

Nome do Game Studio: A Liga

LINK DO REPOSITÓRIO GIT COM TODOS OS CÓDIGOS EM PASTAS DE CADA JOGO:

<https://github.com/carlinaceo28/desenvolvimento-de-jogos-av1>

Carla Maria Santana Lopes - 01440665

Carlos Alberto Ramalho Bezerra Neto - 01585045

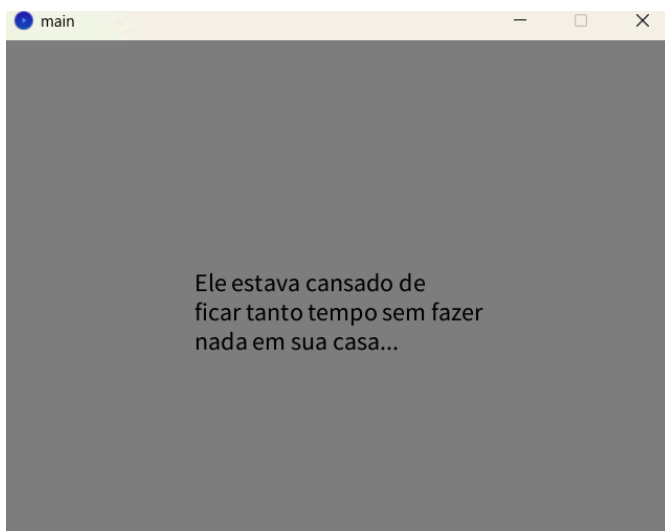
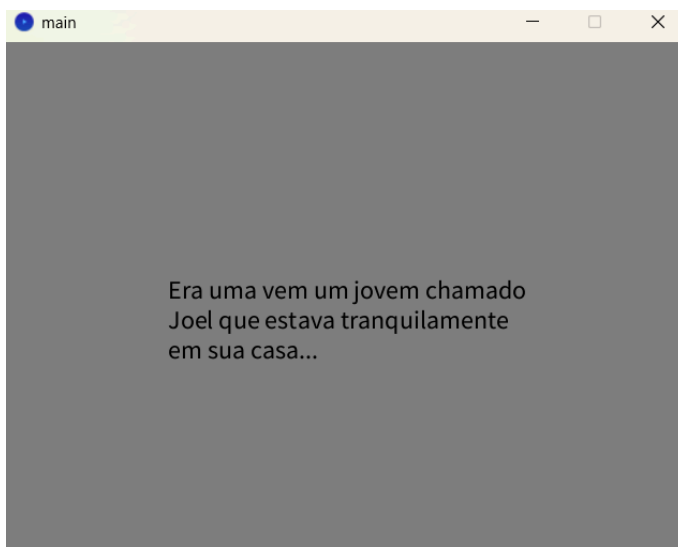
Gustavo Portela Pachêco - 01604533

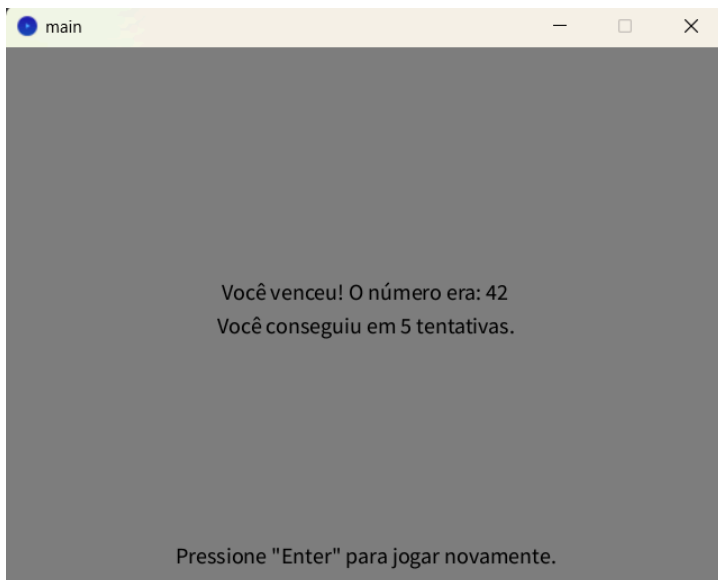
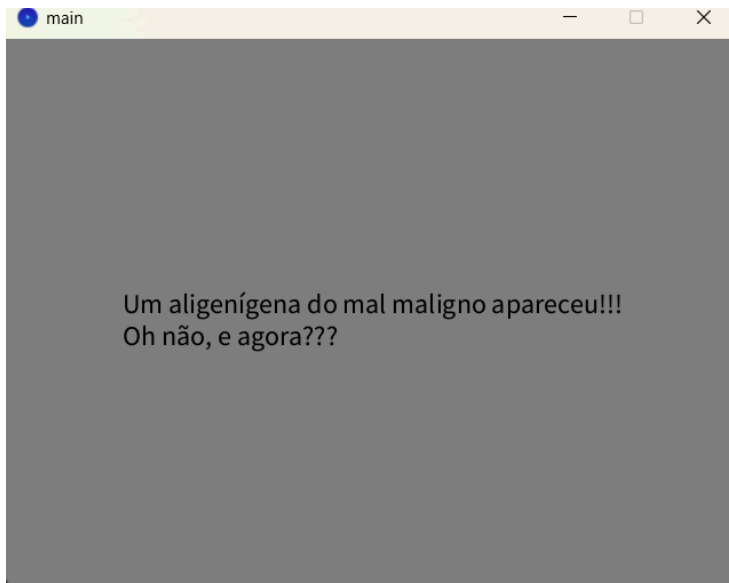
José Gabriel de Oliveira Lino - 01609620

Márcio Cavalcanti Sobel - 01578025

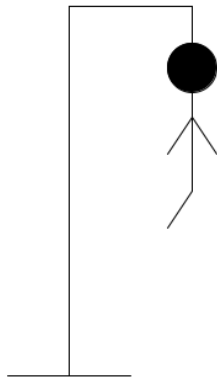
Rafael Antônio Ribeiro Galvão Mendes- 01604007

Jogo Do Marciano





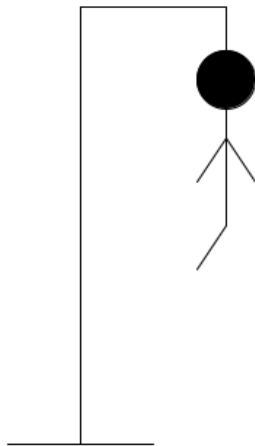
JOGO DA FORCA



JA_A

Tentativas: J A O Q W E R

Parabéns! Você venceu!

