ESTUDO COMPARATIVO ENTRE ALGORITMO A* E BUSCA EM LARGURA PARA PLANEJAMENTO DE PERSONAGENS EM JOGOS DO TIPO PACMAN

Acadêmica: Jeanita Bassani da Silva

Orientador: Prof. Paulo César Rodacki Gomes

ROTEIRO DA APRESENTAÇÃO

- Introdução;
- Objetivos;
- Revisão bibliográfica;
- Desenvolvimento do protótipo do jogo;
- Implementação;
- Resultados;
- Conclusão;
- Trabalhos futuros.

INTRODUÇÃO

- O jogo PacMan;
- Planejamento de caminho de personagens;
- Caminho mínimo;
- Métodos de busca.

O Jogo PacMan

OBJETIVOS

- Estudo comparativo entre algoritmos de busca em largura, busca em profundidade e algoritmo A*;
- Desenvolver protótipo de jogo PacMan:
  - Busca em largura;
  - Busca em profundidade;
  - Busca heurística (A*).
- Comparação de resultados dos algoritmos implementados.

Jogos por computador

- Definição de um jogo;
- Roteiro de um jogo:
  - Sinopse – trama do jogo;
  - Personagem – descrição, características;
  - Cenário – local físico, onde ocorre;
  - Comentários finais – observações, sugestões.
- Tipos de jogos
  - Arcades – interface e o jogo simples;
  - Aventura baseado em texto;
  - Aventura gráficos – “Uma figura vale mais que mil palavras”;
  - Estratégia e de guerra – objetivo de controlar grandes grupos de unidades para atingir uma meta.

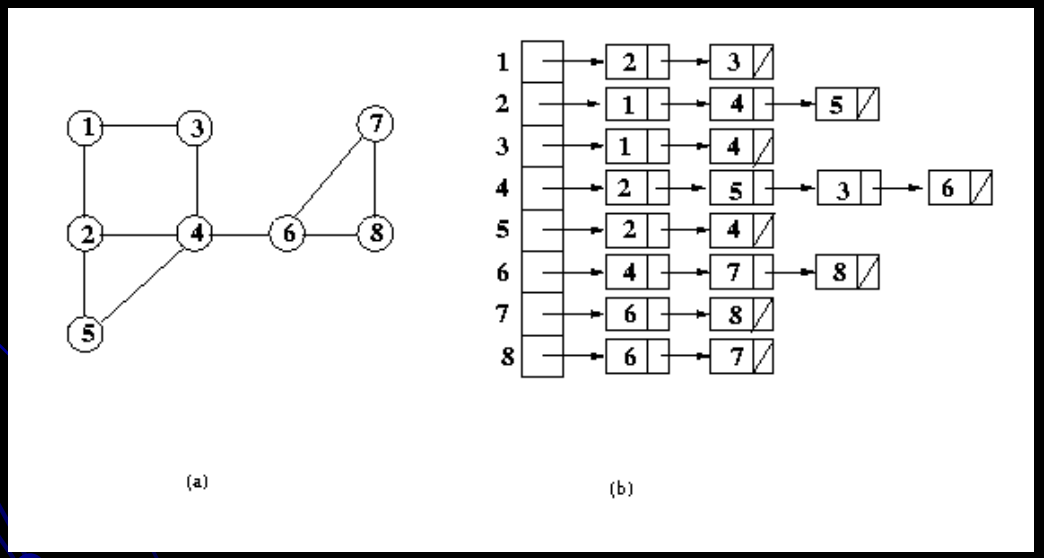
Exemplos de jogos

Civilization III – Estratégia e de guerra

Escape from Monkey Island - Aventura gráfico

Problema de caminho mínimo

- Algoritmos de grafos;
- Grafo G = (V, E);
- Lista de adjacência.



