i = 0; i < cartas.length; i++) {
  int x = 100 + (i % 4) * 130;
  int y = 120 + (i / 4) * 180;
  cartas[i].setPosicao(x, y);
}
```

```
}

boolean jogoVencido() {
  for (Carta c : cartas) {
    if (!c.virada) return false;
  }
  return true;
}

String[] listImageNames(String tema) {
  if (tema.equals("frutas")) {
    return new String[] { "maca1.png", "maca2.png", "banana1.png", "banana2.png" };
  } else if (tema.equals("objetos")) {
    return new String[] { "bola1.png", "bola2.png", "livro1.png", "livro2.png" };
  } else {
    return new String[] { "flash1.png", "flash2.png", "sonic1.png", "sonic2.png" };
  }
}

class Carta {
  PImage img;
  String nome;
  int x, y, w = 100, h = 140;
  boolean virada = false;

  Carta(PImage img, String nome) {
    this.img = img;
    this.nome = nome;
  }

  void setPosicao(int x, int y) {
    this.x = x;
    this.y = y;
  }

  void mostrar() {
    stroke(0);
    if (virada) {
      image(img, x, y, w, h);
    } else {
```

```
      fill(180);
      rect(x, y, w, h);
    }
  }

  boolean foiClicada(int mx, int my) {
    return mx >= x && mx <= x + w && my >= y && my <= y + h;
  }

  boolean combinaCom(Carta outra) {
    return this.nome.equals(outra.nome);
  }
}
```

CÓDIGO-FONTE - JOGO DO PONG:

```
int scoreLeft = 0;
int scoreRight = 0;
int scoreLimit = 5;
boolean gameOver = false;
String winner = "";
boolean menu = true;
int difficulty = 1;

float ballX, ballY, ballSpeedX, ballSpeedY;
int ballSize = 20;
boolean impactEffect = false;
int impactTimer = 0;

float paddleWidth = 10, paddleHeight = 80;
float leftPaddleY, rightPaddleY;
float paddleSpeed = 6;
boolean moveUpLeft = false, moveDownLeft = false;
boolean moveUpRight = false, moveDownRight = false;

int startTime;
```

```
void setup() {
  size(800, 400);
  resetGame();
  startTime = millis();
}

void draw() {
  if (menu) {
    showMenu();
  } else if (!gameOver) {
    playGame();
  } else {
    showWinner();
  }
}

void resetGame() {
  ballX = width / 2;
  ballY = height / 2;
  ballSpeedX = random(3, 5) * (random(1) > 0.5 ? 1 : -1);
  ballSpeedY = random(3, 5) * (random(1) > 0.5 ? 1 : -1);
  leftPaddleY = rightPaddleY = height / 2 - paddleHeight / 2;
  impactEffect = false;
  impactTimer = 0;
  startTime = millis();
}

void playGame() {
  background(0);
  fill(255);
  textSize(32);
  text(scoreLeft, width / 4, 50);
  text(scoreRight, 3 * width / 4, 50);

  int elapsedTime = (millis() - startTime) / 1000;
  textSize(24);
  text("Tempo: " + elapsedTime + "s", width / 2 - 50, 30);

  fill(255, 0, 0);
  rect(20, leftPaddleY, paddleWidth, paddleHeight);
```

```
fill(0, 0, 255);
rect(width - 30, rightPaddleY, paddleWidth, paddleHeight);

if (impactEffect) {
  fill(255, 255, 0);
} else {
  fill(255);
}
ellipse(ballX, ballY, ballSize, ballSize);

ballX += ballSpeedX;
ballY += ballSpeedY;

if (ballY <= 0 || ballY >= height) {
  ballSpeedY *= -1;
  triggerImpact();
}

if (ballX <= 30 && ballY > leftPaddleY && ballY < leftPaddleY + paddleHeight) {
  ballSpeedX *= -1.1;
  triggerImpact();
}
if (ballX >= width - 30 && ballY > rightPaddleY && ballY < rightPaddleY + paddleHeight) {
  ballSpeedX *= -1.1;
  triggerImpact();
}

if (ballX < 0) {
  scoreRight++;
  if (scoreRight >= scoreLimit) {
    gameOver = true;
    winner = "Jogador da direita venceu!";
  } else {
    resetGame();
  }
} else if (ballX > width) {
  scoreLeft++;
  if (scoreLeft >= scoreLimit) {
    gameOver = true;
    winner = "Jogador da esquerda venceu!";
```

```
    } else {
      resetGame();
    }
  }

  if (moveUpLeft && leftPaddleY > 0) leftPaddleY -= paddleSpeed;
  if (moveDownLeft && leftPaddleY < height - paddleHeight) leftPaddleY += paddleSpeed;
  if (moveUpRight && rightPaddleY > 0) rightPaddleY -= paddleSpeed;
  if (moveDownRight && rightPaddleY < height - paddleHeight) rightPaddleY +=
paddleSpeed;

  if (impactEffect) {
    impactTimer++;
    if (impactTimer > 10) {
      impactEffect = false;
    }
  }
}

void triggerImpact() {
  impactEffect = true;
  impactTimer = 0;
}

void showWinner() {
  background(0);
  fill(255);
  textSize(32);
  text(winner, width / 4, height / 2);
  text("Pressione ENTER para reiniciar", width / 4, height / 2 + 40);
}

void showMenu() {
  background(0);
  fill(255);
  textSize(32);
  text("Escolha a dificuldade:", width / 4, height / 3);
  text("1 - Fácil | 2 - Médio | 3 - Difícil", width / 4, height / 2);
}
```

```
void keyPressed() {
  if (key == '1') {
    difficulty = 1;
    paddleHeight = 100;
    ballSpeedX = 3;
    ballSpeedY = 3;
    menu = false;
  } else if (key == '2') {
    difficulty = 2;
    paddleHeight = 80;
    ballSpeedX = 4;
    ballSpeedY = 4;
    menu = false;
  } else if (key == '3') {
    difficulty = 3;
    paddleHeight = 60;
    ballSpeedX = 5;
    ballSpeedY = 5;
    menu = false;
  } else if (keyCode == ENTER && gameOver) {
    scoreLeft = 0;
    scoreRight = 0;
    gameOver = false;
    menu = true;
  } else if (key == 'w') {
    moveUpLeft = true;
  } else if (key == 's') {
    moveDownLeft = true;
  } else if (keyCode == UP) {
    moveUpRight = true;
  } else if (keyCode == DOWN) {
    moveDownRight = true;
  }
}

void keyReleased() {
  if (key == 'w') moveUpLeft = false;
  if (key == 's') moveDownLeft = false;
  if (keyCode == UP) moveUpRight = false;
  if (keyCode == DOWN) moveDownRight = false;
```

```
}
```