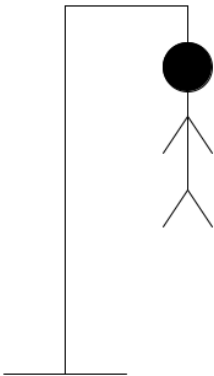


JAVA

Tentativas: J A O Q W E R V



Você perdeu! A palavra era RATO



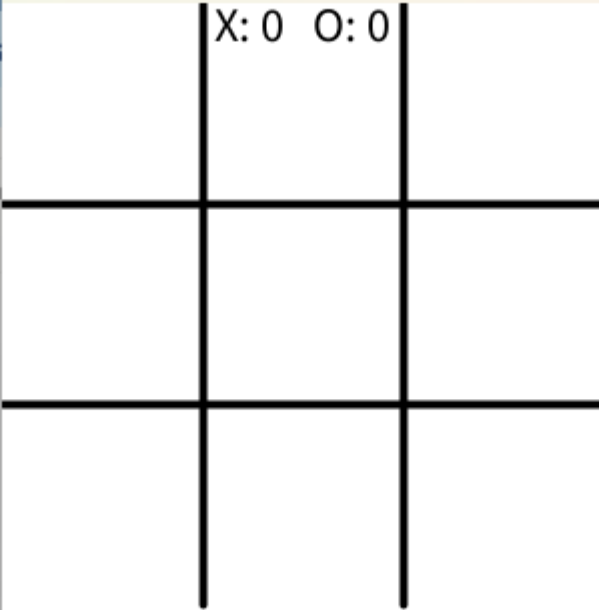
_ATO

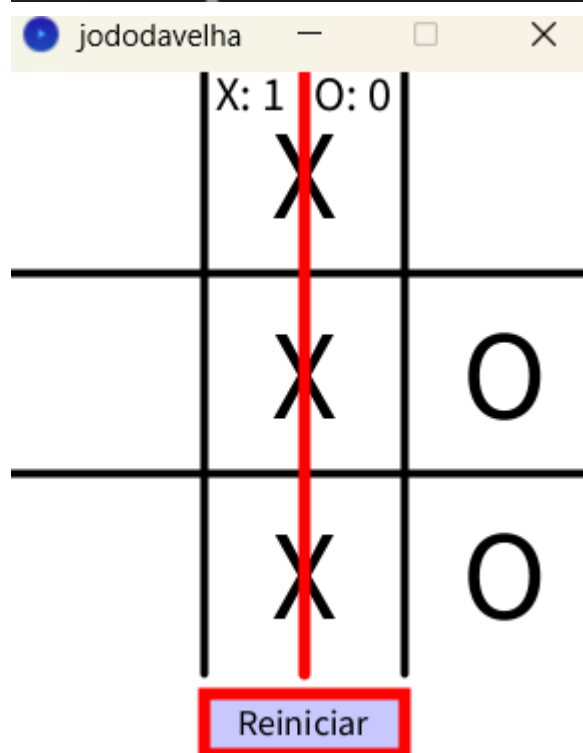
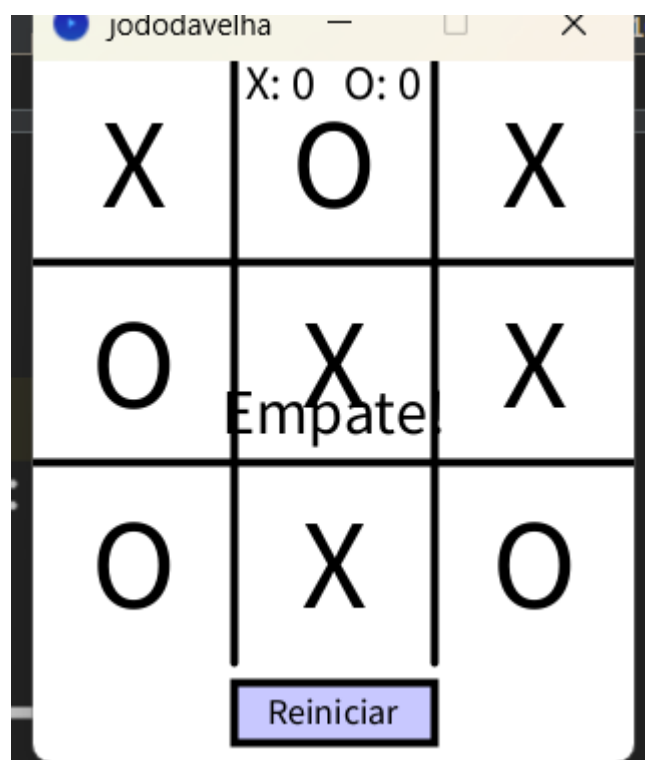
Tentativas: J O G A P T Q W E