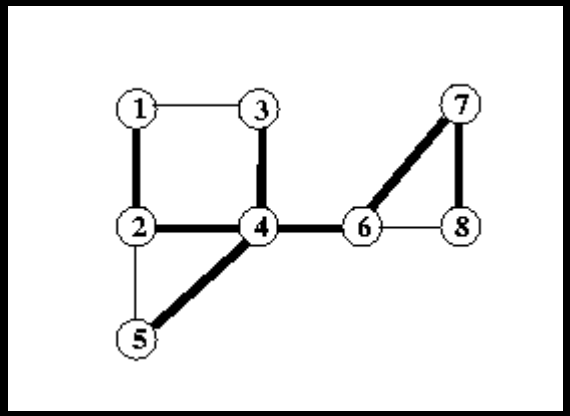
a) Grafo não orientado com 8 vértices
b) Grafo representado como uma lista de adjacência

BUSCA EM GRAFOS

- Localizar um vértice dentro de um grafo;
- “Exploram” a estrutura do grafo;
- Servem de base para outros algoritmos em grafos;
- Métodos mais comuns:
  - Busca em largura;
  - Busca em profundidade.

Busca em profundidade

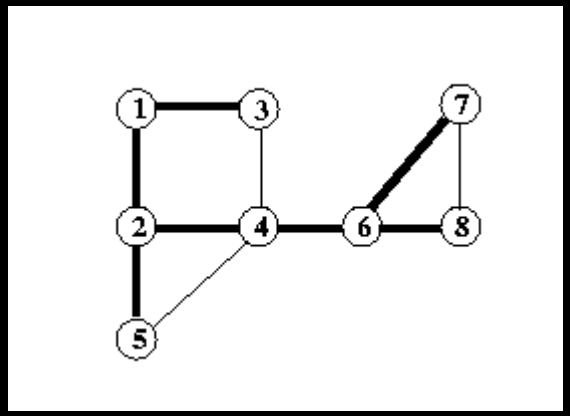
- Pilha de vértices;
- Árvore de busca;
- Dá preferência aos vértices que estão mais distantes da árvore de busca.



Árvore que a busca em profundidade gerou

Busca em largura

- Fila de vértices;
- Percorre caminhos mínimos (em quantidade de arestas percorridas);
- Esta busca examina todos os vértices de certo nível do grafo antes dos vértices do nível abaixo;



Árvore que a busca em largura gerou

BUSCA HEURÍSTICA

- Conhecimento específico do problema;
- O algoritmo A*;
- A* adaptado;
- Busca em profundidade adaptado.

O algoritmo A*

- Caminho entre dois vértices em um grafo;
- Marca as posições exploradas;
- Atributos F, G e H:
  - G - é o custo para chegar até o vértice desejado;
  - H – é um valor estimado;
  - F – é a soma de G + H.
- Mantém duas listas de vértices.

A* adaptado

- Derivado do A*;
- Número limitado de vértices da lista de abertos;
- Aperfeiçoa em velocidade;
- Porém nem sempre encontra o menor caminho.

Busca em profundidade adaptado

- Mescla o busca em profundidade com o A*;
- Testa os vértices na ordem de menor custo.

DESENVOLVIMENTO DO PROTÓTIPO DO JOGO

- Requisitos funcionais:
  - RF – Selecionar o modo de debug;
  - RF – Selecionar uma busca: busca em largura, busca em profundidade, busca em profundidade adaptada, A* e o A* adaptado;
  - RF – Visualização gráfica do ambiente do jogo PacMan.
- Requisito não funcional:
  - RNF – Utilizar a linguagem de programação *Object Pascal* com ambiente Delphi.

