Super Spider: uma ferramenta versátil para exploração de dados
multi-dimensionais representados por malhas de triângulos

Lionis de Souza Watanabe

Data de Depósito: ____ / ____ / ____

Assinatura : _____

Super Spider: uma ferramenta versátil para exploração de dados multi-dimensionais representados por malhas de triângulos

1

Lionis de Souza Watanabe

lionis@icmc.usp.br

Orientadora:

Prof^a. Dr^a. Rosane Minghim

rminghim@icmc.usp.br

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC, USP como parte dos requisitos para a obtenção do título de Mestre em Ciências de Computação e Matemática Computacional.

USP - São Carlos

Fevereiro de 2007

¹Este trabalho tem o apoio financeiro do **CNPq** (Conselho Nacional de Desenvolvimento Científico e Tecnológico).

AGRADECIMENTOS

Em primeiro lugar, quero parar, sentar, e refletir. Nesses 3 anos de mestrado, amadureci 20 anos em experiência de vida. Foi um período muito bom, turbulento, muito ruim. Complicado assim mesmo... E se não perdi o rumo e hoje consegui concluir meu mestrado, devo essa conquista principalmente a Deus, que me fez tornar um homem muito melhor, mais consciente, responsável e que, mais do que nunca, aprendeu a dar mais valor à família.

Agradeço ao meu pai, Dr. Susumo, cirurgião-dentista, que fez medicina na USP Pinheiros, mas preferiu fazer odonto na UNESP de Araraquara, e exercer essa outra profissão. Homem de fibra, trabalhou até seus 84 anos por quase 60 anos, sem nunca ter faltado ao trabalho, inclusive nos finais de semana. Aliás, minto... ele faltou sim, por 1 dia quando quebrou a perna. E em casa, sempre foi um pai muito presente e rígido. Desde cedo aprendi que educar não é espancar ou gritar, mas ser o exemplo e merecer o respeito. Peço a Deus que me ajude a me tornar esta figura para meus filhos, assim como meu pai foi para mim.

Hoje, os papéis se inverteram, e eu cuido dele. Não por obrigação ou pelo dever, mas pela admiração e respeito que cultivou em mim por todos esses anos.

Agradeço à minha mãe, Matilde, que esqueceu de si mesma e viveu por mim, apenas por mim. Amor de mãe é incondicional, inexplicável, divino! De tanto se preocupar com os outros, adoeceu e faleceu há alguns meses. Tantos planos, sonhos, desejos... não deu tempo. Foi nesse momento que perdi o chão, minha base afetiva. Podíamos ter passado mais tempo juntos. Eu queria tê-la beijado mais, abraçado mais, declarado mais meu amor a ela. Podia... Mas mesmo não podendo abraçá-la mais, eu a amo. Simples assim...

E nesse momento, em que mais precisei de amparo, minha noiva Andressa apareceu, e se tornou essa base de apoio para que eu levantasse e continuasse meu trabalho. Agradeço - e muito - a ela por ser realmente minha companheira, e por ter provado isso. Obrigado por tudo.

Agradeço ao meu irmão, Dr. César, por todo o apoio nessa fase tão difícil.

Agradeço à minha orientadora Prof^a Dr^a Rosane Minghim, que foi muito além de uma simples mentora do meu projeto. Foi uma companheira de trabalho, sempre comprando brigas dos seus alunos e entendendo o lado pessoal de cada um. Ela age com humanidade, e tem esse lado mãe de ser com seus orientandos.

Agradeço aos Profs. Drs. Luis Gustavo Nonato e Maria Cristina pela orientação no projeto, pelo profissionalismo e pela infinita paciência comigo.

Agradeço ao Profs. Drs. Castelo e Poti, pela chance que me foi concedida, para trabalhar em um projeto com bolsa de estudos. Infelizmente, a circunstância da ocasião me obrigou a abandonar o projeto, o doutorado iminente, enfim, São Carlos.

Agradeço à República Enxofre, inesquecível, onde vivi os melhores momentos da minha estadia fora de casa! Faiskão, Júlio, Otávio, Flávio... inesquecíveis! Se eu voltar pra

Sanca, volto a morar com eles! O último agregado, o Tony, também é um sujeito difícil de esquecer!

Agradeço aos meus orientandos: Marco Antônio, já falecido, e Renato Rodrigues, o cara mais correto e firme que conheci em São Carlos! Outro grande amigo que vai ficar marcado.

Agradeço aos meus amigos do LCAD, principalmente o Fernando Kurokawa. Dizem que japonês é 8/80! Eu sou o 8, e ele é o 80! Me ajudou muito, com LaTeX, conselhos e caronas!

Agradeço ao Maikel, grande amigo! Me livrou de várias enrascadas, inclusive superlotação do meu carro! rs... Obrigado por tudo, pelos conselhos, lições de vida, e pela irmandade que me ofereceu.

Agradeço ao CNPq pelo suporte financeiro, possibilitando minha permanência em São Carlos e meus estudos.

WATANABE, L. S. *Super Spider: uma ferramenta versátil para exploração de dados multi-dimensionais representados por malhas de triângulos*, São Carlos, fevereiro de 2007. Dissertação de Mestrado - Instituto de Ciências Matemáticas e de Computação - ICMC, USP.

RESUMO

Visualização é o processo genérico que utiliza representações visuais e interativas para facilitar a aquisição de conhecimento humano. Assim, o processo exploratório deve seguir uma sequência de procedimentos, independente da área de aplicação, para que seja bem sucedido. Este trabalho apresenta o *Super Spider*: um sistema de exploração visual baseado no *Spider Cursor* [Salvador e Minghim 2003], que conta com novos recursos de auxílio à investigação visual, além de ser uma ferramenta portátil e flexível. Ambos sistemas compartilham técnicas de um arcabouço comum à área de Visualização Computacional, classicamente dividida em Visualização Científica e Visualização de Informação. Portanto, este trabalho possui 3 objetivos principais: demonstrar a genericidade das técnicas empregadas, propor novos recursos exploratórios, e avaliar a melhor plataforma de desenvolvimento para esta categoria de aplicação. Os novos recursos exploratórios incluem a codificação e apresentação de informações através de atributos de super-quádricas que variam com o movimento do cursor.

WATANABE, L. S. *Super Spider: a versatile tool for multi-dimensional data represented as triangle meshes*, São Carlos, february 2007. Master thesis - Instituto de Ciências Matemáticas e de Computação - ICMC, USP.

ABSTRACT

Visualization consists of a generic process that uses visual and interactive representations to improve human knowledge acquisition. Therefore, the exploratory process should follow a sequence of procedures, regardless of the application area, so it could succeed. This work presents the *Super Spider*: a visual exploration system, based on *Spider Cursor* [Salvador and Minghim 2003], that proposes innovative techniques to aid visual investigation, apart from being a portable and flexible tool. Both systems share the same core, familiar to Computer Visualization, which is classically divided in Scientific Visualization and Information Visualization areas. Thus, this work has 3 main objectives: to show the generality of chosen techniques, to validate the novel exploration tasks, and to evaluate the development environment for this application category. The novel exploratory resources include coding and presenting information through attributes of super-quadrics varying together with the movement of the cursor.

Sumário

Lista de Figuras	v
Lista de Tabelas	vii
1 Introdução	1
1.1 Contextualização	1
1.2 Motivação e Objetivos	2
1.3 Estrutura	3
2 Visualização e Exploração de Dados	5
2.1 Considerações Iniciais	5
2.2 Exploração de Dados	6
2.2.1 Princípio de <i>interfaces</i> visuais	6
2.2.2 Aspectos Humanos	9
2.3 Visualização Científica	12
2.3.1 Técnicas 3D para dados escalares	14
2.4 Visualização de Informação	15
2.5 Considerações Finais	20
3 Sistemas Exploratórios	25
3.1 Considerações iniciais	25
3.2 Técnicas interativas	25
3.2.1 Navegação	25
3.2.2 Seleção e corte	28
3.2.3 Técnicas $F+C$	30
3.2.4 Scatterplots	31
3.3 Sistemas interativos	32
3.3.1 DSVol	32
3.3.2 Voxelplot	34

3.3.3	SimVis	37
3.3.4	<i>Parallel Sets</i>	40
3.3.5	InfoSky	41
3.3.6	InterRing	42
3.4	Considerações finais	43
4	Super Spider	45
4.1	Considerações iniciais	45
4.2	Descrição geral	45
4.3	Funcionalidades	47
4.3.1	<i>Spider Cursor</i>	47
4.3.2	<i>Super Spider</i>	49
4.3.3	Corte no <i>Super Spider</i>	51
4.3.4	Navegação e manipulação	55
4.4	Detalhes de desenvolvimento	56
4.4.1	Primitivas de <i>ray tracing</i>	56
4.4.2	Cônicas, quádricas e super-quádricas	58
4.4.3	Dimensão Fractal	59
4.5	Considerações finais	61
5	Testes e conclusões	63
5.1	Considerações iniciais	63
5.2	Testes comparativos entre VTK e Java3D	63
5.2.1	Testes de FPS (quadros por segundo)	64
5.2.2	Análise do consumo de memória	64
5.3	Análise de dados	67
5.3.1	Notícias internacionais	67
5.3.2	Artigos científicos	68
5.4	Conclusão	70
A	Cônicas, quádricas e super-quádricas	81
A.1	Produto esférico	81
A.1.1	Cônicas	82
A.1.2	Super-cônicas	82
A.1.3	Quádricas	84
A.1.4	Super-Quádricas	86
B	Implementação	93
B.1	Java 3D	93
B.1.1	Elementos de uma aplicação Java3D	95

B.2	<i>Spider Cursor</i> - C++ e VTK/STK	96
B.2.1	Estrutura interna	98
B.3	<i>Super Spider</i> - Java e Java3D	98
B.3.1	Estrutura interna	98