JOGO DA VELHA

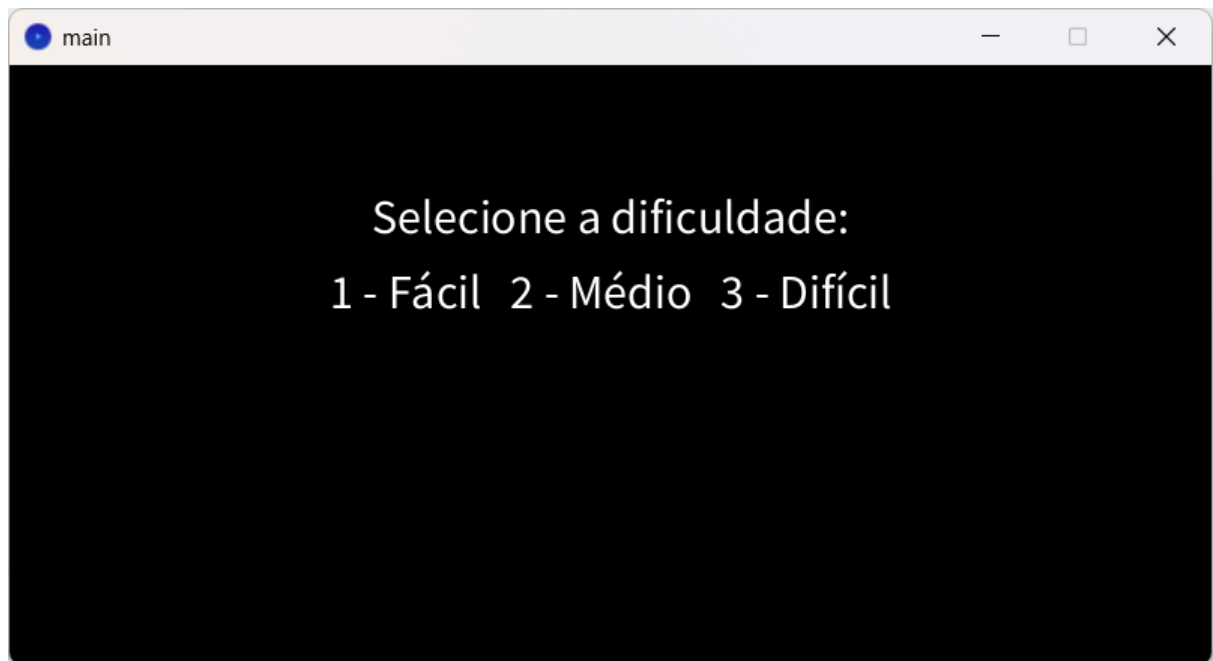
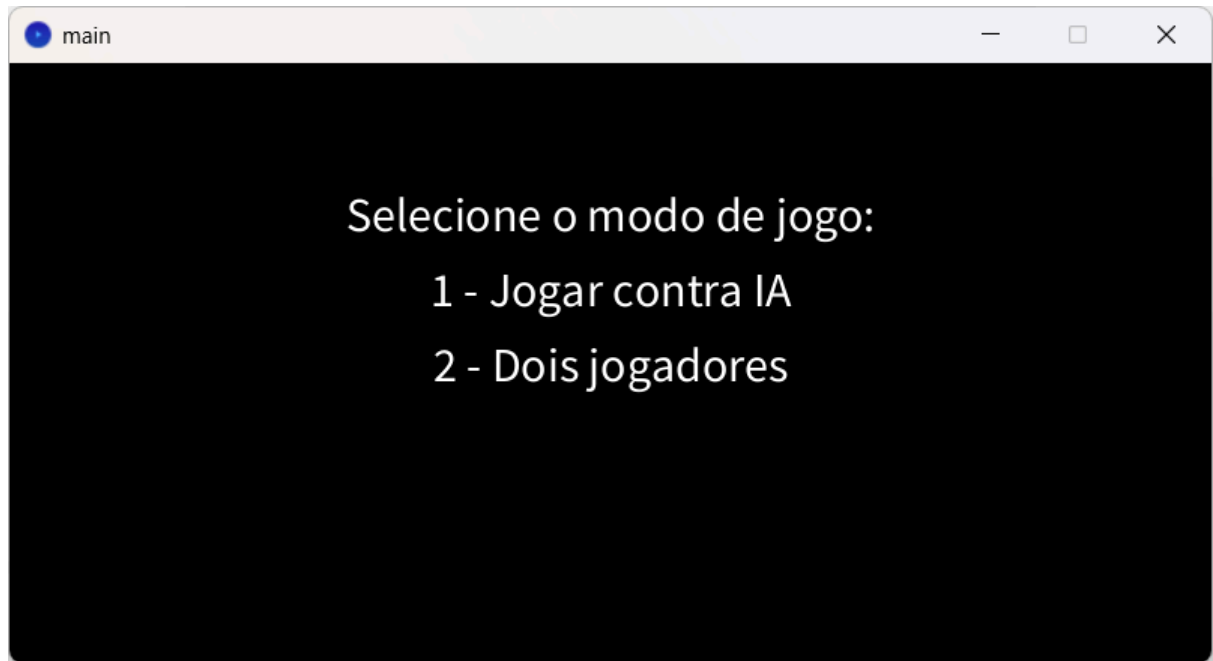


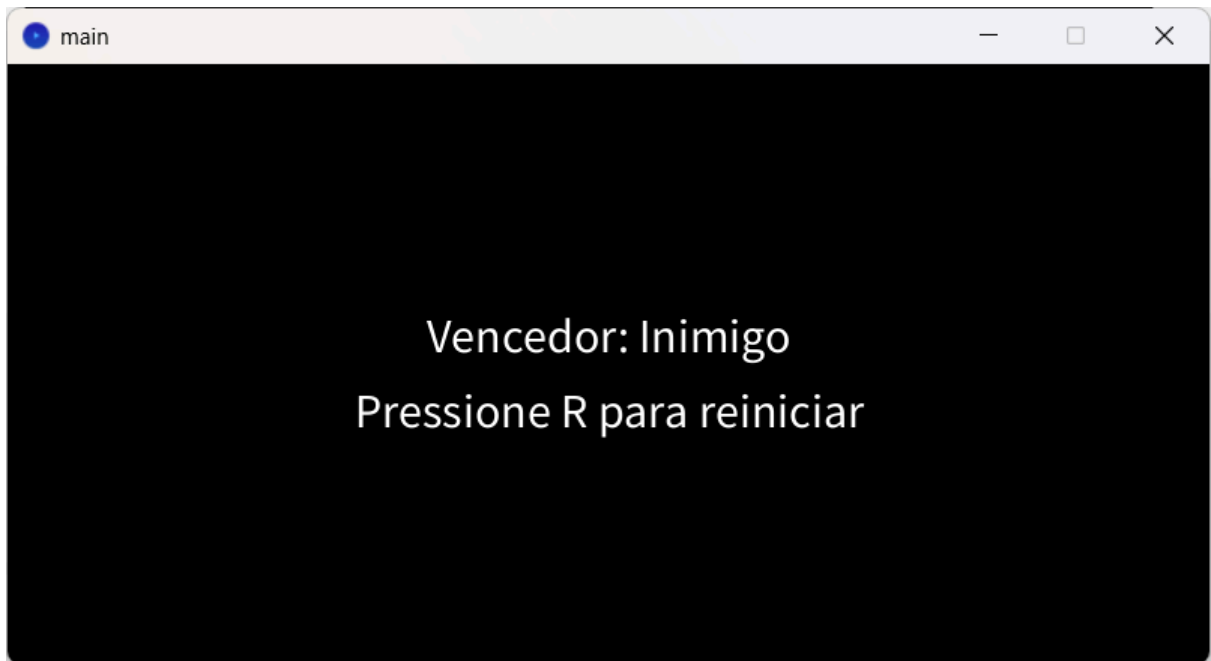
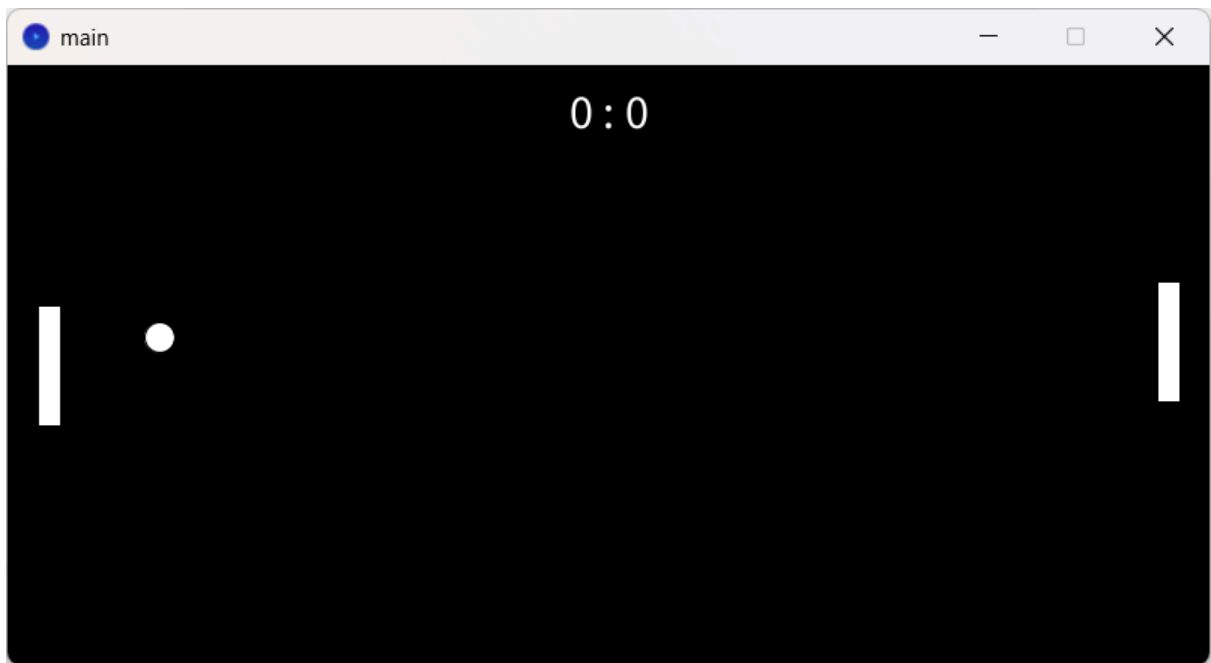
X: 0 O: 0



