# UniGuide: Um modelo de representação do conhecimento espacial

*Reginaldo de Castro Cerqueira Filho*

**Dissertação de Mestrado**

# UniGuide: Um modelo de representação do conhecimento espacial

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Reginaldo de Castro Cerqueira Filho e aprovada pela Banca examinadora

Campinas, 20 de dezembro de 1999

Heloísa Vieira da Rocha ( orientadora)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação

## FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Cerqueira Filho, Reginaldo de Castro

C335u UniGuide: um modelo de representação do conhecimento espacial / Reginaldo de Castro Cerqueira Filho -- Campinas, [S.P. :s.n.], 1999.

Orientadora : Heloísa Vieira da Rocha

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Inteligência artificial. I. Rocha, Heloísa Vieira da. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Instituto de Computação
Universidade Estadual de Campinas

# UniGuide: Um modelo de representação do conhecimento espacial

Reginaldo de Castro Cerqueira Filho

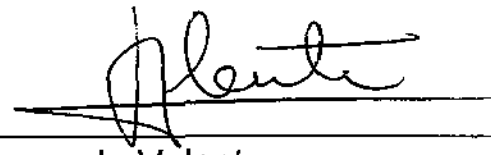
Dezembro de 1999

**Banca Examinadora:**

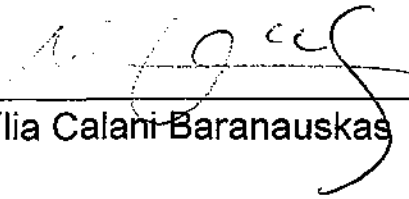
- Heloísa Vieira da Rocha (Orientadora)
- Prof. Dr. José Armando Valente
  Departamento de Multimeios (UNICAMP)
- Profa. Maria Cecília Calani Baranauskas
  Instituto de Computação (IC-UNICAMP)
- Profa. Ariadne M. B. R. carvalho (Suplente)
  Instituto de Computação (IC-UNICAMP)

## TERMO DE APROVAÇÃO

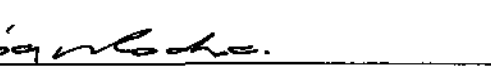
Tese defendida e aprovada em 30 de novembro de 1999, pela Banca Examinadora composta pelos Professores Doutores:



Prof. Dr. José Armando Valente
NIED/UNICAMP



Profa. Dra. Maria Cecília Calani Baranauskas
IC-UNICAMP



Profa. Dra. Heloísa Vieira da Rocha
IC - UNICAMP

# Introdução

O objetivo do trabalho é implementar um modelo do conhecimento que uma pessoa possui sobre a estrutura espacial de um ambiente de larga escala denominado de *mapa cognitivo* e usá-lo para desenvolver uma aplicação prática que versará sobre a orientação geográfica tendo como base a Unicamp.

Um ambiente de larga escala é definido pelos mecanismos de percepção que se usa para explorar o espaço e não pelo seu tamanho físico [Kuipers77]. Assim, um espaço de larga escala é definido como um espaço que não pode ser percebido totalmente de uma só vez. Nesse sentido a Unicamp vista por um pedestre é um ambiente de larga escala, mas, no entanto, uma cidade como Campinas vista de um avião não o é.

Um ambiente de larga escala é caracterizado por uma distinção muito clara entre observações locais e a estrutura global. Uma conseqüência importante disso é que as descrições locais são geralmente muito mais precisas que as descrições globais implicando que o modelo deve ser capaz de trabalhar com informações escassas e imprecisas quando referir-se a escalas maiores. Inconsistências locais são fáceis de se detectar e corrigir, mas inconsistências globais são muito mais difíceis de se localizar e corrigir e são usualmente decorrentes de uma extensão malsucedida de observações locais.

As funções do mapa cognitivo, enquanto modelo de conhecimento, são: assimilar nova informação sobre o ambiente, representar a posição corrente, resolver problemas de encontrar rotas entre pontos específicos e

problemas de posicionamento relativo. O modelo implementado e descrito neste trabalho analisa o mapa cognitivo em termos de descrições simbólicas do ambiente e operações nessas descrições.

O modelo desenvolvido tentará ser o mais fiel possível a como as pessoas processam e armazenam o conhecimento sobre ambientes de larga escala e como fazem uso desse conhecimento para poderem se orientar em tais ambientes. Além disso, a interface também será o mais intuitiva possível, possibilitando o uso do programa por usuários com as mais diversas informações sobre a Unicamp.

Além da parte de orientação geográfica, a ferramenta desenvolvida é preparada para também ter a capacidade de apresentar vários outros tipos de informação sobre departamentos específicos, sobre professores, sobre disciplinas, etc.

Além disso, os modelos implementados são todos absolutamente gerais, permitindo também uma extensão do programa para uma implementação de um sistema semelhante para um ambiente alvo maior, como por exemplo, a cidade de Campinas.

A extensão é grandemente facilitada pela inclusão de uma linguagem descritiva de ambientes de larga escala denominada de GeoDL ( ***Geographic Description Language***). A GeoDL ( explicada em detalhes no capítulo IV) é uma linguagem tipo *script* que, na versão atual, permite descrever praticamente todo tipo de ambiente, inclusive com possibilidade de descrição de ruas curvas e elípticas. Incluso também no programa, um editor/compilador para permitir a expansão da base de conhecimento pelo usuário desenvolvedor.

Essa dissertação está organizada da seguinte forma:

- No capítulo I veremos diversos modelos de conhecimento espacial além de vários conceitos importantes como o de mapa cognitivo.
- No capítulo II veremos os modelos UniGuide e TOUR, suas diferenças e semelhanças bem como seus alcances.
- No capítulo III veremos a linguagem de descrição GeoDL.
- No capítulo IV veremos as estratégias de busca que o UniGuide utiliza para resolver problemas.
- No capítulo V veremos os recursos do UniGuide, sua interface, suas características e sua implementação
- No capítulo VI veremos um apanhado geral do trabalho bem como sugestões de implementações futuras.

# Capítulo I

## Conhecimento Espacial

*Neste capítulo veremos diversos modelos do conhecimento que uma pessoa possui sobre a estrutura de um ambiente de larga escala, além de conceitos importantes, como o de mapa cognitivo.*

### 1.1 Introdução

Este capítulo trata de como a visão do conceito "espaço" evoluiu na Filosofia, Física e Matemática. O restante do capítulo é devotado a temas psicológicos na percepção do espaço e como os humanos memorizam relações espaciais. Fontes diferentes de percepção espacial são vistas, como percepção visual direta, percepção usando os outros sentidos e percepção usando linguagem. A seção sobre conhecimento espacial analisa vários modelos de conhecimento espacial e descreve como estes modelos tentam explicar distorções sistemáticas no conhecimento espacial.

### 1.2 Definição de espaço

Talvez a mais aceita concepção de espaço é aquela de uma estrutura onde as coisas existem. É tão aceita porque concorda com o que as pessoas foram ensinadas na escola e com o que as pessoas esperam na experiência do dia a dia – tanto na vida comum quanto no trabalho técnico tais como engenharia, arquitetura e planejamento. As pessoas podem colocar coisas em algum espaço ou lugar, podem movê-las ao longo deste espaço ou de lugar

para lugar e assim ter a idéia que o espaço é algo que está lá antes que coisas comecem a preenchê-lo. Algo que exista independente de coisas e previamente à sua existência: Ele existe mesmo vazio. [Nune91]

### 1.2.1 Conceitos filosóficos e físicos de espaço

O espaço é freqüentemente visto como uma relação em um conjunto de objetos. É possível criar tipos diferentes de espaço dependendo de como esta relação é definida. A visão ingênua do espaço é vê-lo como um *container* vazio. A moderna filosofia vê o espaço tanto como uma forma necessária de percepção ou como um contínuo que é limitado pela existência da matéria e estruturado pela presença dessa matéria.

Na tradição filosófica européia, o espaço é visto como um contínuo tridimensional e uma entidade homogênea. A filosofia se concentra em três problemas: se o espaço é finito ou infinito, a possibilidade ou impossibilidade do espaço vazio e a "realidade" ou "subjetividade" do espaço.

O conceito de espaço é fortemente relacionado à geometria. Euclides sistematizou e generalizou todo o conhecimento geométrico da antiguidade nos seus "elementos". O espaço que segue aquelas regras geométricas básicas é denominado de espaço euclideano. O conceito de espaço euclideano não é consistente com os conhecimentos atuais de Física e Cosmologia, mas é um conceito adequado de espaço para a grande maioria dos fenômenos contemporâneos.

Demócrito postulou um espaço vazio com átomos móveis. Esta teoria aceita uma existência objetiva de espaço e tempo. Sua teoria envolvia a crença na existência de unidades mínimas e indivisíveis de espaço e tempo. Esta crença era baseada na teoria dos números que afirma que tudo pode ser reduzido a séries de números naturais e frações entre eles. Demócrito via o espaço como infinito e o tempo como eterno.

Na Idade Média foi assumido que o espaço vazio não poderia existir. Esta visão (*horror vacui*) era baseada na visão aristotélica de um espaço que é o lugar de todas as coisas e, portanto, funcionando como um *container*.

Bruno e Galileu advogavam um espaço infinito, fato que os levou a um conflito com a igreja. O princípio da relatividade, formulado por Galileu, afirma que o espaço e tempo são homogêneos por processos mecânicos em cada sistema inercial.

Descartes vê a "extensão" como o princípio básico do espaço e de todas as coisas e reduz o espaço a partículas pequenas que são chamadas "corpúsculos".

Segundo Newton, o espaço tridimensional não tem propriedades físicas inerentes, mas apenas propriedades geométricas euclideanas e é, portanto, absoluto e independente da matéria nele contida. Assim, o espaço é visto como um *container* imaterial para objetos.

Toland, um materialista inglês, rejeitava o espaço absoluto que pode existir independentemente da matéria. Também Leibnitz via a concepção absoluta do espaço como um dos problemas da concepção materialista. Ele postulou que o espaço e o tempo são relações entre objetos e processos. Portanto, espaço e tempo são ligados fortemente à matéria e não possuem uma realidade absoluta na ausência desta. Finalmente, Leibnitz vê o espaço e o tempo como percepções subjetivas apesar de corresponderem a uma ordem objetiva das coisas do mundo.

Kant via o espaço e o tempo como uma forma de percepção puramente subjetiva. Suas idéias de espaço e tempo eram uma forte crítica contra as concepções absoluta e relativa do espaço. Suas idéias, expressadas na obra "*Critique of pure reason*"[1], podem portanto ser vistas com uma tentativa de colocar novas idéias em um velho debate. A tese central de Kant é que o espaço e o tempo não podem ser vistos como se eles fossem objetos ou eventos ordinários, i.e., físicos ou empíricos. Na realidade, espaço e tempo são estruturas ou sistemas para gravar e ordenar as observações sobre coisas e eventos. Como tais, espaço e tempo não pertencem intrinsecamente ao mundo empírico, mas são partes do equipamento mental desenvolvido para capturar e raciocinar sobre o mundo real [Nune94]. Em outras palavras, Kant achava que a ordem que nós vemos na natureza é a ordem que existe em nossas mentes, uma ordem que está entranhada ou que reflete a própria estrutura da mente. Assim, apenas o espaço e o tempo juntos tornam a percepção possível. Na concepção de Kant, o espaço é independente de seus conteúdos.

Einstein posteriormente provou que o espaço e o tempo não são independentes da matéria na sua Teoria Geral da Relatividade. O conteúdo do espaço influencia a sua estrutura, como por exemplo, no caso da presença de grande quantidade de matéria curvando o espaço.

O desenvolvimento do conceito de espaço começou como um conceito geral de um *container*. Do maior interesse para a filosofia sempre foi se o espaço sem conteúdo é ainda espaço ou não. O pensamento atual é que a matéria influencia a estrutura do espaço. O espaço não é mais visto como um conceito tridimensional, mas sim como um espaço-tempo contínuo de quatro dimensões.

---

[1] Crítica da Razão Pura

### 1.2.2 O conceito matemático de espaço

Esta seção examina a definição matemática de espaço a qual é mais geral do que a filosófica ou física. O espaço que se usa quotidianamente para navegação é basicamente um espaço euclideano tridimensional, o qual por sua vez, é um exemplo de um espaço métrico. Um conceito mais geral de espaço pode ser definido através da teoria de conjuntos e o termo "vizinhança". Este lida com o espaço topológico. Uma maneira formal de tratar espaços topológicos é teoria dos grafos. Para uma descrição mais aprofundada de espaços topológicos e teoria dos grafos ver [Gatrell91].

Matematicamente, o espaço euclideano é caracterizado por dimensões ortogonais, onde o número de dimensões é três. Cada ponto no espaço é definido por coordenadas em um sistema de coordenadas ortogonais. Utilizando-se a geometria analítica, distâncias entre pontos podem ser definidas.

Espaços métricos são espaços que definem uma relação métrica nos objetos daquele espaço. Se os elementos são um conjunto de n-tuplas que podem ser vistas como pontos em um espaço n-dimensional, uma função $d(a,b) \rightarrow R$ definida em um par ordenado de tais tuplas, é chamada uma métrica se ela obedecer às seguintes regras para cada três pontos a,b, e c no espaço sendo estudado:

1. $d(a,b) >= 0$ e $d(a,a)=0$
2. $d(a,b) = d(b,a)$
3. $d(a,c) <= d(a,b) + d(b,c)$
4. $d(a,b) > 0$

Um exemplo de uma métrica é a chamada distância euclideana, que em um espaço bidimensional é dada pelo Teorema de Pitágoras.

## 1.3 Percepção espacial

Esta seção se concentra na percepção espacial. A percepção espacial é diferente de outras percepções, pois é sempre possível verificá-la fazendo uso de vários sentidos [Freksa91]. Este fato nos faz tomar qualquer percepção espacial como particularmente real.

### 1.3.1 Percepção espacial utilizando-se a visão

O sistema visual propicia aos humanos perceber objetos e o espaço como objetos tridimensionais. A visão espacial não é a única possibilidade de ver um objeto espacialmente. Utilizando-se perspectiva é possível representar objetos espaciais em uma superfície plana como um desenho. Tal desenho é uma transformação de objetos espaciais em uma representação em um espaço plano. Nosso conhecimento do mundo real ajuda-nos a perceber o desenho plano como uma construção tridimensional.

A representação plana contém menos informação espacial que o objeto real. A informação que falta é adicionada pelo processamento dessa representação em nossos cérebros.

### 1.3.2 Percepção espacial utilizando-se outros sentidos

É possível perceber o espaço sem utilizar os olhos. Segundo Freksa [Freksa91]: " *Nosso conhecimento sobre o espaço físico difere de todos os outros conhecimentos em um modo muito significante : Nós podemos perceber o espaço diretamente através de vários canais levando modalidades distintas de informação. Diferentemente dos casos de outros domínios perceptíveis, o conhecimento espacial obtido através de um canal pode ser verificado ou refutado pela percepção através de outros canais. Como uma conseqüência, nós somos desproporcionalmente confiantes sobre o que sabemos a respeito do espaço.*"

Ouvir sons espacialmente é talvez a percepção espacial mais forte além da visão. A resolução desta percepção é muito menor que a do sistema visual mas nós somos capazes de "ouvir" em qual lugar estamos e podemos localizar fontes de som aproximadamente. A audição permite aos humanos monitorar o meio ambiente, pois percebe-se sons de todas as direções ao mesmo tempo.

### 1.3.3 Percepção espacial utilizando-se abstrações e linguagem

O espaço pode ser percebido através de abstrações. Ícones são desenhos abstratos de um objeto ou situação. Estes ícones freqüentemente não representam um objeto particular ou situação, mas sim uma classe de objetos ou situações.

Outra abstração comumente usada para representar espaço são mapas. Eles representam grandes construções espaciais em um alto nível de abstração. Enquanto todo mundo pode ter percepção espacial de desenhos e fotografias, a leitura de mapas é uma habilidade que precisa ser aprendida.

A forma mais abstrata de representar objetos é através do uso de linguagem tanto na forma falada quanto na forma escrita. A linguagem escrita evoluiu de ícones [Bolt91], [McCl94], tanto assim que ainda hoje temos exemplos de escrita simbólica, como, por exemplo, o Kanji que é a escrita chinesa e japonesa.

Quando lemos descrições textuais de espaços e mapas nós somos capazes de construir uma representação mental do espaço descrito. Perceber espacialmente utilizando-se descrições é possível porque elas podem invocar imagens espaciais na nossa mente que induzem a uma percepção espacial.

A linguagem não especifica o espaço tão precisamente quanto a percepção. Descrições freqüentemente não especificam distâncias entre objetos ou uma orientação exata. Portanto, a percepção espacial de uma descrição baseia-se no conhecimento geral.

### 1.3.3.1 *Frames* de referência

Descrições de relações entre objetos necessitam serem relacionadas a um *frame* de referência. *Frames* de referência estão fortemente acoplados tanto à pessoa quanto ao ambiente em que o indivíduo se move.

Um exemplo para *frames* de referência foi descrito por Kevin Lynch [Lynch60] : " *O sistema utilizado no planalto do norte da China é estritamente regular. Ele tem conotações mágicas profundas : o norte sendo associado ao negro e ao mal, o sul ao vermelho, alegria, vida e ao sol. Ele controla muito estritamente a colocação de todos os objetos religiosos e estruturas permanentes. De fato, o uso principal da bússola, uma invenção chinesa, não era para a navegação no mar, e sim para a orientação de prédios. Tão abrangente é esse sistema que as pessoas dão a sua direção através de pontos cardinais, e não através de direita ou esquerda, como seria natural para nós. Esse sistema não é centralizado no indivíduo, se movendo com ele, ao contrário, é fixo, universal e fora da pessoa.*"

Bryant [Bryant92] propôs uma classificação de *frames* de referência para o seu Esquema de Representação Espacial (SRS[2]). Segundo ele, os *frames* de referência são "sistemas coordenados nos quais as localizações podem ser especificadas ao longo de três dimensões":

- O *frame* de referência egocêntrico é definido pelos três eixos do corpo (cabeça/pé, frente/costas, esquerda/direita).

[2] *Spatial Representation Scheme*

- O *frame* de referência alocêntrico é composto de eixos ortogonais colocados fora do observador. Estes eixos podem ser centrados em um ponto de referência no ambiente ou alinhados aos pontos cardinais.
- O *frame* de referência externo é composto de eixos baseados no corpo mas projetados para frente do campo de visão
- Outro possível *frame* de referência é centrado em um objeto

O *frame* de referência alocêntrico pode ser classificado em vários tipos comuns de *frames* de referência. Segundo [Pederson93]:

- NSLO[3], em direção ao nascer do sol, etc.
- Subindo ou descendo (comum em culturas com um alcance geográfico limitado)
- Em direção a terra ou ao mar (comum em povos vivendo em ilhas)
- Sistemas de referência baseados em um ponto de referência convencional (por exemplo a igreja matriz)

*Frames* de referência podem ser simétricos ou assimétricos. Sistemas assimétricos utilizam um eixo dominante, por exemplo "subindo", enquanto os sistemas simétricos utilizam eixos equivalentes.

A escolha de qual sistema de referência é utilizado depende da linguagem utilizada, da formação cultural e da situação do momento. Isto foi demonstrado em um estudo de Pederson [Pederson93]. O estudo focalizou-se em dois sistemas lingüísticos dos Tamis, um dos quais é utilizado principalmente em cidades enquanto que o outro sistema é utilizado principalmente no campo. Estes dois sistemas mostram diferenças no uso de *frames* de referência e, portanto na referência aos objetos no espaço. Isto mostra que descrições de relações espaciais freqüentemente dependem de contexto e formação cultural do provedor da descrição.

Através de tipos formais de referência é possível apontar objetos e relações espaciais precisamente. A linguagem cotidiana é muito menos precisa. Nossa linguagem provê basicamente palavras para descrições não muito precisas do ambiente como, por exemplo, "perto de", "entre", "a esquerda de". Estas palavras sempre descrevem localização como uma relação a outro elemento. Outras palavras como "dentro de", "contém" são descrições

[3] Norte, Sul, Leste, Oeste

em relação a um *frame* de referência, que é freqüentemente assumido apenas implicitamente.

Ao descrever relações espaciais entre objetos deve-se descrevê-las de acordo com um *frame* de referência. De acordo com um esquema de classificação proposto por Bryant existem vários tipos de *frames* de referência. O mais importante deles é o *frame* de referência alocêntrico. A escolha de qual *frame* de referência vai-se utilizar em uma descrição depende da situação e da formação cultural da pessoa.

### 1.3.3.2 Escala

Muito relacionada à percepção espacial e à escolha de *frames* de referência é a escala. Propriedades do espaço ou relações entre objetos no espaço são normalmente vistos como independentes de escala quando estudados como problemas formais.

Montello [Montello93] define escala como "a relação entre dimensões de uma representação e aquelas da coisa que ela representa". Portanto ele vê a escala como um tamanho relativo de uma representação. De acordo com Montello pode haver confusão entre grande escala e grande tamanho porque em alguns casos o tamanho relativo ao ambiente é considerado, já em outros o tamanho relativo a uma pessoa é que é considerado. Montello então introduz uma terminologia diferente que evita esta ambigüidade.

Ele usa uma classificação de quatro níveis de espaços psicológicos. Ele baseia-se nos conceitos anteriores de escala de Ittelson, Mandler e Zubin. Montello distingue classes de espaços de acordo com o "tamanho projetivo" do espaço relativo ao corpo humano e não de acordo com o tamanho real ou aparente. Portanto, um espaço de grande escala visto à distância pode tornar-se um espaço de menor escala, ou em outras palavras, um ambiente de grande escala é *definido pelo modo como é percebido e não por seu tamanho físico* [Kuipers77].

### 1.3.4 Distorções na percepção espacial

Dependendo do modelo de memória espacial usado, erros sistemáticos na memória espacial podem ser explicados. Como o conhecimento espacial é adquirido em pequenos passos descontínuos, os pedaços de informação da representação cognitiva não são sempre fortemente relacionados uns aos outros. Isto leva a vários erros sistemáticos na estimação

de distâncias e na estimação do alinhamento de objetos. Esta seção descreve dois destes erros sistemáticos.

### 1.3.4.1 Erros sistemáticos na percepção da distância

Em [Tversky93], Tversky analisa vários estudos sobre percepção de distâncias. Por exemplo, distâncias entre prédios agrupados são percebidas como sendo menores que as distâncias entre prédios que não pertencem à mesma classe funcional. Este fato é visto como uma evidência que a representação mental é estruturada hierarquicamente.

Particularmente interessante é uma distorção assimétrica na percepção de distâncias entre um ponto de referência principal ( *landmark* ) e outro objeto: a distância do objeto para o *landmark* é percebida como sendo menor que a distância do *landmark* para o objeto.

Outro erro sistemático relativo à estimação de distâncias ocorre quando o comprimento de uma rota é estimado, que tenha barreiras, desvios e muitas curvas. Tipicamente tal rota é percebida como sendo mais longa que uma rota de mesmo tamanho, mas tendo menos curvas ou desvios.

### 1.3.4.2 Erros sistemáticos na percepção do alinhamento

Outro erro típico na memória espacial é relativo à rotação de áreas de acordo com um *frame* de referência. Como conseqüência, as pessoas mostram uma tendência de colocar objetos ao longo de um eixo mesmo se tais objetos não estejam alinhados na realidade. É também comum endireitar características irregulares tais como ruas ou rios.

Estes erros sistemáticos são observáveis apenas em ambientes que mostram muita liberdade no alinhamento dos objetos. Se um ambiente é descrito usando linguagem, o grau de liberdade para a rotação é reduzido ou a rotação não é mencionada explicitamente. Para ser mais preciso, a linguagem por ela mesma não impõe restrições de como este tipo de informação é descrito. Entretanto, na linguagem natural o conjunto possível de valores para orientações e posições é restrito pelas descrições. Isto só pode ser superado pelo uso de uma linguagem mais formal.

Outro modo de comunicar este tipo inexato de informação é através de gestos. Este tipo de comunicação linguagem-gestos está sendo usada em um sistema em desenvolvimento no MediaLab do MIT.

## 1.4 Modelos de representação do conhecimento espacial

O comportamento espacial humano é dependente da representação mental individual do ambiente espacial [DoSt73], [Shum90]. A representação é utilizada para direcionar a ação e as experiências são usadas, por sua vez, para modificar a representação.

Os vários modelos da habilidade de apreender um ambiente e lembrar relações espaciais formam a base para a compreensão de erros em relações espaciais. Esta seção estudará as diversas representações mentais. Muitos modelos diferem na seqüência em que as propriedades de espaços são apreendidas. Esta seqüência é importante para ambientes virtuais que são projetadas para facilitar tarefas de navegação.

### 1.4.1 Mapas cognitivos

A visão comum de como as pessoas representam mentalmente relações espaciais em uma área é baseada no conceito de mapa cognitivo. Este termo sugere uma construção parecida com um mapa em nossas mentes que possamos consultar para responder questões sobre a área representada. Embora este conceito não seja mais totalmente aceito hoje em dia, adotaremos o termo "mapa cognitivo" para descrever o conceito descrito abaixo [CaTa75]: "*Uma propriedade do ambiente físico de importância psicológica distinta é o fato que o ambiente nos rodeia completamente. Assim, não nos é possível experimentar ou perceber tudo isso de uma só vez. Nós podemos apenas prestar atenção a aspectos discretos do ambiente em pontos sucessivos no tempo. Entretanto, para nosso comportamento ser apropriado, efetivo ou adequado em relação ao ambiente físico, é necessário para ele prosseguir em uma forma contínua. Para explicar o modo no qual essa experiência discreta pode produzir uma interação contínua é necessário postular alguns processos representacionais por parte do indivíduo. Esta representação deve amalgamar a experiência em uma forma que liga descontinuidades na percepção e permite extrapolações para facilitar a preparação para uma ação futura.*

*No caso de um indivíduo lidando com uma cidade, é necessário também assumir que a consolidação postulada de experiências é de alguma forma sumarizada. Este sumário consiste de um padrão ou estrutura cujo efeito resultante é organizar as 'representações' das experiências da cidade e suas implicações.*

*Observe que o termo 'representação' está entre aspas. Isto é porque não há nada em nossa descrição que diga que ele deva ter relação direta com os métodos usuais de representar cidades*".