Diagrama de Classes

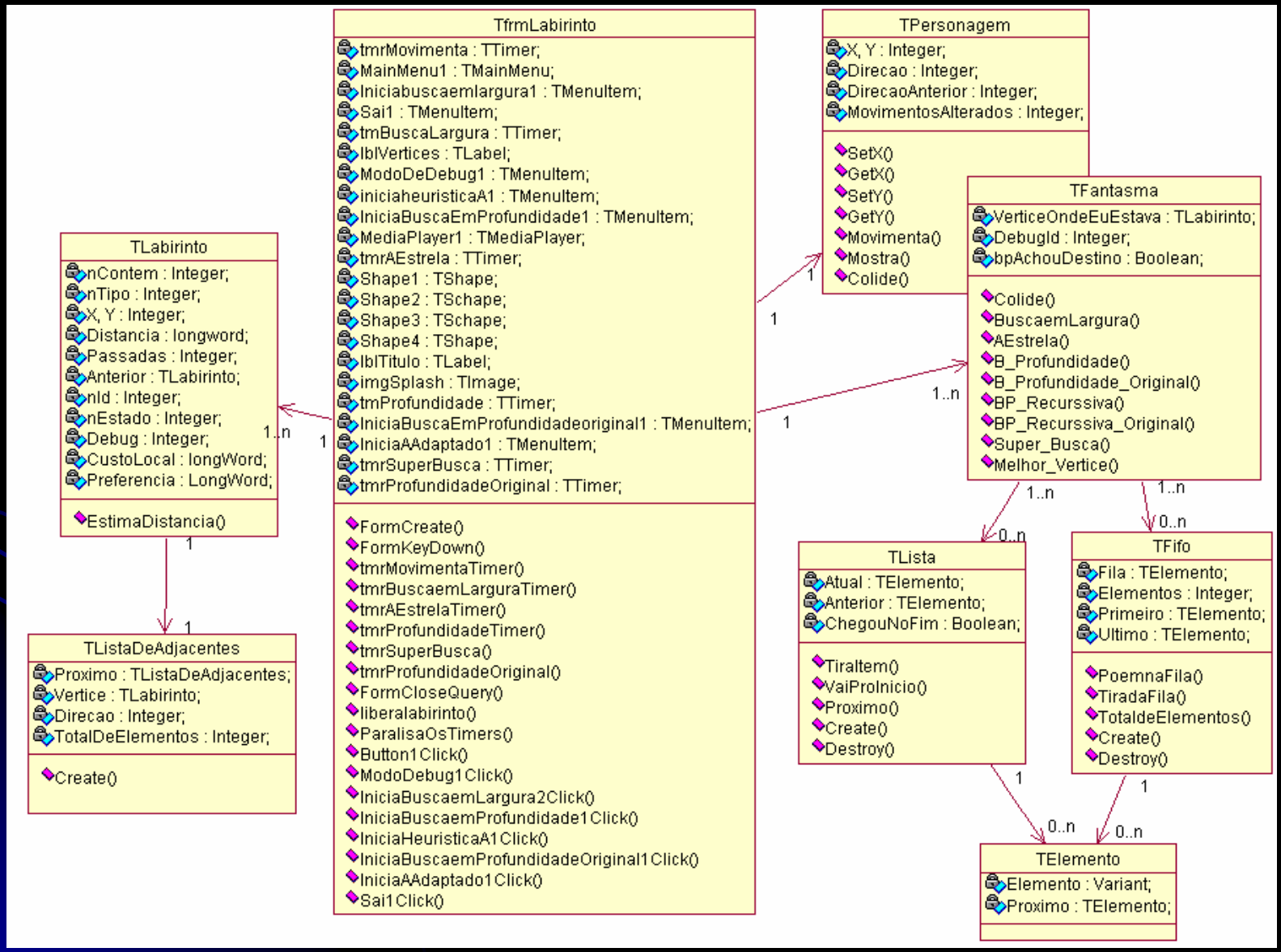


Diagrama de Seqüência