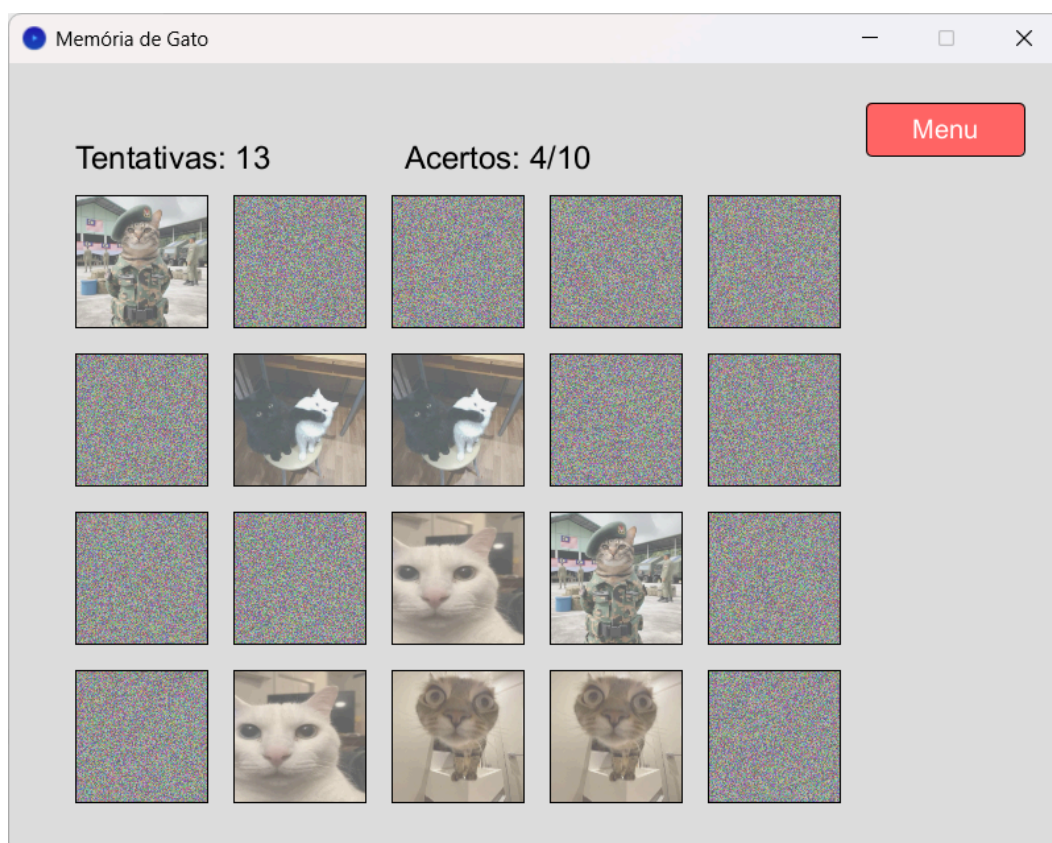


JOGO DO PONG





JOGO DA MEMÓRIA





CÓDIGOS

JOGO DO MARCIANO:

```
main.pde:  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Collections;  
import processing.sound.SoundFile;
```

```
Game game;  
SoundFile music;  
boolean playing;  
int current_speech = 0;  
int MAX_SPEECHES = 5;
```

```
void setup() {  
    size(640, 480);
```

```

game = new Game();
playing = false;
music = new SoundFile(this, "ost.mp3");
music.loop();
music.amp(0.2);
}

void draw() {
    background(125);
    fill(0);

    if (!playing) {
        textSize(24);
        String text = "";
        switch (current_speech) {
            case 0:
                text = "Era uma vez um jovem chamado\nJoel que estava tranquilamente\nem sua casa...";
                break;
            case 1:
                text = "Ele estava cansado de\nficar tanto tempo sem fazer\nnada em sua casa...";
                break;
            case 2:
                text = "Então resolveu sair para dar uma volta,\naté que DE REPENTE...";
                break;
            case 3:
                text = "Um alienígena do mal maligno apareceu!!!\nOh não, e agora???";
                break;
            case 4:
                text = "Ele está tentando se comunicar...????";
                break;
            case 5:
                text = "Ele quer que você adivinhe o número entre 1 e 100.";
                break;
        }

        text_centered(text);
    }
    if (playing) game.draw();
}

void keyPressed() {
    if (playing) game.keyPressed();
    if (!playing && ++current_speech > MAX_SPEECHES) {
        playing = true;
    }
}

```

```
private void text_centered(String text) {  
    float text_width = textWidth(text);  
    text(text, (width / 2) - (text_width / 2), (height / 2));  
}
```