### 1.4.1.1 Aprendendo rotas

Achar um caminho é a habilidade de aprender uma rota através do ambiente. O trabalho de Piaget e outros sobre o conhecimento de rotas forma a base para a maioria das teorias de como achar um caminho que foram aceitas por muito tempo [Blad91]. De acordo com estas teorias o mapa cognitivo é construído em pequenos passos:

1. Primeiro os pontos de referência ( *landmarks* ) são aprendidos. Pontos de referência são objetos ou pontos proeminentes que ajudam na orientação – eles são pontos ou objetos que têm um significado especial o qual pode ser visto de longe e pode ser distinguido facilmente de outros pontos no ambiente.
2. Somente após os pontos de referência serem aprendidos, elementos lineares, como, por exemplo, rotas, o são.
3. Finalmente, informações métricas (por exemplo, distâncias) são aprendidas.

Esta informação é denominada mapa cognitivo para diferenciá-la de mapas reais. O trabalho de Piaget é baseado principalmente na pesquisa de crianças e sugere vários estágios do desenvolvimento do conhecimento sobre o ambiente. Devido parcialmente ao fato de que a pesquisa foi baseada em crianças, esta teoria e suas derivadas não são mais inteiramente aceitas. [Blad91]

Atualmente acha-se que as crianças e adultos podem realizar tarefas de achar caminho com sucesso com pouca experiência e que pontos de referência e rotas são aprendidas conjuntamente e muito rapidamente. Segundo [Tversky93]: "*Assim como os mapas reais, os mapas cognitivos presumivelmente são um todo coerente que refletem relações espaciais entre elementos. Como construções mentais disponíveis para inspeção mental, os mapas cognitivos presumivelmente são como mapas reais disponíveis para uma inspeção real, tal como imagens as quais, de acordo com a visão clássica de imagens mentais, são como representações internalizadas*".

Este descrição afirma que mapas cognitivos presumivelmente são todos coerentes. A memória mental espacial, entretanto, possui certas falhas as quais não são explicáveis utilizando-se uma construção coerente, simples, e

parecida com um mapa. Tversky aponta vários estudos os quais indicam que tais construções parecidas com mapas reais não são muito prováveis. [Tversky93]. Portanto ela descreve duas outras visões construtivistas de como ambientes espaciais poderiam ser representados mentalmente: a *colagem cognitiva* e o *modelo espacial mental.*

### 1.4.2 Colagem cognitiva

A visão construtivista assume que as pessoas adquirem pedaços separados de conhecimento sobre ambientes. Este conhecimento é usado ao descrever rotas e ao fazer julgamentos sobre localizações, direções e distâncias. Estes pedaços de conhecimento incluem vários tipos de informação como memórias de mapas, lembranças de jornadas, direções, fatos e outros. Para ambientes que não sejam conhecidos detalhadamente a informação pode estar em diferentes formas, algumas delas totalmente diferente de mapas.

Tversky [Tversky93] descreve esta representação como: "*Nestes casos, ao invés de parecer com mapas, as representações internas das pessoas parecem mais como colagens. Colagens são sobreposições temáticas de multimídia de diferentes pontos de vista.*"

Como estas construções representam relações espaciais de vários pontos de vista elas não contém informações métricas.

### 1.4.3 Modelos mentais espaciais

As pessoas parecem ter uma representação de leiautes espaciais bastante acuradas quando os ambientes são simples ou já bem aprendidos. Esta informação de leiaute não pode ser explicada usando o modelo de colagem cognitiva. Portanto um modelo mental espacial foi proposto. Tais modelos mentais capturam as relações espaciais coerentemente e permitem reorientação e inferências espaciais. Diferentemente, entretanto, de mapas cognitivos os modelos mentais podem não preservar informação métrica. Para mais informações ver [Tversky93].

#### 1.4.3.1 Sistema de representação espacial – SRS

O SRS foi proposto por Bryant [Bryant92]. Ele difere de quaisquer outros processos de memória. A idéia básica é que "*as pessoas criam os mesmos tipos de mapas cognitivos e modelos espaciais mentais a partir de uma descrição verbal e observação direta. Isto sugere que as pessoas possuem um sistema de representação espacial distinto que cria modelos*

*espaciais de fontes diferentes e é independente de sistemas de memória para outros domínios do conhecimento. O principal papel do SRS é organizar a informação espacial em uma forma geral que possa ser acessada tanto por mecanismos de percepção como por mecanismos lingüísticos.*"

### 1.4.3.2 O modelo TOUR

Kuipers [Kuipers77] baseia seu modelo TOUR de memória espacial em cinco diferentes categorias de conhecimento espacial: rotas, estrutura topológica de ruas, a posição relativa de dois lugares, fronteiras divisórias e regiões. Este conhecimento é melhorado com um conjunto de regras de inferência para se descobrir uma rota no ambiente.

Esta representação mental é como uma rede feita de ruas e cruzamentos onde as formas exatas e comprimentos das ligações da rede são freqüentemente sem importância. Um terceiro componente da representação mental proposta por Kuipers é algo como um catálogo de rotas. O modelo TOUR mostra bem que a representação mental possui distorções, pois os tamanhos e formas exatas são freqüentemente irrelevantes.

Este foi o modelo adotado neste trabalho (com algumas adaptações) porque é bastante completo, explica bem distorções de percepção espacial (estudadas na próxima seção) e, além disso, é o mais adequado para uma implementação em computador. O modelo TOUR será exaustivamente estudado no Capítulo II desta dissertação.

## 1.5 Conclusão

Neste capítulo vimos a definição de ambiente de larga escala, os principais modelos para explicar o conhecimento espacial, inclusive introduzindo o modelo TOUR que servirá de base para a implementação de nossa ferramenta.

No próximo capítulo veremos os modelos UniGuide e TOUR, suas semelhanças e diferenças, a forma utilizada para representação do conhecimento espacial e sua estrutura.

# Capítulo II

## O Modelo UniGuide e o Modelo TOUR

*Neste capítulo veremos o que são os modelos UniGuide e TOUR, suas diferenças, seus objetivos, a forma como foi representada o conhecimento geográfico, as formas de aprendizagem e como os modelos TOUR e UniGuide fazem inferências.*

### 2.1 Introdução

O modelo TOUR para representação de conhecimento geográfico foi proposto por Benjamin Kuipers [Kuipers77] na sua tese de doutoramento pelo MIT. O modelo implementado neste trabalho é uma versão adaptada denominado modelo UniGuide, do modelo proposto por Kuipers.

Ao longo deste capítulo, descreveremos o modelo UniGuide e, quando apropriado, descreveremos as adaptações deste modelo em relação ao modelo originalmente proposto por Kuipers.

### 2.2 Princípios do modelo UniGuide e do modelo TOUR

Os princípios que foram seguidos no desenvolvimento do modelo UniGuide são os que de forma geral foram propostos por David Marr [Marr75] como guia geral para desenvolvimento de sistemas de IA :

• **O princípio da nomeação explícita** - Sempre que uma coleção de dados está para ser descrita, discutida ou manipulada como um todo, ela deve ter uma nome.

• **O princípio do projeto modular** - Todo grande projeto deve ser dividido em pequenas partes e implementado como partes o mais independentes possível.

• **O princípio do menor comprometimento** - Nunca se deve fazer algo que mais tarde tenha que ser desfeito.

• **O princípio da "degradação suave"** ( *graceful degradation*) - Sempre que possível, a degradação dos dados não deve impedir que pelo menos parte da resposta seja dada.

## 2.3 Diferenças entre os modelos TOUR e UniGuide

A principal diferença entre os modelos TOUR e UniGuide reside em seus respectivos objetivos : Enquanto o TOUR foi concebido para ser um modelo teórico geral, que pudesse ser usado em qualquer tipo de mapa e é baseado em pesquisas sobre como as pessoas *realmente* armazenam e utilizam o conhecimento sobre espaços de grande escala, embora nem sempre seja de fácil utilização prática; o modelo UniGuide foi concebido como uma adaptação do modelo TOUR[1] para melhor se adequar a um mapa do tamanho e do tipo do mapa da Unicamp, para facilitar sua implementação, para permitir maior flexibilidade, inclusive utilizando alguns métodos de busca que o modelo original não utilizava e que *não* são utilizados normalmente pelas pessoas.

Outra grande diferença e, neste caso, vantagem do modelo UniGuide em relação ao TOUR é a forma de entrada de dados. Enquanto o UniGuide possui uma linguagem de descrição de mapas ( GeoDL, que vai ser estudada em detalhes no próximo capítulo) que é feito em um arquivo à parte, o TOUR exige que se programe em LISP as descrições dentro do próprio programa, o que o torna bem menos versátil.

O UniGuide, ao contrário do TOUR, possui dois modos de operação: o modo de consulta e o de desenvolvimento. No modo de consulta o usuário apenas tem informações de rotas entre lugares, localizações de lugares, etc. No modo de desenvolvimento, o usuário pode descrever ou alterar a descrição de mapas.

---

[1]Embora possa ser expandido sem maiores problemas

Entretanto o UniGuide não implementa alguns aspectos do TOUR relacionados à regiões, como veremos ao longo desse capítulo.

## 2.4 Representação de conhecimento

O modelo UniGuide é, basicamente, um modelo híbrido de representação baseada em *frames* e sistema de produção.

### 2.4.1 Sistemas de produção

Os sistemas de produção, desenvolvidos por Newell e Simon [Newell72] para o seu modelo de cognição humana, são uma representação de conhecimento modular que tem ganhado crescente aceitação entre pesquisadores de IA.

A idéia básica nestes sistemas é que a base de dados consiste de regras, chamadas de produções, na forma de pares condição-ação, ou seja, SE determinada condição ocorrer ENTÃO faça isto. A utilidade deste formalismo decorre do fato que as condições em que cada regra é válida são explícitas e, pelo menos em teoria, as interações entre as regras são minimizadas. [Feigenbaum89]

Os sistemas de produção consistem basicamente de três partes:

- Um conjunto de regras de produção, ou seja, regras do tipo "SE esta *condição* for verdadeira ENTÃO faça esta *ação*".
- Uma estrutura parecida, em seu funcionamento, com um *buffer*, chamada *contexto*, que é o foco de atenção das regras de produção.
- Um interpretador, que controla a atividade do sistema e cuja função principal é decidir, a cada momento, o que fazer em seguida.

As principais vantagens de um sistema de produção são:

- **Modularidade** - Talvez a maior vantagem de um sistema de produção. Consiste no fato que regras individuais podem ser adicionadas, retiradas ou modificadas independentemente, ou seja, um sistema de produção funciona como se fosse constituído de pedaços independentes de conhecimento e podemos alterar uma regra específica sem nos preocuparmos com efeitos

diretos em outras regras, pois as regras só se comunicam pela estrutura de dados chamada *contexto*. Assim, conhecimento pode ser potencialmente adquirido por uma adição incremental de mais e mais produções sem a necessidade de reescrevermos o programa.

• **Uniformidade** - Outra vantagem de um sistema de produção é a estrutura uniforme imposta à base de conhecimento. Como a informação tem que ser codificada dentro de uma rígida estrutura de regras de produção, ela pode ser mais facilmente entendida por outra pessoa ou por outra parte do sistema.

• **Naturalidade** - Uma vantagem adicional de um sistema de produção é a naturalidade com que se pode expressar certos tipos importantes de conhecimento usando-se regras. Conhecimento do tipo "O que fazer" quando determinadas situações ocorrem são naturalmente expressos em regras de produção e além disso essa é a forma mais freqüentemente usada por especialistas humanos para explicar como eles realizam seus trabalhos.

No entanto, os sistemas de produção têm algumas desvantagens sendo as principais:

• **Ineficiência** - A forte modularidade e uniformidade das produções, resultam em um grande *overhead* no seu uso.

• **Opacidade** - Outra desvantagem é que apesar da naturalidade em que o conhecimento pode ser expresso por produções, algoritmos não são naturalmente expressos desta forma, pois é difícil seguir o fluxo que o programa realiza ao resolver problemas utilizando regras.

### 2.4.2. *Frames*

Há abundante evidência psicológica que as pessoas utilizam conhecimentos de experiências passadas para interpretar novas situações na sua atividade cognitiva diária [Feigenbaum89]. Por exemplo, quando se visita um restaurante onde nunca se esteve antes, tem-se muitas expectativas, baseadas em visitas anteriores a outros restaurantes, do que se irá achar: menus, mesas, garçons etc.

Representar conhecimento sobre objetos e idéias típicos a situações específicas é o foco de idéias sobre a forma de representação de conhecimento denominada de *frame*.

Os *frames* foram originalmente propostos por Marvin Minsky [Minsky75] do MIT em 1975 como uma base para entender a percepção visual, diálogos em linguagem natural e outros comportamentos complexos.

*Frames* são, basicamente, estruturas para abrigar vários tipos de conhecimento [Walters88]. Conceitualmente, um *frame* representa um item (ou seja, um objeto físico) ou um conceito (ou seja, uma idéia). O conteúdo do *frame* então descreve o item de alguma forma (por exemplo, suas características, suas propriedades e seu comportamento).

Apesar de que ao longo desta dissertação este tipo de estrutura será chamada de *frame*, outros autores podem referir-se a estruturas parecidas como *objetos*, *unidades*, *conceitos* ou *entidades* [Walters88], [Feigenbaum89] .

### 2.4.2.1 *Slots*

A estrutura interna de um *frame* pode ser caracterizada provendo-se um conjunto de *slots*[2] nos quais o conhecimento associado com o *frame* possa ser armazenado.

Por exemplo, um *frame* simples para o conceito geral de cadeira poderia ter *slots* para o número de pernas ou estilo do encosto. Um *frame* para um tipo particular de cadeira tem os mesmos *slots*, pois os mesmos são **herdados** do *frame* para cadeira, mas o conteúdo dos *slots* é melhor especificado.

Para melhor visualizarmos isto temos, abaixo, a representação de um *frame* geral para cadeiras [Feigenbaum89]:

**Frame** CADEIRA

**Especialização-de** : Móvel
**Número-de-pernas**: Um inteiro ( Default = 4)
**Estilo-do-encosto** : Reto, alcochoado
**Número-de-braços**: 0,1, ou 2

[2] *Slots* são também chamados de parâmetros ou comportamentos

E para uma cadeira em particular teríamos:

**Frame** MINHA_CADEIRA

**Especialização-de** : CADEIRA
**Número-de-pernas**: 4
**Estilo-do-encosto** : Alcochoado
**Número-de-braços** : 0

Assim, fornecendo um lugar para armazenar conhecimento e desta forma criando a possibilidade de lidar com conhecimento incompleto o mecanismo de *slots* permite o raciocínio baseado na procura de confirmação de expectativas.

### 2.4.2.2 Tipos de conhecimento que um *slot* pode armazenar

Os tipos principais de conhecimento que um *slot* pode armazenar são os valores ou fatos relacionados com o *frame*. Estes dados podem ser armazenados de várias formas dependendo de suas características. Assim, alguns valores podem ser armazenados como números e outros como símbolos ou cadeias de caracteres.

Um *slot* pode armazenar um único valor ou vários dependendo do tipo de sistema em que ele esteja sendo empregado.

Outros tipos de conhecimento que um *slot* pode armazenar são :

- ***Conjunto de regras***: Em alguns sistemas um conjunto inteiro de regras pode ser armazenado em um *slot*, permitindo que conhecimento baseado em regras, assim como fatos e valores ( que são, como dissemos, os conhecimentos básicos armazenados em *slots*) sejam integrados.
- ***Procedimentos***: Os *slots* podem também conter funções, permitindo que o conhecimento procedural seja incorporado na representação de *frames*[3]. Analogamente à inclusão de regras dentro de um *slot*, a inclusão de uma função ou método provê um recurso muito poderoso de estruturar a distribuição de conhecimento procedural ou comportamental dentro de uma aplicação.

[3]Quando usados em programação orientada a objeto, estas funções são chamadas de métodos

Assim, para exemplificar, temos abaixo um *frame* genérico para restaurantes [Feigenbaum89]:

**Frame** Restaurante

**Especialização-de** : Estabelecimento_de_Comércio
**Tipos** :
Alternativas : *fast-food*, *self-service*, tradicional
Default : Tradicional
Se_Necessário : SE existir hamburguers ENTÃO Tipo = *fast-food*
SE existir pilha de bandejas ENTÃO Tipo = *self- service*
SE reservas forem necessárias ENTÃO Tipo = Tradicional
**Localização** :
Form : Um endereço
Se_Necessário : ProcurarNoMenu ( )
**Nome** :
Se_Necessário : ProcurarNoMenu ( )
**TipoDeComida** :
Alternativas: Sanduíches, chinesa, frutos do mar, francesa, brasileira, japonesa
Default : Brasileira
Se_Necessário : ProcurarNoMenu ( )
Se_Novo_Tipo : AdicionarNovoTipo ( )

**Hora_de_Operação** :
Alternativas : Um Hora_Do_Dia
Default : Aberto todas às noites, exceto Segunda-Feira
**Forma_de_Pagamento** :
Alternativas : Dinheiro, cartão de crédito, cheque.
**Alternativas** :
Se_Necessário :
AcheTodosOsRestaurantesComOMesmoEstiloDeComida ( )

Note que neste *frame,* utilizou-se várias formas de representação de conhecimento para preencher-se os seus *slots*. Por exemplo, no *slot* "Tipo" usou-se um conjunto de regras e no *slot* "Alternativas" usou-se uma função ou método para representação do conhecimento.

Note também os tipos especiais de *slots* principalmente o *default* e o Se_Necessário. O *slot default*, como o próprio nome implica, é utilizado se não for informado qual o tipo de restaurante. Assim, um programa utilizando esse *frame* assumiria se nada fosse informado sobre o tipo de restaurante que ele era um restaurante tradicional. Já o *slot* Se_Necessário contém conhecimento armazenado sob a forma de regras de produção que ajudam a deduzir qual tipo de restaurante mesmo se não for explicitamente informado.

### 2.4.2.3 Restrições

Para ser capaz de processar o conhecimento armazenado nos *slots* de um *frame*, um sistema de raciocínio deve ser capaz de interpretar os valores em um *slot* sem ambigüidades. Assim certas restrições aos conteúdos dos *slots* devem ser freqüentemente adotadas. No caso do *frame* que descreve uma cadeira, por exemplo, não teria nenhum sentido preenchermos o *slot* "Número-de-pernas" com um valor fracionário.

As principais formas de restrições são: [Walters88]

- Restrições quanto a forma de representação (por exemplo, só números inteiros seriam permitidos ).
- Restrições quanto aos valores que podem ser armazenados.
- Restrições que são independentes de valores específicos (por exemplo, não mais que três valores em um *slot* ).
- Restrições que especifiquem a quantidade máxima e/ou mínima de valores em um *slot* .
- Restrições que especifiquem o tipo de relação que um *frame* pode ter com um outro *frame*.

### 2.4.2.4 *Facets*

Restrições não são o único tipo de especificação que pode ser usado para controlar aspectos particulares do conteúdo de um *slot*. Por exemplo, o desenvolvedor pode querer especificar o tipo ou o tamanho da letra que determinada informação deva ser mostrada. O mecanismo que permite este tipo de controle chama-se *facet*.

Os *facets* contêm certos tipos de informação que são relacionados com os *slots*. Vários *facets* podem ser alocados para cada *slot*, cada um provendo um tipo diferente de controle ou característica. O conteúdo do *facet*

apropriado é verificado sempre que um tipo particular de ação está para ser realizado em conexão com o valor do respectivo *slot*.

### 2.4.2.5 Classes e subclasses

Classe é um *frame* contendo o conhecimento sobre um conjunto ou classe de objetos de um certo tipo. Considere o seguinte exemplo : Seja um base de conhecimento que lida com automóveis representada, parcialmente, na figura 2.1. Observe que todo o conhecimento relacionado a aspectos particulares de automóveis estão associados a *frames* subordinados ou descendentes do *frame* Automóveis. Entretanto, conhecimentos relacionados a outros tópicos seriam associados a outras classes de *frames*, como por exemplo, barcos ou aviões. Assim o *frame* "Automóveis" seria uma classe.

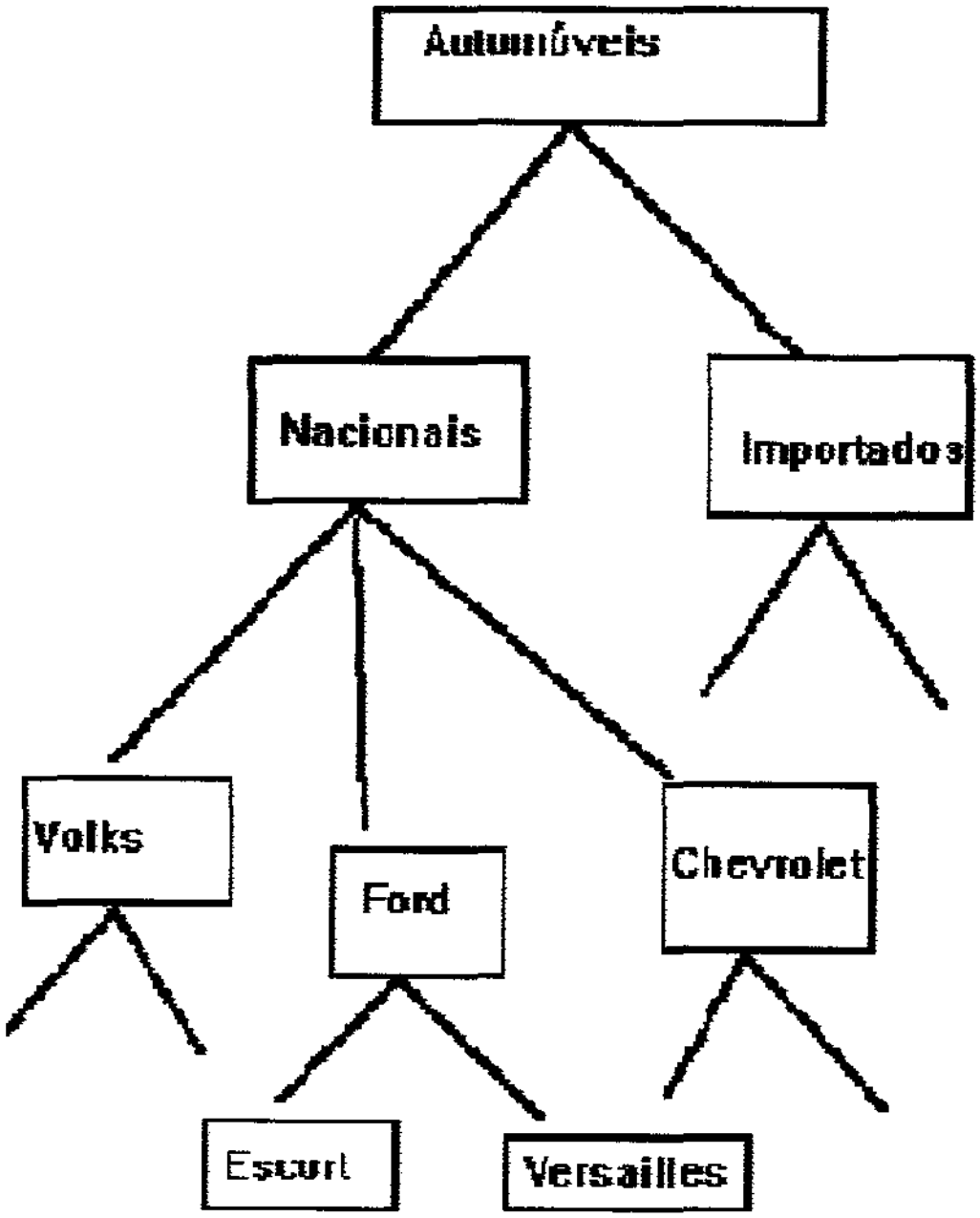



Fig. 2.1 Estrutura de uma base de conhecimento sobre automóveis

Os *frames* em uma subclasse representam uma particularização ou uma especialização da classe à qual eles se relacionam. Assim, a classe "Automóveis", na figura 2.1 tem as seguintes especializações: Nacional e Importado.

Os *frames* representando subclasses contêm informações mais detalhadas e podem, por sua vez, ser classes, pois podem existir especializações posteriores dos conceitos por eles representados. Por exemplo, o *frame* "Nacional" tem como subclasses "Volks", "Chevrolet" e "Ford", enquanto que "Importado" poderia ter como subclasses "Renault", "BMW", etc.

Observe também que "Nacional" é uma subclasse de "Automóveis" mas uma classe para "Volks" e "Chevrolet".

### 2.4.2.6 Instâncias

Mais cedo ou mais tarde, não vai ser mais possível fazer subdivisões sem perder o conceito de uma classe que represente múltiplos objetos. Neste ponto a especialização possível seria uma instância específica. Cada instância representa um exemplo ou caso ou ainda instanciação do conceito geral. Assim teríamos, seguindo o exemplo dos automóveis, o *frame* "Ford" tem como subclasse, o *frame* "Santana" e, este último poderia ter, como instância, um *frame* "MeuSantana".

### 2.4.2.7 Herança

Considere uma hierarquia de *frames*, como, por exemplo, o caso dos automóveis. Seja então um *frame* na base da hierarquia, como é o caso do *frame* "MeuSantana". Este *frame* herda dos *frames* em níveis mais altos da hierarquia várias propriedades que não estão explicitamente descritas nele. O mecanismo que permite esta transferência de propriedades de *frame* para *frame* é chamado de *herança*.