Lista de Figuras

2.1	Ciclo de navegação e exploração visual	10
2.2	<i>Isolinhas</i>	13
2.3	<i>Malhas</i>	14
2.4	<i>Campo de Altura</i>	14
2.5	<i>Marching Cubes</i>	15
2.6	<i>Ray Casting</i>	16
2.7	<i>Scatterplot-Matrices e Landscapes</i>	17
2.8	<i>Prosection Views e Hyperslice</i>	17
2.9	<i>Parallel-Coordinates</i>	18
2.10	<i>Chernoff-Faces e Stick Figures</i>	18
2.11	<i>Shape Coding e Color Icons</i>	19
2.12	<i>Space-Filling Curves</i>	19
2.13	<i>Recursive Pattern Technique</i>	19
2.14	<i>Spiral Technique</i>	21
2.15	<i>Axes Technique</i>	21
2.16	<i>Circle Segments Technique</i>	21
2.17	<i>Dimensional Stacking e Worlds-within-Worlds</i>	22
2.18	<i>Treemap e Conetree</i>	22
2.19	<i>InfoCube</i>	23
2.20	Grafos 2D	23
2.21	Grafos 3D	23
3.1	Dispositivos de Realidade Virtual	27
3.2	<i>Viewpoint</i> centrado no objeto	28
3.3	Exemplos de <i>cutout</i> e <i>breakaway</i>	29
3.4	Diagrama hierárquico das classes de interações	33
3.5	Exemplo de <i>depth of field</i>	35
3.6	Interface do Voxelplot	36

3.7	Exemplo de <i>brushing</i> no Voxelplot	37
3.8	Interface do SimVis	38
3.9	Representação de profundidade no SimVis	39
3.10	Representação de densidade no SimVis	39
3.11	Visualização em <i>Parallel Sets</i>	41
3.12	Composição de dimensões no <i>Parallel Sets</i>	41
3.13	Exibição de galáxia no <i>InfoSky</i>	42
3.14	Técnica RSF	43
4.1	<i>Spider Cursor</i>	46
4.2	Interface do <i>Spider Cursor</i>	48
4.3	Interface gráfica do <i>Super Spider</i>	49
4.4	Corte de uma região relevante	51
4.5	Solução 1: Lançamento de raios	52
4.6	Solução 1: Intersecção de arestas com o raio	53
4.7	Solução 1: Polígono com interior vazio	53
4.8	Solução 2: Análise de convexidade de polígonos	54
4.9	Solução 3: Soma de ângulos em 3D	55
4.10	Primitivas de <i>ray-tracing</i>	58
4.11	Cursor cruzando informações sobre “crime” e “advogado”	59
5.1	Tomografia de um mamão	66
5.2	Mapa gerado pela ferramenta <i>Projection Explorer</i>	68
5.3	Análise da dimensão fractal em notícias internacionais	69
5.4	Segunda análise da dimensão fractal em notícias internacionais	70
5.5	Análise da dimensão fractal em artigos científicos	71
5.6	Análise da dimensão fractal em artigos científicos	72
A.1	Super-elipse 2D de raio r	82
A.2	Super-elipse 2D variando sua forma.	83
A.3	Toro de raio interno r_0 , e raio da seção cruzada r_1	85
A.4	Toro de raio interno $r_0 = 1$, e raio da seção cruzada $r_1 = 0,25$	86
A.5	Tabela de formas de super-elipsóides	88
A.6	Tabela de formas de super-hiperbolóides de 1 lâmina	89
A.7	Tabela de formas de super-hiperbolóides de 2 lâminas	90
A.8	Tabela de formas de super-toróides	91
B.1	Exemplo de <i>Scene graph</i> no Java3D	95
B.2	Exemplo de utilização do <i>Spider Cursor</i>	97
B.3	Diagrama de componentes, enfatizando as trocas de mensagens entre as classes do <i>Super Spider</i>	100

Lista de Tabelas

5.1	Teste: isosuperfícies	65
5.2	Teste: isolinhas	65
5.3	Teste: <i>scatterplots</i>	65
5.4	Arquivo de dados: mamão.vtk, com 28.934 células.	65
A.1	Formas de uma super-elipse 2D	83
A.2	Formas de uma super-elipse 3D	87

Introdução

1.1 Contextualização

Nas últimas décadas, as áreas de conhecimento humano geraram uma grande quantidade de informações, que só puderam ser armazenadas e consultadas pelo poder computacional que possuímos na atualidade. No entanto, tem-se notado que muito conhecimento pode ser adquirido analisando não apenas a informação em si, mas todo seu contexto e conjunto. Daí a necessidade de métodos capazes de sintetizar esse aglomerado de dados para que seja possível interpretá-lo, apresentando-o de forma simples e amigável.

Devido a esta demanda, a área de Visualização de Dados (também chamada de Visualização Computacional [Minghim e Oliveira 1997]) surgiu como uma ferramenta poderosa para exploração de dados, e configura um campo crescente, graças à sua flexibilidade de adaptação às mais variadas aplicações e situações. Outro fator relevante para este crescimento é a característica inerente do ser humano de saber lidar melhor com gráficos e imagens, ao invés de números e textos.

A Visualização Computacional é responsável pelo desenvolvimento e aplicação de técnicas visuais para representar quaisquer conjuntos de dados, e compreendê-los. Desta forma, observa-se que a Visualização Computacional é um campo amplo, aplicando conhecimento de vários setores: Computação Gráfica, Geometria Computacional, Análise Numérica, Estatística, e em geral possui aplicação no estudo e análise de dados que dependam da capacidade de interpretação visual humana.

1.2 Motivação e Objetivos

Nos últimos anos, a área de Visualização Computacional tem sido classificada em duas sub-áreas: Visualização Científica e Visualização de Informação.

Estas áreas utilizam abordagens distintas ao trabalhar com os dados. Por exemplo, em Visualização Científica [Brodie et al. 1992], os dados são tratados espacialmente, pois assume-se que possuam natureza geométrica. Exemplos são dados médicos, de fluidos e meteorológicos. Neste caso, esforços são despendidos na estrutura geométrica do objeto em estudo. Por outro lado, em Visualização de Informação [Keim, Kriegel e Seidl 1994], os dados são abstratos, demandando novas técnicas para apresentá-los visualmente. Exemplos são dados de bioinformática, textos e conjuntos de textos, comércio eletrônico, financeiro, etc.

Apesar das diferentes abordagens, as duas áreas têm convergido para a mesma direção, pois os dados e as aplicações atuais tendem a possuir os dois tipos de informações - espaciais e abstratas - ao mesmo tempo. Desta forma, ferramentas de Visualização Científica necessitam das técnicas bem-difundidas de Visualização de Informação, e vice-versa.

Portanto, este trabalho apresenta o *Super Spider*, que consiste de uma ferramenta de exploração de malhas que compartilha técnicas de ambas as áreas da Visualização Computacional.

Baseada no seu predecessor *Spider Cursor*, a nova ferramenta mantém várias características, como o mesmo formato para arquivo de dados, o mesmo Cursor Aranha e várias técnicas em comum. No entanto, a alteração da biblioteca gráfica e da plataforma de desenvolvimento possibilitaram maior portabilidade, flexibilidade e disponibilidade. Além disso, novas técnicas foram desenvolvidas: *super-quádricas* e *dimensão fractal*. Desta forma, o *Super Spider* é capaz de executar diretamente do navegador Web e analisar as malhas de forma mais eficiente. Assim, este trabalho possui 3 objetivos principais:

- Demonstrar que o processo exploratório independe da natureza dos dados sob investigação, e que técnicas como isolinhas, campos de altura, *glyphs* e super-quádricas, mesmo de áreas distintas, podem ser utilizadas simultaneamente;
- Adicionalmente, o desenvolvimento do trabalho permitiu avaliar qual o melhor arcabouço gráfico (VTK ou Java 3D) para esta categoria de aplicação. Partindo do fato de que a maior diferença entre os sistemas é a tecnologia empregada no seu desenvolvimento, foram realizados testes comparativos entre as ferramentas, sendo consideradas as seguintes características: portabilidade, desempenho, disponibilidade, e principalmente, utilidade dos recursos interativos ao usuário;
- Validar a utilização de novos recursos - super-quádricas (Seção 4.4.2) e cálculo da dimensão fractal (Seção 4.4.3) - para a exploração visual dos dados.

1.3 Estrutura

Este documento apresenta a seguinte organização:

- O **capítulo 2** tem por objetivo esclarecer alguns conceitos básicos sobre o campo da Visualização Computacional, abordando questões referentes ao processo de visualização passo-a-passo e classes de técnicas específicas das sub-áreas da visualização;
- No **capítulo 3** é abordada a questão da interatividade, tratando fatores humanos envolvidos neste processo. Assim, são apresentadas técnicas interativas, que melhoram a compreensão do usuário sobre os dados, e também sistemas interativos, cujas principais características são a intuitividade dos apelos visuais e a usabilidade dos controles e metáforas de interação;
- O **capítulo 4** descreve os 2 sistemas desenvolvidos durante este trabalho de mestrado: *Spider Cursor* e *Super Spider*, seu precursor, tratando suas características e funcionalidades. Define também os fundamentos dos novos recursos de interação: super-quádricas e dimensão fractal;
- No **capítulo 5** os sistemas são comparados de acordo com vários testes realizados: desempenho, ocupação de memória e interatividade. Também é avaliada a utilidade das informações colhidas pela sondagem com o cursor, analisando alguns *corpora*¹ de dados.

Os testes realizados comprovam que a utilização de novos recursos visuais permitem melhor entendimento dos dados e de seu contexto, desde que a resposta ao processo interativo seja rápida e confortável ao usuário. Sendo assim, a investigação de dados multidimensionais é facilitada pela combinação de várias técnicas, que conseguem representar simultaneamente vários atributos dos vértices das malhas. Desta forma, o usuário possui uma visão mais realista dos dados investigados, absorve melhor as informações e assim consegue inferir padrões até então desconhecidos.

A ferramenta, assim como os mapas gerados e utilizados nos testes desta dissertação, podem ser encontrados no site <http://www.icmc.usp.br/~lionis>.

¹Corpo de textos.

Visualização e Exploração de Dados

2.1 Considerações Iniciais

A Visualização Computacional é uma área de pesquisa que estuda estratégias para gerar representações gráficas a partir de dados. Desta forma, grandes conjuntos de dados são melhor compreendidos, bem como o contexto em que estão inseridos.