game.pde:

```
import java.util.Random;
```

```
int MAX_NUMBER = 100;
```

```
int MAX_GUESSES = 10;
```

```
class Game {  
    Random random;  
    int secret_number = -1;  
    int guesses = 0;  
    int current_guess = 0;  
    int last_guess = 0;  
    boolean guessed = false;  
    boolean game_over = false;  
    List<Integer> best_guesses = new ArrayList<Integer>();
```

```
    Game() {  
        random = new Random();  
        this.generate_random_number();  
    }
```

```
    public void draw() {  
        if (guessed) {  
            print_victory_prompt();  
            return;  
        }
```

```
        if (game_over) {  
            print_game_over_prompt();  
            return;  
        }
```

```
        if (guesses > 0) {  
            textSize(24);  
            text("Tentativa: " + guesses, 20, 40);  
        }
```

```
        if (last_guess != 0) {  
            int offset = 15;  
            print_centered_text("Tentativa anterior: " + last_guess, offset);  
            print_guessed_number_status(-offset);  
        }
```

```

    }

    print_guess_prompt();
    print_best_guesses();
}

public void keyPressed() {
    int x = key - '0';
    if (x > -1 && x < 10) append_input(x);
    if (key == BACKSPACE) pop_number();
    if (key == ENTER) try_guess();
}

private void print_best_guesses() {
    int size = best_guesses.size();
    if (size == 0) return;

    float x = 20;
    int gap = 10;

    text("Melhores tentativas:", x, height - 35);

    for (int guess : best_guesses) {
        String guess_str = String.valueOf(guess);
        text(guess_str, x, height - 10);
        x += textWidth(guess_str) + gap;

        stroke(5);
        line(x, height - 25, x, height - 5);
        stroke(1);

        x += gap;
    }
}

private void print_victory_prompt() {
    int offset = 15;
    print_centered_text("Você venceu! O número era: " + secret_number, -offset);
    print_centered_text("Você conseguiu em " + guesses + " tentativas.", offset);

    print_centered_text("Pressione \"Enter\" para jogar novamente.", height / 2 - 20);
}

private void print_game_over_prompt() {
    int offset = 15;
    print_centered_text("Você perdeu! O número era: " + secret_number, -offset);
    print_centered_text("Você atingiu o número máximo de " + MAX_GUESSES + " tentativas.", offset);
}

```

```
    print_centered_text("Pressione \"Enter\" para jogar novamente.", height / 2 - 20);  
}
```

```
private void print_guess_prompt() {  
    float y = height * 0.75;  
    String guess_number_prompt = "Digite sua tentativa: ";  
    textSize(20);  
    text(guess_number_prompt, 20, y);  
    float prompt_width = textWidth(guess_number_prompt);
```

```
    if (current_guess != 0) {  
        text(current_guess, int(prompt_width) + 40, y);  
    }
```

```
    String confirm_text = "Pressione \"Enter\" para confirmar!";  
    float confirm_text_width = textWidth(confirm_text);  
    text(confirm_text, width - confirm_text_width - 20, y);  
}
```

```
private void print_guessed_number_status(int y_offset) {  
    String text = "";
```

```
    if (last_guess > secret_number) text = "Muito alto!";  
    else if (last_guess < secret_number) text = "Muito baixo!";
```

```
    print_centered_text(text, y_offset);  
}
```

```
private void append_input(int x) {  
    int new_guess = (current_guess * 10) + x;  
    if (new_guess > MAX_NUMBER) return;  
    current_guess = new_guess;  
}
```

```
private void pop_number() {  
    current_guess /= 10;  
}
```

```
private void try_guess() {  
    if (current_guess == 0) return;  
    guesses++;
```

```
    if (guessed || game_over) {  
        guessed = false;  
        game_over = false;  
        current_guess = 0;  
        guesses = 0;
```

```

        last_guess = 0;
        generate_random_number();
    }

    if (guesses > MAX_GUESSES) {
        game_over = true;
        return;
    }

    if (current_guess == secret_number) {
        guessed = true;
        best_guesses.add(guesses);
        Collections.sort(best_guesses);
        return;
    }

    last_guess = current_guess;
    current_guess = 0;
}

private void print_centered_text(String text, int y_offset) {
    float text_width = textWidth(text);
    text(text, (width / 2) - (text_width / 2), (height / 2) + y_offset);
}

private void generate_random_number() {
    secret_number = random.nextInt(MAX_NUMBER) + 1;
}
}

```

JOGO DA FORCA

```

String[] palavras = {"PROCESSING", "JAVA", "GRAFICOS", "JOGO", "ANIMACAO",
"AJUDA", "BOLO", "LOJA", "TELHADO", "FACULDADE", "PROJETO", "RATO"};
String palavraEscolhida;
char[] palavraOculta;
boolean[] letrasCorretas;
int erros = 0;
char[] tentativas = new char[26];
int numTentativas = 0;
boolean fimDeJogo = false;

void setup() {
    size(500, 500);
    textSize(32);
    escolherPalavra();
}

```

```

}

void escolherPalavra() {
    palavraEscolhida = palavras[int(random(palavras.length))];
    palavraOcultada = new char[palavraEscolhida.length()];
    letrasCorretas = new boolean[palavraEscolhida.length()];
    tentativas = new char[26];
    numTentativas = 0;
    erros = 0;
    fimDeJogo = false;

    for (int i = 0; i < palavraOcultada.length; i++) {
        palavraOcultada[i] = '_';
    }
    loop(); // Garante que o jogo continue caso tenha parado
}

void draw() {
    background(255);
    desenharForca();
    exibirPalavra();
    exibirTentativas();
    verificarVitoria();
}

void desenharForca() {
    stroke(0);
    line(100, 400, 200, 400); // Base
    line(150, 400, 150, 100); // Poste
    line(150, 100, 250, 100); // Haste superior
    line(250, 100, 250, 130); // Corda

    if (erros > 0) ellipse(250, 150, 40, 40); // Cabeça
    if (erros > 1) line(250, 170, 250, 250); // Corpo
    if (erros > 2) line(250, 190, 230, 220); // Braço esquerdo
    if (erros > 3) line(250, 190, 270, 220); // Braço direito
    if (erros > 4) line(250, 250, 230, 280); // Perna esquerda
    if (erros > 5) line(250, 250, 270, 280); // Perna direita
}

void exibirPalavra() {
    fill(0);
    textAlign(CENTER);
    text(new String(palavraOcultada), width / 2, 450);
}

void exibirTentativas() {
    fill(0);

```

```

    textSize(16);
    textAlign(LEFT);
    String letras = "Tentativas: ";
    for (int i = 0; i < numTentativas; i++) {
        letras += tentativas[i] + " ";
    }
    text(letras, 10, 480);
}

```

```

void keyPressed() {
    if (fimDeJogo) {
        escolherPalavra(); // Reinicia o jogo se estiver finalizado
        return;
    }
}

```

```

    char letra = Character.toUpperCase(key);
    if (letra >= 'A' && letra <= 'Z') {
        if (!tentativaJaFeita(letra)) {
            tentativas[numTentativas++] = letra;
            verificarLetra(letra);
        }
    }
}

```

```

boolean tentativaJaFeita(char letra) {
    for (int i = 0; i < numTentativas; i++) {
        if (tentativas[i] == letra) return true;
    }
    return false;
}

```

```

void verificarLetra(char letra) {
    boolean acertou = false;
    for (int i = 0; i < palavraEscolhida.length(); i++) {
        if (palavraEscolhida.charAt(i) == letra) {
            palavraOcultada[i] = letra;
            letrasCorretas[i] = true;
            acertou = true;
        }
    }
    if (!acertou) erros++;
}

```

```

void verificarVitoria() {
    if (erros >= 6) {
        fill(255, 0, 0);
        textAlign(CENTER);
        text("Você perdeu! A palavra era " + palavraEscolhida, width / 2, 50);
    }
}