Os tipos principais de herança são:

- ***Slot*** - Acontece quando o próprio *slot*, mas não seu valor, é herdado de um *frame* pai.
- **Valor** - Acontece quando o valor de um *slot* em um *frame* é herdado do *slot* correspondente do *frame* pai.
- **Restrição** - Acontece quando as restrições aos valores e não os valores em si, é que são herdadas.
- **Múltipla** - Acontece quando um *frame* herda propriedades de duas ou mais classes distintas.

- **Vazia** - Acontece quando nenhum tipo de herança é desejável, e, portanto, todas as suas formas devem ser inibidas.

Funções ou métodos podem ser herdados das maneiras tradicionais, mas alguns tipos especiais de herança são desejáveis para o conhecimento procedural:

- **Herança Anterior** - Acontece quando a função armazenada no *slot* do filho é aplicada *antes* da função herdada do *frame* pai. A herança anterior pode ser usada para vários propósitos. A função do filho pode ser usada como um filtro, verificando se as condições são tais que a rotina geral não deva ser utilizada ou pode ser usada como um pré-processador, modificando ou ajustando os valores antes de fornecê-los para a rotina geral.
- **Herança Posterior** - Acontece quando a função armazenada no *slot* do filho é chamada *após* a função herdada do pai ter sido aplicada. Pode ser usada para que a função do *frame* filho funcione como um pós-processador, modificando ou ajustando os valores antes que eles sejam retornados pela função.

A capacidade de herança é particularmente útil, quando o processo de raciocínio envolve a criação de novas instâncias e a modificação ou adição de relacionamentos. A modificação do conteúdo de um *slot* em um *frame* pertencente a uma subclasse, como resultado do processo de raciocínio se estenderia, através da herança, por todas as subclasses descendentes ou subordinadas, sem que o sistema tenha que procurar por todas as instâncias e fazer a modificação apropriada.

### 2.4.2.8 Aplicabilidade de *frames*

Os *frames* são úteis na categorização de conhecimentos, quando este conhecimento tem alguma estrutura subjacente. Se o conhecimento pode ser relacionado a um conjunto de objetos ou conceitos, então pelo menos alguns fatos contidos na base de conhecimento podem ser aglutinados em torno destes objetos e conceitos.

Para ilustrar melhor este conceito, retornemos ao exemplo dos automóveis: aglutinando o conhecimento em torno de cada tipo de carro, um usuário poderia obter informações sobre, por exemplo, os carros da Chevrolet

sem ter que procurar por toda a base de conhecimento sobre automóveis. Este tipo de consulta poderia obter vantagem do uso de *frames*, referindo-se apenas aos *slots* de um *frame* específico ( no caso o *frame* Chevrolet). Similarmente, o usuário poderia obter informações sobre tipos de pneus sem ter que procurar por toda a base de conhecimento, bastando procurar por um *slot* específico em vários *frames* diferentes.

Assim ao se projetar um sistema baseado em *frames*, o projetista deve procurar as características da base de conhecimento, identificando se vai consistir de uma coleção solta de fatos não-relacionados ou se vai consistir de vários aglomerados de fatos relacionados. Quanto maior o número de objetos e conceitos possíveis de se aglutinar, maior a vantagem da utilização de *frames*.

### 2.4.2.9 Raciocínio usando *frames*

Devido ao fato do conhecimento armazenado em *slots* de *frames* estar disponível a mecanismos de inferência, uma regra poderia raciocinar sobre as características do *frame*, referindo-se aos valores de seus *slots*. Suponha, por exemplo, que uma aplicação lide com planejamento de rotas e que determinada ponte (chamada Ponte1) deva ser evitada porque o veículo é muito pesado. Uma regra como a especificada a seguir poderia fazer parte da base de conhecimento:

**Regra 1** - SE o peso do MeuVeículo > 3 toneladas
ENTÃO Desvie da Ponte1

Ou seja, se o valor do *slot* "Peso" do *frame* MeuVeículo for maior que 3 toneladas, então a rota deve incluir um desvio da ponte1.

Similarmente, regras podem ser usadas para processar o conhecimento estrutural da base de conhecimento. A regra seguinte mostra como isto poderia ser realizado:

**Regra 2** - SE o VEÍCULO for um membro de AUTOMÓVEIS
E SE o VEÍCULO tiver um MOTOR
E SE o MOTOR for um membro de DIESEL
ENTÃO prencha o *slot* REABASTECIMENTO de VEÍCULO com POSTO_DE_CAMINHÕES

Na primeira linha, a máquina de inferência tenta instanciar a variável VEÍCULO com cada *frame*, mas apenas as instâncias que são membros da classe AUTOMÓVEIS são mantidos. Neste ponto, vários veículos podem ser mantidos. A segunda linha instancia a variável MOTOR com qualquer *frame* que VEÍCULO seja relacionado. A terceira linha descarta qualquer instância que não tenha motor que pertence à classe DIESEL. A quarta linha preenche o *slot* REABASTECIMENTO de *cada* veículo associado com o valor POSTO_DE_CAMINHÕES. Observe que se houvesse, por exemplo, cinco veículos com motores diesel, a regra preencheria o *slot* REABASTECIMENTO de *todos* com POSTO_DE_CAMINHÕES.

### 2.4.2.10 Implementação de frames usando orientação a objetos

Nos primórdios da programação, usava-se uma perspectiva orientada aos problemas em si. Programadores e projetistas concentravam-se na resolução de problemas imediatos. A metodologia aplicada na prática era a seguinte: começava-se com um pequeno pedaço de código que parecia funcionar, então acrescentava-se mais código em camadas sucessivas. Esta metodologia funcionava para pequenos projetos, mas trazia enormes dificuldades em projetos maiores, principalmente no interfaceamento dos pedaços de código.

Depois, surgiu a Programação Procedural, que é exemplificada por linguagens do tipo FORTRAN e COBOL. As funções a serem realizadas eram embutidas em procedimentos e funções que, então, poderiam ser chamadas por outros códigos. Ainda hoje é o paradigma mais comum de programação e funciona bem para aplicações que se baseiam fortemente na implementação de algoritmos. No entanto, este paradigma tem problemas quando se tenta reusar o código para funções ligeiramente diferentes das originais e é fraco na análise orientada a dados.

O projeto e análise orientada a dados foram desenvolvidos para suportar aplicações que exigiam intenso tratamento de dados, particularmente na área empresarial. Técnicas tais como diagramas de fluxos de dados e análises entidade-relacionamento tornaram-se populares com o surgimento de linguagens de quarta geração.

Mais recentemente, técnicas orientadas a objeto combinaram a visão orientada a dados, a visão procedural e princípios de encapsulamento em uma única metodologia que se aplica ao projeto e ao desenvolvimento de sistemas e resultados recentes tendem a considerá-la superior às outras [Stroustrup97].

Mas, o que são objetos? Objetos são conceitos, abstrações ou algo com uma fronteira bem definida e significado para o problema a ser solucionado [Rumbaugh91] e servem a dois propósitos: promover entendimento do mundo real e prover uma base prática para a implementação em computadores.

Uma *classe* descreve um grupo de objetos com propriedades semelhantes, comportamento comum, relações com outros objetos e semântica comuns. Para ser considerada orientada a objeto, uma classe deve possuir quatro características:

- **Encapsulamento**: Dados e funções ( também chamadas, neste caso, métodos ) devem ser encapsuladas em uma única entidade - um objeto.
- **Abstração**: Significa que os objetos devem exibir o comportamento esperado quando os operadores padrões forem aplicados. Por exemplo, se você define um objeto que é um número complexo, é razoável que os usuários esperem que possam somar e subtrair estes objetos como o fariam usualmente.
- **Herança**: Objetos novos que sejam semelhantes a outros previamente definidos, devem reutilizar as características do objeto pai.
- **Polimorfismo**: O polimorfismo significa que funções (ou métodos) com o mesmo nome exibem comportamento diferente baseado no tipo de objeto em que forem aplicadas.

Outra idéia inerentemente ligada à programação orientada a objetos é a **ligação dinâmica** ( *dynamic binding* ) [Gorlen90], que ajuda a tornar os programas mais gerais, deixando cada classe de um grupo relacionado de classes ter uma implementação diferente de uma função particular. Os programas clientes podem, então, aplicar a função em um objeto, sem se preocupar em saber a sua classe específica. O sistema determinará, em tempo de execução, a classe específica do objeto e chamará a implementação da função desta classe.

#### 2.4.2.11 *Frames* e orientação a objetos

Pode-se ver claramente a partir das definições anteriores que *frames* e orientação a objetos são conceitos, do ponto de vista prático, intimamente entrelaçados. Mais especificamente, apesar de objetos e *frames* não serem teoricamente iguais, *frames* se adequam a ser implementados usando técnicas e linguagens orientadas a objetos.

Assim, a implementação de um *frame* seria uma classe. Seus *slots* seriam implementados como funções-membro ou elementos de dados (*data members*). As restrições são implementadas pelo compilador, que realiza uma verificação de tipos durante o processo de compilação para evitar erros em tempo de execução.

### 2.4.2.12 Vantagens e desvantagens do uso de *frames*

A maior vantagem do raciocínio baseado em *frames* é que ele provê um meio para estruturar vários tipos de dados na base de conhecimento e um arcabouço onde se pode raciocinar não só sobre os dados, mas também sobre a estrutura destes dados. Estas características tornam esta estrutura de dados particularmente útil para a representação do conhecimento espacial seguindo-se o modelo TOUR de Kuipers [Kuipers77].

As principais vantagens de uma representação baseada em *frames*: [Walters88]

- Ajuda enormemente na estruturação do projeto para uma aplicação baseada em conhecimento.
- Permite que as regras e procedimentos em uma aplicação sejam mais genéricos, portanto reduzindo o conjunto de regras e/ou o tamanho dos procedimentos e tornando a base de dados mais fácil de entender e testar.
- Agrupa os dados na base de conhecimento, freqüentemente reduzindo a complexidade de tal forma que uma aplicação pode ser feita e testada em menos tempo.
- Permite melhor entendimento da estrutura do conhecimento e suas inter-relações.
- Facilita a manutenção da base de conhecimento.
- Provê muitas das características essenciais de um objeto desde que o tipo daquele objeto seja identificado, eliminando, assim, a necessidade de derivar as características individualmente.

No entanto, existem algumas desvantagens na utilização de *frames* e a maior delas é a relacionada com sua eficiência. Todas estas vantagens e benefícios explicitados anteriormente têm um preço para serem conseguidos. A capacidade de ligação dinâmica, a capacidade dinâmica de modificar a estrutura da base de conhecimento ou modificar os *facets* associados a um *slot* durante a execução do sistema requerem que um significativo número de testes seja feito em tempo de execução e isso, obviamente, resulta em degradação da performance.

Outra desvantagem da representação por *frames* é que as técnicas orientadas a objetos, ideais para este tipo de representação, ainda não estão suficientemente disseminadas (embora caminhem a passos largos para isto) e a sua curva de aprendizado é um tanto lenta, em geral demorando alguns meses para se atingir um grau de aperfeiçoamento adequado [Rumbaugh91].

### 2.4.3 Estrutura do modelo UniGuide

Constitui-se de:

1- Representações para o conhecimento sobre um ambiente particular.
2- A descrição da posição corrente.
3- Regras de inferência para manipular 1 e 2.

As representações no modelo UniGuide, que codificam informações sobre ambientes particulares são :

- Uma representação de uma rota imperativa que dirige o usuário ao longo da rota sobre o mapa cognitivo.
- Uma representação para propriedades topológicas locais de ruas, incluindo a ordem dos lugares na rua e a geometria local das ruas em um cruzamento.
- *Frames* de referência que definem a posição relativa dos objetos geográficos, ou seja, a posição dos lugares, ruas, cruzamentos, etc, com relação a outros lugares, ruas, cruzamentos etc.
- Fronteiras divisórias, que provêem uma segunda forma de representação para a posição relativa.

A posição corrente do usuário é representada por uma estrutura *(frame)* chamada de "VocêEstáAqui". A estrutura "VocêEstáAqui" contém vários elementos, e pode descrever a posição corrente em termos de lugar, rua,

orientação na rua e *frame* de referência corrente. Esta descrição pode ser mais ou menos completa, dependendo das descrições do ambiente e da rota até aquele ponto. A estrutura "VocêEstáAqui" possui os seguintes *slots* :

- **Place** - Indica o lugar atual
- **Path** - Indica a rua atual
- **Direção** - Indica a direção atual
- *Frame* de orientação - Indica o *frame* de orientação, ou seja, o *frame* que indica qual sistema de referência está sendo utilizado no momento.
- **X** - Posição X
- **Y**- Posição Y

Por exemplo, suponha que uma pessoa esteja em uma rua A, em frente a um monumento chamado Monumento A e indo em direção ao sul. Então a estrutura "VocêEstáAqui" seria preenchida da seguinte forma :

- **Place** - Monumento A
- **Path** - Rua A
- **Direção** - Sul

Os outros *slots* (**X**, **Y** e o *frame* orientação) não seriam preenchidos simplesmente porque não temos informações suficientes. Esta é, aliás, uma das principais características dos modelos UniGuide e TOUR: poder lidar com informações apenas parcialmente especificadas. O processo de aquisição das informações restantes é feito à medida que o UniGuide navega pelo mapa e vai preenchendo as informações que faltam em elementos parcialmente especificados, transferindo informações de outras descrições que, apesar de incompletas por si sós, têm informações que completam as descrições dos elementos anteriormente apenas parcialmente especificados.

A maior parte do conhecimento no modelo UniGuide é armazenado proceduralmente nas regras de inferência. Estas regras são compostas de testes e ações, onde os testes são aplicados a informações que são acessíveis a apenas

uma pequena memória de trabalho. Este tipo de estrutura é modelada como um sistema de produção, visto no capítulo anterior.

As regras de inferência do modelo UniGuide correspondem aos seguintes tipos de representações :

- Regras que comparam a instrução de rota corrente, o ponteiro "VocêEstáAqui" e as descrições topológicas do ambiente. Podem servir para preencher espaços em cada representação com informação dos outros *slots*.
- Regras para manter a orientação atualizada ao longo da rota. Comparam informações no ponteiro "VocêEstáAqui" com as descrições do lugar e rua corrente e com base nestas informações preenchem as descrições que faltam em cada um.
- Regras que detectam características especiais do ambiente, como por exemplo ruas que dividem regiões[4].
- Regras que resolvem problemas de achar rotas.

Uma característica extremamente importante do modelo UniGuide é que ele nunca realiza uma busca sem restrições no mapa cognitivo. A qualquer ponto da operação, ele pode acessar no máximo duas instruções de rota, o ponteiro "VocêEstáAqui" e as descrições topológicas que são acessíveis a partir deles. Isto siginifica que o tempo para realizar uma operação (com exceção de algumas operações de busca) não aumenta com o tamanho do mapa cognitivo. Ao contrário, como mais conhecimento é representado, mais regras de inferência poderosas podem ser aplicadas.

## 2.4.4 Objetos geográficos

Um dos pilares fundamentais do modelo UniGuide é a descrição topológica do ambiente. O modelo UniGuide considera cinco tipos de objetos geográficos diferentes: Lugares, cruzamentos, ruas, regiões e rotas. A seguir iremos detalhar cada um deles.

### 2.4.4.1 Lugares

As características fundamentais de um ambiente são dadas pelos lugares que ele possui. Um lugar é caracterizado e reconhecido por suas

[4]O conceito de regiões, bem como dos outros objetos geográficos será explicado na próxima seção.

características visuais. A descrição de um lugar no modelo UniGuide é implementada por um *frame* chamado PLACE. Possui os seguintes *slots*:

**Name** - Contém o nome do lugar.

**On** - Contém o(s) nome(s) da(s) rua(s) em que o lugar está.

**Connect** - Contém a lista dos lugares que são conectados ao lugar sendo descrito, bem como a descrição da rota entre eles. Usualmente, este *slot* é preenchido pelo próprio sistema à medida que vai aprendendo sobre um determinado mapa.

**View** - Contém a descrição da vista a partir do lugar sendo descrito. Esta informação é fundamental para a melhor orientação do usuário ao longo da rota, bem como para a aquisição de novas informações (a partir das descrições dos lugares visíveis) no caso de lugares apenas parcialmente descritos. Consiste de três informações: O lugar visto, o ângulo aproximado (para se saber se o lugar está à frente, por trás etc.) e a distância aproximada. Por exemplo, a vista do Instituto de Computação poderia ser descrita da seguinte forma: (Instituto de Economia, ATRÁS, PERTO); (Faculdade de Engenharia Civil, DIREITA, PERTO). (Figura 2.1)

**In** - Contém uma lista de regiões nas quais o lugar sendo descrito se encontra. Serve para se ter uma hierarquia de regiões. Assim, em uma descrição da Unicamp, por exemplo, poderíamos escrever - In: (Barão Geraldo) ; (Campinas) ; (São Paulo) ; (Brasil). As regiões são descritas da mais interna para a mais externa. Isto é muito útil para resolver problemas de achar rotas em uma escala maior, como por exemplo, para achar uma rota entre a USP e a Unicamp. O sistema, sabendo que a USP está na cidade de São Paulo procuraria, então, primeiro uma rota entre São Paulo e Campinas para então procurar, na região de Campinas, uma rota para a Unicamp. Note que isso reduz sensivelmente o espaço de busca e, por conseguinte, o custo computacional. No caso de nossa implementação dividimos a Unicamp em quatro regiões e a busca, quando necessária, se resumirá à menor região possível que englobe o lugar de origem e o de destino.

**Region** - Contém o nome da região (a mais interna) em que o lugar se encontra.

**Contained** - Contém uma lista com os nomes dos lugares que estão dentro do lugar sendo descrito. Assim, no caso da descrição do Hospital das Clínicas, por exemplo, o Pronto-Socorro estaria neste *slot*. Assim se estivéssemos procurando uma rota para o Pronto-Socorro, bastaria acharmos a rota para o Hospital das Clínicas, que automaticamente encontraríamos para o Pronto-Socorro.

Como um exemplo final, a descrição do IMECC seria: (ver figura abaixo)

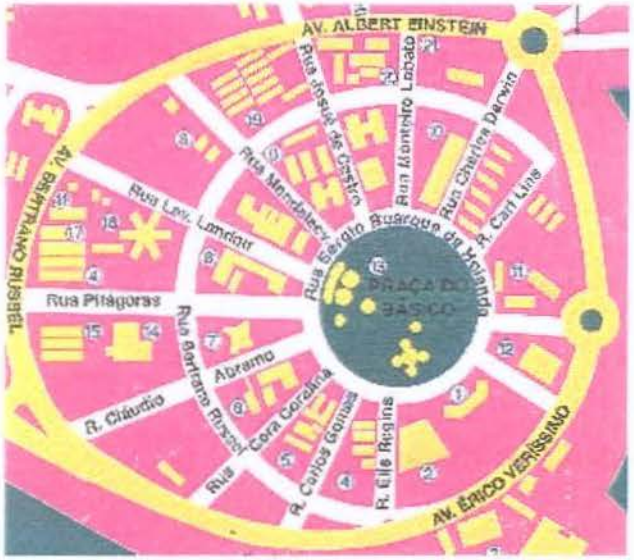



Figura 2.1 – Mapa da Unicamp (parcial)

Place <IMECC>
On:<Rua Sergio Buarque de Holanda>.
View:(<Praca do Ciclo Básico>, FRENTE, PERTO);
(<InstitutodeFilosofia>, ESQUERDA, LONGE); (<Instituto de Fisica>, DIREITA, LONGE).
In:<InternalArc>;<ExternalArc>;<LeftRegion>.
Region: <InternalArc>.

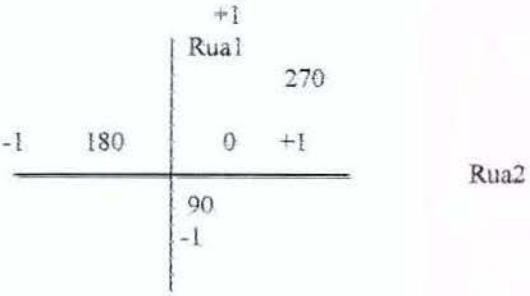
Observe que o *slot* **View** é preenchido com a vista do lugar sendo descrito a partir de uma referência diferente para cada lugar. No caso do IME escolheu-se a entrada da rua Sérgio Buarque de Holanda. Note também que a descrição "LONGE" na realidade significa que existe um objeto geográfico visível mais perto.

Os *slots* **In** e **Region** são preenchidos com nomes de regiões da Unicamp, que são explicadas, em detalhes, no apêndice II.

### 2.4.4.2 Cruzamentos

Cruzamentos são um tipo especial de lugar, herdam as propriedades, mas, no entanto, têm uma específica que o caracterizam e é de extrema importância para a caracterização das rotas. Esta propriedade é a geometria local do cruzamento.

Considere, por exemplo, um cruzamento perpendicular entre duas ruas, chamadas Rua1 e Rua2, mostrado abaixo.




O *frame* que representa o cruzamento acima seria:

Crossing <Cruzamento1>
On:<Rua1>;<Rua2>.
Star:(<Rua2>,0,+1);(<Rua2>,180,-1);
(<Rua1>,270,-1); (<Rua1>,90,+1).
In:<Region1>;<Region2>.
Region: <Region1>.

Como se pode ver, o *slot* **Star** descreve a geometria local do cruzamento. As informações do *slot* **Star** são da seguinte forma: As direções radiais (0, 90, 180, 270) são rotuladas de acordo com o sentido do relógio, mas só por uma questão de método de trabalho, pois elas são arbitrárias e locais a cada cruzamento. Seus valores absolutos são irrelevantes, pois são usadas apenas para calcular os resultados de instruções de rota TURN (Dobre), que serão vistas adiante. Assim como os valores absolutos das direções radiais são arbitrários e irrelevantes, os valores de direção das ruas (+1 e -1) também o são.

Além destas informações, os *frames* Crossing ainda possuem, herdadas do *frame* Place o *slot* **View**, explicado na seção anterior, para descrever o lugar mais destacado visível a partir do cruzamento, com o objetivo de dar maiores informações ao usuário ao longo da rota.

### 2.4.4.3 Ruas

Ruas são representadas no modelo UniGuide pelo *frame* **Path**. O *frame* **Path** possui os seguintes *slots*:

**Name** : Contém o nome da rua.

**Row** : Contém os principais lugares da rua. Assim na descrição da Av. James Clerk Maxwell, por exemplo, teríamos a Faculdade de Engenharia Elétrica e o Instituto de Geociências. (vide mapa da Unicamp no anexo)

**Shape** : Descreve a forma da rua. Na nossa implementação, aceita-se a forma reta e a circular. Serve para dois principais propósitos: Apresentação da rota graficamente e para manter a orientação correta durante a rota. Já no modelo TOUR este slot não existe, pois a implementação de Kuipers não tinha uma interface gráfica.

**Right** : Contém o nome da região que fica à direita da rua, no caso de esta rua ser uma fronteira divisória. Serve para facilitar a busca de uma rota entre lugares que estão em regiões distintas, pois se deseja saber a rota entre um lugar X que está na região X1 e um lugar Y que está na região X2, e sabe-se que a rua Z é uma fronteira divisória entre as regiões X1 e X2 então a busca da rota resumir-se-á a achar a rota entre o lugar X e a rua Z e entre a rua Z e o lugar Y. Este tipo de heurística é chamada de *stepping stone* e é explicada em detalhes no Capítulo IV.

**Left** : Contém o nome da região que fica à esquerda da rua, no caso de esta rua ser uma fronteira divisória.

**In** : Contém a lista das regiões nas quais a rua sendo descrita se encontra.

**Parallel** : Contém a lista de ruas paralelas à rua sendo descrita. Tem uma grande importância para se achar rotas, pois se duas ruas são paralelas então, provavelmente, haverá uma rua que as liga, e então a tarefa de se achar uma rota entre as duas ruas paralelas se resumirá a achar esta rua. Esta e outras estratégias heurísticas de busca serão examinadas com profundidade no capítulo IV.

### 2.4.4.4 Regiões

Regiões têm um papel muito importante no mapa cognitivo por permitir que objetos geográficos sejam referidos coletivamente. Outro papel fundamental das regiões é funcionar como níveis de abstração ao se descrever a geometria do ambiente. Assim eu poderia descrever a minha posição corrente de várias maneiras, por exemplo, estou na Unicamp, Campinas, Brasil ou América do Sul.

Como já foi dito, isto tem uma grande importância para facilitar o trabalho de se achar rotas entre lugares descritos por diferentes níveis de abstração, pois primeiro se acharia a rota entre níveis iguais de abstração e só então se concentraria a busca no menor nível. Entretanto, mesmo em mapas com um só ou com poucos níveis de abstração, como, por exemplo, a Unicamp, o conceito de regiões é útil para limitar o espaço de busca, já que se o problema envolver lugares situados em uma mesma região, a busca só precisará ser efetuada, obviamente, nela.

As regiões são representadas pelo seguinte *frame*:

**Region**

- **Name** - Contém o nome da região.
- **Boundary** - Contém o nome das ruas que formam as fronteiras da região sendo descrita.
- **Places** - Lista dos lugares da região.
- **Paths** - Lista das ruas da região.

Outro uso para o conceito de regiões é responder a perguntas do tipo se um lugar é "longe" ou "perto" de outro. Segundo a Teoria do Quarto de Rumelhart [Rumelhart74], o processo de decidir se um lugar é longe ou perto de outro, consiste em achar a distância entre os lugares e compará-la ao tamanho da menor região que os englobe. Se a distância for pequena em relação ao tamanho da região, o lugar é considerado perto, caso contrário, é considerado longe. Isto é implementado, no modelo TOUR, através da inclusão de um *slot* extra, denominado **Scale**, que contém o tamanho aproximado da região. Este *slot* não foi implementado no modelo UniGuide pelos motivos apresentados na introdução.