Nota-se que a quantidade de informações armazenadas e processadas atualmente é demasiadamente grande, gerando a necessidade de utilizar o aparato de ferramentas da Visualização Computacional a fim de interpretar estes dados e inferir padrões e tendências neles codificados.

Entretanto, as ferramentas existentes não são de propósito genérico, pois os sistemas de visualização da atualidade encontram-se classificados em uma das duas sub-áreas da Visualização Computacional: Visualização Científica ou Visualização de Informação. A primeira trata de visualizar dados de natureza geométrica (ex: corpo humano), enquanto para a segunda uma representação visual deve ser extraída antes da exploração (ex: dados de censo demográfico). Apesar da distinção da natureza dos dados, que inicialmente promoveu o desenvolvimento da visualização em dois ramos diferentes, a forma de exploração de informação do ser humano é semelhante em ambos os casos. Cada vez mais os dois tipos de informação estão presentes concomitantemente numa mesma aplicação. As principais diretrizes da visualização exploratória, sob essa perspectiva, são discutidos na Seção 2.2. Nas Seções 2.3 e 2.4, são visitadas as principais classes de técnicas de Visualização Científica e de Informação existentes.

2.2 Exploração de Dados

Apesar da crescente quantidade de informações produzidas pelos sistemas computacionais, há uma tendência de que as próximas gerações de bibliotecas digitais para bases estruturadas, documentos textuais e de dados multimídia permitirão um processo de exploração mais eficiente e automatizado, permitindo que um maior número de usuários executem tal tarefa.

Os objetivos da exploração variam desde encontrar um amplo conjunto de itens (*clusters*¹) que satisfaçam um critério bem definido, até inferir padrões desconhecidos, aprimorando o entendimento sobre a coleção de dados. Entretanto, este processo exploratório tende a tornar-se mais difícil à medida que o volume de dados cresce.

Assim, a visualização tem o poder de tornar fenômenos físicos, como a propagação do calor nos motores, visíveis e compreensíveis. Por outro lado, também é capaz de revelar padrões, *clusters*, *gaps*² e contornos em dados estatísticos, por exemplo. A utilização de *interfaces* visuais tem se tornado cada vez mais atrativa, provendo orientação, contexto, seleção de regiões, e um retorno imediato das alterações requisitadas, tendo a capacidade de unificar num único contexto as classes de visualização.

2.2.1 Princípio de *interfaces* visuais

Partindo da característica humana de que informações processadas visualmente são melhor absorvidas do que por qualquer outro sentido humano, foi elaborado o *Visual Information Seeking Mantra* [Shneiderman 1996], que consiste de um princípio básico para a confecção de *interfaces* visuais. O trabalho de *Shneiderman (1996)* tem sido bastante aceito pela comunidade de HCI (*Human-Computer Interaction*) e será a base desta Seção. Este guia enumera os seguintes passos na descoberta de conhecimento:

1. *Overview* (Visão geral): Tem-se uma visão geral de todo o conjunto de dados. Esta visualização possui um campo de visão controlável, que pode ser movimentado de acordo com o interesse do usuário;
2. *Zoom* (Foco): O usuário geralmente manifesta interesse por apenas uma região da coleção, e necessita, portanto, de ferramentas que o permitam controlar o foco e o fator de ampliação. Este método amplia os itens de interesse dentro do campo de visão. A ampliação pode ser realizada de forma homogênea, preservando a sensação da posição e do contexto do usuário;
3. *Filter* (Filtro): Retira itens irrelevantes à visualização. Isto permite que o usuário possa enfatizar rapidamente seus dados de interesse, eliminando o lixo visual;

¹Agrupamentos de dados que possuam alguma semelhança

²Buracos

4. *Details-on-demand* (Detalhes sob demanda): Exibe mais detalhes sobre os itens, apenas se forem requisitados. Uma vez que a coleção de dados foi lapidada, sobra mais espaço para visualizar mais informações sobre os dados de interesse.

Outras tarefas complementares também relevantes ao desenvolvimento de *interfaces* visuais são:

- *Relate* (Relação): Visualiza relacionamentos entre os itens, onde um usuário pode escolher um atributo, como o nome de um diretor de filmes, e fazer com que, na visualização dos filmes, apenas os dirigidos pelo suposto diretor sejam relacionados na lista;
- *History* (História): Armazena um histórico para permitir a volta de uma ação, seguida de sua repetição ou refinamento progressivo;
- *Extract* (Extração): Permite extração de sub-conjuntos dentro de um conjunto, por exemplo gravando-os para um arquivo ou distribuindo-os por meio de alguma mídia de comunicação.

Segundo o mesmo trabalho, em aplicações típicas de visualização, nota-se que o número de tarefas, na realidade, é muito maior (ex: técnicas de distorção, de omissão, de ampliação instantânea, de múltiplas janelas, entre outras [Ware 2000]). Porém, todas as outras são derivadas dessas 7 citadas anteriormente. Apesar de básicas, tais tarefas encontram-se em um alto nível de abstração, e são previstas para trabalharem com os seguintes tipos de dados:

- *1-dimensional* (1D): Consiste de tipos de dados lineares, códigos-fonte de programas, lista de nomes em ordem alfabética, entre outros. As operações mais comuns sobre esses dados incluem encontrar o número de itens da coleção, visualizar apenas os que possuam certa característica ou visualizar apenas um item, porém com todo detalhamento possível;
- *2-dimensional* (2D): Esta categoria engloba dados planares, como mapas geográficos e plantas de construções. Possíveis interações com esses dados são: encontrar itens adjacentes, verificar inclusão de um pelo outro, analisar caminhos entre os itens, além das tarefas básicas de contagem, filtragem, e detalhes sob demanda;
- *3-dimensional* (3D): Esta categoria lida com objetos do mundo real, como moléculas, corpo humano e construções. Tais entidades possuem volume e relacionamentos complexos. Com esses dados é possível observar adjacência entre os objetos, bem como o posicionamento em relação aos outros. A compreensão de posição e de orientação tornam-se mais difíceis quando imersos neste ambiente, adicionando complexidade à exploração;

- *Temporal*: Compreende dados unidimensionais, entretanto com evolução ao longo do tempo. Linhas de tempo são largamente utilizadas, e cruciais para gravações médicas, gerenciamento de projetos e estudos históricos. A principal característica destes dados é sua limitação temporal: possuem um tempo de início e de fim, e nesse intervalo pode haver sobreposição de itens. Aplicações sobre esses dados incluem tarefas como encontrar eventos antes, durante, ou depois de um dado instante ou intervalo de tempo. As interações básicas também são possíveis neste domínio;
- *Multi-dimensional*: Lida com dados de alta dimensionalidade, como bases de dados relacionais com muitos atributos. Tais dados, com n atributos, são representados em um espaço n -dimensional como pontos. Quanto maior a complexidade dos dados, maior o número de aplicações possíveis sobre os mesmos, como encontrar padrões, *clusters*, co-relações entre pares de atributos, *gaps*, contornos, entre outras. Graficamente, estes dados n -dimensionais são representados em um espaço tridimensional, entretanto a desorientação (causada por um ponto de vista ruim do usuário) e a oclusão (principalmente se objetos mais próximos ao usuário são maiores) são agravantes na visualização. Em vista desses problemas, uma das técnicas mais difundidas para este tipo de visualização são as Coordenadas Paralelas [Inselberg 1985];
- *Tree* (Árvore): Fazem parte desta categoria coleções de itens que, estruturalmente, se assemelhem a uma árvore, com cada um dos itens possuindo ligações para seus pais. Tais ligações entre pais e filhos podem possuir múltiplos atributos, e desta forma pode-se requisitar o número de níveis da árvore, bem como o número de filhos de um dado nó;
- *Network* (Rede): Para hierarquias mais complexas, ou relacionamentos desordenados entre vários itens, a estrutura de árvore não é adequada. Assim, essa nova categoria é capaz de suportar itens ligados a um número arbitrário de outros itens, e que constituem um dos vários tipos de rede: acíclica, simétrica, com ou sem raiz, direcionada ou não-direcionada. Uma representação típica de redes é um grafo. Além das tarefas básicas sobre nós e ligações, é possível calcular o caminho menos custoso entre dois nós, ou então possíveis caminhos Euleriano e Hamiltoniano.

Trabalhar com representações gráficas, ao invés de representações textuais, é uma forma de trabalho bem mais atrativa, uma vez que aproveita a habilidade sensorial humana em absorver melhor informações visuais. Assim, técnicas de marcação (por exemplo, sublinhar, piscar, enquadrar ou colorir) podem ser utilizadas para chamar a atenção do usuário em meio a milhares de itens.

A utilização de tais recursos visuais permitem uma rápida seleção dos dados de interesse, e o retorno das requisições é imediato. Desta forma, o olho, a mão e a mente

parecem trabalhar sincronizadamente, e o usuário tem a sensação de estar interagindo diretamente com os dados, o que diminui o cansaço e aumenta a produtividade.

A filtragem é outra etapa do processo de visualização, que pode revelar propriedades globais bem como ajudar usuários na resposta de questões específicas. Com a utilização de operadores lógicos (OR, AND e NOT), é possível construir consultas mais complexas, sendo, portanto, mais eficazes no processo exploratório em grandes bases de dados.

Algumas metáforas foram desenvolvidas para suportar este tipo de consulta, como *Venn diagrams* [Michard 1982], *decision tables* [Greene et al. 1990], *InfoCrystal* [Spoerri 1993] e *water flowing* [Young e Shneiderman 1993]. Esta última aplica uma metáfora de correnteza d'água da esquerda para a direita, através de uma rede de encanamentos e filtros, onde cada filtro permite a passagem de apenas certos documentos e os canos representam as combinações de AND e OR.

2.2.2 Aspectos Humanos

O processo de visualização pode ser considerado como a *interface* de um sistema de resolução de problemas [Ware 2000]. Tal sistema é composto por duas componentes: de um lado, o computador que suporta um grande acúmulo de informações, e do outro, o homem capaz de utilizar a visualização para desvendar novos relacionamentos estruturais, intrínsecos no volume de dados em estudo.