```



```

    fimDeJogo = true;
    noLoop();
}
boolean venceu = true;
for (boolean letraCorreta : letrasCorretas) {
    if (!letraCorreta) {
        venceu = false;
        break;
    }
}
if (venceu) {
    fill(0, 255, 0);
    textAlign(CENTER);
    text("Parabéns! Você venceu!", width / 2, 50);
    fimDeJogo = true;
    noLoop();
}
}

```

JOGO DA VELHA

```

int[][] board = new int[3][3]; // Matriz do tabuleiro (0 = vazio, 1 = X, 2 = O)
int currentPlayer = 1; // Jogador atual (1 = X, 2 = O)
boolean gameOver = false; // Controla se o jogo terminou
boolean winnerFound = false; // Indica se houve vencedor (para desenhar a linha vencedora)
int[] winnerLine = new int[4]; // [0] = tipo (0=horizontal, 1=vertical, 2=diagonal principal, 3=diagonal secundária)

// Variáveis para o placar
int player1Score = 0; // Placar do jogador 1
int player2Score = 0; // Placar do jogador 2

void setup() {
    size(300, 350); // Espaço extra para o botão de reiniciar
}

void draw() {
    background(255);

    // Desenha o placar no topo, com o ajuste para não cobrir o tabuleiro
    textSize(24);
    textAlign(CENTER, CENTER);
    fill(0);

```

```
text("X: " + player1Score + " O: " + player2Score, width / 2, 10); // Placar ajustado para y = 40
```

```
stroke(0);
strokeWeight(4);
// Desenha as linhas do tabuleiro
line(100, 0, 100, 300);
line(200, 0, 200, 300);
line(0, 100, 300, 100);
line(0, 200, 300, 200);
```

```
textSize(64);
textAlign(CENTER, CENTER);
fill(0);
// Desenha os símbolos no tabuleiro
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
        int x = j * 100 + 50;
        int y = i * 100 + 50;
        if (board[i][j] == 1) {
            text("X", x, y);
        } else if (board[i][j] == 2) {
            text("O", x, y);
        }
    }
}
```

```
// Se o jogo acabou, exibe a linha vencedora ou mensagem de empate e o botão de reiniciar
```

```
if (gameOver) {
    if (winnerFound) {
        stroke(255, 0, 0);
        strokeWeight(6);
        float startX = 0, startY = 0, endX = 0, endY = 0;
        if (winnerLine[0] == 0) { // Horizontal
            startX = 0;
            startY = winnerLine[2] * 100 + 50;
            endX = 300;
            endY = startY;
        } else if (winnerLine[0] == 1) { // Vertical
            startX = winnerLine[1] * 100 + 50;
            startY = 0;
            endX = startX;
            endY = 300;
        } else if (winnerLine[0] == 2) { // Diagonal principal
            startX = 0;
            startY = 0;
            endX = 300;
            endY = 300;
        }
    }
}
```

```

        endY = 300;
    } else if (winnerLine[0] == 3) { // Diagonal secundária
        startX = 300;
        startY = 0;
        endX = 0;
        endY = 300;
    }
    line(startX, startY, endX, endY);
} else { // Se não houve vencedor, é empate
    fill(0);
    textSize(32);
    text("Empate!", width / 2, height / 2);
}
// Desenha o botão de reiniciar
fill(200, 200, 255);
rect(100, 310, 100, 30);
fill(0);
textSize(18);
text("Reiniciar", 150, 325);
}
}

void mousePressed() {
    // Se o jogo terminou, verifica se o clique foi no botão de reiniciar
    if (gameOver) {
        if (mouseX > 100 && mouseX < 200 && mouseY > 310 && mouseY < 340) {
            resetGame();
        }
        return;
    }
}

int col = mouseX / 100;
int row = mouseY / 100;

if (row >= 0 && row < 3 && col >= 0 && col < 3 && board[row][col] == 0) {
    board[row][col] = currentPlayer;

    if (checkWinner(currentPlayer)) {
        gameOver = true;
        winnerFound = true;
        if (currentPlayer == 1) {
            player1Score++; // Incrementa o placar de X
        } else {
            player2Score++; // Incrementa o placar de O
        }
    }
} else if (checkDraw()) {
    gameOver = true;
    winnerFound = false;
}

```

```

    } else {
        currentPlayer = (currentPlayer == 1) ? 2 : 1;
    }
}
}

```

```

boolean checkWinner(int player) {
    // Verifica linhas e colunas
    for (int i = 0; i < 3; i++) {
        if (board[i][0] == player && board[i][1] == player && board[i][2] == player) {
            winnerLine = new int[]{0, 0, i, 2}; // Linha horizontal
            return true;
        }
        if (board[0][i] == player && board[1][i] == player && board[2][i] == player) {
            winnerLine = new int[]{1, i, 0, 2}; // Linha vertical
            return true;
        }
    }

    // Verifica diagonais
    if (board[0][0] == player && board[1][1] == player && board[2][2] == player) {
        winnerLine = new int[]{2, 0, 0, 2}; // Diagonal principal
        return true;
    }
    if (board[0][2] == player && board[1][1] == player && board[2][0] == player) {
        winnerLine = new int[]{3, 2, 0, 0}; // Diagonal secundária
        return true;
    }
    return false;
}

```

```

boolean checkDraw() {
    // Se houver alguma casa vazia, não é empate
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (board[i][j] == 0) {
                return false;
            }
        }
    }
    return true;
}

```

```

void resetGame() {
    board = new int[3][3]; // Limpa o tabuleiro
    currentPlayer = 1;      // Reinicia para o jogador 1
    gameOver = false;      // Reseta o estado do jogo
    winnerFound = false;   // Reseta a flag do vencedor
}

```

```
winnerLine = new int[4]; // Limpa a linha vencedora  
}
```

JOGO DO PONG

```
int modoSelecionado = -1;  
int dificuldadeSelecionada = -1;
```

```
float ballX, ballY;
```

```
float ballSpeedX, ballSpeedY;  
float ballSize = 20;  
float velocidadeInicialX, velocidadeInicialY;
```

```
float paddleWidth = 15, paddleHeight;  
float playerX = 20, playerY;  
float enemyX, enemyY;  
float enemySpeed;
```

```
int playerScore = 0;  
int enemyScore = 0;  
int scoreLimit = 5;
```

```
boolean gameOver = false;  
String winner = "";
```

```
int tempoUltimoReset;  
int intervaloAumento = 5000;
```

```
int velocidadePlayer = 10;
```

```
void setup() {  
    size(800, 400);  
    textAlign(CENTER, CENTER);  
    textSize(32);  
}
```

```
void draw() {  
    background(0);  
  
    if (modoSelecionado == -1) {  
        mostrarMenuModo();  
        return;  
    }  
}
```

```

if (modoSeleccionado == 1 && dificuldadeSeleccionada == -1) {
    mostrarMenuDificuldade();
    return;
}

if (gameOver) {
    fill(255);
    text("Vencedor: " + winner, width / 2, height / 2 - 20);
    text("Pressione R para reiniciar", width / 2, height / 2 + 30);
    return;
}

if (millis() - tempoUltimoReset > intervaloAumento) {
    ballSpeedX *= 1.2;
    ballSpeedY *= 1.2;
    tempoUltimoReset = millis();
}

fill(255);
ellipse(ballX, ballY, ballSize, ballSize);
rect(playerX, playerY, paddleWidth, paddleHeight);
rect(enemyX, enemyY, paddleWidth, paddleHeight);
text(playerScore + " : " + enemyScore, width / 2, 30);

ballX += ballSpeedX;
ballY += ballSpeedY;

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

if (ballX < playerX + paddleWidth && ballY > playerY && ballY < playerY + paddleHeight) {
    ballSpeedX *= -1;
    ballX = playerX + paddleWidth;
}

if (ballX > enemyX - ballSize && ballY > enemyY && ballY < enemyY + paddleHeight) {
    ballSpeedX *= -1;
    ballX = enemyX - ballSize;
}

if (ballX < 0) {
    enemyScore++;
    checkWinner();
    resetBall(true);
}