### 2.4.4.5 Rotas

As rotas são representadas pelo modelo UniGuide, como uma sequência de instruções que devem ser seguidas pelo ponteiro "VocêEstáAqui", o qual deve ir atualizando, ao longo da rota, os seus parâmetros.

Assim, um *frame* Route seria da seguinte forma:

**Route** :

•**From** : Este *slot* armazena o lugar de origem.

•**To** : Este *slot* armazena o lugar de destino.

•**Sequence** : Este *slot* armazena a lista de instruções que o ponteiro "VocêEstáAqui" deve seguir. Os tipos de instruções são:

**GoTo** - Esta instrução move incondicionalmente o ponteiro "VocêEstáAqui" para um determinado lugar. Possui os seguintes *slots*:

- From - Lugar[5] de origem.
- To - Lugar de destino.
- On - Rua a ser seguida.
- Dir - Direção a ser seguida.

**Turn** - Esta instrução faz com que o ponteiro "VocêEstáAqui" dobre em um certo lugar. Possui os seguintes *slots*:

- At - Especifica o lugar onde dobrar.
- OldPath - Especifica a rua original (antes de dobrar).
- OldDir - Especifica a direção original (antes de dobrar).
- Amount - Quantidade de graus a dobrar ( Em geral, 90°).
- NewPath - Nova rua a seguir após dobrar.
- NewDir - Nova direção a seguir.

**Take** - Esta instrução faz com que o ponteiro "VocêEstáAqui" toma uma sub-rota já conhecida da principal. Possui os seguintes *slots*:

- Route - Sub-rota a ser tomada.
- From - Lugar de origem.
- To - Lugar de destino.

[5] "Lugar" refere-se tanto a lugares propriamente ditos como a cruzamentos.

**Notice** - Esta instrução serve para orientar melhor o usuário ao longo da rota, fornecendo pontos de referência. Possui os seguintes *slots*:

- From - Se o usuário estiver presente neste lugar, ele verá o lugar especificado no *slot* **Remote.**
- Remote - Ponto de referência.
- Heading - Direção na qual o usuário deve olhar para ver o ponto de referência especificado em **Remote.**

**GetTo** - Esta instrução formula um problema de se achar uma rota a ser resolvido pelo UniGuide, especificando o estado do ponteiro "VocêEstáAqui" na origem e no destino. No caso de nenhuma solução ser achada, a instrução é deixada na rota e o UniGuide continua da origem especificada nesta instrução.

Seus *slots* são:

- From : Lugar de origem.
- Path1 : Rua.
- Dir1 : Direção.
- To : Lugar.
- Dir2 : Direção.

## 2.5 Aprendizagem

A capacidade de aprendizagem é fundamental para um programa ser considerado inteligente e está intimamente ligada a como se representou o conhecimento. Segundo Boden [Boden77] existem, basicamente, três tipos de aprendizagem: Aprendizagem através de exemplos (ver [Winston92] para uma descrição bastante interessante), aprendizagem supervisionada *(by being told)* e aprendizagem por experiência. O modelo UniGuide ( bem como o TOUR ) possui duas formas de aprendizagem : a aprendizagem supervisionada e a por experiência.

### 2.5.1 Aprendizagem supervisionada (*by being told*)

Neste tipo de aprendizagem o programa aprende através de informações entradas por um "instrutor". Em geral, as informações *devem* ser restritas a um certo campo de atuação para serem bem compreendidas.

O modelo TOUR aprende de forma supervisionada, a partir de descrições de rotas fornecidas por um usuário e assimila-as, incorporando este

conhecimento aos preexistentes e até mesmo construindo sua base de dados a partir destas descrições.

Este tipo de aprendizagem é feito da seguinte forma no modelo TOUR: um processador extremamente simples de linguagem natural converte a descrição em instruções de rota para o modelo TOUR (vistas em 2.4.4.5). Com estas instruções, o TOUR segue uma série de regras [Kuipers77] para assimilação e expansão de seus conhecimentos. O processador converte as instruções da seguinte forma: (Tabela 2.1)

| *Palavra - chave* | *Instrução correspondente* |
|---|---|
| ***Siga pela / Vá pela*** | GoTo/ On |
| ***Siga até / Vá até*** | GoTo/To |
| ***Começando / Iniciando*** | GoTo/From |
| ***Dobre*** | Turn |
| ***à direita*** | Turn/Dir |
| ***à esquerda*** | Turn/Dir |
| ***em/ no*** | Turn/At |

Tabela 2.1 – Instruções de rota e respectivas palavras-chave

O modelo UniGuide, ao contrário do TOUR, aprende de forma supervisionada pela descrição do novo lugar em GeoDL, que é a linguagem de descrição para o sistema implementado e que será minuciosamente explicada no próximo capítulo. Sendo assim, o UniGuide precisa da descrição de um ambiente em GeoDL, para que após a compilação dessa descrição, possa fazer inferências.

### 2.5.2 Aprendizagem por experiência

Neste tipo, o programa aprende "fazendo", ou seja, aprende com cada situação nova que enfrenta. Um exemplo de programa que utiliza este tipo de aprendizagem é um programa de A. L. Samuels [Winston92] para o jogo de damas. Samuels dotou-o de um sistema de 38 parâmetros (tais como ameaça de ataque duplo ou controle do centro) com pesos associados, que aumentavam ou diminuíam de acordo com quão útil foram estes parâmetros para o sucesso do

programa em um jogo. Outros exemplos podem ser encontrados nos programas que utilizam redes neurais.

O modelo UniGuide implementa este tipo de aprendizagem da seguinte forma : toda vez que o usuário solicita uma informação sobre uma determinada rota entre dois pontos, ele a armazena em um *frame* **Route** após a solução ser achada, além de inserir no *slot* **Connect** da origem e do destino esta informação, aprendendo, assim, cada vez mais sobre o mapa através de sua própria experiência. Utilizando as instruções de rota armazenadas no *slot* **Connect,** o UniGuide reconstitui uma rota em linguagem natural convertendo as instruções na forma da tabela 2.1.

## 2.6 - Considerações finais

Neste capítulo vimos as principais características dos modelos UniGuide e TOUR, suas semelhanças e diferenças e a forma como representam o conhecimento espacial. De uma forma geral, o TOUR é mais geral, mas, no entanto, possui algumas dificuldades de ordem prática, especialmente na entrada de dados, enquanto que o UniGuide tem aplicabilidade mais restrita, sendo, entretanto, muito mais prático e versátil que o TOUR, inclusive utilizando mecanismos de busca mais sofisticados que o TOUR, embora não sendo totalmente fiel à forma que as pessoas normalmente realizam .

Mais especificamente, os dois modelos possuem a mesma forma de representação do conhecimento, mas diferem na forma de entrada dos dados e em alguns mecanismos de busca utilizados.

No próximo capítulo veremos a linguagem GeoDL, que é a linguagem adotada pelo UniGuide para a descrição de mapas.

# Capítulo III

## A linguagem de descrição GeoDL

*Este capítulo justifica o desenvolvimento de um linguagem de descrição geográfica, apresenta suas características em profundidade, incluindo detalhes técnicos de sua implementação e termina apresentando uma descrição completa em GeoDL de um mapa exemplo.*

### 3.1 - Introdução

Uma das maiores deficiências do modelo TOUR é a dificuldade de entrada dos dados, pois, como foi originalmente proposto, a entrada de dados só era possível através de descrições em LISP dos objetos. Além disso, esta forma de entrada de dados impediria que outros mapas fossem descritos sem modificação do programa em si, o que limitaria sensivelmente a utilização da ferramenta.

Por estas razões, resolveu-se desenvolver uma linguagem de descrição que fornecesse a capacidade, a qualquer pessoa que a dominasse, de descrever qualquer mapa utilizando-a sem qualquer modificação no programa em si.

Os principais objetivos ao projetar-se a linguagem foram:

- A linguagem deveria descrever um mapa usando os conceitos do modelo UniGuide.
- A linguagem deveria ser capaz de descrever mapas com características as mais diversas possíveis, como mapas com ruas curvas e retas, mapas

grandes com diversos níveis hierárquicos (como por exemplo o mapa de um país) etc.

• A linguagem deveria ser fácil de aprender, o que foi pretendido através de sua sintaxe extremamente simples.

• A linguagem deveria ser fácil de utilizar, o que originou a inclusão de um poderoso editor/compilador integrado ao programa, de tal forma que o usuário pode, sem sair do programa, modificar ou adicionar alguma descrição no mapa. Além disto, o editor possui modelos de estruturas GeoDL (*templates*) prontos para serem inseridos pelo usuário no arquivo contendo a descrição.

## 3.2 - A linguagem GeoDL

### 3.2.1 Forma Normal de Backus

A forma normal de Backus para GeoDL é mostrada a seguir:

1. <programa>::=
   **BeginFile** < Bloco> **EndFile**.
2. <bloco> ::=
   [ <objeto place> ]
   [ <objeto crossing> ]
   [ <objeto path > ]
   [ <objeto region > ]

3. <objeto place > ::=
   **Place** <nome>
   **BeginDescription**
   [ **On :** <nome> {; <nome>}.]
   [ **View :** ( <nome>,<número> | <direções>,<número> | <distâncias>) {;(<nome>,<número> | <direções>,<número> | <distâncias>) }.]
   [ **Region :** <nome> {; <nome>}.]
   [ **In :** <nome> .]
   [ **Contained :** <nome> {; <nome>}.]
   [ **XPos :** <número>.]
   [ **YPos :** <número>.]
   [ **Connect :** <nome> {; <nome>}.]

**EndDescription**

4. <objeto crossing > ::=
**Crossing** <nome>
**BeginDescription**
[ **On :** <nome> {; <nome>}.]
[ **Star :** ( <nome>,<número>,<sentido>)
{;(<nome>,<número>,<sentido>) }.]
[ **View :** ( <nome>,<número> | <direções>,<número> | <distâncias>)
{;(<nome>,<número> | <direções>,<número> | <distâncias>) }.]
[ **Region :** <nome> {; <nome>}.]
[ **In :** <nome> .]
[ **Contained :** <nome> {; <nome>}.]
[ **XPos :** <número>.]
[ **YPos :** <número>.]

**EndDescription**

5. <objeto path > ::=
**Path** <nome>
**BeginDescription**
[ **Row :** <nome> {; <nome>}.]
[ **Shape :** <forma> [, <número>, <número>, <número>].]
[ **Right :** <nome> {; <nome>}.]
[ **Left :** <nome> {; <nome>}.]
[ **Parallel :** <nome> {; <nome>}.]
[ **In :** <nome> {; <nome>}.]
**EndDescription**
6. <objeto region > ::=
**Region** <nome>
**BeginDescription**
[ **Type:** <tipo de região>.]
[ **Scales:** <número>.]
[ **Boundary:** <nome> {; <nome>}.]
[ **Places:** <nome> {; <nome>}.]
[ **Paths :** <nome> {; <nome>}.]
**EndDescription**
7. <nome> ::= < <letra> { <letra>|<dígito>| <espaço>|<ponto>} >
8. <letra> ::= a|b|c|d|e|f|g|h|i|j|l|k|m|n|o|p|q|r|s|t|u|v|w|x|y|z

9. <número> ::= <dígito>{<dígito>}
10. <dígito>::= 0|1|2|3|4|5|6|7|8|9
11. <direções> ::= **front** | **rear** | **right** | **left**
12. <distâncias> ::= **near** | **far**
13. <sentido> ::= **up** | **down**
14. <forma> ::= **straight** | **curved**
15. <tipo de região> ::= **country** | **state** | **city** | **quarter** | **local**
16. <espaço > ::= .
17. <ponto> ::= . .

### 3.2.2 Características Gerais

A linguagem GeoDL ( Geographic Description Language) é uma linguagem de descrição de *frames* e segue fielmente a estrutura dos *frames* de descrição de objetos geográficos explicados no Capítulo II.

A seguir veremos detalhadamente como descrever todos os tipos de objetos geográficos.

### 3.2.2.1 – Place

O campo **On** indica a(s) rua(s) em que o lugar se encontra. Assim, a seguinte declaração de um campo **On** seria válida:

**On :** <Rua1>;<Rua2>;<Rua3>.

O campo **View** descreve a vista de um determinado lugar. É constituído por uma lista de tuplas indicando o nome do lugar, sua distância aproximada e sua direção.

A direção deve ser um dos seguintes valores pré-definidos: RIGHT para lugares situados à direita, LEFT para lugares situados à esquerda, FRONT para lugares em frente e REAR para lugares atrás ou qualquer valor entre 0 e 359, sendo que 0 corresponde a FRONT, 90 a RIGHT, 180 a REAR e 270 a LEFT.

A distância deve ser um inteiro positivo, cujo valor máximo não deve ultrapassar 32768. É importante destacar que o sistema de orientação usado deve ser o *local*, ou seja, cada lugar tem um ponto (usualmente sua

entrada) que vai servir de base para a localização dos outros em relação a este ponto.[Retz-Schmidt88]

Para facilitar a descrição o usuário pode usar as palavras-chave NEAR e FAR[1] para indicar a distância. Assim a seguinte declaração seria válida:

**View :** (<lugar1>,FRONT, NEAR) ; (<lugar2>, 180, FAR).

O campo **In** descreve a(s) região(ões) em que o lugar sendo descrito está. Assim a seguinte descrição seria válida:

**In :** <Region1>; <Region2>.

O campo **Region** indica a região mais interna a qual o lugar pertence. Uma descrição válida seria:

**Region :** <Region1>.

O campo **Contained** descreve uma lista de lugares que o lugar sendo descrito contém. Uma declaração válida seria:

**Contained :** <Lugar1>;<Lugar2>.

O campo **Connected** contém as rotas conhecidas do lugar sendo descrito. Geralmente é preenchido automaticamente pelo sistema. É composto por uma lista de nomes de rota. Uma descrição válida seria:

**Connected :** <Rota1>;<Rota2>;<Rota3>. É importante notar que as rotas especificadas neste campo devem ter sua descrição no mesmo arquivo.

Os campos XPos e YPos representam a posição no mapa (em coordenadas lógicas de tela ) do objeto. Na implementação está disponível uma função para a automatização da entrada destes valores através do *mouse* (ver capitulo V).

### 3.2.2.2 Crossing

[1]Novamente com a ressalva que FAR, na realidade, significa "menos perto".

Como um objeto **Crossing** é descendente de um objeto **Place**, ele herda todas as suas características e, portanto, pode ser descrito exatamente igual a um **Place**. Além das características comuns ao objeto **Place**, um objeto **Crossing** possui uma específica, que o caracteriza: A descrição da geometria local do cruzamento. A geometria é descrita no campo **Star** e consiste de uma lista de tuplas contendo, cada uma, três informações: O nome da rua, direção e o sentido da rua. A direção representa a direção de cada rua em relação ao cruzamento sendo descrito, deve ser um inteiro entre 0 e 359. O sentido deve ser a palavra-chave UP ( representando +1) ou DOWN (representando -1). Assim, uma descrição válida seria:

**Star :** (Rua1, 0, UP); (rua2, 180, DOWN).

Observe também que o campo **View**, na descrição de um cruzamento, serve para informar ao usuário um ponto de referência ao longo da rota.

### 3.2.2.3 Path

Analogamente a um objeto **Place**, a descrição de um objeto **Path** começa da seguinte forma: Path <NomeDoObjeto>. Após BeginDescription, os seguintes campos são válidos : **Row**, **Shape**, **Right**, **Left**, **In** e **Parallel**.

O campo **Row** contém a lista dos lugares que estão na rua sendo descrita (inclusive cruzamentos). É descrito, então, por uma lista de nomes. Uma descrição válida seria:

**Row :** <lugar1> ; <cruzamento1>;<lugar2>.

O campo **Shape** representa a forma da rua. Pode ser STRAIGHT, para uma rua reta e CURVED, para uma rua curva. No caso de **Shape** ser CURVED, as coordenadas do centro e o raio aproximado devem ser indicados (para permitir a rota ser mostrada graficamente), após uma vírgula.

Uma descrição válida seria:

**Shape :** Straight ou **Shape :** Curved, 100,200,100.

Os campos **Right** e **Left** contêm o nome da região que está à direita ou à esquerda da rua sendo descrita (se esta for uma rua de fronteira[2]). Uma descrição válida seria:

**Right :** < Region2>.

O campo **In** contém a região à qual a rua pertence. Uma descrição válida seria:

**In :** < EstaRegião>.

O campo **Parallel** contém a lista das ruas que são paralelas à rua sendo descrita. Consiste em uma lista com os nomes das ruas. Uma descrição possível seria:

**Parallel :** <rua1>; <rua2>; <rua3>.

Para finalizar temos um exemplo de descrição da Av. James Clerk Maxwell na Unicamp (ver figura 3.1):


Figura 3.1 – Mapa da Unicamp (parcial)

**Path** <Av. James Clerk Maxwell>

**BeginDescription**

**Row:**<CROSSING33>;<Praca da Paz>;<Faculdade de Engenharia Eletrica>;<CROSSING36>;<CROSSING39>;<Instituto de Geociencias>.

**Parallel :** <Rua da Reitoria>;<Av. Martin Luther King>.

---

[2]Para maiores detalhes ver capitulo II

**Shape:** Straight.
**In :** <RightRegion>.
**EndDescription**

Observe que como a Av. James Clerk Maxwell não é fronteira de nenhuma região, os campos Right e Left não são declarados. CROSSING33, CROSSING36 e CROSSING39 são respectivamente os cruzamentos com a Av. Albert Einstein, com a Rua 6 de Agosto e com a Av. Oswaldo Cruz.

### 3.2.2.4 Regiões

Analogamente a todos os outros objetos geográficos, as regiões têm sua descrição iniciada do seguinte modo: **Region** <Nome da região>. Os seguintes campos são válidos:

**Type** - Tipo da região. Pode ser país (COUNTRY), estado (STATE), cidade (CITY), bairro (QUARTER) ou local (LOCAL).
**Boundary** - Lista com os nomes das ruas que formam a sua fronteira.
**Places** - Lista dos lugares da região.
**Paths** - Lista das ruas da região.
**Scale** - Tamanho aproximado da região.

Assim, uma descrição válida do arco interno (ver figura 3.2) da Unicamp seria:

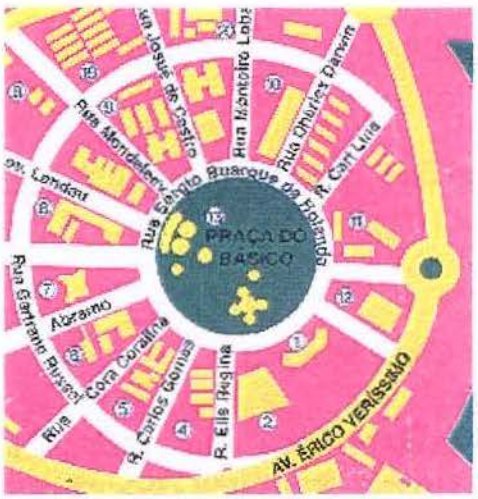


Figura 3.2 - Arco interno da Unicamp

**Region** < ArcoInterno>
**BeginDescription**
**Boundary :** <Av. Bertrand Russell>; <Av. Erico Verissimo>.
**Places :** <IME>; <IEL>; <Biblioteca Central>.
**Paths :** <Rua Sergio Buarque de Holanda>; < Rua Monteiro Lobato>;
<Rua Elis Regina>.

### 3.2.3 O compilador GeoDL

Na implementação é fornecido um compilador integrado. O compilador é do tipo descendente recursivo, com uma única passagem e com os analisadores léxico e sintático/semântico separados. Para informações gerais sobre compiladores ver [Vieira81], [Kowaltowski83], [Aho86] e [Cormen91]. Para uma descrição completa e extremamente detalhada do compilador desenvolvido para GeoDL, inclusive com código-fonte ver o Apêndice II.

## 3.3 Modo de uso

Para finalizarmos o assunto sobre GeoDL, iremos agora dar um exemplo de como descrever um mapa usando a linguagem. Considere a figura 3.2. Como vimos, este mapa corresponde ao arco interno da Unicamp. Para uma compreensão mais profunda dos mecanismos de descrição de mapas, o Apêndice III fornece a descrição completa da Unicamp em GeoDL. Abaixo segue uma parte da descrição:

```
BeginFile
{Descricao dos locais do arco interno da Unicamp}
```

Place <Biblioteca central>

BeginDescription

On:<Rua Sergio Buarque de Holanda>.

View: (<Teatro de Arena>,90,10);(<Restaurante II>,0,10);

(<Ginasio Multidisciplinar>,180,10);

(<Faculdade de Educacao Fisica>,270,10).

In:<InternalArc>;<ExternalArc>;<LeftRegion>.

Region: <InternalArc>.

XPos:152.

YPos:493.

EndDescription

Place <Teatro de Arena>

BeginDescription

On:<Rua Sergio Buarque de Holanda>.

View: (<Biblioteca central>,90,10);(<Ginasio Multidisciplinar>,0,10);

(<Restaurante II>,180,10).

In:<InternalArc>;<ExternalArc>;<LeftRegion>.

Region: <InternalArc>.

XPos:157.

YPos:461.

EndDescription

Place <Ciclo Basico>

BeginDescription

On:<Rua Sergio Buarque de Holanda>.

View: (<Teatro de Arena>,270,10);

(<Instituto de Fisica>,90,10).

In:<InternalArc>;<ExternalArc>;<LeftRegion>.

Region: <InternalArc>.

XPos:211.

YPos:388.

EndDescription

Place <Praca Henfil>

BeginDescription

On:<Av. Erico Verissimo>;<Rua Euclides da Cunha>;<Av. Martin Luther King>.

View: (<Diretoria Academica>,90,10);

(<Faculdade de Educacao Fisica>,180,10).

In:<InternalArc>;<ExternalArc>;<LeftRegion>.

Region: <InternalArc>.

XPos:191.

YPos:584.

EndDescription

Place <Ginasio Multidisciplinar>

BeginDescription

On:<Rua Elis Regina>.

View: (<Instituto de Artes>,90,20);(<Biblioteca Central>,0,10);

(<PLACE13>,180,10).

In:<InternalArc>;<ExternalArc>;<LeftRegion>.

Region: <InternalArc>.

XPos:90.

YPos:448.

EndDescription

Place <Instituto de Artes>

BeginDescription

On: <Rua Sergio Buarque de Holanda>.

View: (<Praca do ciclo basico>,90,10);(<PLACE2>,0,5);
(<Ginasio Multidisciplinar>,0,10);(<PLACE3>,180,5);
(<IEL>,180,10).
In:<InternalArc>;<ExternalArc>;<LeftRegion>.
Region: <InternalArc>.
XPos:130.
YPos:433.
EndDescription.

Place <IEL>
BeginDescription.
On:<Rua Sergio Buarque de Holanda>.
View: (<Praca do ciclo basico>,90,10);(<PLACE3>,0,5);
(<PLACE4>,180,5);(<Instituto de Artes>,0,10);
(<Instituto de Filosofia>,180,10).
In:<InternalArc>;<ExternalArc>;<LeftRegion>.
Region: <InternalArc>.
XPos:137.
YPos:408.
EndDescription.

Place <Instituto de Filosofia>
BeginDescription.
On:<Rua Sergio Buarque de Holanda>.
View: (<Praca do ciclo basico>,90,10);(<PLACE4>,0,5);
(<PLACE5>,180,5);(<IEL>,0,10);
(<IMECC>,180,10).
In:<InternalArc>;<ExternalArc>;<LeftRegion>.
Region: <InternalArc>.
XPos:156.

YPos:390.

EndDescription.

Place <IMECC>

BeginDescription.

On:<Rua Sergio Buarque de Holanda>.

View: (<Praca do ciclo basico>,90,20);(<PLACE5>,0,5);

(<PLACE6>,180,5);(<Instituto de Filosofia>,0,10);

(<Instituto de Fisica>,180,10).

In:<InternalArc>;<ExternalArc>;<LeftRegion>.

Region: <InternalArc>.

XPos:179.

YPos:379.

EndDescription.

Place <Instituto de Fisica>

BeginDescription.

On:<Rua Sergio Buarque de Holanda>.

View: (<Praca do Ciclo Basico>,90,20);(<PLACE6>,0,5);

(<PLACE7>,180,5);(<IMECC>,0,10);(<Instituto de Quimica>,180,10).

In: <InternalArc>;<ExternalArc>;<LeftRegion>.

Region:<InternalArc>.

XPos:224.

YPos:382.

EndDescription.

Place <Instituto de Quimica>

BeginDescription.

On:<Rua Sergio Buarque de Holanda>.

View: (<Praca do Ciclo Basico>,90,20);(<PLACE8>,0,5);

(<PLACE9>,180,5);(<Instituto de Fisica>,0,10);

(<Instituto de Biologia>,180,10);

(<Faculdade de Engenharia Mecanica>,270,10).

In: <InternalArc>;<ExternalArc>;<LeftRegion>.

Region:<InternalArc>.

XPos:262.

YPos:426.

EndDescription

Place <Instituto de Biologia>

BeginDescription

On:<Rua Sergio Buarque de Holanda>.

View: (<Praca do Ciclo Basico>,90,20);(<PLACE10>,0,5);

(<PLACE12>,180,5);(<Instituto de Quimica>,0,10);

(<Diretoria Academica>,180,20);

(<Faculdade de Engenharia de Alimentos>,270,10).

In: <InternalArc>;<ExternalArc>;<LeftRegion>.

Region:<InternalArc>.

XPos:254.

YPos:537.

EndDescription

Place <Diretoria Academica>

BeginDescription

On:<Rua Sergio Buarque de Holanda>.