Segundo Ware (2000), no processo de conhecimento, a interação faz-se necessária no sentido de aperfeiçoar o processo cognitivo humano, e consiste de *feedback loops*³ Estes ciclos estão divididos em três níveis: ciclo de controle visual-manual, ciclo de navegação e refinamento visual e ciclo de resolução de problemas.

Ciclo de controle visual-manual

Trata-se do nível mais baixo da interação. Engloba a manipulação de dados, os quais são trabalhados usando as habilidades naturais do ser humano, como a interpretação visual de um estímulo e a movimentação da mão, num processo coordenado.

O tempo de reação (Equação 2.1) a uma escolha, por exemplo, é modelado pela lei de Hick-Hyman:

$$\text{Tempo de reação} = a + b \log_2 C \quad (2.1)$$

onde C é o número de escolhas e a e b são constantes empiricamente determinadas. A expressão $\log_2 C$ representa a quantidade de informações processadas pelo operador humano, expressa em *bits* de informação.

Outro fator humano relevante é o tempo de seleção e posicionamento em um espaço

³Ciclos de re-alimentação.

2D. O tempo estimado que um usuário leva utilizando um *mouse*, por exemplo, é dado pela lei de Fitt, segundo a Equação 2.2.

$$\text{Tempo de seleção} = a + b \log_2 \left(\frac{D}{W} + 1.0 \right) \quad (2.2)$$

onde D é a distância para o centro do alvo, W é o comprimento do alvo e a e b são constantes determinadas empiricamente, as quais variam de acordo com o dispositivo de entrada. O termo $\log_2 \left(\frac{D}{W} + 1.0 \right)$ é conhecido como o índice de dificuldade. Portanto, quanto maior a distância do ponteiro do *mouse* para o alvo, ou quanto menor for o tamanho deste alvo, maior será a dificuldade e o tempo gasto na tarefa.

Em contrapartida dos tempos de atraso, inerentes à natureza humana, há uma característica que reduz o tempo gasto com tarefas que se tornam rotineiras. A equação 2.3 mostra o aumento no desempenho em certa atividade conforme o número de vezes que é realizada.

$$\log(T_n) = C - \alpha \log(n) \quad (2.3)$$

onde $C = T_1$ é o tempo gasto para realizar a tarefa pela primeira vez e α é uma constante que representa o declive do tempo na curvatura de aprendizado.

Ciclo de navegação e refinamento visual

Este ciclo consiste do nível intermediário do processo interativo. Neste ciclo, o usuário se familiariza com o espaço visual que lhe é apresentado (Figura 2.1).

Figura 2.1: Este esquema representa o ciclo de navegação e exploração visual. À direita, do lado humano, tem-se um modelo espacial e lógico-cognitivo, onde o usuário compreende o espaço de dados e seu progresso nele. À esquerda, o computador atualiza e refina a visualização a partir dos dados mapeados para o modelo espacial (com os quais o usuário interage). Figura adaptada de [Ware 2000].

Uma maneira de facilitar o processo de navegação, fazendo uso das habilidades humanas de navegação e interpretação espacial, que já possuímos do mundo real - é a utilização de metáforas espaciais de navegação.

Metáforas visuais interativas são modelos cognitivos que alteram totalmente o *design* das *interfaces* dos sistemas para espaços de dados. Desta forma, o usuário lida com um modelo visual familiar, que o permite prever o comportamento do sistema.

Portanto, uma boa metáfora deve ser apropriada, modelar bem o sistema e ser facilmente absorvida pelo usuário. Em espaços virtuais 3D, são utilizadas basicamente quatro classes de metáforas:

- *World-in-hand*: Nesta metáfora, o usuário pode manipular o volume de dados como um todo, como se o possuísse entre as mãos. Trata-se de uma metáfora eficiente apenas na visualização de objetos relativamente compactos, pois não possui recursos sofisticados para navegação;
- *Eyeball-in-hand*: Neste ambiente quem muda de posição é o usuário, e não a cena sob observação. Assim, as coordenadas do objeto em cena são estáticas, enquanto na metáfora *world-in-hand* o objeto é constantemente reposicionado;
- *Walking*: Trata-se de uma das formas mais naturais de interação. Nesta metáfora o usuário tem a sensação de estar caminhando pelo volume de dados. O problema desta metáfora está em evitar que as barreiras do mundo real atrapalhem a interação. Dispositivos especiais podem apoiar este tipo de *interface*;
- *Flying*: Esta metáfora dá ao usuário a sensação de estar voando, facilitando a movimentação dentro do volume. Apesar da falta de realismo, esta metáfora é considerada a mais flexível e útil entre as apresentadas anteriormente [Ware 2000], pela sua liberdade de movimentos e fácil adequação.

Ciclo de resolução de problemas

No nível superior do processo de exploração, tem-se o ciclo de solução de problemas, no qual o usuário forma hipóteses sobre os dados, construindo um modelo conceitual sobre o problema.

A partir de então, uma estratégia para a resolução do problema é mentalizada. Isto leva a um ciclo (Figura 2.1), em que o usuário realiza ajustes na visualização, a fim de procurar mais informações ou alterar o modelo do computador, e este, por sua vez, realimenta a visualização com os novos resultados.

Neste ciclo, há uma intensa utilização da memória. O ser humano trabalha, aproximadamente, com três tipos de memória: sensorial, de curta duração e de longa duração.

A memória sensorial é um tipo de *buffer* que armazena temporariamente informações fornecidas pelos sentidos, que no caso da visualização, são as informações presentes na

retina. A partir do momento em que esta informação não ocupar mais a retina, ela também desaloca o *buffer*. Trata-se da *interface* entre a tela do computador e o sistema de processamento humano.

A memória de curta duração é um repositório temporário, que guarda informações por frações de segundo ou poucos minutos. Esta memória compreende a região ativa do cérebro: a atenção e a solução do problema são trabalhadas nesta região. Entretanto, esta memória possui capacidade limitada, e no processo de resolução do problema, parte dessa memória ainda pode ser ocupada por estímulos sensoriais ou pelas memórias de longa duração.

Já a memória de longa duração contém informação de toda uma vida. Geralmente, esta memória armazena eventos e episódios dos quais conseguimos nos lembrar, além de incluir habilidades motoras, tais como digitação e direção de um veículo. Habilidades de percepção, como o sistema visual, também fazem parte desta memória, e permitem uma rápida identificação de palavras e objetos.

Conhecer as habilidades e limitações do ser humano é um pré-requisito para a criação de novas ferramentas exploratórias e de visualização, como será visto nas Seções 2.3 e 2.4.

2.3 Visualização Científica

A Visualização Científica surgiu por volta de 1980, e trabalha com a visualização de dados espaciais associados a fenômenos científicos, como a ligação de moléculas estudadas em bioinformática. Portanto, esta área lida com objetos do mundo real, cujos dados são originalmente geométricos.

O desenvolvimento de uma técnica de visualização envolve vários passos. Inicialmente, tem-se a construção de um modelo empírico a partir dos dados, seguido pela adoção de um mecanismo que mapeie tal modelo para um objeto de visualização abstrato, como uma imagem ou um mapa de contornos. Finalmente, tem-se a *renderização* da imagem no dispositivo gráfico.

A visualização envolve a exploração, transformação e mapeamento de dados em objetos gráficos. Brodlie e outros (1992) desenvolveram uma taxonomia baseada em técnicas de visualização, as quais são classificadas de acordo com a dimensionalidade e o tipo dos dados.

A entidade criada a partir dos dados pode ser expressa como uma função $F(X)$, onde $X = (x_1, x_2, x_3, \dots, x_n)$ é o domínio e n é a dimensão. A função F pode ser um ponto, um escalar, um vetor ou um tensor. Desta forma, a classificação das técnicas é baseada no tipo de função e na dimensão do domínio. O tempo ainda pode ser um fator atuante sobre os dados.

Para dados pontuais, onde cada elemento é considerado um ponto em um espaço n -dimensional, a principal técnica utilizada são os *scatterplots*. Para dados com valores

escalares associados, são utilizadas várias técnicas, para diferentes domínios:

- 1D: Neste domínio, dados são geralmente obtidos de uma função $F(x_1)$ qualquer. Exemplos de técnicas são: gráficos de linha, histogramas e gráficos de barra;
- 2D: Nesta categoria, dados são gerados de funções bidimensionais do tipo $F(x_1, x_2)$. Linhas de contorno (Figura 2.2), visualização de superfícies (Figura 2.3) e campo de altura (Figura 2.4) constituem exemplos desta categoria;

Figura 2.2: Mapa topográfico exibindo forma e elevação de um terreno [http://interactive2.usgs.gov/learningweb/textonly/teachers/mapsshow_act4.htm].

- 3D: Funções do tipo $F(x_1, x_2, x_3)$ compreendem a extração de isosuperfícies e o *rendering* volumétrico direto (DVR). Exemplos de técnicas serão discutidos na Seção 2.3.1.

Dados vetoriais são geralmente representados pelo uso de setas ou outros objetos (cones e *glyphs*). Em aplicações que envolvem fluxo, os pontos de dados possuem vetores ao invés de simples escalares. Os campos vetoriais a serem visualizados podem possuir duas, três ou mais dimensões. Tanto no campo planar como no volumétrico, as técnicas mais utilizadas são as setas sobre os pontos e as *streamlines*⁴.

Para entidades tensoriais, também muito utilizadas em Dinâmica dos Flúidos, a visualização direta é mais complicada. Tensores de segunda ordem, no espaço 3D, são

⁴Linhas que descrevem a trajetória de partículas.

Figura 2.3: Exemplo de uma visualização de superfície de malha [Brodliet al. 1992].

Figura 2.4: Exemplo de Campo de Altura [Brodliet al. 1992]. Nesta visualização, 2 escalares são exibidos: um para altura, e outro para cor da superfície.

representados por matrizes quadradas 3×3 , ou seja, com 9 elementos. Para facilitar a representação, há a possibilidade de mapear tensores para escalares, entretanto haveria uma grande perda de informação.