```

```

    if (ballX > width) {
        playerScore++;
        checkWinner();
        resetBall(true);
    }

    if (modoSelecionado == 1) {
        if (ballY > enemyY + paddleHeight / 2) {
            enemyY += enemySpeed;
        } else {
            enemyY -= enemySpeed;
        }
        enemyY = constrain(enemyY, 0, height - paddleHeight);
    }

    handlePlayerMovement();
}

void mostrarMenuModo() {
    fill(255);
    text("Selecione o modo de jogo:", width / 2, 100);
    text("1 - Jogar contra IA", width / 2, 150);
    text("2 - Dois jogadores", width / 2, 200);
}

void mostrarMenuDificuldade() {
    fill(255);
    text("Selecione a dificuldade:", width / 2, 100);
    text("1 - Fácil  2 - Médio  3 - Difícil", width / 2, 150);
}

void escolherDificuldade(int nivel) {
    if (nivel == 1) {
        paddleHeight = 120;
        velocidadeInicialX = 3;
        velocidadeInicialY = 2;
        enemySpeed = 2;
    } else if (nivel == 2) {
        paddleHeight = 80;
        velocidadeInicialX = 5;
        velocidadeInicialY = 3;
        enemySpeed = 4;
    } else if (nivel == 3) {
        paddleHeight = 60;
        velocidadeInicialX = 7;
        velocidadeInicialY = 4;
        enemySpeed = 6;
    }
}

```

```

    iniciarJogo();
    dificuldadeSelecionada = nivel;
}

```

```

void iniciarJogo() {
    ballX = width / 2;
    ballY = height / 2;
    playerY = height / 2 - paddleHeight / 2;
    enemyX = width - playerX - paddleWidth;
    enemyY = height / 2 - paddleHeight / 2;
    resetBall(false);
}

```

```

void resetBall(boolean reiniciarVelocidade) {
    ballX = width / 2;
    ballY = height / 2;
    ballSpeedX = velocidadeInicialX * (random(1) > 0.5 ? 1 : -1);
    ballSpeedY = random(-velocidadeInicialY, velocidadeInicialY);

```

```

    if (reiniciarVelocidade) {
        tempoUltimoReset = millis(); // zera o tempo de contagem de aumento
    }
}

```

```

void checkWinner() {
    if (playerScore >= scoreLimit) {
        gameOver = true;
        winner = "Jogador 1";
    } else if (enemyScore >= scoreLimit) {
        winner = (modoSelecionado == 2) ? "Jogador 2" : "Inimigo";
        gameOver = true;
    }
}

```

```

void handlePlayerMovement() {
    if (!keyPressed) return;
    if (key == 'w' || key == 'W') {
        playerY -= velocidadePlayer;
    } else if (key == 's' || key == 'S') {
        playerY += velocidadePlayer;
    }
    playerY = constrain(playerY, 0, height - paddleHeight);

```

```

    if (modoSelecionado == 2) {
        if (keyCode == UP) {
            enemyY -= velocidadePlayer;
        } else if (keyCode == DOWN) {

```



```

        enemyY += velocidadePlayer;
    }
    enemyY = constrain(enemyY, 0, height - paddleHeight);
}
}

void keyPressed() {
    if (modoSelecionado == -1) {
        if (key == '1') modoSelecionado = 1;
        else if (key == '2') {
            modoSelecionado = 2;
            escolherDificuldade(2);
        }
        return;
    }

    if (modoSelecionado == 1 && dificuldadeSelecionada == -1) {
        if (key == '1') escolherDificuldade(1);
        else if (key == '2') escolherDificuldade(2);
        else if (key == '3') escolherDificuldade(3);
        return;
    }

    if (modoSelecionado == 2) {
        if (keyCode == UP) {
            enemyY -= 20;
        } else if (keyCode == DOWN) {
            enemyY += 20;
        }
        enemyY = constrain(enemyY, 0, height - paddleHeight);
    }

    if (gameOver && (key == 'r' || key == 'R')) {
        playerScore = 0;
        enemyScore = 0;
        gameOver = false;
        winner = "";
        resetBall(true);
    }
}
}

```

JOGO DA MEMORIA

```
import processing.sound.*;
```

```
SoundFile acertoSound;
SoundFile erroSound;
SoundFile musicaMenu;
PImage[] catImages = new PImage[10];
int[] indicesCartas = new int[20];
PImage[] cardBacks;
boolean[] cardFlipped;
boolean[] cardMatched;
int firstCard = -1;
int secondCard = -1;
int tentativas = 0;
int acertos = 0;
int estadoJogo = 0;
int[] records = new int[10];
PFont font;
```

```
int tempoVirada = 1000;
int tempoUltimaVirada = 0;
boolean esperandoParaDesvirar = false;
```

```
void setup() {
  size(800, 600);
  surface.setTitle("Memória de Gato");
```

```
  try {
    acertoSound = new SoundFile(this, "acerto.wav");
    erroSound = new SoundFile(this, "erro.wav");
    musicaMenu = new SoundFile(this, "menuSound.wav");
    musicaMenu.loop();
  } catch (Exception e) {
    println("Erro ao carregar sons: " + e.getMessage());
  }
```

```
  font = createFont("Arial", 24);
  textFont(font);
```

```
  for (int i = 0; i < records.length; i++) {
    records[i] = 999;
  }
```

```
  for (int i = 0; i < 10; i++) {
    catImages[i] = loadImage("imagem" + (i+1) + ".jpg");
```

```

        catImages[i].resize(100, 100);
    }

    prepararJogo();
}

void prepararJogo() {

    cardBacks = new PImage[20];
    for (int i = 0; i < 20; i++) {
        cardBacks[i] = createImage(100, 100, RGB);
        cardBacks[i].loadPixels();
        for (int j = 0; j < cardBacks[i].pixels.length; j++) {
            cardBacks[i].pixels[j] = color(random(100, 200), random(100, 200), random(100,
200));
        }
        cardBacks[i].updatePixels();
    }

    cardFlipped = new boolean[20];
    cardMatched = new boolean[20];
    for (int i = 0; i < 20; i++) {
        cardFlipped[i] = false;
        cardMatched[i] = false;
    }

    tentativas = 0;
    acertos = 0;
    firstCard = -1;
    secondCard = -1;
    esperandoParaDesvirar = false;

    embaralharCartas();
}

void embaralharCartas() {

    for (int i = 0; i < 10; i++) {
        indicesCartas[i] = i;
        indicesCartas[i+10] = i;
    }

    for (int i = indicesCartas.length - 1; i > 0; i--) {