View: (<Praca do Ciclo Basico>,90,20);(<PLACE12>,0,5);

(<PLACE1>,180,5);(<Instituto de Biologia>,0,10);

(<Restaurante II>,180,20);

(<Praca Henfil>,270,10).

In: <InternalArc>;<ExternalArc>;<LeftRegion>.

Region:<InternalArc>.

XPos:214.

YPos:506.

EndDescription.

Place <Restaurante II>

BeginDescription.

On: <Rua Sergio Buarque de Holanda>.

View: (<Praca do Ciclo Basico>,90,20);(<PLACE1>,0,5);

(<PLACE2>,180,20);(<Diretoria Academica>,0,10);

(<Biblioteca Central>,180,5).

In: <InternalArc>;<ExternalArc>;<LeftRegion>.

Region:<InternalArc>.

XPos:172.

YPos:504.

EndDescription.

Place <Praca do Ciclo Basico>

BeginDescription.

On:<Rua Sergio Buarque de Holanda>.

In:<InternalArc>;<ExternalArc>;<LeftRegion>.

Region: <InternalArc>.

XPos:192.

YPos:441.

EndDescription.

Place <Faculdade de Educacao Fisica>

BeginDescription.

On:<Av. Erico Verissimo>.

View:(<Ginasio Multidisciplinar>,90,10);(<Biblioteca Central>,90,10);

(<Praca Henfil>,0,10).

In:<ExternalArc>;<LeftRegion>.

Region:<ExternalArc>.

XPos:77.

YPos:537.

EndDescription

Place <DCC>

BeginDescription

On:<Av. Albert Einstein>.

View:(<PLACE25>,0,5);(<Instituto de Economia>,270,10);

(<Faculdade de Engenharia Civil>,0,10).

In:<ExternalArc>;<LeftRegion>.

Region:<ExternalArc>.

XPos:159.

YPos:181.

EndDescription

Place <Instituto de Economia>

BeginDescription

On:<Av. Bertrand Russell>.

View:(<IMECC>,90,5);(<DCC>,270,10);

(<Faculdade de Educacao>,180,10).

In:<ExternalArc>;<LeftRegion>.

Region:<ExternalArc>.

XPos:155.

YPos:291.

EndDescription

{Descricoes dos cruzamentos}

Crossing <PLACE1>

BeginDescription

On:<Rua Sergio Buarque de Holanda>;<Rua Euclides da Cunha>.

View:(<Diretoria Academica>,90,10).

XPos:190.

YPos:505.

EndDescription.

Crossing <PLACE2>

BeginDescription.

On:<Rua Sergio Buarque de Holanda>;<Rua Elis Regina>.

View:(<Instituto de Artes>,90,10).

XPos:133.

YPos:442.

EndDescription.

Crossing <PLACE3>

BeginDescription.

On:<Rua Sergio Buarque de Holanda>;<Rua Carlos Gomes>.

View:(<IEL>,90,10).

XPos:137.

YPos:419.

EndDescription.

Crossing <PLACE4>

BeginDescription.

On:<Rua Sergio Buarque de Holanda>;<Rua Cora Coralina>.

View:(<Instituto de Filosofia>,90,10).

XPos:149.

YPos:400.

EndDescription.

Crossing <PLACE5>

BeginDescription.

On:<Rua Sergio Buarque de Holanda>;<Rua Claudio Abramo>.

View:(<IMECC>,90,10).

XPos:170.

YPos:386.

EndDescription.

Crossing <PLACE6>

BeginDescription.

On:<Rua Sergio Buarque de Holanda>;<Rua Pitagoras>.

View:(<Instituto de Fisica>,90,10).

XPos:192.

YPos:380.

EndDescription.

Crossing <PLACE7>

BeginDescription.

On:<Rua Sergio Buarque de Holanda>;<Rua Lev Landau>.

View:(<Instituto de Fisica>,90,10).

XPos:215.

YPos:381.

EndDescription.

Crossing <PLACE8>

BeginDescription.

On:<Rua Sergio Buarque de Holanda>;<Rua Mendeleiev>.

View:(<Instituto de Quimica>,90,10).

XPos:238.

YPos:395.

EndDescription.

Crossing <PLACE9>

BeginDescription.

On:<Rua Sergio Buarque de Holanda>;<Rua Josue de Castro>.

View:(<Instituto de Quimica>,90,10).

XPos:256.

YPos:414.

EndDescription.

Crossing <PLACE10>

BeginDescription.

On:<Rua Sergio Buarque de Holanda>;<Rua Monteiro Lobato>.

View:(<Instituto de Biologia>,90,10).

XPos:261.

YPos:440.

EndDescription.

Crossing <PLACE11>

BeginDescription.

On:<Rua Sergio Buarque de Holanda>;<Rua Charles Darwin>.

View:(<Instituto de Biologia>,90,10).

XPos:256.

YPos:468.

EndDescription.

Crossing <PLACE12>

BeginDescription.

On:<Rua Sergio Buarque de Holanda>;<Rua Carl Linne>.

View:(<Diretoria Academica>,90,10).

XPos:239.

YPos:442.

EndDescription.

Crossing <PLACE13>

BeginDescription.

On:<Av. Erico Verissimo>;<Rua Elis Regina>;<Av. Bertrand Russell>;
<Av. Albert Einstein>.

View:(<Ginasio Multidisciplinar>,90,10).

XPos:43.

YPos:447.

EndDescription.

Path <Rua Sergio Buarque de Holanda>

BeginDescription.

Row:<IMECC>;<PLACE5>;<Instituto de Filosofia>;<PLACE4>;<IEL>;<PLACE3>;
<Instituto de Artes>;<PLACE2>;<Praca do Ciclo basico>;
<Teatro de Arena>;<Biblioteca Central>;<Ciclo basico>;
<Restaurante II>;<PLACE1>;<PLACE12>;<PLACE11>;
<Instituto de Biologia>;<PLACE10>;<Instituto de Quimica>;<PLACE8>;
<Instituto de Fisica>;<PLACE6>;<Diretoria Academica>.

Shape:Curved,193,441,65.

EndDescription

Path <Rua Elis Regina>

BeginDescription

Row:<PLACE2>;<Ginasio Multidisciplinar>;<PLACE13>.

Shape:Straight.

EndDescription

Path <Rua Carlos Gomes>

BeginDescription

Row:<PLACE3>;<IEL>;<PLACE18>;<PLACE28>.

Shape:Straight.

EndDescription

Path <Rua Cora Coralina>

BeginDescription

Row:<PLACE4>;<PLACE17>;<PLACE27>.

Shape:Straight.

EndDescription

Path <Rua Claudio Abramo>

BeginDescription

Row:<PLACE5>;<PLACE16>;<PLACE26>.

Shape:Straight.

EndDescription

Path <Rua Pitagoras>

BeginDescription

Row:<PLACE6>;<PLACE15>;<PLACE25>.

Shape:Straight.

EndDescription

Path <Rua Lev Landau>

BeginDescription

Row:<PLACE7>;<PLACE19>.

Shape: Straight.

EndDescription

Path <Rua Mendeleiev>

BeginDescription

Row:<PLACE8>;

<PLACE20>;<PLACE29>.

Shape: Straight.

EndDescription

Path <Rua Josue de Castro>

BeginDescription

Row:<PLACE9>;<PLACE21>;<PLACE30>.

Shape: Straight.

EndDescription

Path <Rua Monteiro Lobato>

BeginDescription

Row:<PLACE10>;<PLACE22>;<PLACE31>;<Creche>.

Shape: Straight.

EndDescription

Path <Rua Charles Darwin>

BeginDescription

Row:<PLACE11>;<PLACE23>;<PLACE32>;<Praca Carlos Drummond de Andrade>.

Shape: Straight.

EndDescription

Path <Rua Carl Linne>

BeginDescription

Row:<PLACE12>;<PLACE24>.

Shape: Straight.

EndDescription

Path <Rua Euclides da Cunha>

BeginDescription

Row:<PLACE1>;<PLACE14>;<Praca Henfil>.

Shape: Straight.

EndDescription

Path <Av. Bertrand Russell>

BeginDescription

Row:<Instituto de Economia>;<Faculdade de Educacao>;
<Faculdade de Engenharia Mecanica>;<Faculdade de Engenharia de Alimentos>;
<PLACE15>;<PLACE13>;<PLACE18>;<PLACE17>;<PLACE16>;
<PLACE19>;<PLACE20>;<PLACE21>;<PLACE22>;<PLACE23>;<PLACE24>.

Shape: Curved,193,441,150.

EndDescription

Path <Av. Albert Einstein>

BeginDescription

Row:<DCC>;<Nucleos Interdisciplinares>;

<Faculdade de Engenharia Civil>;

<Praca Carlos Drummond de Andrade>;

<PLACE28>;<PLACE27>;<PLACE26>;<PLACE25>;<PLACE29>;

<PLACE30>;<PLACE31>;<PLACE32>;<PLACE13>;<PLACE33>.

Shape: Straight.

EndDescription

## 3.4 Considerações finais

A linguagem GeoDL, da forma que foi implementada, serve bastante bem aos nossos propósitos, descrevendo de forma completa e elegante um mapa razoavelmente complexo. Entretanto para mapas maiores e extremamente complexos ou para utilização em GIS's[3] [Gatrell93] é necessário sua expansão, o que é facilitado pelas suas características e implementação orientada a objeto de seu compilador.

Vários outros objetos poderiam ser acrescentados aos já manejados por GeoDL, entre os quais poderíamos citar :

- Viadutos (uma especialização de RUA).
- Ruas com formas diferentes de reta e círculo.
- Objetos de hierarquia maior, por exemplo, cidades ou países.
- Objetos que suportem características especiais, como por exemplo, ruas asfaltadas ou só com calçamento ou bairros com rede de esgotos e telefone. Ou ainda informações sobre densidade populacional e economia da região. O objetivo disto é permitir que o usuário disponha de informações suficientes para consultas tais como "Localize e mostre (possivelmente através de uma mudança de cor) os bairros que tenham rede de esgotos". Para maiores informações sobre estruturas de dados adequadas a este objetivo ver [Egenhofer91] e [Franklin91].

No próximo capítulo veremos os métodos de busca que o UniGuide utiliza para achar uma rota.

---

[3]Geographic Information Systems

# Capítulo IV

## Estratégias de Busca

*Neste capítulo mostraremos detalhadamente as estratégias de busca utilizadas pelo UniGuide para resolver problemas em que se pede uma rota, os tipos de situações em que cada uma pode ser aplicada, bem como várias comparações de eficácia entre estas estratégias.*

### 4.1 Introdução

O modelo UniGuide possui várias estratégias de busca, sendo que a maioria baseia-se em métodos fortemente heurísticos, principalmente para assegurar a maior utilização possível dos conhecimentos armazenados nos *frames*, durante a busca, e também para assegurar que o espaço de busca seja o mais reduzido possível e relativamente independente do tamanho do mapa.

Basicamente, o modelo UniGuide, para realizar uma busca, utiliza duas categorias principais de algoritmos : algoritmos que utilizam propriedades especiais do mapa e/ou dos lugares de destino e origem e algoritmos baseados em busca heurística ou não em nós de uma árvore de busca.

Os pertencentes ao primeiro grupo são mais poderosos, utilizam ao máximo os conhecimentos armazenados nos *frames*, restringindo ao mínimo qualquer busca não-limitada e são relativamente insensíveis ao tamanho total do mapa. Entretanto não são de uso geral e dependem fortemente das características específicas do mapa.

Já os pertencentes ao segundo grupo, são de uso geral, mas, no entanto, não utilizam intensivamente os conhecimentos armazenados nos *frames* e seu desempenho é mais sensível ao tamanho do mapa. O UniGuide só os utiliza quando os do primeiro grupo não forem suficientes para achar uma solução.

Uma outra diferença importante entre os dois grupos é que os algoritmos do primeiro grupo tendem a fornecer uma solução mais "natural", ou seja, uma solução que uma pessoa teria maior probabilidade de fornecer, mesmo que na maioria das vezes esta não seja a solução ótima do ponto de vista computacional.

## 4.2 Algoritmos que utilizam propriedades especiais do mapa

Os tipos de problemas em que estes algoritmos podem ser utilizados são os seguintes:

- Problemas cuja solução seja uma rota já armazenada em um *frame*.
- Problemas cuja solução seja uma rota "trivial", ou seja, uma rota possível de ser achada somente utilizando informações encontradas nos lugares de origem e destino.
- Problemas nos quais os lugares de origem e destino estejam localizados em ruas paralelas.
- Problemas nos quais sub-rotas intermediárias entre os pontos de origem e destino sejam conhecidas.
- Problemas nos quais os pontos de origem e destino estejam em regiões diferentes mas com fronteira comum.
- Problemas nos quais os pontos de origem e destino estejam em regiões de hierarquia diferente.

Veremos agora detalhadamente cada um destes tipos de problema.

### 4.2.1 Rotas pré-conhecidas

Este tipo de situação surge quando o UniGuide, ao examinar o *slot* **Connect** do *frame* que representa o lugar de origem descobre que já existe uma rota conhecida ao lugar de destino. Assim, o UniGuide agora só tem que seguir

as instruções contidas na descrição da rota transformando-as em linguagem natural de uma forma direta, como está especificado na seção de aprendizagem do capítulo II e desta forma já temos a solução.

### 4.2.2 Rotas "triviais"

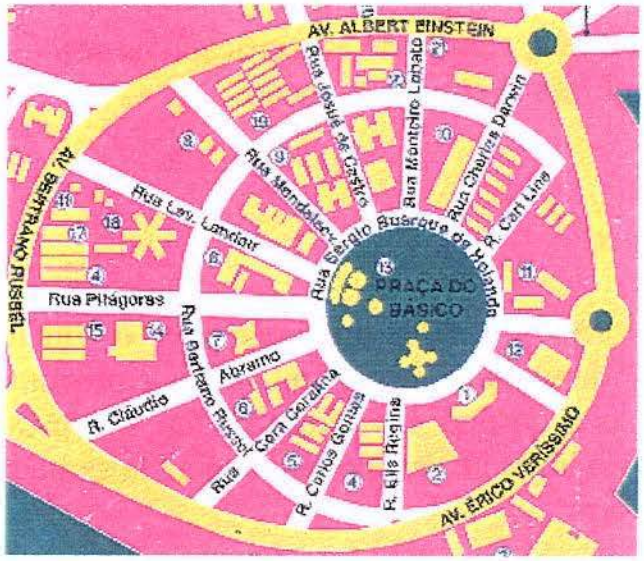
Se a rota desejada não for pré-conhecida, o UniGuide testará para ver se a rota é "trivial" da seguinte forma : o UniGuide procura no campo **View** do lugar de origem se o destino é visível. Se for, o UniGuide informa ao usuário, a localização do lugar de destino baseado na informação de **Heading** e **Distance** do campo **View**.

Se o lugar de destino não for visível, o UniGuide verifica, pelo campo **On**, se ambos estão em uma mesma rua. Se estiverem, O UniGuide sugere ao usuário seguir por esta rua comum até encontrar o objetivo. É interessante observar que o UniGuide, baseado no campo **View** do lugar de destino, dá uma descrição detalhada de suas redondezas.

### 4.2.3 Rotas entre ruas paralelas

Não conseguindo achar uma rota "trivial", o UniGuide verifica se as duas ruas, nas quais estão localizados a origem e o destino, através de uma pesquisa no campo **Parallel**, são paralelas, e então, utiliza um algoritmo baseado no seguinte raciocício heurístico: Se duas ruas são paralelas então, provavelmente, existe uma rua transversal a ambas ligando-as. Se acharmos esta rua, então teremos a rota entre a origem e o destino simplesmente achando a rota entre a origem e a rua de ligação e entre a rua de ligação e o destino, usando qualquer um dos métodos vistos neste capítulo, inclusive este próprio, recursivamente. O UniGuide acha, se existir, esta rua de ligação através de uma análise da geometria local dos cruzamentos da região.

Como um exemplo, temos o seguinte problema: Achar uma rota entre o IC e o IMECC (ver mapa abaixo).




Apesar da Av. Albert Einstein (onde está situado o IC) e da Rua Sérgio Buarque de Holanda (uma das ruas onde o IMECC está situado) não serem paralelas no sentido geométrico do termo, elas podem ser, para a nossa aplicação, consideradas "paralelas"[1] pois possuem ruas transversais ligando-as e não se cruzam. Assim o algoritmo descobriria que a origem e destino estariam em ruas "paralelas" e através da análise da geometria dos cruzamentos descobriria que a Av. Albert Einstein e a Rua Sérgio Buarque de Holanda estão ligadas pela Rua Pitágoras[2] e agora fica muito fácil descobrir a rota completa entre os dois pontos.

### 4.2.4 Rota entre lugares com sub-rotas intermediárias conhecidas

Se os lugares de origem e destino não estiverem em ruas paralelas, o UniGuide tenta, então, achar sub-rotas intermediárias de tal forma que ligando-as consiga chegar à rota desejada. Isto é realizado pesquisando os campos **Connect** da origem e do destino e verificando se os lugares que são referenciados nestas rotas estão, por sua vez, ligados ao destino.

---

[1]Isto pode, entretanto, causar pequenas distorções no mapa cognitivo [Kuipers77].
[2]Na realidade esta é apenas uma das ruas que as ligam (ver apêndice III).

### 4.2.5 Rota entre regiões com fronteira comum

Este algoritmo funciona quando o lugar de origem e o lugar de destino encontram-se em regiões que possuem uma rua como fronteira comum. A heurística na qual ele se baseia é a seguinte: para achar uma rota entre dois lugares A e B, ache uma rua R tal que A e B estejam em lados opostos de R. Então ache um caminho de A até R e de R até B.

Este método é essencialmente igual ao método para achar rotas entre ruas paralelas e pode também ser recursivo para lidar com situações que necessitem de objetivos intermediários para achar a solução final.

Mas como achar a rua R? Para cada um dos lugares A e B considera-se as regiões (obtidas nos seus *slots* **In**) cujos *slots* **Boundary** não sejam vazios. Estas duas listas são comparadas para achar pares de regiões, uma de cada lista, que tenham como fronteira, as mesmas ruas.

A forma mais fácil de se fazer isto é examinar todos os pares possíveis de regiões para ver se elas têm uma fronteira comum. Este método examina, no máximo, o produto dos comprimentos das duas listas, o que em mapas maiores pode torná-lo inviável. Por isso, na implementação, foi utilizado um algoritmo um pouco melhor [Kuipers77]:

1- Põe-se uma marca temporária na(s) rua(s) fronteiriça(s) de cada região da lista de A.
2- Então se examina cada região da lista de B e verifica-se se a rua fronteiriça já está marcada.
3- Se estiver, esta é uma rua de fronteira comum.
4- Quando terminam as buscas, apagam-se as marcas temporárias.

Com isto, examina-se a *soma* das duas listas e não o produto.

### 4.2.6 Rotas entre regiões de hierarquia diferente

Como dissemos no Capítulo II, em mapas maiores é necessário ter uma hierarquia de regiões baseada em uma escala ascendente de seus tamanhos. Assim, um problema de achar uma rota entre regiões A e B com hierarquia diferente seria resolvido da seguinte maneira:

1- "Nivelamento" das regiões - Acha-se, entre as duas regiões originais, a de maior hierarquia (região R). Então se acha uma região de hierarquia equivalente à R (região R1) que contenha a outra região original.

2- Acha-se uma rota entre R e R1.

3- Acha-se uma rota entre R1 e a região original de menor hierarquia.

## 4.3 Algoritmos gerais de busca

Quando nenhuma das situações anteriores se aplica, o UniGuide utiliza algoritmos gerais de busca[3]( [Winston92], [Charniak87]). Entretanto, vale ressaltar que o espaço de busca é sempre restrito, pois só são incluídos os objetos geográficos pertencentes à região da menor hierarquia possível, que contenha os lugares de origem e destino.

A seguir, veremos em detalhes todos estes algoritmos.

### 4.3.1 Algoritmos não-heurísticos

Os algoritmos não-heurísticos são aqueles que realizam uma busca cega na árvore de busca. Os implementados foram: Busca em profundidade, busca em largura e busca não-determinística.

A estratégia seguida pelo algoritmo de ***busca em profundidade***, como o próprio nome diz, é pesquisar a árvore de busca para o "fundo" sempre que possível. Optou-se pela implementação da sua forma não-recursiva por ser muito mais rápida e pelo fato de, em mapas maiores, existir a possibilidade da forma recursiva causar estouro de pilha ( *stack overflow*) .

O algoritmo da busca em profundidade, bem como de todos os outros, em pseudo-código, está no Apêndice IV.

A ***busca em largura*** é bastante semelhante à busca em profundidade, com a exceção que ao invés de tentar sempre ir mais "fundo" na árvore de busca, o algoritmo realiza a busca em todos os nós do mesmo nível antes de passar para o próximo nível (ou seja, antes de se "aprofundar" na árvore). Assim como a busca em profundidade e pelas mesmas razões, implementou-se a forma não-recursiva da busca em largura.

A ***busca não-determinística*** tenta tirar proveito de ambos os métodos anteriormente descritos. O algoritmo usa um número aleatório para determinar, *a cada passo*, qual dos dois métodos, busca em profundidade ou busca em largura, vai utilizar para prosseguir com a busca.

---

[3]O usuário tem a possibilidade de escolher qual algoritmo o programa usará. (ver Capítulo V).

O problema deste algoritmo é gerar um número *realmente* aleatório, pois apesar da linguagem C++ ter uma função padrão para isto, na realidade ela gera apenas números pseudo-aleatórios, devido à própria natureza determinística de um computador operando normalmente. Para resolver este problema adotou-se um algoritmo baseado em [Plumb94].

O algoritmo é, essencialmente o mesmo, que gera números aleatórios para o programa de segurança de *e-mails* chamado de PGP (Pretty Good Privacy) e baseia-se, principalmente, em amostragens do temporizador do PC, que tem a capacidade de marcar tempos de até 1 microsegundo.

### 4.3.2 Algoritmos heurísticos

Os algoritmos heurísticos são aqueles que usam informações adicionais sobre os nós (as heurísticas) da árvore de busca para diminuir seu tamanho (ou, como preferem alguns, "podá-la"). Os implementados foram: *hill climbing*, *best first* e *beam search*.[Winston92]

O algoritmo ***hill climbing*** utiliza uma busca em profundidade orientada, ou seja, quando os filhos de um dado nó são gerados, eles sofrem uma reordenação em função da distância euclideana à origem, de tal forma que o nó-filho escolhido para continuar a busca, no caso de nossa implementação, é aquele mais próximo ao lugar de destino. Observe que isto corresponde a uma reordenação localizada, pois os únicos nós reordenados são os filhos do nó sendo pesquisado *atualmente*.

O algoritmo *hill climbing*, em pseudo-código, bem como de todos os outros está no apêndice VI.

O algoritmo ***best first*** é bastante semelhante ao *hill climbing*, com a diferença que ao invés de reordenar apenas a lista dos filhos de um dado nó como o *hill climbing*, o *best first* reordena *todos* os nós abertos (ou seja, reordena a lista Open). Com isto, o *best first* faz uma reordenação global e por isto é mais eficaz, em geral, que o *hill climbing*.

O algoritmo ***beam search*** é um melhoramento do *best first*. A melhoria é que ao se gerar a lista dos filhos de determinado nó, verifica-se se estes já foram examinados anteriormente. Se algum nó já foi examinado, ele é eliminado da lista.

O algoritmo *beam search*, por seu melhor desempenho, foi escolhido como o algoritmo padrão para a nossa implementação.

### 4.3.3 Quadro comparativo de desempenho

Para termos uma melhor idéia de como estes algoritmos se comportam na prática, fizemos as seguintes simulações (desabilitando todas as outras formas de achar rotas)[4] :

Quadro 1 - Rota entre o IC e o Hospital da Mulher

| *Método* | *Total de nós da árvore* | *Nós pesquisados* | *Tempo(Relativo ao Beam search )* |
|---|---|---|---|
| ***Depth First*** | 224 | 24 | 2.98 |
| ***Breadth First*** | 224 | 47 | 6 |
| ***Não Determ.*** | 224 | 47 | 7 |
| ***Hill Climbing*** | 224 | 16 | 1.98 |
| ***Best First*** | 224 | 9 | 1 |
| ***Beam Search*** | 224 | 9 | 1 |

Quadro 2- Rota entre o Ginásio Multidisciplinar e a Cirurgia Experimental

| *Método* | *Total de nós da árvore* | *Nós pesquisados* | *Tempo (Relativo ao Beam search )* |
|---|---|---|---|
| ***Depth First*** | 215 | 24 | 2.98 |
| ***Breadth First*** | 215 | 46 | 5.89 |
| ***Não Determ.*** | 215 | 22 | 1.98 |
| ***Hill Climbing*** | 215 | 8 | 1.31 |
| ***Best First*** | 215 | 8 | 1 |
| ***Beam Search*** | 215 | 8 | 1 |

[4]Estas estatísticas estão disponíveis no modo desenvolvimento da implementação. O programa as computa toda vez que o usuário pede uma rota. ( Ver capítulo V)

Quadro 3- Rota entre o IMECC e Faculdade de Engenharia Elétrica

| *Método* | *Total de nós da árvore* | *Nós pesquisados* | *Tempo(relativo ao Beam search )* |
|---|---|---|---|
| ***Depth First*** | 225 | 20 | 3.05 |
| ***Breadth First*** | 225 | 38 | 5.07 |
| ***Não Determ.*** | 225 | 17 | 2.02 |
| ***Hill Climbing*** | 225 | 5 | 1.12 |
| ***Best First*** | 225 | 5 | 1 |
| ***Beam Search*** | 225 | 5 | 1 |

Observe que o *Best first* e o *beam search* se comportaram consistentemente muito melhor que os métodos não-heurísticos (como era de se esperar) e um pouco melhor que o *hill climbing*, sendo que em mapas maiores esta diferença de desempenho tende a aumentar significativamente.