2.3.1 Técnicas 3D para dados escalares

Com relação à classe de técnicas 3D para entidades escalares, que são as mais relevantes para este trabalho, pode-se destacar como técnicas clássicas:

- Técnicas de isosuperfície: Nesta classe, a metodologia geral consiste em aplicar um detector de superfícies sobre a amostra de dados, ajustando primitivas geométricas às superfícies detectadas. Este processo assemelha-se à detecção de linhas de contorno, citado para o espaço 2D. As superfícies são então *renderizadas* por uma técnica apropriada.

Um exemplo de método que utiliza extração de isosuperfícies é o *Marching Cubes*. Este processo classifica os 8 vértices do *voxel* como acima ou abaixo de um valor de fronteira. Desta forma, o algoritmo “marcha” por todos os cubos do espaço 3D, classificando seus vértices, e trocando o cubo por um conjunto apropriado de polígonos (Figura 2.5).

Figura 2.5: As 15 possíveis combinações de intersecção da superfície com um *voxel* [<http://www.exaflop.org/docs/marchcubes/ind.html>].

- Técnicas de DVR (*Direct Volumetric Rendering*): Como sugere o nome, estas técnicas realizam a *renderização* direta de conjuntos de dados 3D, sem a geração de primitivas geométricas intermediárias. Tais técnicas não lidam com superfícies, mas diretamente com os *voxels* do volume. Assim, o usuário tem a possibilidade de visualizar o volume de dados de várias formas e perspectivas; observar partes internas, externas ou até através de camadas.

A técnica mais difundida em DVR é o *ray casting*, que consiste no disparo de raios sobre a malha de dados, resultando em imagens de alta qualidade. No *ray casting* volumétrico (Figura 2.6), um raio continua o seu caminho pelo interior do volume em linha reta, até que a opacidade acumulada no caminho do raio atinja o máximo, ou seja, o valor 1.

2.4 Visualização de Informação

A Visualização de Informação consiste de uma área mais recente, que amadureceu em meados da década de 90. Ela se interessa pelo desenvolvimento de metáforas visuais bem como outras formas de exibição dos dados, que por não possuírem natureza geométrica, devem receber tratamento especial para facilitar a interpretação de dados. Como exemplo, tem-se a exploração de bases de documentos de texto, onde se buscam padrões e novas classificações que, de forma visual, são melhor identificadas.

Figura 2.6: Algoritmo de *ray casting* para *rendering* volumétrico, demonstrando a composição da opacidade, que é acumulada ao longo do raio. Figura adaptada de [<http://smohith.tripod.com/proj/vhp/thesis/node92.html>].

Ainda assim, encontrar formas de representação de grandes quantidades de dados multivariados (ou n -dimensionais), que sejam realmente eficazes no processo interativo de análise e interpretação humanos, permanece um desafio [Keim, Kriegel e Seidl 1994].

Segundo a abordagem adotada por Keim e Kriegel (1996) para o mapeamento dos dados em uma representação visual, as técnicas de exploração visual são categorizadas em 6 grupos:

- **Técnicas geométricas:** Possibilitam a visualização de transformações geométricas e projeções dos dados. Exemplos destas técnicas são: *Scatterplot-Matrices* [Andrews 1972, Cleveland 1993] e *Landscapes* [Wright 1999] (Figura 2.7), *Projection Pursuit Techniques* [Huber 1985], *Prosection Views* [Furnas e Buja 1994, Su et al. 1995] e *Hyperslice* [Wijk e Liere 1993] (Figura 2.8), e *Parallel Coordinates* (Coordenadas Paralelas) [Inselberg 1985, Inselberg e Dimsdale 1990] (Figura 2.9);
- **Técnicas iconográficas:** Nesta categoria, os atributos dos dados são visualizados como características de ícones. Utilizam esta técnica: *Chernoff-Faces* [Chernoff 1973, Tufte 1983] e *Stick Figures* [Pickett 1970, Pickett e Grinstein 1988] (Figura 2.10), *Shape Coding* [Beddow 1990] e *Color Icons* [Levkowitz 1991, Keim e Kriegel 1994] (Figura 2.11), e *TileBars* [Hearst 1995];
- **Técnicas orientadas a *pixel*:** Esta classe compreende ferramentas que representam cada atributo dos dados como um *pixel* colorido, em janelas separadas. Estas técnicas são sub-categorizadas em:
 - **Técnicas independentes de consulta:** Os atributos dos dados são diretamente mapeados para cor. Exemplos são: *Space-Filling Curves* (Figura 2.12) e *Recursive Pattern Technique* [Keim, Ankerst e Kriegel 1995] (Figura 2.13);

Figura 2.7: Exemplos de técnicas geométricas. À esquerda, *Scatterplot-Matrices* de n dimensões. À direita, uma representação dos dados como *Landscapes* em perspectiva [Keim 1997].

Figura 2.8: As técnicas geométricas *Prosection Views*, à esquerda, e *Hyperslices*, à direita. A técnica *Prosection Views* é apresentada com todas as projeções ortogonais, onde o resultado da seleção multidimensional é destacada com uma cor diferente. À direita, *Hyperslices* com n^2 fatias do conjunto n -dimensional [Keim 1997].

Figura 2.9: A técnica geométrica *Parallel-Coordinates* exibindo 15.000 itens de dados [Keim 1997].

Figura 2.10: Exemplos de Técnicas iconográficas. À esquerda, *Chernoff-Faces* representando dados multivariados, cujos atributos são mapeados para características dos rostos (formato do nariz, boca, olhos e a própria forma do rosto). À direita, uma imagem obtida com a técnica *Stick Figures*, a partir de dados com 5 dimensões da região dos Grandes Lagos, nos EUA [Keim 1997].

Figura 2.11: Exemplos de técnicas iconográficas. À esquerda, a representação foi obtida com o uso da técnica *Shape Coding*, sobre dados de observação da NASA sobre a Terra. À direita, *Color Icons* representando dados de 6 dimensões [Keim 1997].

Figura 2.12: Exemplos de técnicas orientadas a *pixel*, da classe *Space-Filling Curves*. À esquerda, foi utilizada a técnica de *Peano-Hilbert*, e à direita, a técnica de *Morton*, também conhecida como *Z-Curve* [Keim 1997].

Figura 2.13: A técnica *Recursive Pattern*, orientada a *pixel*. À esquerda, um ciclo linha a linha. À direita, um ciclo que alterna entre o início e o final das linhas [Keim 1997].

- Técnicas dependentes de consulta: Os dados são visualizados no contexto escolhido pelo usuário, que, através de consultas, configura a visualização de forma direcionada, de acordo com seus interesses. Técnicas que utilizam esta abordagem são: *Spiral Technique* [Keim e Kriegel 1994] (Figura 2.14), *Axes Technique* [Keim e Kriegel 1994] (Figura 2.15) e *Circle Segments Technique* [Keim 1997] (Figura 2.16);
- Técnicas hierárquicas: É realizado o particionamento da área visual em sub-áreas, as quais acomodarão os dados. Como exemplos, têm-se: *Dimensional Stacking* [LeBlanc, Ward e Wittels 1990] e *Worlds-within-Worlds* [Feiner e Beshers 1990] (Figura 2.17), *Treemaps* [Shneiderman 1992, Johnson 1993], *Cone Trees* [Robertson, Mackinlay e Card 1991] (Figura 2.18) e *InfoCube* [Rekimoto e Green 1993] (Figura 2.19);
- Técnicas baseadas em grafos: Lidam com grafos esparsos, almejando manter a clareza e a facilidade de navegação. Tem-se uma grande variedade de técnicas nesta categoria, como grafos 2D (Figura 2.20), 3D (Figura 2.21) e variantes destes;
- Técnicas híbridas: Uso integrado de múltiplas técnicas, em uma ou mais janelas, a fim de melhorar a expressividade das visualizações.

2.5 Considerações Finais

Mesmo que o computador tenha contribuído para a explosão de informações geradas e armazenadas, é a única ferramenta capaz de lidar com estas, realizando buscas, ordenação, filtragem, estruturação, e assim apresentando somente dados relevantes.

Neste contexto, a Exploração Visual de Dados configura uma área desafiante. As técnicas de exploração de dados complexos, como gráficos, imagens, sons, vídeos ou modelos geométricos têm se mostrado eficazes no tratamento de apenas alguns casos previstos. Entretanto, com o crescimento dos dados e de seus atributos, tornam-se incompletas, o que fomenta o desenvolvimento de novas técnicas exploratórias e de caráter genérico. O próximo capítulo descreve técnicas interativas e sistemas que as utilizam, a fim de propiciar ao usuário um melhor entendimento dos dados.

Figura 2.14: A técnica orientada a *pixel* e dependente de consulta, *Spiral Technique*, ordena os *pixels* de acordo com dada característica, organizando-os a partir do centro [Keim 1997].

Figura 2.15: Similar à *Spiral Technique*, a técnica orientada a *pixel* e dependente de consulta, *Axes Technique*, também aproxima os *pixels* semelhantes, entretanto são consideradas 2 características. Da comparação dessas características, são obtidos 4 quadrantes, os quais acomodarão os *pixels* correspondentes [Keim 1997].

Figura 2.16: *Circle Segments Technique*, orientada a *pixel* e dependente de consulta, representando dados de tempo [Keim 1997].

Figura 2.17: Técnicas hierárquicas aplicadas a conjuntos de dados multivariados. À esquerda, a técnica *Dimensional Stacking* é utilizada para visualizar dados de extração de óleo mineral, com longitude e latitude mapeadas para os eixos x e y externos, e a classificação do mineral e a profundidade são mapeadas para os x e y internos. À direita, a técnica *Worlds-within-Worlds* exibe uma função de 6 dimensões, entretanto particionada em sub-espacos 3D [Keim 1997].

Figura 2.18: Técnicas hierárquicas aplicadas a sistemas de arquivo. À esquerda, a visualização de um sistema com aproximadamente 1000 arquivos com a técnica *Treemap*. À direita, a arquitetura de um sistema é visualizada, utilizando a técnica *Conetree* [Keim 1997].

Figura 2.19: A técnica hierárquica *InfoCube* exibe a estrutura de um sistema de arquivos [Keim 1997].

Figura 2.20: À esquerda, um grafo 2D ortogonal, e à direita, um grafo 2D simetrizado [Keim 1997].

