



PROJETO DE EDII

PALAVRAS CRUZADAS COM BACKTRACKING

DOCENTE: RAPHAEL OLIVEIRA
DISCENTES: DAIANE, GUILHERME, IGOR
DIAS, MATHEUS E YASMIM

O QUE É O PROJETO

E COMO JOGAR



O QUE É

Jogo de preencher uma grade com palavras usando pistas.



COMO JOGAR

Use as pistas para descobrir palavras e encaixá-las na grade, cruzando letras corretamente.

OBJETIVO DO PROJETO

O objetivo não é apenas jogar, mas demonstrar a aplicação de algoritmos de Backtracking, para resolver ou validar o preenchimento do tabuleiro.

TECNOLOGIAS IMPLEMENTADAS

Um jogo de Palavras Cruzadas desenvolvido em C puro, utilizando a biblioteca gráfica Raylib.

O QUE FOI DESENVOLVIDO

PALAVRAS CRUZADAS

INTERFACE GRÁFICA (GUI)

Um tabuleiro interativo estilo "jornal" (azul/branco) onde o usuário pode clicar e digitar.

SISTEMA DE NÍVEIS

Use as pistas para descobrir palavras e encaixá-las na grade, cruzando letras corretamente.

AJUDA INTERATIVA

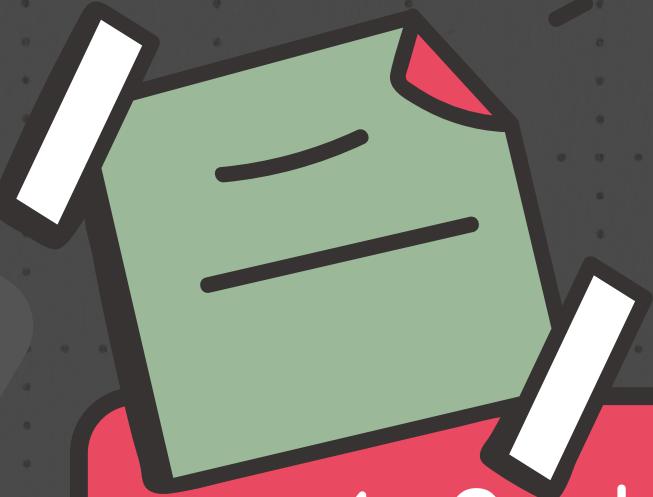
Dicas (Horizontais/Verticais) e validação de respostas.

NAVEGAÇÃO INTELIGENTE

Setas do teclado pulam automaticamente os espaços vazios (blocos pretos). Navegação no puzzle por meio de teclado e mouse.

DESTAQUES DO CÓDIGO

PALAVRAS CRUZADAS



ALGORÍTMO DE BACKTRACKING (`SRC/GAME/SOLVER.C`)

- 
- 
1. O algoritmo pega a primeira palavra vazia do grid.
 2. Busca no Dicionário todas as palavras que cabem naquele espaço (ex: 5 letras).
 3. Tenta colocar uma palavra ("Candidata").
 4. Verifica Conflitos: A letra cruza corretamente com as palavras que já estão lá?
 5. Recursão: Se deu certo, chama a si mesmo para a próxima palavra.
 6. Backtrack: Se chegar num beco sem saída (nenhuma palavra cabe), ele desfaz a última escolha (apaga a palavra) e tenta a próxima candidata.

ALGORÍTMO DE BACKTRACKING (`SRC/GAME/SOLVER.C`)

```
static bool backtrack(Grid *grid, int indicePalavra, EstadoSolver *estado) {
    // CASO BASE: Todas as palavras foram preenchidas
    if (indicePalavra >= grid->numPalavras) {
        return verificarSolucao(grid);
    }

    Palavra *palavra = obterPalavra(grid, indicePalavra);
    // Tentar cada candidato do dicionário
    for (int i = 0; i < numCandidatos; i++) {
        // 1. Tentar colocar
        if (colocarPalavra(grid, palavra, candidatos[i])) {
            // 2. Recursão (Próxima palavra)
            if (backtrack(grid, indicePalavra + 1, estado)) {
                return true; // Sucesso!
            }
            // 3. Backtrack (Falhou, desfazer)
            removerPalavra(grid, palavra);
        }
    }
    return false;
}
```