Quanto aos métodos não-heurísticos, o pior, na média, foi a busca em largura. O não-determinístico, por sua natureza aleatória, às vezes se comporta muito bem, sendo melhor inclusive que a busca em profundidade, e às vezes muito mal, sendo até o pior de todos. Já a busca em profundidade teve um desempenho mais consistente e seria o método não-heurístico de escolha *para este tipo de mapa*, pois a performance destes métodos é muito influenciada pelas características da árvore de busca.

### 4.3.4 Métodos de reordenação

Como vimos, todos os algoritmos heurísticos, na nossa implementação, possuem uma reordenação como ponto principal. Dentre as várias técnicas existentes implementou-se três: O *quicksort*, o *bubble sort* e o *shell sort*. Por ser, na prática, o mais rápido, o *quicksort* foi o escolhido como padrão. Para maiores detalhes sobre estes métodos ver [Sedgewick88] ou [Cormen91].

## 4.4 Considerações finais

Neste capítulo vimos os métodos utilizados pelo UniGuide para realizar uma busca. Os métodos implementados são bastante eficazes na resolução de problemas de achar rota, em mapas de tamanho médio. No

entanto, em mapas maiores seria necessário encontrar novas e mais poderosas heurísticas para melhoria de performance.

Um campo interessante de pesquisa, é utilizar um modelo de rede neural para o mapa e realizar buscas sobre ele. Vários modelos para a rede estão sendo testados, principalmente os mapas auto-organizados de Kohonen e o modelo de Hopfield ([Rao93], [Kosko92]). Os resultados preliminares são bastante promissores, principalmente em problemas de otimização não-linear, como por exemplo, o problema do caixeiro-viajante, que, como é sabido, não possui um algoritmo geral que o resolva satisfatoriamente.

Outro campo de pesquisa é a utilização de algoritmos genéticos para a realização de buscas. Para uma visão geral sobre algoritmos genéticos ver [Winston92].

No próximo capítulo, veremos a implementação do UniGuide.

# Capítulo V

## Descrição dos recursos do UniGuide e sua implementação

*Este capítulo descreve o sistema UniGuide através de sua interface, apresentando o objetivo, as características e a implementação de cada uma das suas ferramentas.*

### 5.1 Introdução

O UniGuide é um protótipo de um sistema que objetiva permitir que um usuário obtenha facilmente informações sobre determinados lugares e rotas, inclusive com informações adicionais que o guiam ao longo do caminho.

Suas principais características e ferramentas são:

- *Interface gráfica* - Permite uma grande interação com o usuário. Praticamente todas as operações podem ser realizadas de duas ou mais formas diferentes, seja através de menus, de botões da barra de ferramentas ou de cliques do *mouse*. Para uma discussão aprofundada sobre como projetar uma interface para uso de mapas ver [Vertelney92] e [Booker92].
- *Barra de ferramentas* - Seu uso facilita muito o trabalho do usuário, possibilitando o acesso aos recursos com um simples clique do *mouse*, bem como mostra visualmente se os recursos estão ou não disponíveis no momento.
- *Barra de status* - Apresenta uma mensagem contendo uma ajuda rápida sobre a função desempenhada pelo elemento sobre o qual o ponteiro do *mouse* se encontra.
- *Sistema de ajuda em hipertexto* - O UniGuide possui um sistema integrado de ajuda on-line em hipertexto que possibilita ao usuário, a qualquer momento, consultá-lo para dirimir eventuais dúvidas sobre a operação do sistema. O fato da ajuda estar organizada como hipertexto facilita enormemente sua consulta. [Conklin87]
- *Editor / Compilador integrado* - O UniGuide possui um editor/compilador integrado que permite ao usuário fazer modificações na descrição do mapa ( em GeoDL), recompilá-lo e em seguida fazer consultas

sobre esta nova base de dados. O editor, além de todas as características básicas, como, por exemplo, procura e substituição de palavras, cópia, colagem e transferência para o *clipboard* (compatíveis com o padrão CUA[1]), possui também modelos prontos ( *templates*) de descrições em GeoDL para os principais objetos geográficos.

- *Apresentação do mapa na tela* - O UniGuide apresenta o mapa descrito na tela, possibilitando, assim, que o usuário tenha uma melhor compreensão dos seus arredores.
- *Procura de lugares no mapa* - Possibilita ao usuário entrar com informações sobre um lugar que ele deseje saber a localização e o UniGuide automaticamente o acha e ajusta o mapa para que esta localização torne-se visível. Esta ferramenta é particularmente útil em mapas grandes.
- *Apresentação da rota graficamente* - Possibilita ao usuário uma perfeita visão de toda a rota desejada, inclusive com informações extras ao longo do caminho, principalmente de lugares importantes. Se assim o desejar, o usuário pode imprimir esta rota.
- *Apresentação da rota em linguagem natural* - Se o usuário assim o preferir a rota é apresentada em linguagem natural, possibilitando uma melhor compreensão.
- *Resposta a consultas em linguagem natural* - O UniGuide responde a consultas em linguagem natural, facilitando a operação do sistema pelo usuário.
- *Aprendizagem supervisionada e não-supervisionada* - O UniGuide possui a capacidade de aprender tanto de forma supervisionada como de forma não-supervisionada (automaticamente), podendo, assim, aumentar seu conhecimento incrementalmente e melhorar sua performance com o uso.
- *Versatilidade de uso* - Como sua base de conhecimento é descrita em GeoDL, o UniGuide pode manusear qualquer mapa cuja descrição, em GeoDL, esteja disponível.
- *Dois modos de operação (usuário e desenvolvimento)* - Esta característica foi acrescentada principalmente por motivos de segurança. No modo usuário as opções de aprendizagem supervisionada, estatísticas, configuração e o editor/compilador ficam inacessíveis, para evitar que um usuário comum, inadvertidamente, modifique erroneamente a descrição do mapa. O modo desenvolvimento (protegido por senha) dá acesso a todas as ferramentas do UniGuide.

---

[1]Common User Access

• *Sistemas de informações em hipertexto sobre lugares* - Esta característica possibilita ao usuário obter as mais variadas informações sobre determinado lugar com um simples clique do mouse sobre o ponto correspondente ao lugar no mapa.
• *Escolha do método de busca* - O usuário desenvolvedor pode escolher, através de uma caixa de diálogo, qual método de busca o sistema utilizará.
• *Fornecimento de estatísticas* - O UniGuide fornece várias estatísticas sobre o mapa descrito e sobre seu estado de conhecimento atual, além de informações sobre a performance dos métodos de busca. Só disponível no modo desenvolvimento.
• *Totalmente configurável* - O UniGuide, através de caixas de diálogos apropriadas, permite ao usuário configurá-lo totalmente, desde os arquivos do mapa, dos dados e da ajuda até as cores e largura da caneta utilizada na apresentação gráfica da rota.

## 5.2 Descrição da ferramenta

O sistema possui, basicamente, quatro janelas: a janela principal, a janela do editor/compilador, a janela de apresentação da rota e a janela de localização de lugares, sendo que estas três últimas são acessadas somente a partir da principal. Além destas mais importantes, o sistema conta com várias caixas de diálogo e janelas de apresentação de resultado.

### 5.2.1 A Janela Principal

Como foi visto na Introdução, o UniGuide possui dois modos de operação :

- Para usuários finais (Modo consulta)
- Para desenvolvedores (Modo desenvolvimento)

A escolha do modo de operação é feito no início da execução do programa através de uma caixa de diálogo, como mostra a figura 5.1.

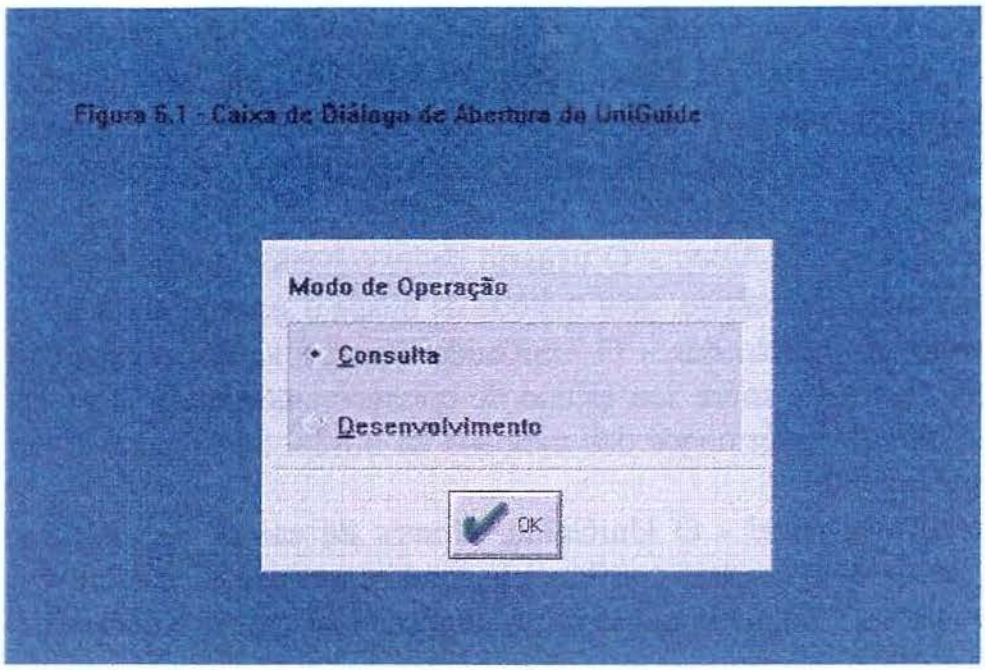



Figura 5.1 - Caixa de Diálogo de Abertura do UniGuide

No modo consulta, como o próprio nome está dizendo, o usuário só poderá ter acesso às funções de consulta do UniGuide, que são as seguintes:

- Obter informações sobre o sistema.
- Achar uma rota.
- Procurar por um determinado lugar.
- Obter informações sobre um determinado lugar.
- Obter ajuda on-line sobre o sistema.

Na figura 5.2 temos a janela principal do UniGuide no modo consulta.

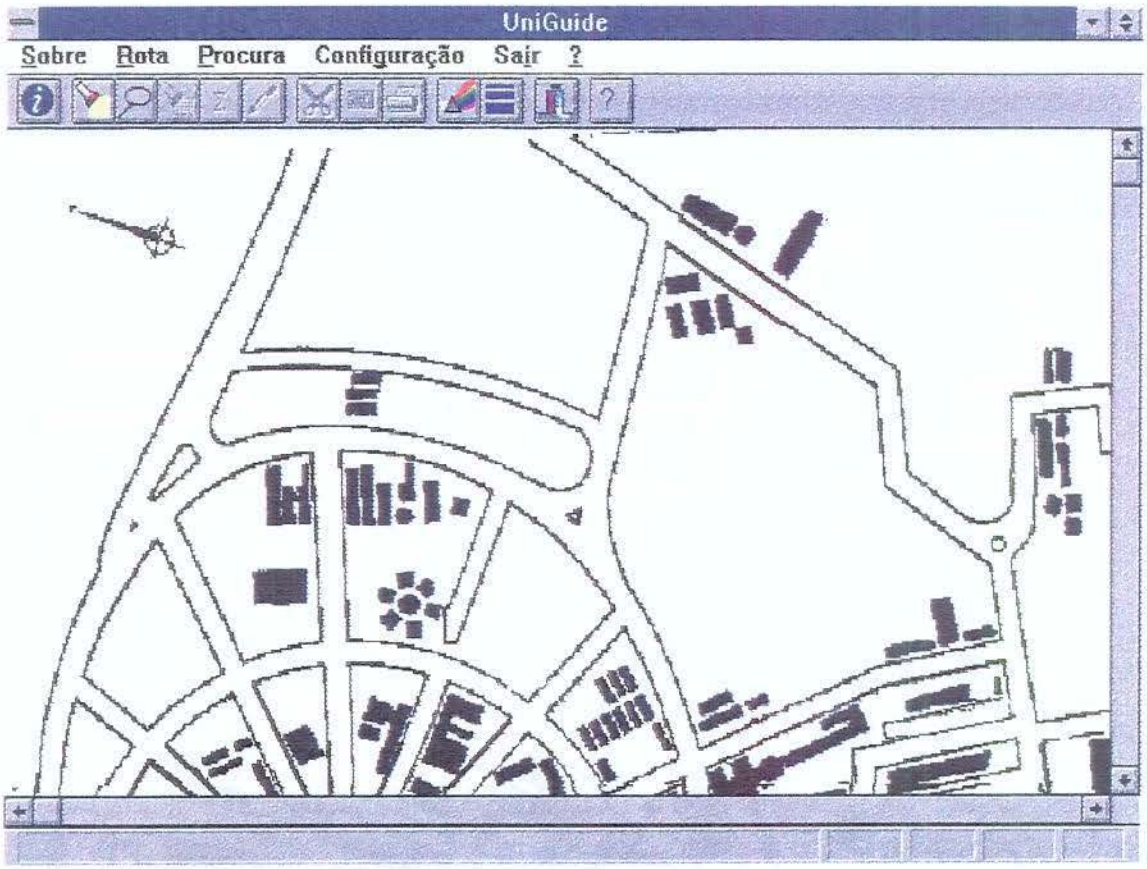



Figura 5.2 Janela Principal do UniGuide no modo Consulta

No modo Desenvolvimento, tem-se acesso a todas as ferramentas do UniGuide e a janela principal, neste modo, pode ser vista na figura 5.3.

Os elementos principais da janela principal[2] são: (vistos na figura 5.3)

- O menu
- A barra de ferramentas
- A sua área cliente, onde o mapa é mostrado
- A barra de status

[2]A partir de agora, o termo janela principal refere-se à janela principal no modo de desenvolvimento.

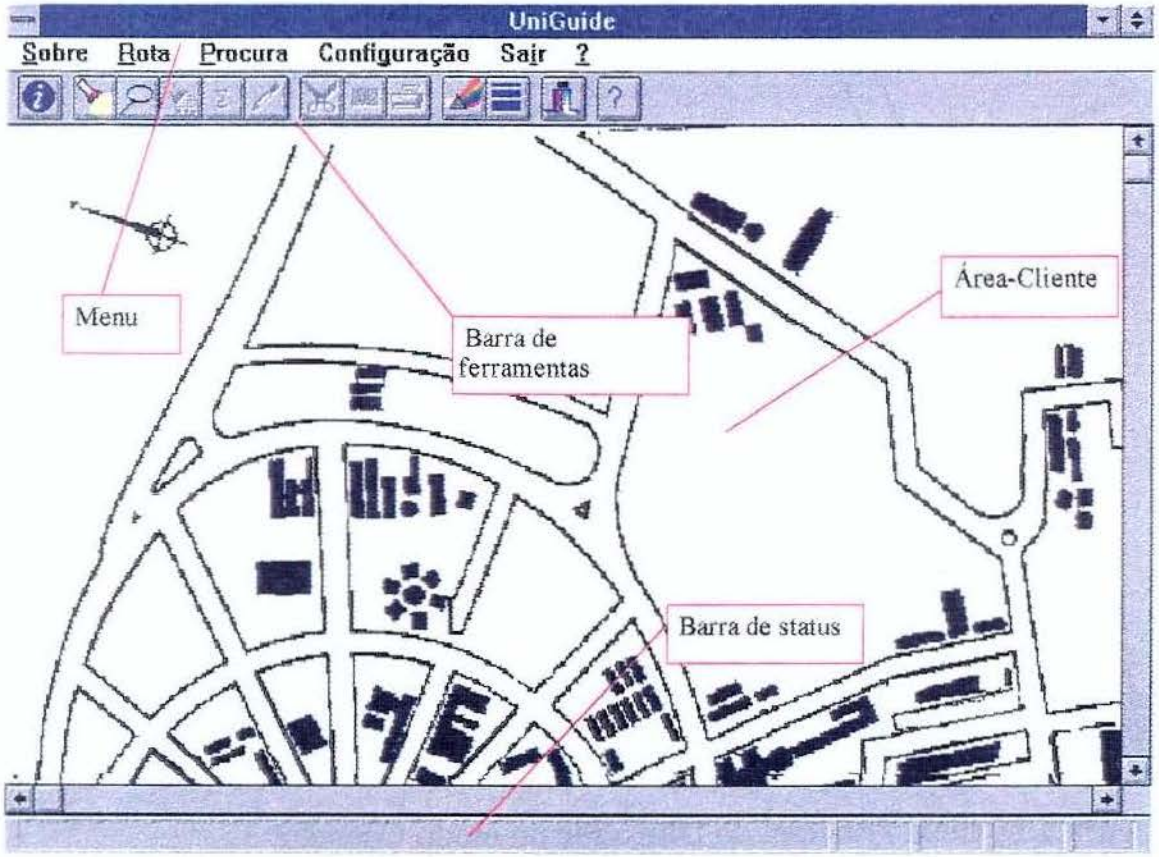



Figura 5.3 - Elementos principais da janela principal

### 5.3.1.1 O menu

O menu principal possui as seguintes opções[3] :

• *Sobre* - Mostra uma caixa de diálogo com informações sobre o programa.
• *Rota* - Mostra uma caixa de diálogo (figura 5.4) que permite ao usuário inserir os dados referentes ao lugar de origem e destino. A caixa de diálogo apresenta três botões:

[3]Cada item do menu pode ser acessado pressionando-se a tecla ALT e a letra sublinhada de seu nome.

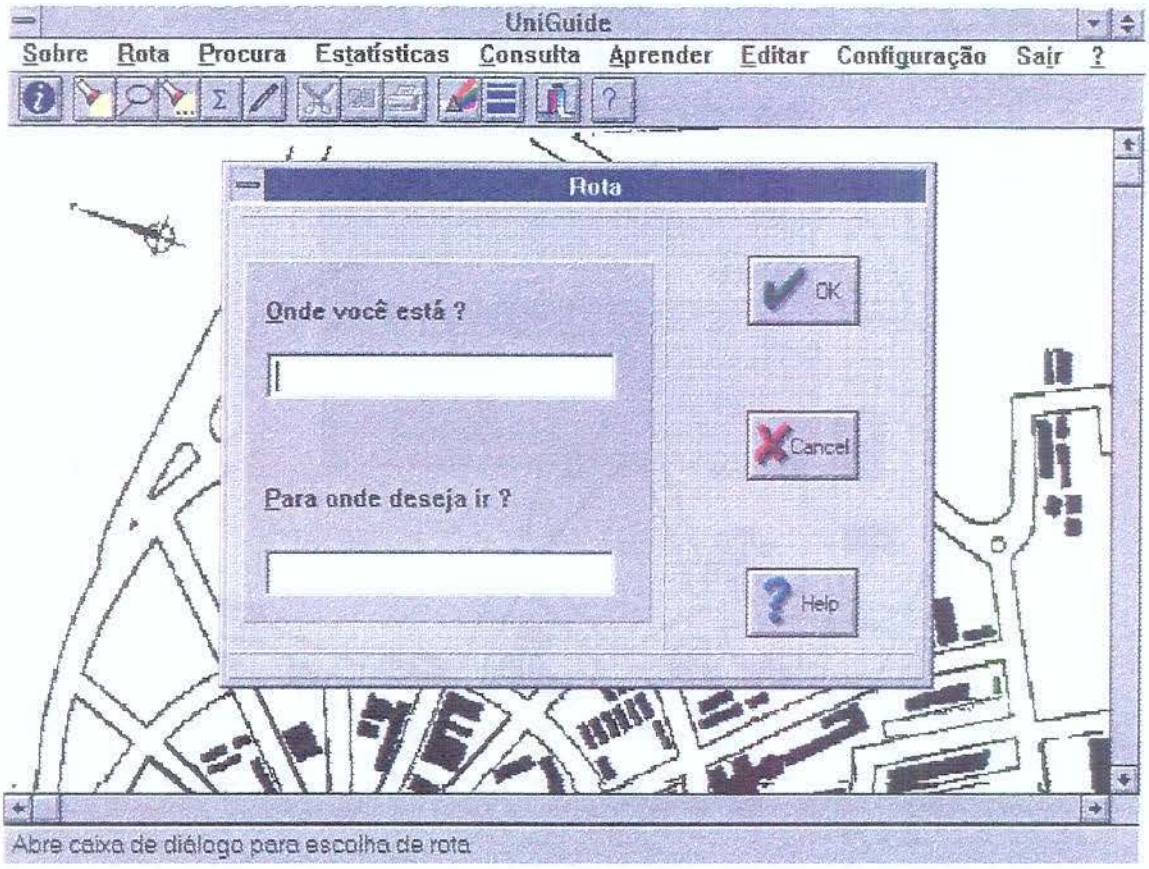



Figura 5.4 - Caixa de Diálogo "Rota"

1) *OK* - Fecha a caixa de diálogo e tenta achar a rota com os dados fornecidos pelo usuário.
2) *Cancela* - Fecha a caixa de diálogo.
3) *Help* - Mostra uma janela contendo uma ajuda de como usar a caixa de diálogo.

• *Procura* - Mostra uma caixa de diálogo (figura 5.5) em que o usuário deve entrar com um nome de um lugar que ele deseje saber sua localização. O UniGuide automaticamente o acha e rearranja o mapa de forma que o ponto onde está localizado torne-se visível e então o mostra através de uma seta. (figura 5.6)

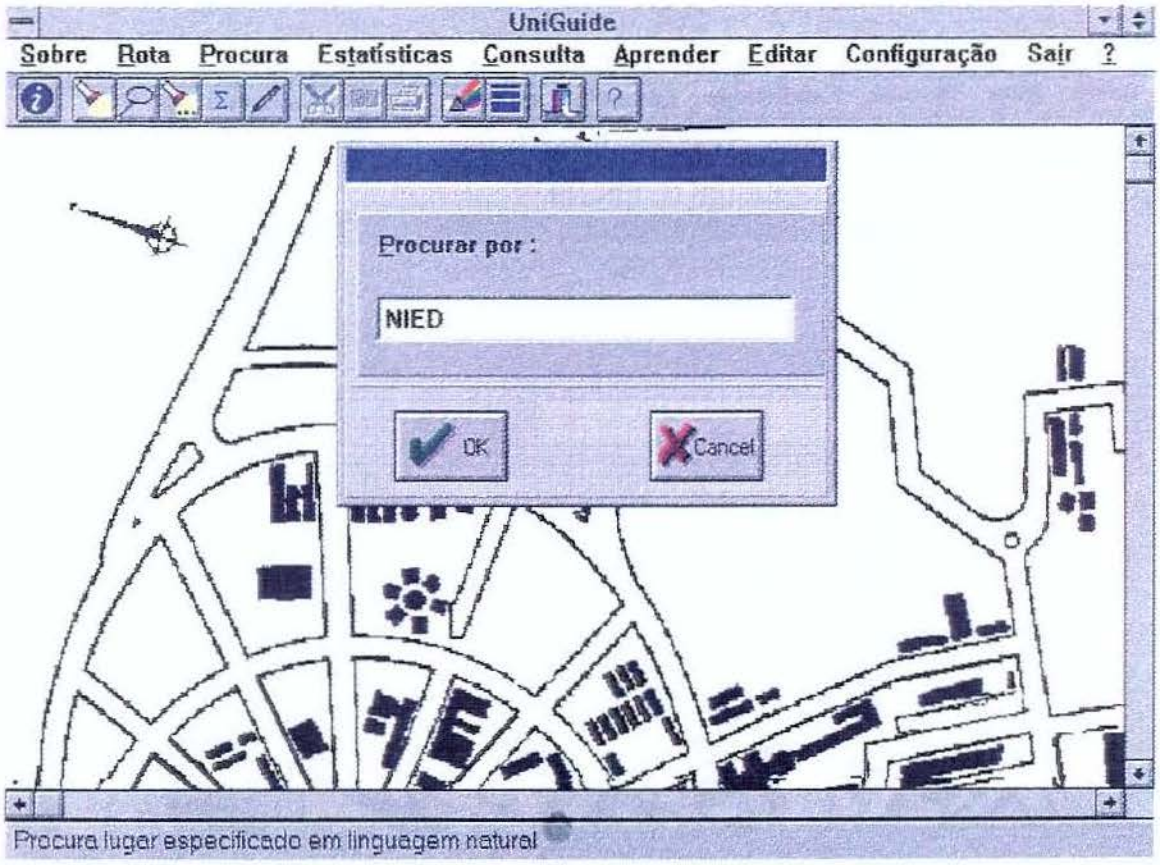



Figura 5.5 - Caixa de Diálogo "Procura"

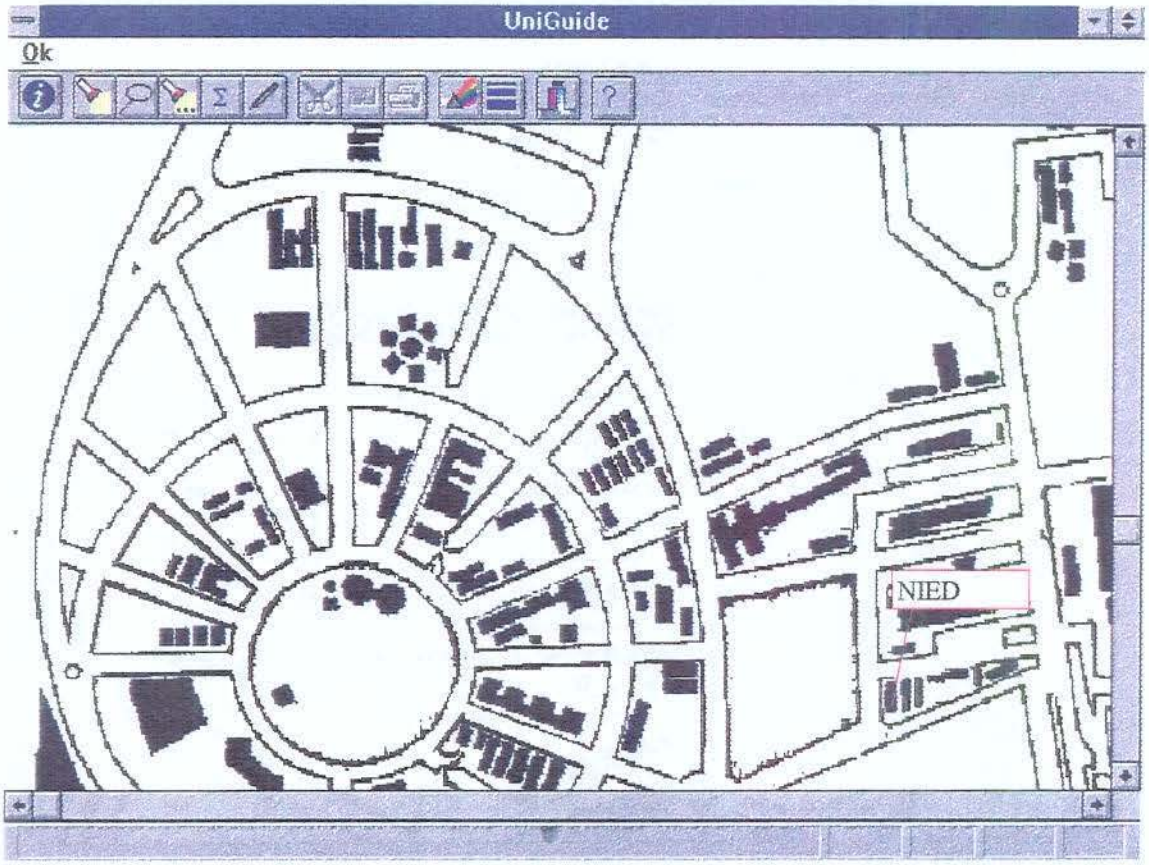



Figura 5.6 - Janela indicando a localização do local especificado

• *Estatísticas* - Esta opção mostra uma janela com as estatísticas sobre o conhecimento atual do programa, conforme mostra a figura 5.7. A janela mostra quantos lugares, ruas, cruzamentos e rotas são atualmente conhecidos pelo UniGuide.