Figura 2.21: Grafo 3D otimizado para verificação de *clusters* [Keim 1997].

Sistemas Exploratórios

3.1 Considerações iniciais

Assim como foi discutido na Seção 2.2.1, a necessidade da interação como um recurso de auxílio à visualização é imprescindível. Através da interação, é possível melhorar a eficiência do sistema visual humano e, desta forma, desvendar a semântica das informações mais rapidamente.

Para a Visualização Computacional, há várias formas de refinamento visual e interação possíveis, algumas das quais serão descritas a seguir, conforme a relevância para este trabalho.

3.2 Técnicas interativas

A possibilidade de se aproveitar o conhecimento intuitivo do usuário sobre este tipo de manipulação é o grande desafio da área de HCI (*Human-Computer Interaction*¹). Em um sistema interativo, características como confortabilidade, navegabilidade, intuitividade e tempo de resposta são consideradas durante o desenvolvimento de novas técnicas, com o intuito de fornecer um melhor suporte ao processo de interação homem-máquina.

3.2.1 Navegação

Existem, na literatura, inúmeras metáforas de interação e navegação, as quais são baseadas em interação direta utilizando a mão, a extensão do braço, o apontamento por raio e outros dispositivos de Realidade Virtual [Bowman e Hodges 1999].

¹Interação Homem-Computador.

De acordo com Shneiderman (1996), o princípio de exploração visual deve seguir o seguinte roteiro: *Overview first, Zoom and Filter, then Details-on-demand*². Assim, os 2 primeiros passos dizem respeito à visualização e, conseqüentemente, à navegação no espaço em que o objeto está imerso.

Em várias bibliotecas gráficas, o *viewpoint*³ é descrito analogamente como uma câmera, assumindo integralmente esta metáfora. Desta forma, a região de visão e a visibilidade dos dados é definida por parâmetros de uma câmera real, como localização, direção, *zoom*⁴, foco, entre outros.

Especificamente, para um espaço 3D, o movimento da câmera é definido por 6 parâmetros:

- Posição: A localização da câmera no decorrer dos eixos x , y e z define sua posição no espaço 3D;
- Angulação: Os movimentos *pitch*⁵, *yaw*⁶ e *roll*⁷ definem a orientação da câmera, e sua movimentação ao redor de um objeto.

O benefício primário da navegação, ou seja, da manipulação do *viewpoint*, é oferecer perspectivas alternativas, facilitando a compreensão da visualização. E para que a navegação tenha maior eficiência no controle do *viewpoint*, metáforas (Seção 2.2.2) de movimentação e de formas interativas devem ser aplicadas. Assim, com o mapeamento de controles para tarefas operacionais, a utilização do sistema torna-se espontânea e o aprendizado do usuário é direcionado aos resultados da aplicação, e não em como usá-la.

Dispositivos especiais de Realidade Virtual criam metáforas de interação, ou seja, formas não-convencionais de interagir em uma visualização. Na Figura 3.1 são exibidos alguns dispositivos: *Ware Bat*, *SpaceMaster*, *SpaceOrb*, *CrystalEyes 3* e *Head Tracker*, *Hand Tracker* e *Phantom Desktop* (caneta 3D).

Em [Elcacho, Dingel e Klein 2001] é apresentado um novo conceito para navegação em ambientes virtuais. Enquanto uma variedade de metáforas de navegação são propostas para Realidade Virtual imersiva, cenários de *desktop* são tipicamente baseados em navegação de *mouse*, utilizando uma metáfora de *flying* (voar), *walking* (caminhar) ou *driving* (dirigir). No entanto, guiar por todos os 6 graus de liberdade não é trivial, mesmo para um usuário treinado. Elcacho (2001) descreve um conceito de navegação centrado no objeto, o qual permite explorar um ambiente desconhecido, partindo diretamente para objetos desejados pelo usuário, observando-os de perto. O usuário especifica o objeto pelo seu nome (por exemplo, "mesa"), um aspecto ("frente") e um nível de distância

²Visualizar tudo, ampliar e filtrar, e então exibir detalhes sob demanda.

³Ponto de visão

⁴Fator de ampliação.

⁵Rotacionar em torno do eixo esquerda-direita.

⁶Rotacionar em torno do eixo vertical.

⁷Rotacionar em torno do eixo frente-trás.

Figura 3.1: (a) *Ware Bat*, (b) *SpaceMaster*, (c) *SpaceOrb* (Figuras obtidas em [Hughes 2005]), (d) *CrystalEyes3* e *Head Tracker*, (e) *Hand Tracker* (Figuras obtidas em [<http://www.est-kl.de/hardware/tracking/intersense/isminitrax.html>]), (f) *Phantom Desktop* (Figura obtida em [<http://www.sensable.com/haptic-phantom-desktop.htm>]).

(“perto”), e a visão é imediatamente transferida para o ponto de observação desejado. Assim, com um espaço “discretizado”, comandos de voz são mais eficazes, pelo baixo número de alternativas.

O conceito de *viewpoints* centrados nos objetos faz referência à metáfora de *world-in-hand* (Seção 2.2.2), que permite ao usuário selecionar os objetos de interesse, indo diretamente a eles e então utilizando posicionamento relativo para se mover em torno deles. Todos os *viewpoints* são calculados dinamicamente durante a navegação, não sendo necessário pré-defini-los.

Cada objeto na cena é associado com um conjunto de *viewpoints* para uma orientação, por exemplo: atrás, frente, esquerda, direita, abaixo e acima. E para cada uma destas orientações, um nível de distância do objeto pode ser especificado: perto, longe e intermediário (Figura 3.2).

Desta forma, cada objeto provê um total de 18 *viewpoints* centrados no objeto. Para navegação da cena, o observador pode facilmente ir de uma posição a outra, apenas informando o endereço do próximo objeto a ser visualizado.

Figura 3.2: Modelo de *viewpoint* centrado no objeto [Elcacho, Dingel e Klein 2001].

3.2.2 Seleção e corte

A técnica de seleção (ou *brushing*) define o grau de interesse do usuário em determinada região da imagem, ou seja, tem a função de filtragem (3º passo do *Visualization Seeking Mantra*) para futuras consultas e interações. Em visões coordenadas (Seções 3.3.2 e 3.3.3), a seleção de objetos em uma visão pode influenciar sua exibição nas outras visões, que podem exibir dados e/ou dimensões diferentes para os mesmos dados.

Já o recurso do corte possibilita realizar *details-on-demand* - último passo sugerido por Shneiderman (1996) - e obter 2 resultados:

1. Separação e isolamento das regiões interna e externa da superfície, sendo possível explorá-las individualmente;
2. Visualização de entidades interiores ao objeto (quando houver).

Em NPR (*Non-Photo Realistic*), onde o principal objetivo do corte é a visualização do interior de um objeto opaco, ao invés de simplesmente transparecer a casca que envolve a entidade interna, partes da superfície são removidas. Isto ocorre de 2 formas, como mostrado na Figura 3.3:

1. *Cutout*: A remoção de uma área da superfície deixa um aspecto de corte, como um serrilhado. Esta técnica consiste na utilização de planos cortantes, e é melhor

aproveitada em aplicações que necessitam da distinção entre muitos objetos internos e externos, produzindo um alto nível de abstração (devido aos apelos visuais);

2. *Breakaway*: Ao contrário do *cutout*, não é um corte grosseiro; o corte é apenas suficiente para visualizar o objeto desejado. Para poucas entidades internas, este método é recomendado, pois mantém o aspecto inicial da visualização.

Figura 3.3: Comparação entre um *cutout* (à esquerda) e um *breakaway* (à direita) [Diepstraten, Weiskopf e Ertl 2003].

Nota-se que objetos internos não são atingidos pelo corte; apenas os objetos externos, mais próximos do observador. Desta forma, as regras para realizar um *cutout* são:

- *R1* - Objetos internos e externos devem ser distinguidos uns dos outros;
- *R2* - A geometria cortada é representada pela intersecção de 2 ou mais planos;
- *R3* - O corte é localizado no próprio eixo principal do objeto externo, ou próximo a ele;
- *R4* - Um mecanismo de *jittering* aplica um comportamento grosseiro ao corte, dando a impressão de um serrilhado;
- *R5* - Há a possibilidade de criar paredes em objetos que não representam explicitamente suas paredes internas, a fim de corrigir a iluminação interna durante a visualização.

Por outro lado, para realizar um *breakaway*, deve-se respeitar as seguintes premissas:

- *R1* - Utiliza-se o mesmo requisito R1 de *cutouts*, para a distinção de objetos internos e externos;
- *R6* - O corte deve ser realizado através de um simples buraco no objeto externo;
- *R7* - Todos os objetos interiores devem ser visíveis a partir de qualquer ângulo.

No entanto, as regras aplicadas variam de aplicação para aplicação, dependendo da ordem de relevância na visualização: o objeto externo, o objeto interno ou a região cortada.

3.2.3 Técnicas $F+C$

Na área de Visualização Computacional, é comum que aplicações tentem direcionar a atenção do usuário para certos objetos ou regiões do gráfico. Isto alerta o usuário de um problema, uma característica ou lhe mostra objetos de interesse, satisfazendo uma consulta. Entretanto, é interessante permitir ao usuário não apenas visualizar os resultados de sua consulta, mas também visualizar e entender prontamente a informação destacada no contexto de outra informação. Este tipo de visualização é definido como *Focus+Context*, o qual provê a visualização de ambos - foco e contexto - fornecendo ao usuário maior poder de interpretação para seus dados de interesse.

Uma visualização impõe uma estrutura gráfica - um mapeamento dos dados para a tela - dos dados que os usuários devem assimilar. No entanto, a forma com que a visualização é organizada pode não ser a mais confortável e intuitiva ao usuário. Tornar-se, então, necessário mudar este mapeamento, conforme o interesse do usuário em obter mais informações; mas geralmente não há espaço na tela para exibir todos os dados com detalhes suficientes. Métodos baseados em $F+C$ (*Focus+Context*) tornam isto possível, mostrando mais detalhes dos objetos sob observação, ao mesmo tempo que dá aos usuários o senso de quão ampliado, transladado ou deformado o objeto está.

