Universidade de Vila Velha - UVV

PROJETO JOGO DO MILHÃO

Rafael Ferreira Bassul - 202305395

Vila Velha

# SUMÁRIO

1	INTRODUÇÃO	3
1.1	Objetivos	3
2	DESENVOLVIMENTO	4
2.1	Alterações	4
2.2	Fluxo	5
2.3	Abordando o Codigo	5
3	CONCLUSÕES	<b>2</b>
4	IMAGENS 1	13

## **RESUMO**

O jogo apresenta uma sequência de perguntas e respostas de múltipla escolha. O jogador deve responder corretamente para avançar para a próxima pergunta. Caso erre, automaticamente é GameOver. Se o jogador completar todas as perguntas, ele é parabenizado por ter completado o jogo.

## 1 INTRODUÇÃO

Este código é uma recriação do clássico "Show do Milhão"do Silvio Santos, implementada em C# com Windows Forms. Assim como no famoso programa de TV, o jogador precisa responder corretamente a uma série de perguntas para avançar e conquistar a pontuação máxima. A cada pergunta, ele enfrenta múltiplas alternativas, com sons e imagens para deixar a experiência mais imersiva.

Assim como no programa original, o jogador tem recursos de ajuda (dicas limitadas) e a possibilidade de pular uma pergunta uma vez. Se responder certo, avança para a próxima pergunta, mas se errar, o jogo termina e uma tela de fim é mostrada. Se o jogador acertar todas as perguntas, recebe uma mensagem especial de parabéns!

O charme da aplicação está na utilização de sons e imagens inspirados no programa de auditório, criando um ambiente divertido e nostálgico, perfeito para testar conhecimentos e se sentir no palco com Silvio Santos.

### 1.1 Objetivos

Este projeto é baseado em um jogo de perguntas e respostas genérico que foi previamente fornecido durante uma atividade em sala de aula. A partir do código original, foram realizadas diversas adaptações e melhorias para transformá-lo em uma versão personalizada, inspirada no famoso programa "Show do Milhão", apresentado por Silvio Santos.

## 2 DESENVOLVIMENTO

### 2.1 Alterações

Com o codigo em mãos, realizei diversas alterações e resoluções de bugs que faziam o jogador poder trapacear no jogo. As seguintes alterações foram:

#### 1. Sistema de perguntas:

- O jogo possui 11 perguntas, apresentadas uma por vez, usando imagens e sons.
- As alternativas s\(\tilde{a}\) exibidas em bot\(\tilde{o}\) es o jogador deve clicar na resposta correta.
- A função askQuestion() controla a exibição de cada pergunta e suas alternativas.

#### 2. Verificação da resposta:

- Quando o jogador clica em um botão de resposta, o evento ClickAnswerEvent verifica se a resposta está correta.
- Acerto: Toca um som de acerto e avança para a próxima pergunta.
- Erro: Toca um som de erro e leva o jogador para a tela de fim de jogo.

#### 3. Pontuação e progresso:

- A cada acerto, a pontuação aumenta e o número da próxima pergunta é carregado.
- Se todas as perguntas forem respondidas corretamente, toca um som de parabéns e exibe uma mensagem final.

#### 4. Ajuda e Pular Perguntas:

- O botão "**Ajuda**"dá uma dica ao jogador (usando a função button6\_ Click), mas ele pode ser usado apenas 3 vezes.
- O jogador pode pular uma pergunta (apenas uma vez) clicando em outro botão.

#### 5. Sons e recursos multimídia:

- Cada pergunta tem um som associado (como "1.wav", "2.wav"...) e uma imagem correspondente.
- O som de fundo é reproduzido em loop até que o jogador responda a pergunta.

### 2.2 Fluxo

- 1. O jogo inicia: A primeira pergunta é exibida, e os botões estão ativados.
- 2. O jogador responde: Se acerta, o jogo avança; se erra, a tela de fim é exibida.
- 3. Ajuda disponível: O jogador pode pedir dicas limitadas para algumas perguntas.
- 4. **Fim de jogo:** Ao completar todas as perguntas, o jogador é parabenizado e o jogo termina.