Criação do labirinto

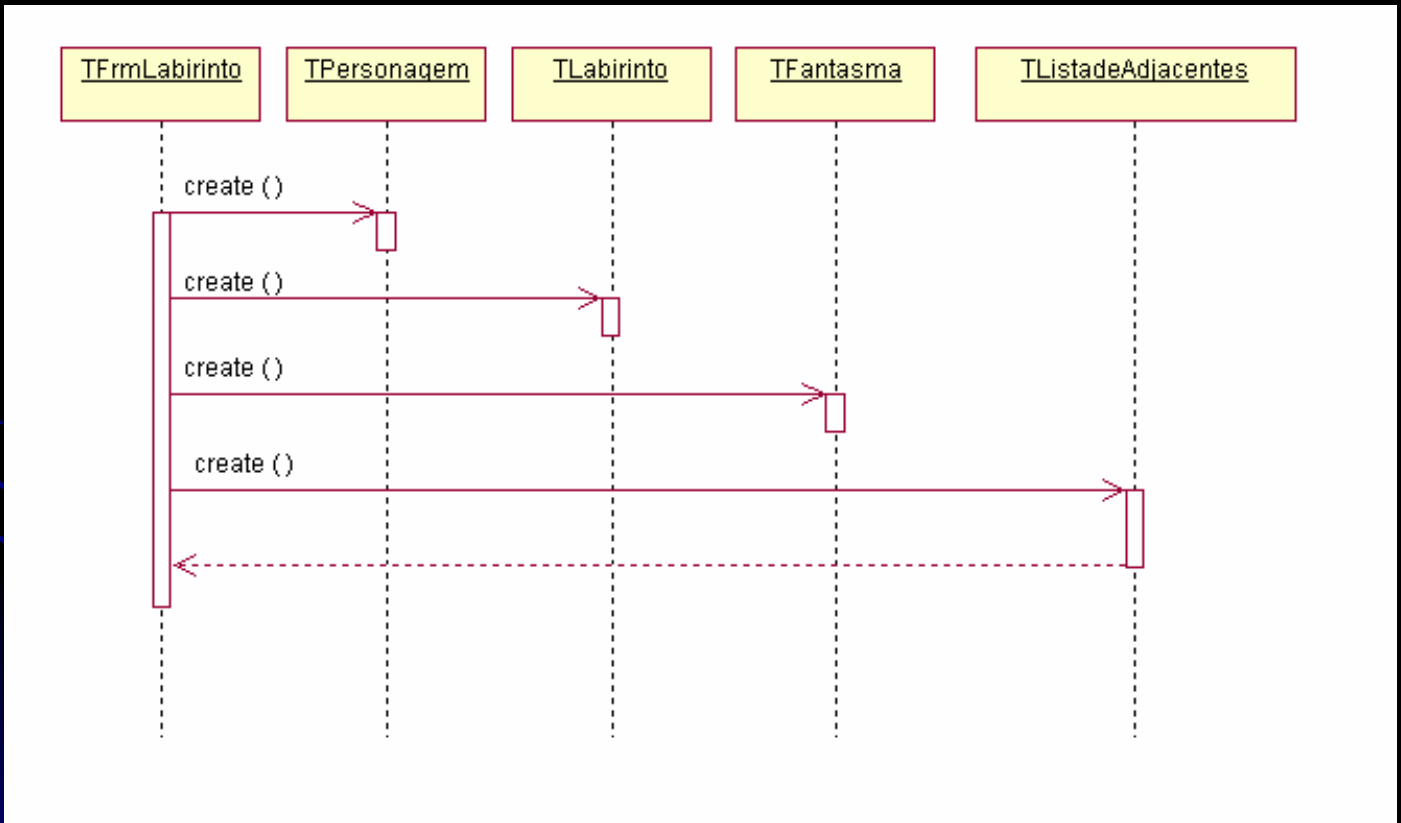


Diagrama de Seqüência