Em [Kosara, Miksch e Hauser 2002] são descritos métodos e aplicações de $F+C$, que podem ser divididos em 3 grupos:

- *Métodos espaciais*: Este grupo engloba técnicas que realizam modificações estritamente espaciais, seja no espaço visual ou seja na geometria dos dados, a fim de exibir foco e contexto. Pode-se citar 4 sub-classes de técnicas desse grupo:
 - Técnicas de *rapid zooming* [Parker, Franck e Ware 1998]: Um grande terreno de informações é disponibilizado, mesmo que apenas uma parte dele esteja visível em um dado instante. O usuário tem a possibilidade de ter uma visão ampliada dos seus pontos de interesse, bem como alternar para uma visão geral do contexto. O intuito dessa classe de técnicas é possibilitar essa alternância de visões de forma rápida, mantendo a característica $F+C$;
 - Janelas múltiplas [Card, Pirolli e Mackinlay 1994]: A técnica consiste em possuir várias janelas exibindo a estrutura dos objetos, porém em diferentes graus de abstração. Por exemplo, pontos de dados organizados por datas são eficientemente representados por esta técnica: em uma janela constam os dias, em outra as semanas, em outra os meses, e assim por diante. Desta maneira, é possível especificar o nível de organização desejado;
 - Técnicas de omissão [Bartram et al. 1994]: Estas técnicas escondem partes das estruturas até que sejam estritamente necessárias. Basicamente, isto é obtido agrupando uma complexa estrutura em um simples objeto gráfico;

- Técnicas de distorção [Bartram 1997]: Compreende o grupo mais distinto dos métodos de $F+C$. Com estas técnicas, dados de interesse ganham mais espaço e são centralizados, enquanto o contexto é comprimido e movido para a periferia. Exemplos destes métodos são *Fish-eye Views* [Furnas 1986], *Hyperbolic trees* [Kreuseler, Lopez e Schumann 2000], *Document lens* [Robertson e Mackinlay 1993], *Stretchable rubber sheets* [Sarkar et al. 1993], e outros métodos [Leung e Apperley 1994];
- *Métodos dimensionais*: Nesta classe de técnicas, outra dimensão é apresentada ao usuário, ou seja, a região sob investigação é exibida de forma não-convencional, com mais atributos ou um conjunto diferente destes. Assim como uma lupa, o usuário observa uma região de interesse com outra perspectiva. Desta forma, estes métodos não exibem mais objetos, mas permitem que mais dimensões, ou dimensões diferentes sejam exibidas. Exemplos são *Magic lenses* [Stone, Fishkin e Bier 1994], *Tool glasses* [Bier et al. 1993] e outras;
- *Métodos de marcação*: Em uma visualização existente, objetos que se enquadram em certos critérios são enfatizados através da atribuição de marcas visuais. Desta forma eles se destacam aos olhos do usuário, sem que seja necessário esconder o contexto. Um exemplo simples é usar saturação de cor e brilho [Lokuge e Ishizaki 1995]. Outro exemplo é um sistema que permite que 26 camadas de informação geográfica sejam exibidas ao mesmo tempo [Colby e Scholl 1991]. Usuários podem mover o foco entre estas camadas simplesmente alterando o nível de borramento e transparência de cada uma delas.

3.2.4 Scatterplots

Os *scatterplots*⁸ representam uma das técnicas mais simples de exibição de dados, por isso são largamente utilizados tanto em Visualização de Informação como em Visualização Científica.

No plano bidimensional (2D) este método consiste em exibir, num gráfico, um ponto para cada item de dado da base. A posição de cada ponto depende de quais atributos as duas dimensões representam, as quais definem os eixos de coordenadas do plano. Desta forma, um *scatterplot* pode exibir dados abstratos de forma muito eficiente, além de também exibir imagens, bastando para isso alternar os tipos de atributos referenciados pelos eixos: coordenadas ou outras características.

Caso um conjunto de dados possua mais dimensões, apenas duas delas poderão ser exibidas ao mesmo tempo, resultando em uma projeção dos pontos no plano. Um recurso alternativo, que possibilita a exibição de mais dimensões simultaneamente, é a utilização

⁸Nuvem de pontos.

de *glyphs* [Ward 2003] que, por exemplo, representam outras dimensões através de cor, tamanho e forma. Portanto, são naturalmente aplicáveis a dados genéricos.

Têm sido propostas várias extensões para os *2D scatterplots*, como *prosection views* [Furnas e Buja 1994], que permite ao usuário selecionar um intervalo de valores em um ou mais eixos não exibidos. Outra extensão é a *scatterplot matrix* [Cleveland 1985], que consiste de um conjunto de vários *scatterplots* que exibem todas as combinações possíveis de atributos da base de dados em duas dimensões. Os *plots* são colocados em uma matriz, de forma que todos que possuam mesmo eixo *y* estão em uma linha, e os que possuam o mesmo eixo *x* estão na mesma coluna. Ambas as técnicas foram ilustradas na Seção 2.4.

Partindo para os *3D scatterplots*, vários sistemas têm sido desenvolvidos [Becker 1997, Reina, G. e Ertl 2004], e tentam reduzir falhas e limitações inerentes à projeção 2D, no que se refere ao *overplotting* de pontos⁹ e em facilitar o processo cognitivo humano na busca de *clusters*, que são mais nebulosos no espaço 2D.

Os *3D scatterplots*, juntamente com os 2D, configuram uma tentativa de integração da Visualização Científica com a Visualização de Informação, que ocorre em 2 sistemas descritos a seguir: Voxelplo (Seção 3.3.2) e SimVis (Seção 3.3.3). Tal integração é obtida pela combinação de múltiplas visões do mesmo conjunto de dados, com interação simultânea e coordenada entre as visões.

3.3 Sistemas interativos

Descritas algumas das principais técnicas de interação, as quais são genéricas ao tipo de aplicação, nas Seções seguintes serão apresentados sistemas voltados para uma ou ambas as áreas da Visualização Computacional, que contribuíram para este trabalho.

3.3.1 DSVol

O *DSVol* (*Distributed Sound for Volumes*) [Salvador e Minghim 2003] consiste de um sistema de visualização com uso de sonificação¹⁰, originado no sistema *SVol* [Salvador, Minghim e Pacheco 1998], e que atualmente permite a escolha de servidores para realizar a visualização e a sonificação via XML.

O *DSVol* utiliza como *renderer* a biblioteca de visualização VTK (*The Visualization Toolkit*) [Schroeder, Martin e Lorensen 2002], que também permite interações. Já a sonificação é realizada pelo sintetizador STK (*The Synthesis Toolkit*) [Cook e Scavone 1999].

Durante o desenvolvimento do *DSVol*, foi proposto um modelo básico de interação para Visualização Científica. Em um primeiro instante, as técnicas de interação pré-existent

⁹Sobreposição de pontos

¹⁰Uso de som como um recurso para representar dados [Salvador e Minghim 2003]

de Visualização Científica foram categorizadas e um modelo básico de interação 3D foi criado. Em seguida, foram definidos ferramentas e esquemas de sonificação para as classes de interação previamente criadas. A Figura 3.4 exibe o modelo básico de interação.

Figura 3.4: Diagrama hierárquico das classes de interações para Visualização Científica [Salvador 2003].

Em Manipulação do Ambiente, as tarefas são relacionadas com o ponto de vista do usuário. Assim, esta classe define a maneira como o usuário visualiza o volume de dados, e se movimenta durante a interação no volume. São apresentadas quatro metáforas de navegação: *walking*, *flying*, *eyeball-in-hand* e *world-in-hand*, previamente explicadas na Seção 2.2.2.

A classe Manipulação dos Dados contempla operações realizadas diretamente sobre o volume de dados, ou seja, manipulando-o (definindo subvolumes, traçando caminhos ou realizando cortes). Três tipos básicos de interação são definidos para esta classe:

- Animação: O usuário pré-configura parâmetros que o interessam, e então assiste passivamente a ocorrência de uma animação;
- Manipulação de parâmetros: O usuário também pré-configura parâmetros de interesse, porém visualiza os resultados logo após a alteração (em tempo real) ou após um dado comando;
- Sondagem de Dados: O usuário pode analisar estruturas ou padrões de partes do volume de dados apresentado, utilizando-se de vários objetos de sondagem dentro do volume. Assim, pode-se observar pontos, planos ou subvolumes, e de acordo com a dimensão da sonda utilizada, esta é classificada como 0D, 1D, 2D ou 3D.

O DSvol possui 7 modos de interação com o volume de dados:

1. *Progressive Display*: Esta ferramenta de interação é um exemplo da classe de ferramentas de dimensão animada. Fatias dos dados, perpendiculares a qualquer um dos eixos de coordenadas, são apresentadas como uma animação, uma a uma;
2. *Grid Display*: Representando a classe de sondagem pontual, realiza sondagem 0D, permitindo ao usuário analisar valores ou características sobre o volume de dados como um todo;
3. *2D Scan*: Esta outra ferramenta representa a classe de rastreamento 2D, com sondagem 1D. Os usuários escolhem uma fatia específica do volume, e navegam pelo plano com uma pequena sonda quadrada;
4. *Plane Scan*: Representa a classe de ferramentas de sonda de superfícies. Nesta ferramenta 2D, a sonda é um plano que pode ser movido dentro do volume de dados;
5. *3D Lens*: Representa a classe de lentes 3D, em que os usuários definem qual tipo de características ou valores necessitam analisar apenas em partes da imagem;
6. *Sound Probe*: A ferramenta de interação *Sound Probe* representa a classe de sonda volumétrica. Uma sonda geométrica é apresentada dentro do volume, que pode ser redimensionada e movida pelo usuário;
7. *2D Spider Cursor*: O *Spider Cursor* representa a classe de sondagem de vizinhança 2D, que consiste de um *cursor* em forma de aranha, que navega sobre uma superfície 2D, apresentando informações relacionadas ao ponto central e seus vizinhos.

O *2D Spider Cursor* é a base deste trabalho, o qual possui atualmente uma versão estendida, e outra versão re-estruturada, com novos recursos (Seção 4.3.2).