```

```

        int j = (int)random(i + 1);

        int temp = indicesCartas[i];
        indicesCartas[i] = indicesCartas[j];
        indicesCartas[j] = temp;
    }
}

void draw() {
    background(220);

    if (esperandoParaDesvirar && millis() - tempoUltimaVirada > tempoVirada) {
        cardFlipped[firstCard] = false;
        cardFlipped[secondCard] = false;
        firstCard = -1;
        secondCard = -1;
        esperandoParaDesvirar = false;
    }

    if (estadoJogo == 0 && musicaMenu != null && !musicaMenu.isPlaying()) {
        musicaMenu.loop();
    } else if (estadoJogo != 0 && musicaMenu != null && musicaMenu.isPlaying()) {
        musicaMenu.stop();
    }

    switch(estadoJogo) {
        case 0: desenharMenu(); break;
        case 1: desenharJogo(); break;
        case 2: desenharRecords(); break;
    }
}

void desenharMenu() {

    fill(0);
    textSize(48);
    textAlign(CENTER, CENTER);
    text("Memória de Gato", width/2, 100);

    fill(100, 100, 255);
    rect(width/2 - 100, 200, 200, 50, 10);
    fill(255);
    textSize(32);
    text("Novo Jogo", width/2, 225);

```

```
fill(100, 255, 100);
rect(width/2 - 100, 280, 200, 50, 10);
fill(255);
text("Records", width/2, 305);
```

```
fill(255, 100, 100);
rect(width/2 - 100, 360, 200, 50, 10);
fill(255);
text("Sair", width/2, 385);
}
```

```
void desenharJogo() {

  for (int i = 0; i < 20; i++) {
    int x = 50 + (i % 5) * 120;
    int y = 100 + (i / 5) * 120;

    if (cardMatched[i]) {

      tint(255, 150);
      image(catImages[indicesCartas[i]], x, y, 100, 100);
      noTint();
    } else if (cardFlipped[i]) {

      image(catImages[indicesCartas[i]], x, y, 100, 100);
    } else {

      image(cardBacks[i], x, y, 100, 100);
    }

    stroke(0);
    noFill();
    rect(x, y, 100, 100);
  }
}
```

```
fill(0);
textSize(24);
textAlign(LEFT);
text("Tentativas: " + tentativas, 50, 80);
text("Acertos: " + acertos + "/10", 300, 80);
```

```
fill(255, 100, 100);
```

```

rect(width - 150, 30, 120, 40, 5);
fill(255);
textSize(20);
textAlign(CENTER, CENTER);
text("Menu", width - 90, 50);
}

```

```

void desenharRecords() {

```

```

    fill(0);
    textSize(48);
    textAlign(CENTER, CENTER);
    text("Melhores Recordes", width/2, 80);

```

```

    textSize(24);
    for (int i = 0; i < records.length; i++) {
        if (records[i] != 999) {
            text((i+1) + ". " + records[i] + " tentativas", width/2, 150 + i * 30);
        } else {
            text((i+1) + ". ---", width/2, 150 + i * 30);
        }
    }
}

```

```

    fill(100, 100, 255);
    rect(width/2 - 60, 500, 120, 40, 5);
    fill(255);
    textSize(20);
    text("Voltar", width/2, 520);
}

```

```

void mousePressed() {
    switch(estadosJogo) {
        case 0: verificarCliqueMenu(); break;
        case 1: verificarCliqueJogo(); break;
        case 2: verificarCliqueRecords(); break;
    }
}

```

```

void verificarCliqueMenu() {

```

```

    if (mouseX > width/2 - 100 && mouseX < width/2 + 100 &&
        mouseY > 200 && mouseY < 250) {
        estadosJogo = 1;
        prepararJogo();
    }
}

```

```
if (mouseX > width/2 - 100 && mouseX < width/2 + 100 &&
    mouseY > 280 && mouseY < 330) {
    estadoJogo = 2;
}
```

```
if (mouseX > width/2 - 100 && mouseX < width/2 + 100 &&
    mouseY > 360 && mouseY < 410) {
    exit();
}
}
```

```
void verificarCliqueJogo() {
```

```
    if (mouseX > width - 150 && mouseX < width - 30 &&
        mouseY > 30 && mouseY < 70) {
        estadoJogo = 0;
        return;
    }
```

```
    if (esperandoParaDesvirar) {
        return;
    }
```

```
    for (int i = 0; i < 20; i++) {
        int x = 50 + (i % 5) * 120;
        int y = 100 + (i / 5) * 120;
```

```
        if (mouseX > x && mouseX < x + 100 &&
            mouseY > y && mouseY < y + 100 &&
            !cardMatched[i] && !cardFlipped[i]) {
```

```
            if (acertoSound != null && acertoSound.isPlaying()) acertoSound.stop();
            if (erroSound != null && erroSound.isPlaying()) erroSound.stop();
```

```
            cardFlipped[i] = true;
```

```
            if (firstCard == -1) {
                firstCard = i;
            } else {
                secondCard = i;
                tentativas++;
```

```
            if (indicesCartas[firstCard] == indicesCartas[secondCard]) {
```

```

        if (acertoSound != null) acertoSound.play();
        cardMatched[firstCard] = true;
        cardMatched[secondCard] = true;
        acertos++;

        firstCard = -1;
        secondCard = -1;

        if (acertos == 10) {
            adicionarRecord(tentativas);
            delay(1000);
            estadoJogo = 0;
        }
        } else {
            if (erroSound != null) erroSound.play();
            esperandoParaDesvirar = true;
            tempoUltimaVirada = millis();
        }
        }
        break;
    }
}

void verificarCliqueRecords() {

    if (mouseX > width/2 - 60 && mouseX < width/2 + 60 &&
        mouseY > 500 && mouseY < 540) {
        estadoJogo = 0;
    }
}

void adicionarRecord(int tentativas) {

    for (int i = 0; i < records.length; i++) {
        if (tentativas < records[i] || records[i] == 999) {

            for (int j = records.length - 1; j > i; j--) {
                records[j] = records[j-1];
            }
            records[i] = tentativas;
            break;
        }
    }
}

```