Busca em profundidade

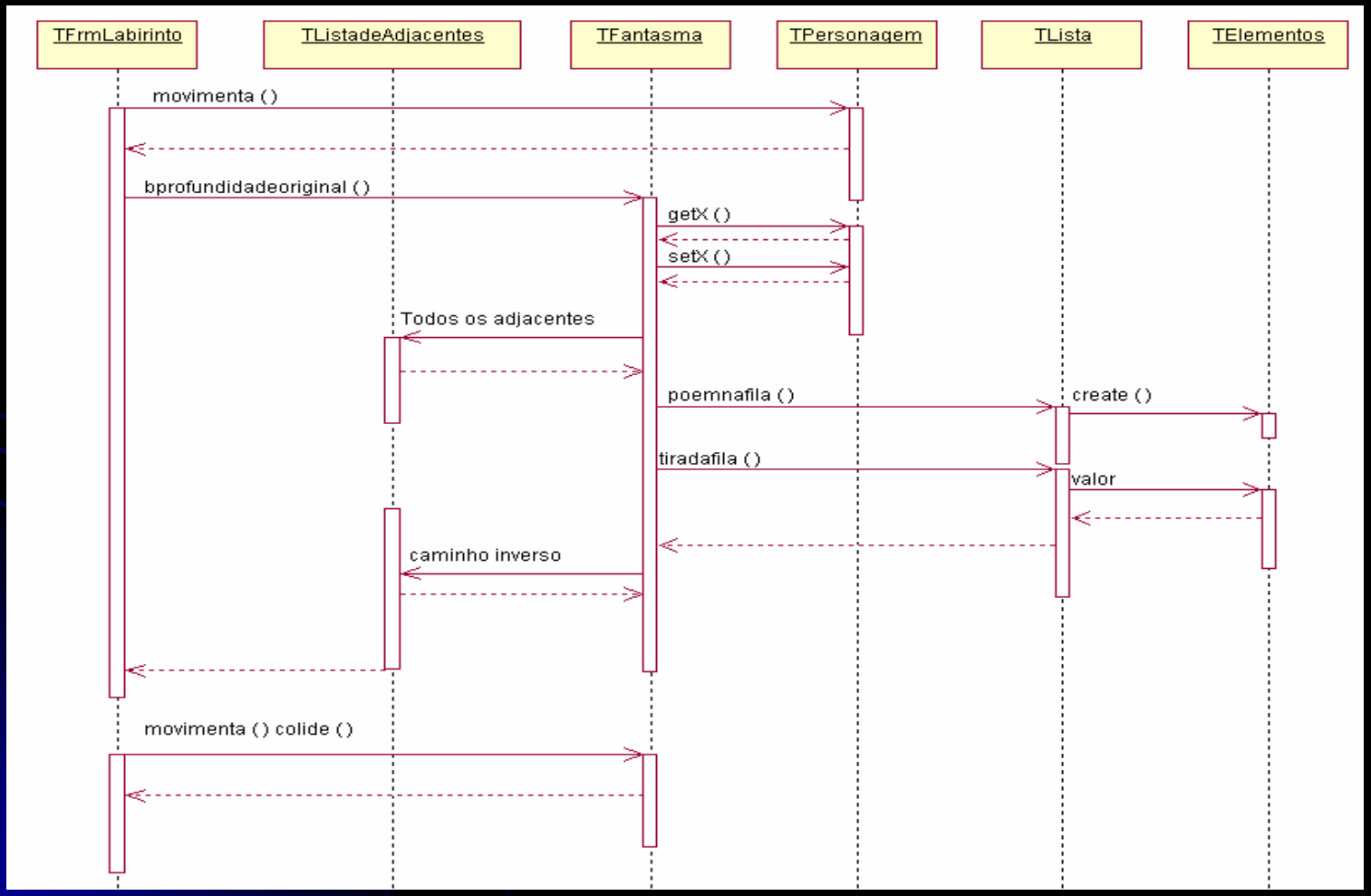


Diagrama de Seqüência

Busca em largura