Figura 5.7 - Estatísticas

• *Consulta* - Mostra uma caixa de diálogo em que o usuário entra com uma frase em linguagem natural (figura 5.8) realizando uma consulta à base de dados, como por exemplo "Onde se encontra o IC?". Feito isto, o UniGuide apresenta uma janela mostrando a descrição das redondezas do lugar pedido.

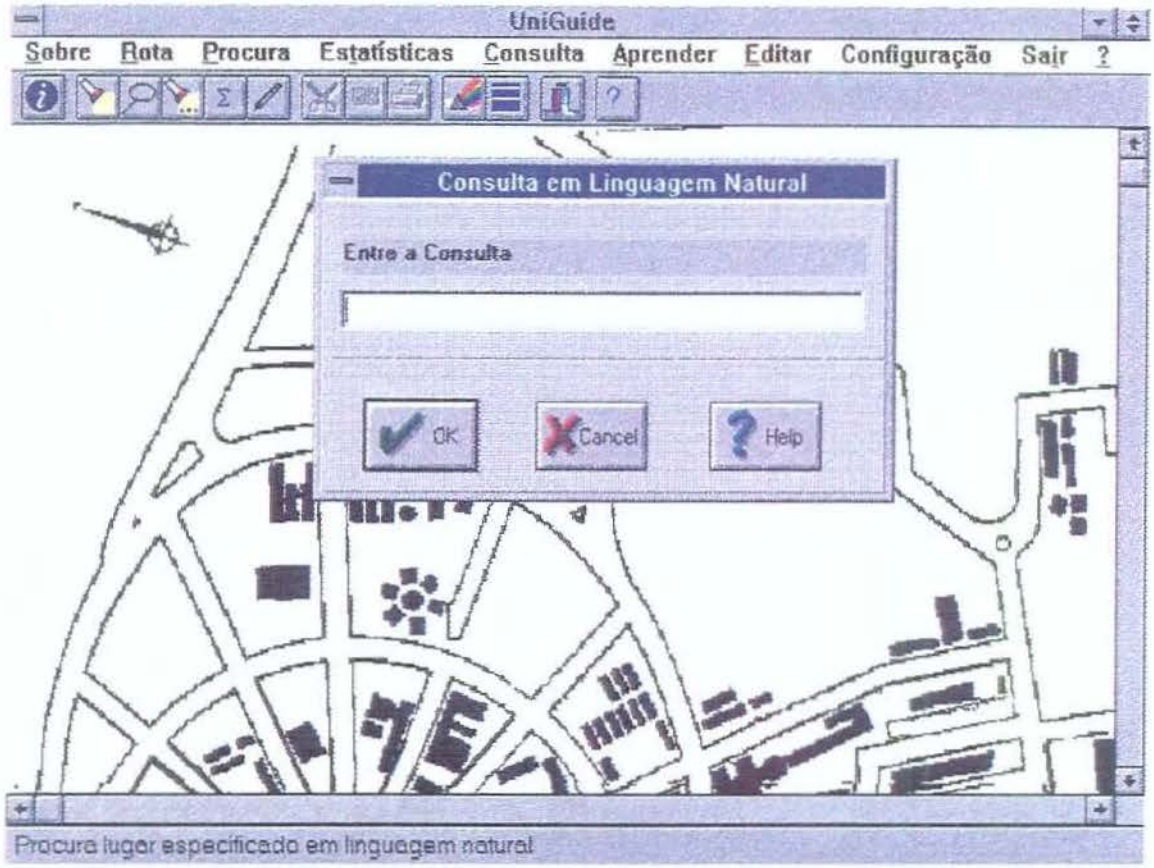



Figura 5.8 - Caixa de Diálogo "Consulta"

• *Editar* - Cria a janela do editor/compilador (figura 5.9) e carrega para edição o arquivo GeoDL especificado no arquivo de inicialização uniguide.ini.

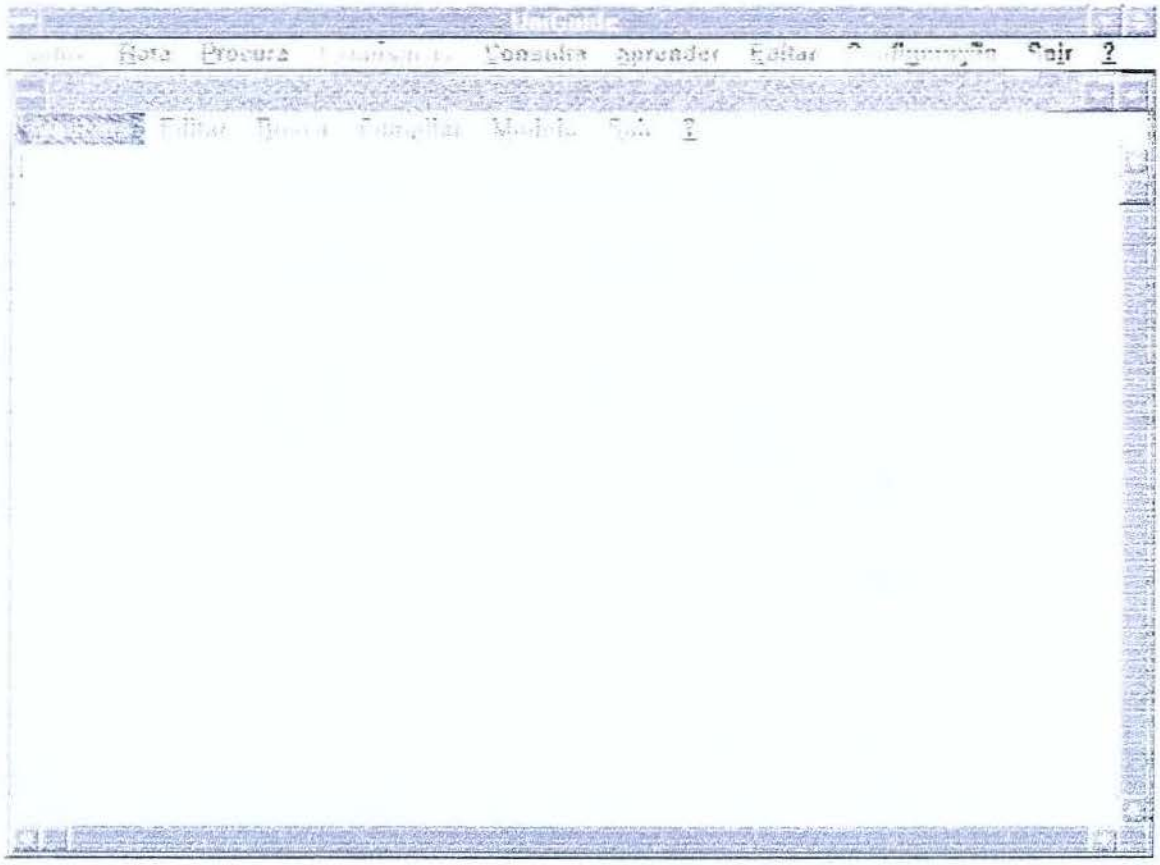


Figura 5.9 - Janela do Editor/Compilador

• *Configura* - Este item tem seis subitens : *Opções* ... ( o símbolo ..., no padrão de interface Windows, significa que este item, ao ser escolhido, dá origem a uma caixa de diálogo), *Arquivo de Ajuda* ..., *Arquivo de Dados* ..., *Arquivo do mapa* ..., *Cor*... e *Espessura* ... .

1) *Opções* ... - Mostra uma caixa de diálogo que permite ao usuário escolher qual método de busca e de ordenação o UniGuide deve utilizar e se o sistema deve ou não salvar as configurações correntes ao terminar. A operação da caixa de diálogo é extremamente simples, bastando clicar-se o mouse para seleção do método preferido.
2) *Arquivo da Ajuda..., Arquivo de Dados..., Arquivo do Mapa...* - Permitem a escolha respectivamente dos arquivos da ajuda, de dados e do mapa. Sendo que cada um deles dá origem a uma caixa de diálogo, que permite a escolha e/ou mudança dos arquivos de dados, do mapa e da ajuda.

3) *Cor...* - Dá origem a uma caixa de diálogo em que se pode escolher, visualmente, a cor desejada para o traçado da rota. ( o padrão é o vermelho)
4) *Espessura...*- Dá origem a uma caixa de diálogo onde se pode escolher a espessura da caneta usada para o traçado da rota. ( o padrão é 3)

- *Sair* - Termina o programa.

- *?* - Este item possui 2 subitens : *Índice* e *Usando a Ajuda*. O subitem *Índice* mostra uma janela com os itens disponíveis para ajuda (figura 5.10) e o subitem *Usando a Ajuda* mostra um tutorial completo sobre como se deve usar a ajuda. (figura 5.11)

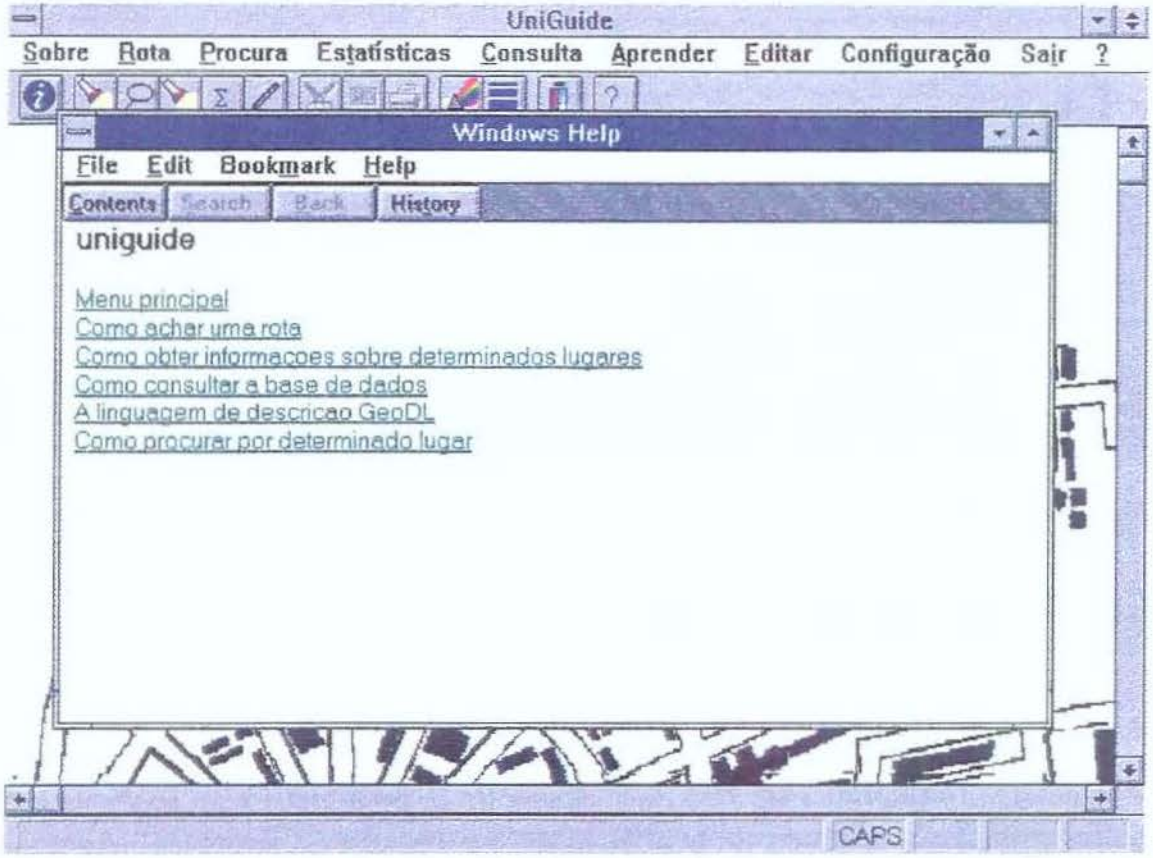



Figura 5.10 - Índice da ajuda

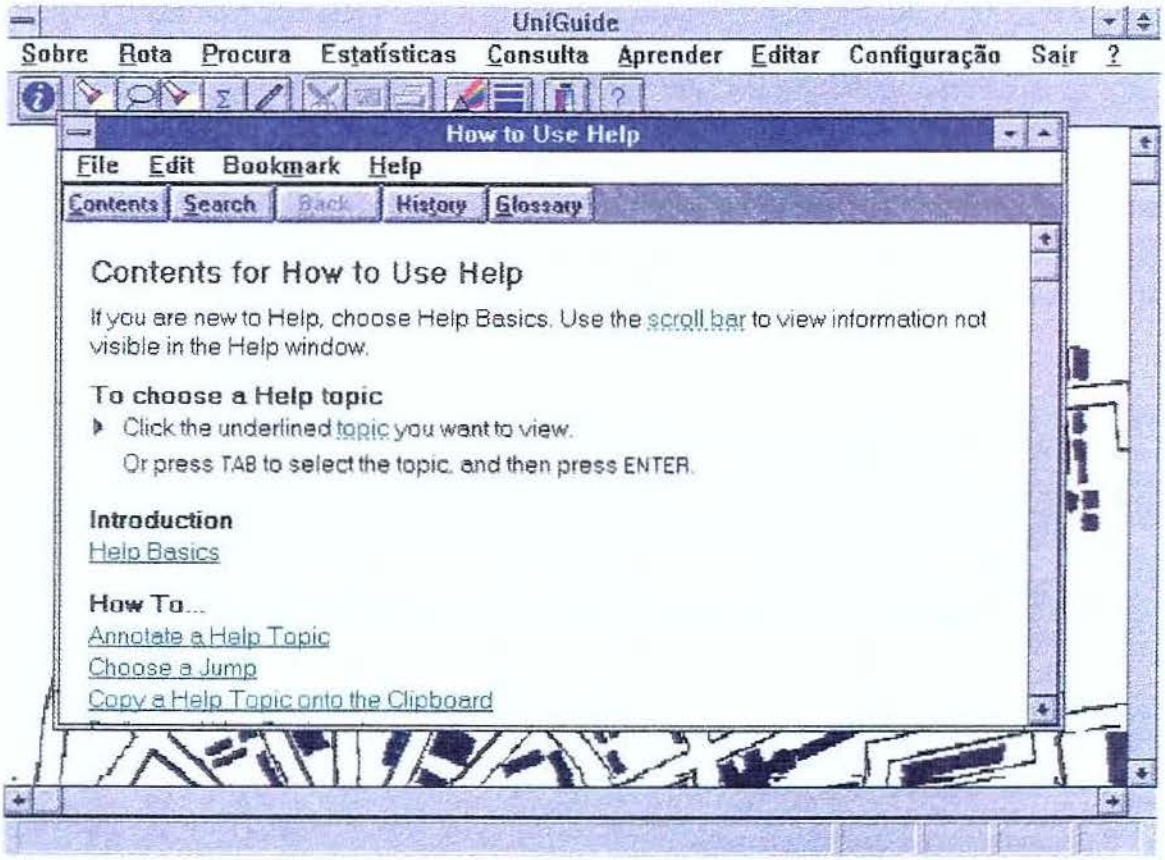



Figura 5.11 - Tutorial sobre como utilizar a ajuda

### 5.3.1.2 A Barra de Ferramentas

As barras de ferramentas são, atualmente, presença quase obrigatória em qualquer tipo de interface gráfica, pois melhoram bastante os quatro principais critérios de avaliação de um projeto de interface, que são [Vertelney92]: A usabilidade, a funcionalidade, a comunicação visual e a estética.

A barra de ferramentas do UniGuide é constituída por 13 botões com ícones, situados logo abaixo do menu e agrupados de acordo com as funções que desempenham. Cada botão corresponde a um comando do

menu, mas com a vantagem de maior facilidade, e conseqüente rapidez, de acesso.

Além disto, a barra de ferramentas indica, visualmente, se determinada opção está disponível no momento, pois os botões que as representam tornam-se esmaecidos se elas não estiverem disponíveis.

A seguir a descrição de cada botão da barra[4] :

- - Mostra uma caixa de diálogo com informações sobre o programa. Corresponde ao item *Sobre* do menu principal.

- - Mostra uma caixa de diálogo em que o usuário deve entrar com um nome de um lugar que ele deseje saber sua localização. Corresponde ao item *Procura* do menu principal.

- - Mostra uma caixa de diálogo que permite ao usuário inserir os dados referentes ao lugar de origem e destino. Corresponde ao item *Rota* do menu principal.

- - Mostra uma caixa de diálogo em que o usuário entra com uma frase em linguagem natural realizando uma consulta à base de dados. Corresponde ao item *Consulta* do menu principal.

- - Abre a janela de aprendizado. Corresponde ao item *Aprender*.

- - Cria a janela do editor/compilador e carrega para edição o arquivo GeoDL especificado no arquivo de inicialização uniguide.ini. Corresponde ao item *Editar* do menu principal.

- - Volta ao modo normal. Só está ativo quando uma rota estiver sendo apresentada na janela principal.

- - Apresenta a rota textualmente.

- - Imprime a rota.

---

[4]Uma pequena descrição aparece na barra de status ao se colocar o cursor do mouse sobre o botão.

• - Dá origem a uma caixa de diálogo em que se pode escolher, visualmente, a cor desejada para o traçado da rota. Corresponde ao item *Cor...* do menu principal.

• - Dá origem a uma caixa de diálogo onde se pode escolher a espessura da caneta usada para o traçado da rota. Corresponde ao item *Espessura...* do menu principal.

• - Termina o programa. Corresponde ao item *Sair* do menu principal.

• - Ativa o sistema de ajuda em hipertexto. Corresponde ao item *?* do menu principal.

### 5.3.1.3 A Área Cliente

A área cliente da janela principal tem, no caso do UniGuide, as funções de mostrar o mapa, de mostrar o local corrente quando o cursor do mouse passa por um objeto conhecido, de mostrar a rota graficamente, bem como a de realizar algumas tarefas relacionadas ao mapa de forma alternativa.

As principais tarefas que podem ser realizadas na janela principal são:

• *Nomes de lugares* - Ao se passar o cursor do mouse sobre um lugar conhecido, o UniGuide mostra, na tela, o seu nome (figura 5.12). Isto facilita enormemente o trabalho e a localização do usuário.

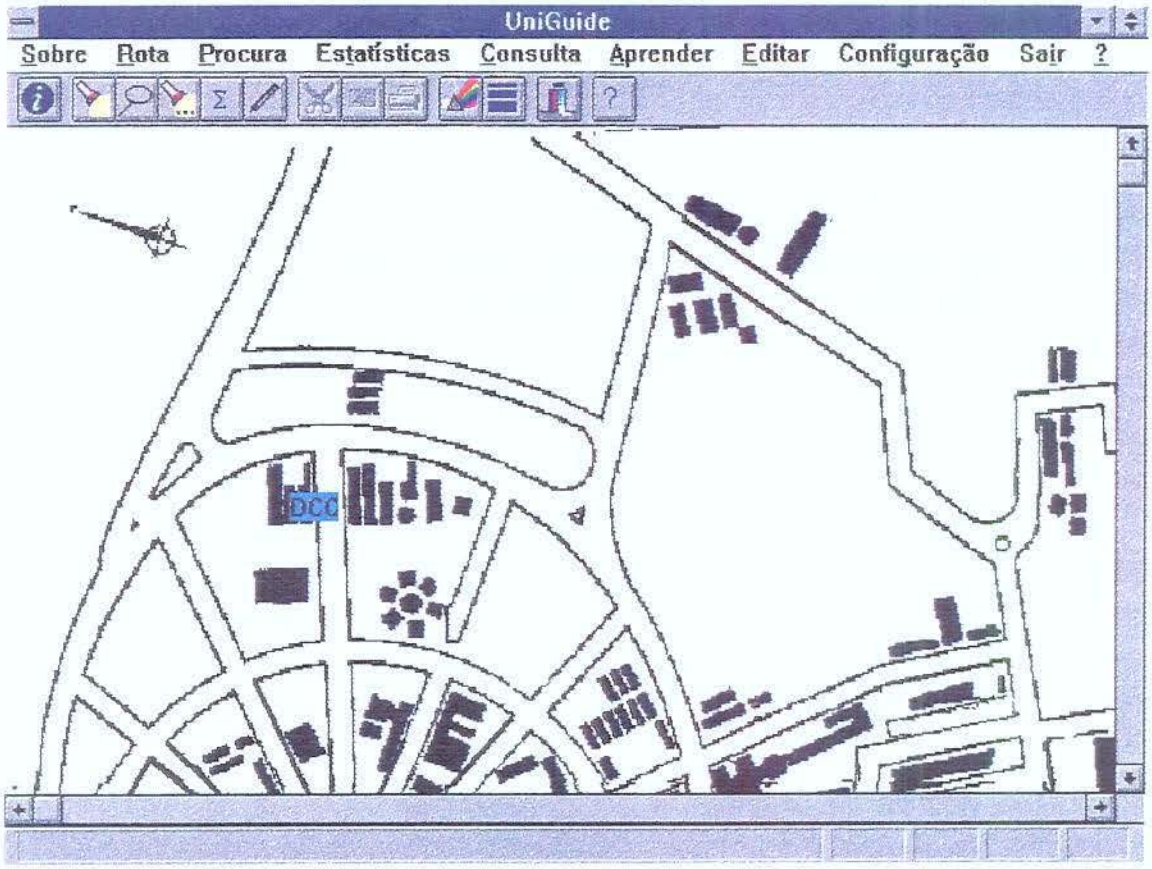



Figura 5.12 - Apresentação do nome do local na tela

• *Informações sobre rotas* - Além das duas maneiras apresentadas anteriormente de se entrar as informações sobre os lugares de origem e destino para que o UniGuide ache uma rota, pode-se conseguir o mesmo objetivo operando unicamente na área cliente da janela principal. O procedimento consiste no seguinte: Posicionar o cursor do mouse (que tem a forma de uma mão) sobre o lugar de origem. O UniGuide, então, deve apresentar o nome do local escolhido na tela. Feito isto, clicar o botão *direito* do mouse, cujo cursor, se a operação foi bem-sucedida, deve assumir a forma de uma lupa. A seguir, posicionar o cursor sobre o local de destino e clicar novamente com o botão *direito*. Feito isto, o UniGuide automaticamente dará início ao processamento da rota.

• *Informações sobre lugares* - O UniGuide possui, também, a capacidade de aprsentar informações em hipertexto sobre os lugares de um mapa. O procedimento para isto é bastante simples e consiste no seguinte: Posicionar o cursor do mouse sobre o lugar sobre o qual se deseja obter

informações e dar um duplo clique com o botão *direito* do mouse. Feito isto, o UniGuide apresentará as informações desejadas.

• *Posicionamento dos lugares*[5] - Um dos elementos fundamentais da descrição de um mapa, em GeoDL ou em qualquer outra linguagem, é informar a posição de determinado local. Para permitir que um desenvolvedor consiga estes dados facilmente, o UniGuide possui uma função que informa a posição de qualquer local. Para obter esta informação, é preciso apenas posicionar o cursor do mouse sobre a posição que se quer saber as coordenadas e clicar o botão *do meio*. Feito isto, o UniGuide mostra uma janela com estas coordenadas.

Outro elemento a se destacar na área cliente é a presença de barras de rolagem, situadas à direita e embaixo. Servem para rolar a imagem do mapa, de forma que o programa tenha a capacidade de lidar com mapas de qualquer tamanho. As barras, automaticamente, ajustam seus intervalos de rolagem para acomodar mapas de diversos tamanhos.

### 5.3.1.4 A barra de status

A barra de status possui, basicamente, duas funções: Mostrar mensagens rápidas de ajuda sempre que o cursor do mouse estiver sobre um botão da barra de ferramentas ou item de menu, informando o que tal botão ou item de menu realiza e mostrar o estado atual do programa e se as teclas NumLock, CapsLock e ScrollLock estão ativas ou não.