ALGORÍTMO DE BACKTRACKING (`SRC/GAME/SOLVER.C`)

```
bool resolverPuzzle(Grid *grid, EstadoSolver *estado) {
    if (grid == NULL || estado == NULL) {
        return false;
    }
    // Inicializar estado
    inicializarSolver(grid, estado);
    // Executar backtracking a partir da primeira palavra
    bool sucesso = backtrack(grid, 0, estado);
    // Marcar como concluído
    estado->concluido = true;
    estado->encontrouSolucao = sucesso;
    return sucesso;
}
```



INTEGRACÃO COM DICIONÁRIO (`SRC/DATA/DICTIONARY.C`)

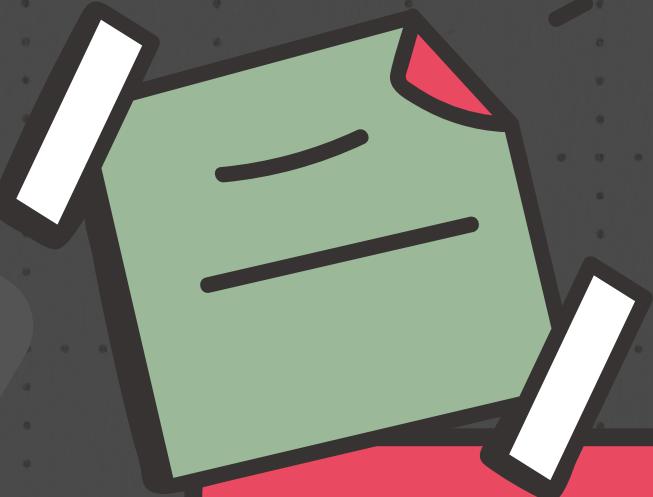
- Simula uma API de palavras.
- Carrega bancos de dados de texto (`pt.txt`, `en.txt`) na memória para o caso de API falhar ou não ter Internet.
- Permite buscas rápidas por tamanho dentro do algoritmo para montar o puzzle.

ALGORÍTMO DE BACKTRACKING (`SRC/GAME/SOLVER.C`)

```
bool dict_check_word(const char* word, char* out_def, int max_len) {
    // 1. checa API
    char temp_def[TAMANHO_MAX_DICA];
    if (fetch_word_definition(word, temp_def, sizeof(temp_def))) {
        if (out_def) {
            strncpy(out_def, temp_def, max_len - 1);
            out_def[max_len - 1] = '\0';
        }
        return true;
    }
    // 2. Arquivos como segunda opção caso API falhe
    // Tenta multiplos caminho para achar o dicionário
    const char* paths[] = {
        "data_files/dictionaries/en.txt",    "../data_files/dictionaries/en.txt",
        "../../data_files/dictionaries/en.txt"
    };
    const char* pathsPT[] = {
        "data_files/dictionaries/pt.txt",
        "../data_files/dictionaries/pt.txt",
        "../../data_files/dictionaries/pt.txt"
    };
}
```

ALGORÍTMO DE BACKTRACKING (`SRC/GAME/SOLVER.C`)

```
bool found = false;
for(int i=0; i<3; i++) {
    if(check_local_dictionary(paths[i], word, out_def, max_len)) {
        found = true;
        break;
    }
}
if(!found) {
    for(int i=0; i<3; i++) {
        if(check_local_dictionary(pathsPT[i], word, out_def, max_len)) {
            found = true;
            break;
        }
    }
}
if (found) return true;
return false;
```



BIBLIOTECA GRÁFICA (`RAYLIB`) E INTERFACE

- Utilizamos **Raylib** por ser leve e direta para C.
- **Loop do Jogo:** O `main.c` roda 60 vezes por segundo, desenhando o grid (`DrawCrosswordGrid`) e escutando cliques/teclado (`UpdateInterface`).
- **Renderização:** Desenhamos retângulo por retângulo, calculando coordenadas X/Y baseadas na matriz do tabuleiro.



OBRIGADO!