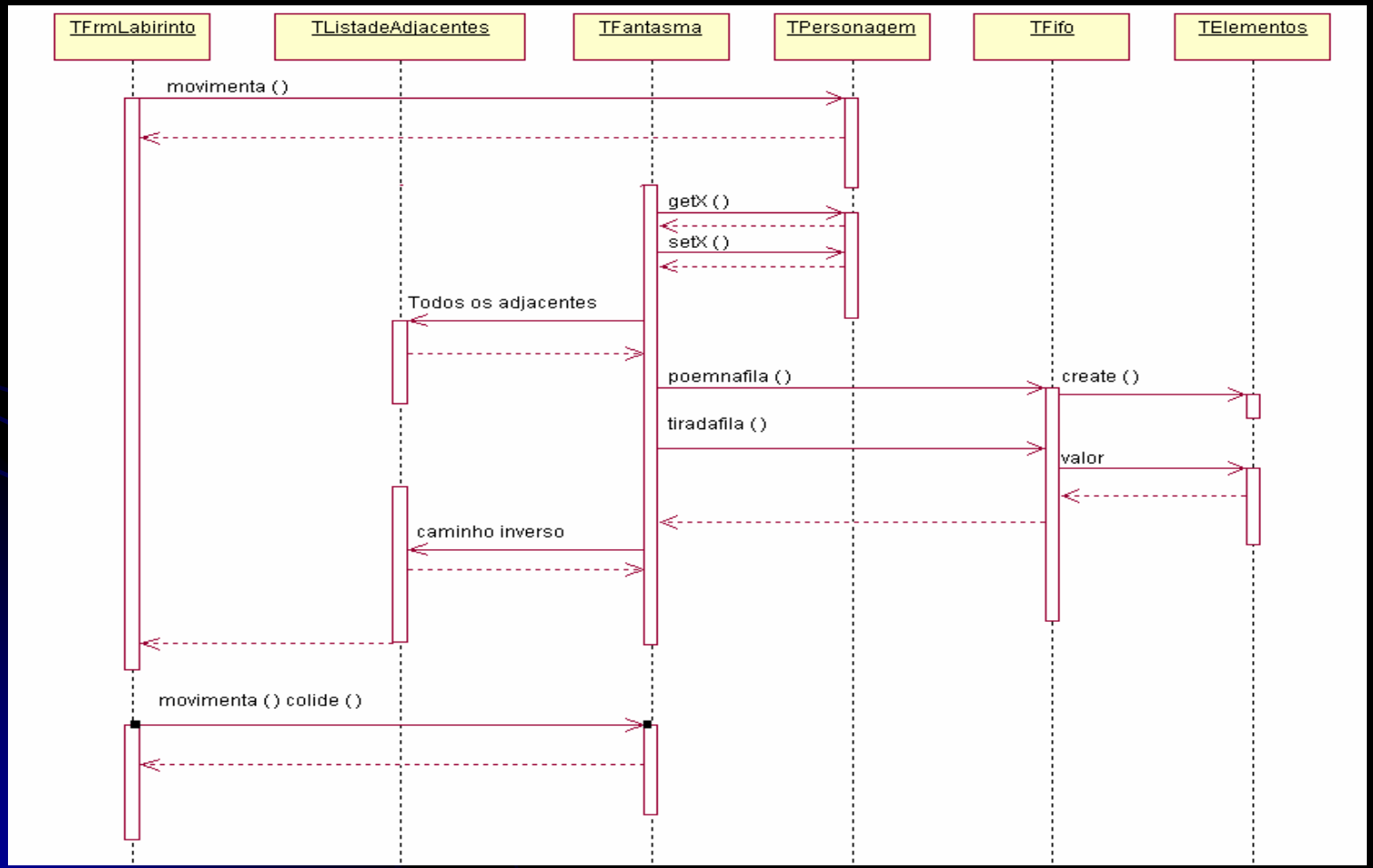


Diagrama de Seqüência

Algoritmo A* e A* adaptado