Com a utilização da barra de status, o usuário tem à sua disposição um poderoso auxiliar para dirimir suas dúvidas, bastando para isto posicionar o cursor do mouse sobre o botão ou item de menu, cuja função ele desconheça.

## 5.3.2 A Janela do Editor/ Compilador

Esta janela[6] concentra todas as operações relativas à edição/compilação de um arquivo GeoDL. A janela do editor foi mostrada na figura 5.9.

O editor do UniGuide é completo, inclusive com capacidade de busca e/ou substituição de palavras, interface padrão CUA ( *Common User*

[5]Disponivel apenas no modo de Desenvolvimento

[6]Disponivel apenas no modo de desenvolvimento.

*Access*), capacidade de copiar, recortar e colar para e a partir da área de transferência (*clipboard*), via mouse ou teclado, ou seja, possui todas as capacidades básicas dos editores de texto mais conhecidos.

Uma característica adicional deste editor/compilador é que ele permite adicionar ao código GeoDL modelos ( *templates*) de descrições de objetos geográficos como Place, Crossing, etc. Estas descrições foram exaustivamente discutidas no Capítulo III.

### 5.3.3 A janela de apresentação da rota

A janela de apresentação da rota (figura 5.13) tem como principal finalidade, como o próprio nome diz, apresentar ao usuário a rota desejada tanto na forma gráfica, como na textual. Ela não é uma janela nova propriamente dita, pois é apenas uma modificação da janela principal. As modificações são as seguintes:

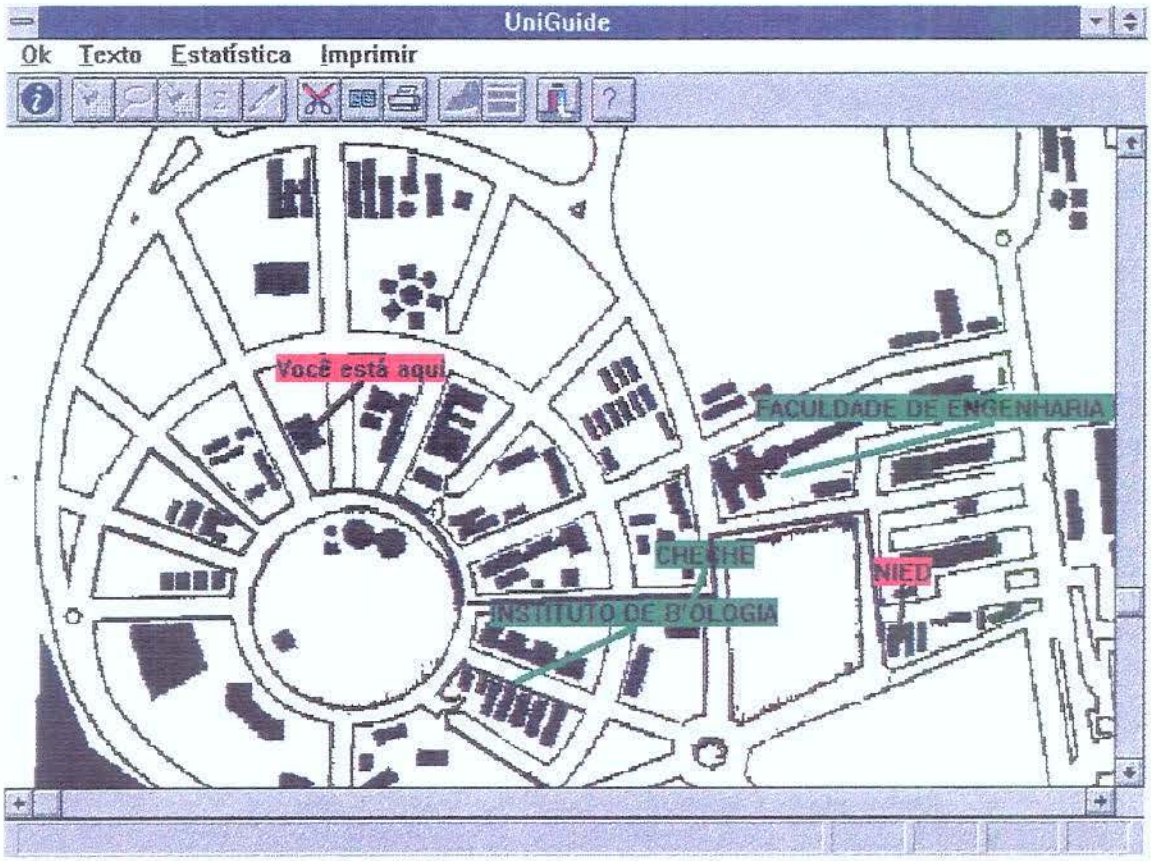



Figura 5.13 - Janela de Apresentação da Rota

• O menu é alterado. Agora possui os seguintes itens :

- *Ok* - Retorna ao modo normal, ou seja, à janela principal original.
- *Texto* - Apresenta a rota textualmente, em linguagem natural.
- *Imprimir* - Imprime a rota.

• A área cliente agora mostra, ao invés do mapa padrão, a rota pedida pelo usuário, apresentada graficamente.

• As operações com o mouse na área cliente são desabilitadas.

• Na barra de ferramentas, os botões representativos dos itens *Ok*, *Texto* e *Imprimir* tornam-se ativos e os que representam os itens do menu da janela principal normal tornam-se inativos (com exceção do que apresenta informações sobre o programa, do que representa a saída do programa e do que mostra a ajuda).

### 5.3.4 A janela de localização de lugares

Esta janela, a exemplo da anterior, também é apenas uma modificação da janela principal, cuja única função é localizar e mostrar um local especificado pelo usuário.

O UniGuide, para conseguir isto, rola, se necessário, a imagem do mapa tanto horizontalmente quanto verticalmente para que o usuário possa ver o local desejado na tela. Além disto o UniGuide também escreve o nome e aponta a localização com uma seta. (figura 5.13)

Além da área cliente diferente (como explicado acima), o menu principal também é modificado, possuindo agora apenas uma opção : *Ok*, que retorna o programa ao modo normal.

## 5.4 Plataforma e linguagem

A plataforma utilizada para a implementação do sistema UniGuide foi um microcomputador pessoal (PC) com processador Intel® Pentium II MMX 300 MHz, memória RAM de 64 MB e monitor SVGA de 15 polegadas. No entanto, o UniGuide roda em qualquer PC com processador 80386 ou superior com pelo menos 8 MB de memória RAM.

O sistema operacional escolhido foi o Microsoft Windows® 95. Por ser um aplicativo nativo de 32 bits o UniGuide roda também sem problemas no Windows NT 4.0 ou superior.

A linguagem utilizada para a implementação foi o C++, escolhida por sua portabilidade e por possibilitar a geração de código extremamente rápido.

## 5.5 Arquivos

Os arquivos que compõe o UniGuide, com as respectivas descrições e quantidade de linhas de código são :

| Arquivos | Descrição | Linhas de Código |
|---|---|---|
| uniguide.cpp<br>uniguide.h<br>unidefs.h | Arquivos principais do programa, implementam a janela principal. | 420<br>184<br>52 |
| editor.cpp<br>editor.h | Implementam o editor/compilador integrado | 170<br>41 |
| map.cpp<br>map.h | Implementam as operações com o mapa | 693<br>77 |
| dlgsrc.cpp<br>dlgsrc.h | Implementam as caixas de diálogo | 328<br>104 |
| routewnd.cpp<br>routewnd.h | Implementam a janela que mostra a rota sugerida como texto | 46<br>17 |
| route.cpp<br>route.h | Implementão a contrução e busca da rota. | 616<br>42 |
| frames.cpp<br>frames.h | Implementam os *frames* | 742<br>363 |
| searcher.cpp<br>searcher.h | Implementam a busca | 599<br>53 |
| symbtab.cpp<br>symbtab.h | Implementam a classe genérica para o compilador. | 228<br>74 |
| scanner.cpp<br>scanner.h | Implementam o analisador léxico do | 127<br>25 |

| | | |
|---|---|---|
| | compilador. | |
| parser.cpp<br>parser.h | Implementa o analisador sintático / semântico. | 327<br>39 |
| compiler.cpp | Módulo principal do compilador. | 88 |
| uniguide.rc | Arquivo de recursos | 80 |
| Total | - | 5550 |
| Total geral,incluindo arquivos *header* e VCL, compilado | | 201063 |

## 5.6 Considerações finais

Neste capítulo foram vistas, detalhadamente e com várias ilustrações, as principais características e como utilizar o UniGuide.

No próximo capítulo veremos um apanhado geral do trabalho e algumas sugestões para melhoramentos futuros.

# Capítulo VI

## Conclusões e trabalhos futuros

Neste trabalho desenvolvemos um modelo de conhecimento espacial denominado UniGuide baseado no modelo desenvolvido por Kuipers [Kuipers77] no MIT denominado TOUR.

Ao contrário de um GIS, nosso modelo acha as rotas de um modo similar ao que seres humanos utilizam inclusive com a aplicação de estruturas de dados que tentam mimetizar os processos mentais humanos.

A implementação é completa e atingiu, em testes preliminares, os seus objetivos principais, que são:

- Robustez, no sentido que o sistema consegue achar qualquer rota dentro do mapa descrito.
- Capacidade de lidar com vários tipos de mapas, desde aqueles relativamente pequenos (ex. Unicamp) como mapas maiores (por exemplo, o da cidade de Campinas). Isso é devido à grande flexibilidade da GeoDL. Com a descrição apropriada em GeoDL, o sistema pode aprender e informar rotas, a princípio, em qualquer mapa.
- Capacidade de aprender.

Entretanto, alguns melhoramentos podem ser implementados em versões posteriores do sistema, como por exemplo:

- *Suporte à multimídia* - o UniGuide pode beneficiar-se em vários aspectos do uso da multimídia. Por exemplo, a rota na forma textual poderia ser *falada* ao usuário. Além disto, na janela de apresentação de informações sobre lugares, poder-se-ia incluir além do texto, fotos e vídeos do local. Isto é particularmente importante em mapas de lugares turísticos.

- *Uso do GPS* - o GPS (sigla, em inglês, para Sistema de Posicionamento Global) seria extremamente útil se fosse acoplado a um computador portátil rodando o UniGuide. Desta forma, o usuário não precisaria informar o lugar de origem já que o UniGuide, automaticamente o localizaria pelas coordenadas obtidas por um aparelho de GPS. Mas a principal aplicação disto seria o uso do sistema como um guia contínuo, ou seja, o usuário especificaria o lugar de origem e o UniGuide o guiaria continuamente ao longo do caminho, sempre baseado nas coordenadas fornecidas pelo GPS. Um sistema semelhante a este é descrito em [Siewiorek93], que descreve a implementação de um computador especial equipado com GPS pela Universidade Carnegie-Mellon.

- *Desenvolvimento de uma versão para a Internet* – com a massificação da Internet, uma versão especificamente desenvolvida para ela seria de muita utilidade, pois os usuários, via rede, poderiam acessar todas as informações do mapa desejado. Isto poderia ser realizado, por exemplo, portando o código fonte para Java.

- *Implementar extensões para GeoDL* – Apesar de nossa implementação de GeoDL servir bastante bem a nossos propósitos algumas extensões seriam úteis para mapas mais complexos tais como viadutos, ruas com outros formatos ( diferentes de retas e círculos), etc.

- *Desenvolvimento de uma versão para um mapa maior* – por exemplo, para a cidade de Campinas.

- *Testes com usuários* – para possibilitar um melhor entendimento de como as pessoas armazenam e utilizam as informações sobre o conhecimento espacial visando a uma eventual melhoria do modelo empregado. Isto poderia ser realizado com a instalação de quiosques para alunos e/ou visitantes ou com a disponibilização da versão para a Internet do UniGuide.

- *Aprimoramento da interface* - Realizando-se testes com usuários a interface poderia ser melhorada visando a atender a necessidades específicas identificadas nos testes. Além disso, pode-se integrar capacidade multimídia à interface.

# Referências bibliográficas


[Aho86] Aho A. *et al*, *Compilers*, Addison-Wesley, 1986

[Appl92] Apple Computers Inc., *Macintosh Human Interface Guidelines*, Addison-Wesley, 1992

[Barstow87] Barstow D., *Artificial Intelligence and Software Engineering*, 1987

[Baum77] Baum D. R., *Cognitive Maps: The Mental Representation of Geographic Distance*, PhD Thesis, University of Michigan, 1977

[Begeman88] Begeman M. & Conklin J., Byte magazine, outubro/88, pp. 255-268

[Binstock96] Binstock A ., *Hashing Rehashed*, Dr. Dobb's Journal, Maio/96, pp.24-33.

[Blad91] Blades M., *Wayfinding Theory and Research: The Need for a New Approach*, in: Mark D.M., Frank A.U. (Eds.): "Cognitive and Linuistic Aspects of Geographic Space", Kluwer, 1991, pp. 137-165

[Boehm87] Boehm B.W., *Improving Software Productivity*, Survey and Tutorial Series, IEEE, Setembro/87, pp. 43-57

[Bolt91] Bolter J.D. *Writing Space*, Lawrence Erlbaum Associates, 1991

[Bonnet85] Bonnet A ., *AI - Promise and Performance*, Prentice-Hall, 1985

[Booker92] Booker, S. *et al*, *Designing The Whole-Product User Interface*, in Layrel B. (ed.): *The Art Of Human-Computer Interface Design*, Addison-Wesley, 1992

[BorlandC94] Borland International, *BorlandC++ 4.5 User's Guide*, Borland Press, 1994

[Bower72] Bower G. H., *Mental Imagery and Associative Learning*, Cognition in Learning and Memory, New York, 1972

[Brown91] Brown P., *Structure, Navigation And Hypertext: The Status Of The Navigation Problem*, proc. of hypertext 91, p. 363-366

[Bryant92] Bryant D.J.*A Spatial Representation System in Humans*, target paper in Psycoloquy, psycoloquy.92.3.16.space.1.bryant, psyc@pucc.bitnet, 23/5/1992

[Calvert97] Calvert, C., *Borland C++ Builder Unleashed*, Borland press1997

[Card93] Card S.K. e Newell A ., *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum Publishers, 1983

[CaTa75] Canter D. & Tagg S.K., *Distance Estimation in Cities*, Environment and Behavior, Vol.7, No.1, March 1975, pp. 59-80

[Charniak87] Charniak E. *et al*, *Introduction to Artificial Intelligence*, Addison-Wesley, 1987

[Chorefus88] Chorefus D., *Applying Expert Systems in Business*, MacGraw-Hill, 1988

[Conklin87] Conklin J., *A Survey Of Hypertext*, MCG Technical Report, 1987

[Conklin87A] Conklin J., *Hypertext: An Introduction and Survey*, Survey and Tutorial Series, , IEEE Setembro/87, pp.17-41

[CoOl93] Cotton B., Oliver R., *Understanding Hypermedia*, Phaidon Press, 1993

[Cormen91] Cormen T. *et al*, *Introduction to Algorithms*, Addison-Wesley, 1991

[DeGrost86] DeGrost D. *et al*, *Logic Programming*, Prentice-Hall, 1986

[Diaz86] Diaz B e Bell S (eds.), *Spatial Data Processing Using Tesseral Methods*, 1986

[Dieberger94] Dieberger A., *Navigation in Spatial Information Environments - User Interface DesignIissues for Hypertext and VR Systems*, ECHT'94

[DoSt73] Downs R.M. & Stea D., *Cognitive Maps and Spatial Behaviour: Process and Products*, Downs R.M., Stea D. (Eds.): *Image And Environment; Cognitive Mapping And Spatial Behaviour*, London, Edward Arnold, 1973

[Eckel93] Eckel B., *C++ Inside and Out*, McGraw-Hill, 1993

[Egenhofer91] Egenhofer M J e Herring J R, *High-Level Spatial Data Structures For GIS*, in Maguire D J, Goodchild M F, Rhind D W (eds.) *Geographical Information Systems: principles and applications*, Vol I, pp 227-36, Longman, London, 1991

[Faison95] Faison T., *Examining BorlandC++ 4.5*, Dr. Dobb's Journal, Abril/95, pp. 68-76

[Feigenbaum89] Feigenbaum E. & Barr A ., *The Handbook of Artificial Intelligence*, Heuristech Press & William Kaufmann, 1989

[Fiderio88] Fiderio J., *A Grand Vision*, Byte magazine, outubro/88, pp.237-244

[Frank84] Frank A U, *Computer Assisted Cartography - Graphics Or Geometry*, in *Journal of Surveying Engineering*, 110(2), pp. 159-62, 1984

[Frank91] Frank A U, Mark D M, *LanguageIissues for GIS*, em Maguire D J, in Goodchild M F, Rhind D W (eds.): *Geographical Information Systems: Principles And Applications*, Vol I, pp 147-63, Longman, London, 1991

[Franklin91] Franklin WM R, *Computer Systems And Low Level Data Structures for GIS*, in Maguire D J, Goodchild M F, Rhind D W (eds.): *Geographical Information Systems: principles and applications*, Vol I, pp 215-24, Longman, London, 1991

[Freksa91] Freksa C.: *Qualitative Spatial Reasoning*, in Mark D.M., Frank A.U. (Eds.): *Cognitive and Linuistic Aspects of Geographic Space*, Kluwer, 1991, pp. 361-372

[Gardner83] Gardner H., *Frames of Mind: The Theory of Multiple Intelligences*, 1983

[Gardner85] Gardner H., *The Mind's New Science*, Basic Books, 1985

[Gatrell91] Gatrell A C, *Concepts Of Space And Geographical Data*, in Maguire D J, Goodchild M F, Rhind D W (eds.): *Geographical Information Systems: Principles And Applications*, Vol I, pp 119-34, Longman, London, 1991

[Genaro87] Genaro S., *Sistemas Especialistas*, CTC, 1987

[Georgeff88] Georgeff M, *Reasoning about Plans and Actions*, in Exploring Artificial Intelligence, Morgan Kaufman, 1988

[Gessner90] Gessner R., *Building a Hypertext System*, Dr. Dobb's Journal, Junho/90, pp. 22-32

[Glass90] Glass A . et al, *Cognition*, Addison-Wesley Publishing, 1990

[Gorlen90] Gorlen K. *et al*, *Data Abstraction and Object-Oriented Programming in C++*, B.G. Teubner, 1990

[Halfhill95] Halfhill T., *Inside Microsoft's Mind*, Byte magazine, Agosto/95, pp. 46-52

[Heintze94] Heintze S., *Programming C++ as a Better C*, Software Development, Nov/94, Miller-Friedman

[Holtzner94] Holtzner S., *BorlandC++ for Windows Programming*, Makron Books, 1994

[Horn90] Horn R., *Mapping Hypertext*, The Lexington Institute, 1990

[Horowitz94] Horowitz E. & Sahni S., *Fundamentals of Data Structures*, Computer Science Press, 1994

[Jacobson92] Jacobson B., *The Ultimate User Interface*, Byte 4/1992, pp. 175-188

[Johnson90] Johnson J.S., *The DDJ Help Project*, Dr. Dobb's Journal, Junho/90, pp. 16-18

[Kaplan73] Kaplan S., *Cognitive Maps In Perception And Thought*, Downs and Stea, 1973

[King90] King T., *A Self-Referencial Hypertext Engine*, Dr. Dobb's Journal, Junho/90, pp. 34-38

[Kosko92] Kosko B., *Nerural Networks and Fuzzy Systems*, Prentice-Hall, 1992

[Kowaltowski83] Kowaltowski T., *Implementação de Linguagens de Programação*, Guanabara Dois, 1983

[Kuipers77] Kuipers B. J., *Representing Knowledge of Large-Scale Space*, PhD Thesis, MIT Press, 1977

[Kuipers78] Kuipers, B. J., *Modeling Spatial Knowledge*, Cognitive Science, Vol.2, pp. 129-153, 1978

[Lehnert88] Lehnert W., *Knowledge-based Natural Language Understanding*, in Exploring Artificial Intelligence, Morgan Kaufman, 1988

[Levine88] Levine R., *A Comprehensive Guide to AI and Expert Systems*, McGraw-Hill, 1988

[Lynch60] Lynch K.,*The Image of the City*, MIT Press, 1960

[Lynch91] Lynch K., *Managing the Sense of a Region*, MIT Press , 1991

[MarDam91] Marcus A., van Dam A., *User-Interface Developments For The Nineties*, IEEE Computer, Vol.24 (9), pp. 49-57, 1991

[Maling91] Maling D H, *Coordinate Systems And Map Projections For GIS*, in Maguire D J, Goodchild M F, Rhind D W (eds.): *Geographical Information Systems: principles and applications*, Vol I, pp 135-46, Longman, London, 1991

[McCl94] McCloud S. *Understanding Comics - the invisible art*, Kitchen Sink Press, 1994

[Minsky75] Minsky M., *A Framework For Representing Knowledge*, in *The Psychology of Computer Vision*, P.H. Winston (Ed.), McGraw-Hill, 1975

[Montello93] Montello D.R. *Scale and Multiple Psychologies of Space*, Proc. of the COSIT'93, pp. 312-321, LNCS 716, 1993

[Mountford92] Mountford, S., *Tools And Techniques For Creative Design*, in Layrel B. (ed.): *The Art Of Human-Computer Interface Design*, Addison-Wesley, 1992

[MuFH94] Mukherjea S., Foley J.D., Hudson S.E., *Interactive Clustering for Navigating in Hypermedia Systems*, Proc. ECHT'94, pp. 136-145

[Mullin89]Mullin M., *Object-oriented Program Design*, Addison-Wesley, 1989

[Newell72] Newell A ., *Human Problem Solving*, Prentice-Hall, 1972

[Newquist94] Newquist R., *Lisp for Lunch*, AI Expert, Maio/94

[Nielsen90] Nielsen J., *The Art of Navigating through Hypertext*, Communications of the ACM, Março/90, Vol. 33, Número 3

[Notenboom90] Notenboom L. & Vose M., *Building an efficient help system*, Dr. Dobb's Journal, Junho/90, pp. 40-48

[Nune91] Nunes, J. *Geographic Space as a set of concrete Geographical Entities*, in Mark D.M., Frank A.U. (Eds.): *Cognitive and Linuistic Aspects of Geographic Space*, Kluwer, 1991, pp. 9-33

[Nune94] Nunes J, *Spatial Concepts and the representation of spatial knowledge*, 1994

[Pappas94] Pappas C. & Murray W., *BorlandC++ handbook*, Osborne, 1994

[Pederson93] Pederson E, *Geographic and manipulable space in two Tamil linguistic systems*, Proc. COSIT'93, pp. 294-311

[Perrault88] Perrault C. R. & Grosz B., *Natural Language Interfaces*, Exploring Artificial Intelligence, Morgan Kaufman, 1988

[Petzhold96] Petzhold C., *Programming Windows 95*, Microsoft Press, 1996

[Plumb94] Plumb D.., *An Algorithm for Generating Real Random Numbers*, Dr. Dobb's Journal, Abril/94

[Retz- Schmidt88] Retz-Schimidt G., *Various Views on Spatial Prepositions*, AI Magazine, Summer 88, pp. 95-105

[Rumbaugh91] Rumbaugh J. et al, *Object-oriented modeling and design*, Prentice-Hall, 1991

[Rumelhart74] Rumelhart D., *The Room Theory*, 1974

[Schutzer87] Schutzer D., *AI - An Application oriented approach*, Van Nostrand, 1987

[Sedgewick88] Sedgewick R., *Algorithms*, Addison-Wesley, 1988

[Schneiderman83] Shneiderman B., *Direct Manipulation: A Step Beyond Programming Languages*, IEEE Computer, August 1983, pp. 57-69, 1983

[Schneiderman92] Shneiderman B., *Designing the User Interface*, 2nd Edition, Addison Wesley, 1992

[Shum90] Shum S.B., *Real and Virtual Spaces: Mapping from spatial cognition to Hypertext*, Hypermedia, Vol.2, No.2, 1990, pp. 133-158

[Siewiorek93] Siewiorek D. et al, *The VuMan 2 Wearable Computer*, Desogn & Test of Computers , Setembro 1993, IEEE

[Stroustrup97] Stroustrup, B., The C++ Programming Language, Addison-Wesley, 1997

[Swan94] Swan T., *Mastering Windows Programming*, Berkeley, 1994

[Telles97] Telles, M., *High Performance C++ Builder*, 1997

[Tello89] Tello E., *Object-oriented programming for AI*, Addison-Wesley, 1989

[Tognazzini92] Tognazzini, B., *Tog on Interface*, Addison Wesley, 1992

[Tversky93] Tversky B., *Cognitive Maps, Cognitive Collages and Spatial Mental Models*, Proc. of the COSIT'93, pp. 14-24, LNCS 716, 1993

[Valduriez94] Valduriez P., *Some Hints to Improve Writing of Technical Papers*, Engineering of Information Systems Vol. 3, Nº 3, Hermes, 1994

[Vertelney92] Vertelney, L. *et al, Two Disciplines In Search Of An Interface*, in Layrel B. (ed.): *The Art Of Human-Computer Interface Design*, Addison-Wesley, 1992

[Vieira81] Vieira H., *Recuperação de Erros em Analisadores Sintáticos Descendentes*, IMECC-UNICAMP, Dissertação de Mestrado, 1981

[Walters88] Walters J. & Nielsen N., *Crafting Knowledge-Based Systems*, Wiley Interscience, 1988

[Winston92] Winston P., *Artificial Intelligence*, Addison-Wesley, 1992

[ZiBL94] Zizi M., Beaudouin-Lafon M., *Accessing Hyperdocuments through Interactive Dynamic Maps*, Proc. ECHT'94, pp. 126-135