### 2.3 Abordando o Codigo

• Este trecho de código em C# é responsável por controlar a interação da interface gráfica do usuário em um jogo de perguntas e respostas. Ele garante que, em diferentes momentos (como durante o processamento ou a escolha de uma resposta), a interface se comporte adequadamente, ativando e desativando botões para evitar cliques indevidos e proporcionar feedback visual claro ao jogador.

```
public void botoesDesativados()
{
    button1.Enabled = false;
    button2.Enabled = false;
    button3.Enabled = false;
    button4.Enabled = false;
    button5.Enabled = false;
    button6.Enabled = false;
}
public void botoesAtivados()
{
    button1.Enabled = true;
    button2.Enabled = true;
    button3.Enabled = true;
    button4.Enabled = true;
}
public void processando()
{
    botoesDesativados();
    lblQuestion.Text = "Processando...";
    button1.Text = "Processando...";
    button2.Text = "Processando...";
    button3.Text = "Processando...";
    button4.Text = "Processando...";
}
```

• Esta função chamada **ajuda()** controla a habilitação ou desabilitação do botão 6 ( associado a ajuda no jogo, como "ajuda nerd"). O comportamento depende do valor de uma variável chamada flag, que parece atuar como um contador ou limitador para o uso da ajuda.

```
public void ajuda()
{
    if (flag >= 0 && flag < 3)
    {
       button6.Enabled = true;
    }
    else if (flag > 2)
    {
       button6.Enabled = false;
    }
}
```

• A função **passar**() é responsável por habilitar ou desabilitar o botão 5. O comportamento dessa função depende do valor da variável flagdois, que atua como um indicador binário (0 ou 1) para decidir se o jogador ainda pode usar essa funcionalidade.

```
public void passar()
{
    if (flagdois == 0)
    {
        button5.Enabled = true;
    }
    else
    {
        button5.Enabled = false;
    }
}
```

• A função ClickAnswerEvent é responsável por gerenciar as interações do jogador com as respostas em um jogo de perguntas e respostas. Quando um botão é clicado, o evento captura qual botão foi acionado e verifica se a resposta está correta. Se a resposta for correta, a função desabilita os botões, para o tema de fundo, toca um som de acerto, incrementa a pontuação e verifica se o jogador completou todas as perguntas. Caso afirmativo, toca uma música de vitória, exibe uma mensagem de parabéns e navega para uma nova tela.

Se o botão clicado for um botão especial (com valor 100), ele desabilita um botão adicional e atualiza uma flag. Em caso de resposta errada, a função toca um som de erro, aguarda um tempo para permitir que o jogador ouça o feedback, e então navega para uma tela de erro.

Por fim, a função prepara a próxima pergunta, garantindo uma experiência fluida e interativa, enquanto gerencia as transições entre perguntas e respostas.

```
private async void ClickAnswerEvent(object sender, EventArgs e)
{
    var senderObject = (Button)sender;
    int buttonTag = Convert.ToInt32(senderObject.Tag);
    if (buttonTag == correctAnswer)
    {
        botoesDesativados();
        Tema.Stop();
        certa.Load();
        certa.Play();
        score++;
        await Task.Delay(1000);
        if (questionNumber == totalQuestions)
        {
            parabens.Load();
            parabens.Play();
            MessageBox.Show("Parabens!!! tirou onda!");
            await Task.Delay(5742);
            Form3 form3 = new Form3();
            this.Hide();
            form3.ShowDialog();
            this.Close();
        }
    }
    else if (buttonTag == 100)
    {
        button5.Enabled = false;
        flagdois = 1;
    }
    else
    {
        errou.Load();
        errou.Play();
        await Task.Delay(2000);
        Form4 form4 = new Form4();
        this.Hide();
        form4.ShowDialog();
        this.Close();
    }
```

```
if (questionNumber == totalQuestions)
    {
        parabens.Load();
        parabens.Play();
    }
    questionNumber++;
    askQuestion(questionNumber);
}
```

 Cada caso neste código é um exemplo de como o jogo fornece perguntas de maneira interativa, controlando o fluxo de jogo e a experiência do usuário com som e imagens.
 A lógica para carregar e exibir a pergunta, ativar os botões e verificar as respostas é bem estruturada, permitindo uma navegação fluida pelas perguntas.

```
private async void askQuestion(int qnum)
{
    switch (qnum)
    {
        case 1:
            um.Load();
            um.Play();
            pictureBox1.Image = Image.FromFile(@"Resources/questions.png");
            processando();
            await Task.Delay(8300);
            Tema.Load();
            Tema.PlayLooping();
            pictureBox1.Image = Image.FromFile(@"Resources/tomate.jpg");
            lblQuestion.Text = "Que tipo de alimento é o tomate?";
            button1.Text = "Fruta";
            button2.Text = "Legume";
            button3.Text = "Verdura";
            button4.Text = "Vegetal";
            botoesAtivados();
            ajuda();
            passar();
            correctAnswer = 1;
            break;
        [....]
    }
}
```

• O código apresentado é um manipulador de eventos para um botão (button6) em um jogo de perguntas e respostas. Abaixo, explico cada parte do método button6\_Click, que fornece dicas ao jogador com base na pergunta atual:

[...]