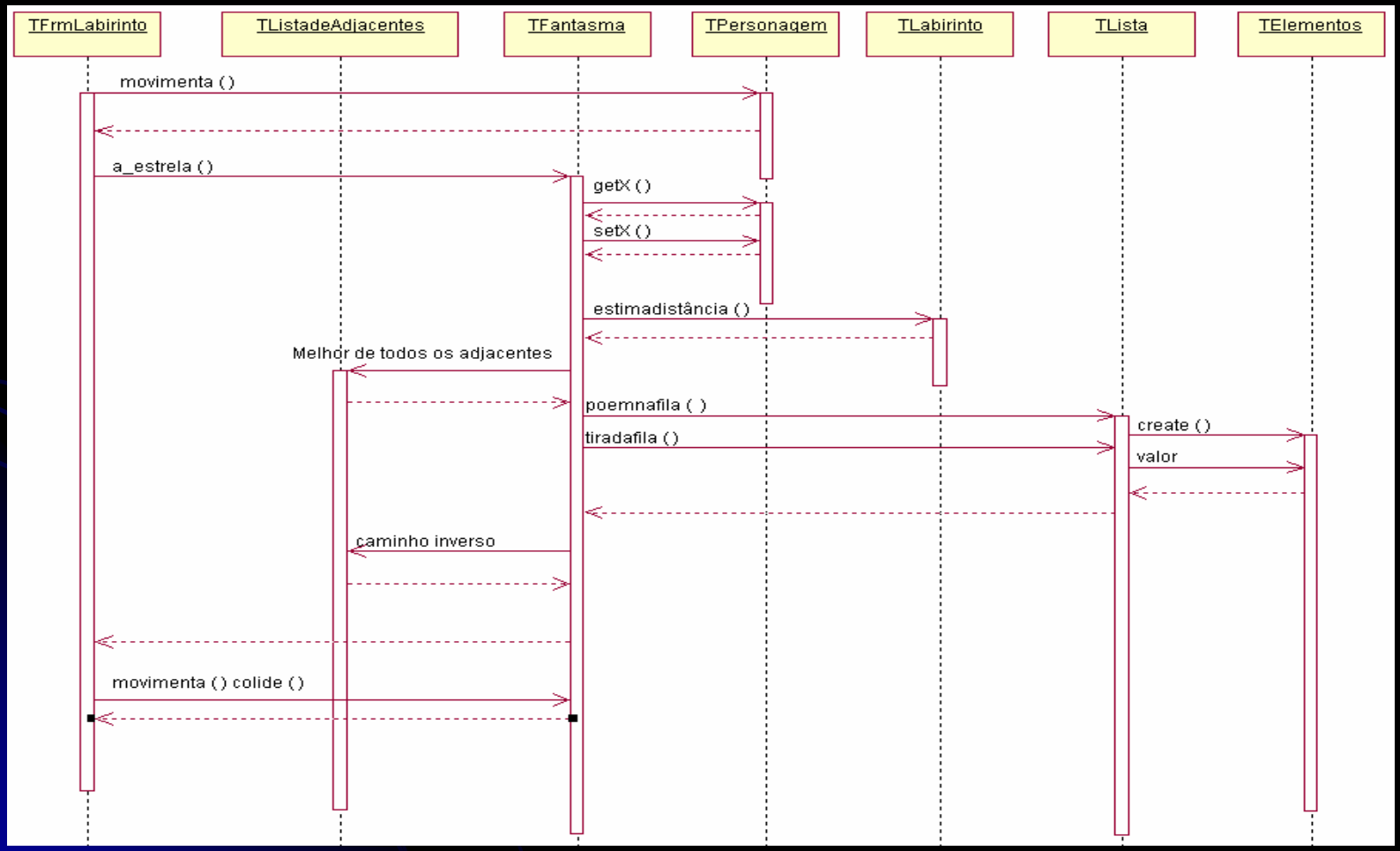
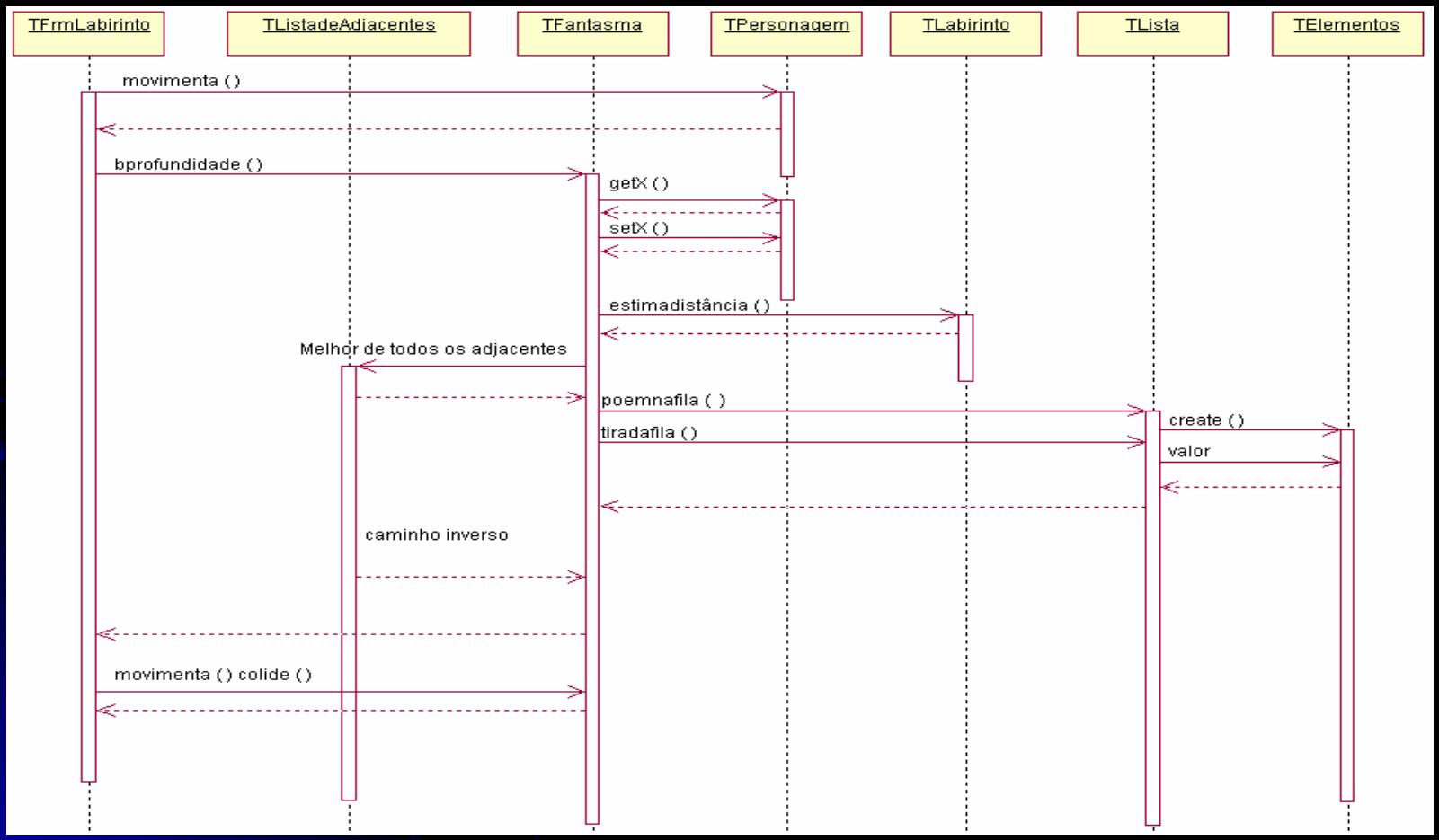


Diagrama de Seqüência

Busca em profundidade adaptado



IMPLEMENTAÇÃO

Ferramentas utilizadas:

- Rational Rose Requisite-Pro;
- *Object Pascal* com ambiente Delphi.

O labirinto

```
//Inicia a string que será o labirinto
    // P = Parede
    // V = Vitamina
    // C = PacMan
    // F = Fantasma
    //' '= Espaço livre com comida

Const Labirinto: AnsiString = 'PPPPPPPPPPPPPPPPPPPP' +
                              'PV                VP' +
                              'P PPP PPP P PP PPP P' +
                              'P P     P P       P' +
                              'P PPP P P P PP PPP P' +
                              'P       P         P' +
                              'P PPP PPP PPPP PPP P' +
                              'P P       P      P P' +
                              'P PPP P P P PP PPP P' +
                              'P                 P' +
                              'P PPP P P P PP PPP P' +
                              'P P F   P P      PCP' +
                              'P P PPPPP PPPPPP P P' +
                              'P       P P       P' +
                              'P PPP P P P PP PPP P' +
                              'P P     P P      P P' +
                              'P P PPPPP PPPPPP P P' +
                              'P p ppppp pppppp   P' +
                              'PV                VP' +
                              'PPPPPPPPPPPPPPPPPPPP' ;
```

- Classe TFrmLabirinto;
- TfrmLabirinto.FormCreate – timers são definidos;
- O labirinto é criado através de texturas carregadas dinamicamente;
- Matriz de 20 x 20;
- A medida que a matriz é lida é montado o labirinto.

RESULTADOS

- Foram realizado dez testes com cada algoritmo de busca com exceção do busca em profundidade;
- Busca em profundidade não obteve êxito;
- Média de vértices testados;

Busca em largura	Busca em profundidade adaptado	Busca heurística A*	A* adaptado
94	19	30	35
118	20	38	34
76	18	35	35
118	10	33	38
94	17	44	32
108	21	36	41
78	23	36	31
90	18	38	43
96	18	37	42
107	18	31	32

Resultados e Conclusões

- Busca em largura encontra a solução do problema;
- Busca em profundidade não é eficiente para o problema proposto;
- O algoritmo A* encontra sempre uma solução eficiente;
- A* adaptado às vezes testa menos vértices que o A*;
- Busca em profundidade adaptado obteve o melhor desempenho.

CONCLUSÃO

- Busca heurística possui melhor desempenho;
- Busca em profundidade pode não encontrar caminho;
- Linguagem utilizada foi eficiente.

Trabalhos futuros

- Implementação de outros algoritmos;
- Estudo e implementação de caminho mínimo 3D.