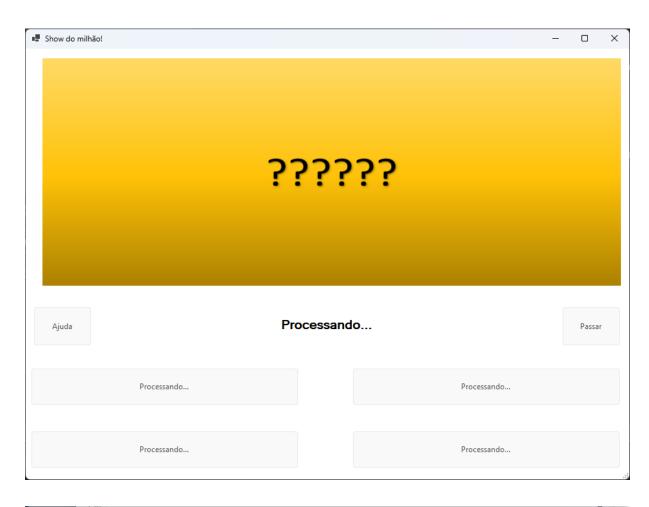
## 3 CONCLUSÕES

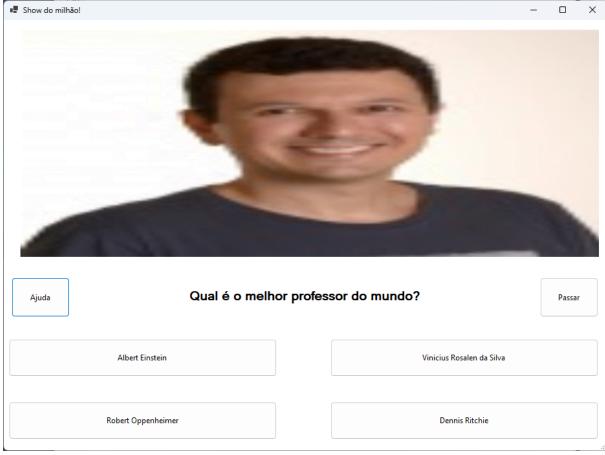
Essas modificações foram feitas com o objetivo de aproximar a experiência do usuário àquela vivida pelos participantes do Show do Milhão, ao mesmo tempo em que mantive a estrutura lógica do jogo original. Dessa forma, o projeto evoluiu de um jogo básico para uma versão mais temática e divertida, refletindo tanto os conceitos de programação abordados na sala de aula quanto a criatividade aplicada ao adaptar o conteúdo para um contexto específico.

## 4 IMAGENS

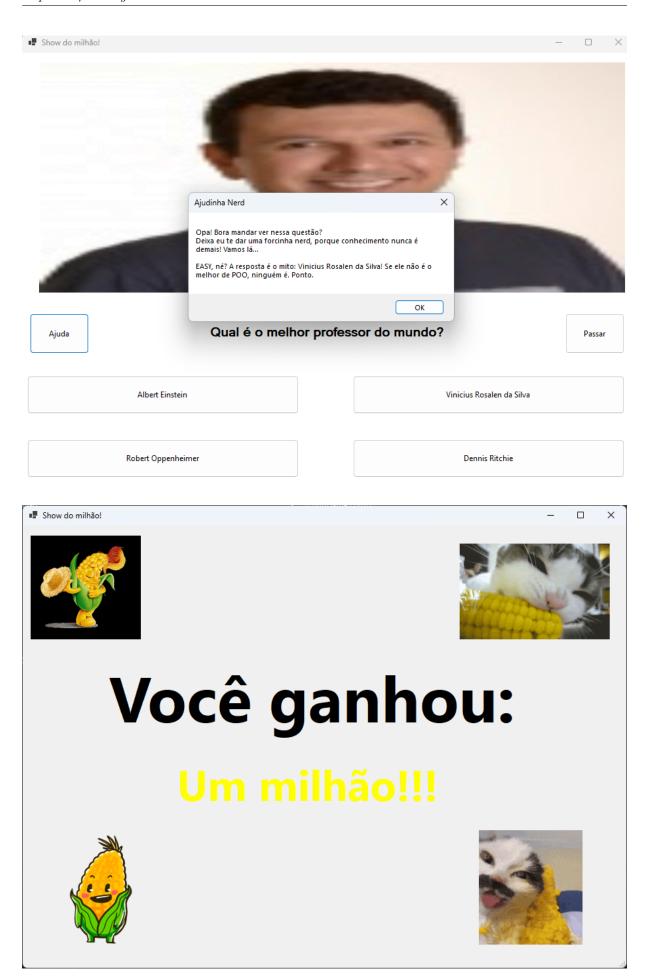


Capítulo 4. Imagens





Capítulo 4. Imagens



Capítulo 4. Imagens