3.3.2 Voxelplot

O sistema Voxelplot [Kosara, Sahling e Hauser 2004] é uma implementação de *3D scatter-plots* que utiliza RTVR (*Real-Time Volume Rendering*) [Mroz e Hauser 2001], e consiste de uma biblioteca Java para *rendering* volumétrico direto (DVR). Ao invés de *pixels*, cada item de dado é mapeado para um *voxel*¹¹.

O Voxelplot utiliza uma técnica de *depth of field* [Kosara, Miksch e Hauser 2002], classificada como *Focus+Context*, que borra o contexto ou as informações irrelevantes à visualização, ao mesmo tempo que torna nítida a visão dos dados ou objetos de interesse. Este método tem sido utilizado em fotografia há um longo tempo, onde a profundidade de campo determina qual o alcance da profundidade para fotografar com boa definição (Figura 3.5).

¹¹Trata-se do menor elemento em uma visualização 3D.

Figura 3.5: Exemplo de *depth of field* [Kosara, Miksch e Hauser 2002], utilizado em fotografia.

Desta forma, ao invés de borrar partes da imagem baseado em sua distância do observador, utiliza-se a relevância da informação como critério para definição da imagem.

Neste sistema, a integração entre Visualização Científica e Visualização de Informação é obtida pela combinação de múltiplas visões do mesmo conjunto de dados, mantendo interação entre as visões. Desta maneira, é possível obter visualizações com atributos abstratos, ao mesmo tempo que se tem a disposição física dos pontos no espaço, promovendo uma ligação entre o espaço de características e o espaço real, onde se encontra o objeto. Isto é obtido através do uso de um *software* de DVR, e combinando-o com métodos de interação de Visualização de Informação, como *linking* e *brushing*.

Em *rendering* volumétrico, a profundidade é indicada pelo grau de transparência. Esta é a única forma que o RTVR utiliza para indicar a profundidade, o que é uma limitação no que se refere a aplicações de Visualização de Informação. Em Visualização Científica, onde os dados a serem visualizados possuem forma geométrica conhecida, isto não acarreta em um problema, pois os objetos são altamente intuitivos, mesmo em 3D. Entretanto, em Visualização de Informação, é uma tarefa mais complicada inferir formas, ou mesmo distância e profundidade nas visualizações. Portanto, neste caso, a ferramenta necessitaria de mais apelos visuais para representar profundidade, além do grau de transparência.

Este sistema permite ao usuário o uso de dois tipos de interação: o primeiro consiste dos tipos básicos, como rotação e *zoom*, que são realizados implicitamente pela biblioteca RTVR; o segundo consiste de mudanças nas dimensões mapeadas, no próprio mapeamento, *brushing*¹², etc., que são realizados em um painel de controle adicional.

¹²Seleção de partes dos dados [Piringer, Kosara e Hauser 2004]

Como visto na Figura 3.6, o Voxelplot exibe simultaneamente quatro *3D scatterplots*, fornecendo ao usuário a opção de aumentar uma das visões.

Figura 3.6: Interface do Voxelplot. A visão selecionada tem a borda vermelha [Kosara, Sahling e Hauser 2004].

Figura 3.7: À esquerda, um exemplo de *range brush*, e à direita, um *beam brush* selecionando parte de um cérebro [Kosara, Sahling e Hauser 2004].

A técnica de *brushing* realiza marcação de pontos, destacando-os. Ela define um grau de interesse (DOI) nos pontos, afetando todas as visões. Como um produto dessa

característica, pode-se observar a co-relação entre a visualização do volume e a visualização das informações, comparando os *scatterplots*. Seu objetivo é atribuir um valor escalar, representando seu DOI, para cada item de dado.

No Voxelpplot, duas técnicas de *brushing* foram implementadas, como pode ser visto na Figura 3.7:

- *Range Brush*: O usuário especifica um intervalo de valores referentes a um determinado atributo, selecionando os valores mínimo e máximo com uma barra de rolagem. Trabalhando com *brushes* multi-dimensionais, em vários eixos, é possível criar uma seleção *hyper-box*, como um cubo;
- *Beam Brush*: Este tipo de *brush* marca todos os pontos que estão contidos em um cilindro que corta ortogonalmente o plano de visão, e cujo raio o usuário pode alterar.

Trabalhando em 3D, *brushes* simples não são suficientes, então técnicas alternativas como *composite brushes*, visualização *focus+context* e *smooth brushing* devem ser usadas.

- *Composite brushes*: Para realizar *brushes* de formas mais complexas, a combinação de vários *brushes* é uma operação interessante. Isto consiste principalmente na combinação de três operadores lógicos: OR, AND e SUB. Por exemplo, após aplicar o primeiro *brush*, aplica-se o segundo sobre o primeiro. Então define-se o que acontece com a área na intersecção, e fora dela;
- Visualização de *focus+context*: Dados marcados com o *brush* são o foco da visualização. Entretanto, a vizinhança destes dados de interesse também são relevantes para a análise do contexto, facilitando a percepção e interpretação das informações;
- *Smooth brushing*: Este tipo de *brushing* possibilita uma maior variação do nível de interesse (DOI); existe a possibilidade de marcar os pontos com outros escalares que não sejam os binários 0 e 1. Assim, no Voxelpplot é possível definir várias funções de interesse, utilizando também os operadores lógicos.

O resultado da técnica de *brushing* é a definição do DOI, que é realizado sobre os dados originais, ao invés dos *voxels* projetados. Desta forma, os respectivos *voxels* nas outras visões - projetados sobre diferentes dimensões - serão remapeados e então marcados.

3.3.3 SimVis

Outro sistema que implementa os *3D scatterplots* é o SimVis [Piringer, Kosara e Hauser 2004], que também utiliza múltiplas visões interligadas (Figura 3.8) para a visualização de dados de alta dimensionalidade, especialmente de simulações de Dinâmica de Fluidos Computacional (CFD). Este sistema emprega variação

de tamanho, cor e *halos*¹³ sobre os pontos para indicar profundidade nas visualizações 3D, e também utiliza histogramas como uma técnica para destacar a distribuição dos pontos e sua densidade.

Figura 3.8: Interface do SimVis [Piringer, Kosara e Hauser 2004].

Os recursos usados para representar profundidade são:

- **Tamanho:** Como o usuário está acostumado à projeção perspectiva, na qual objetos mais distantes parecem menores do que os mais próximos ao observador, o tamanho se torna naturalmente um indicador de profundidade. Desta forma a percepção de profundidade melhora consideravelmente. Por exemplo, diminui-se o tamanho do ponto à medida que a distância para o observador aumenta;
- **Cor:** Cores claras e quentes, por exemplo, vermelho e amarelo, são indicadas para indicar proximidade com o observador, enquanto cores escuras e frias, como azul e cinza, são intuitivamente identificadas como mais longínquas. Este fenômeno é chamado *chromo-stereoscopy* [Toutin 1997]. Desta forma, alterar a matiz e o contraste dos elementos da cena que estão distantes do observador é uma técnica que melhora a noção de profundidade;
- **Halos:** Trata-se de uma técnica conhecida da pintura, que intuitivamente indica a presença de discontinuidades. Ou seja, elementos contínuos - numa projeção 2D - porém com profundidades diferentes são destacados na projeção

¹³Auréola

[Interrante e Grosch 1997]. Esta técnica desenha círculos semi-transparentes em volta de cada ponto, com muito menos brilho do que a cor original do ponto, realçando-o com um efeito de auréola.

Figura 3.9: À esquerda, nenhum apelo visual para profundidade é utilizado. No meio, a profundidade é apenas indicada pelo tamanho do ponto e a utilização de *halos*, facilitando a discriminação entre pontos. À direita, todos os recursos citados são empregados na visualização: tamanho do ponto, cor e *halos* [Piringer, Kosara e Hauser 2004].

Neste sistema, a densidade de pontos é também representada: cada ponto exibido é ortogonalmente projetado para cada um dos três planos formados pelos eixos x , y e z , e os histogramas 2D são amostrados em grades uniformes, com células de mesmo tamanho, de acordo com a resolução desejada.

Figura 3.10: À esquerda, a densidade não é claramente interpretada. No meio, a representação baseada em textura apenas usa opacidade para representar a densidade. À direita, a densidade é ilustrada pela opacidade juntamente com as barras volumétricas, fornecendo a melhor visualização dessa característica [Piringer, Kosara e Hauser 2004].

De forma semelhante ao Voxelplo, o SimVis apresenta quatro visões do mesmo conjunto de dados: três delas são 2D, cada uma exibindo projeções ortogonais das combinações dos eixos x , y e z . E a última visão 3D é obtida a partir da manipulação das

outras visões. O SimVis admite a interligação com visões externas, sendo possível importar *brushes* definidos em outras visualizações, como uma simulação de Mecânica dos Flúidos Computacional. Desta forma, dados científicos também podem ser marcados, em visões externas, e mapeados para o espaço de características dos *scatterplots*.

Assim, o uso de visões múltiplas e interligadas é a idéia-chave para trabalhar com dados de alta dimensionalidade, e a interação é realizada em 2D, sendo seus resultados exibidos em 3D. *Brushes* simples são desenhados ao arrastar o *mouse* em qualquer uma das visões 2D, onde as duas dimensões mapeadas por esta visão são delimitadas pelo usuário, enquanto a terceira dimensão é inicialmente definida pelo tamanho do foco no espaço.

3.3.4 *Parallel Sets*

A ferramenta *Parallel Sets* [Kosara, Bendix e Hauser 2006] trata de outra abordagem para visualização interativa de dados categóricos. Consiste da fusão de 2 técnicas bem difundidas: Coordenadas Paralelas [Keim 1997], tratando a visualização de todas as dimensões independentemente, e a exibição das freqüências das categorias, ao invés de exibí-las uma a uma separadamente.

Dessa forma, as intersecções dos pontos são substituídas por conjunto de caixas que representam as categorias (Figura 3.11). Estas caixas são ajustadas de acordo com as freqüências das categorias correspondentes e são inicialmente ordenadas de acordo com metadados. Usando a informação de freqüência significa utilizar uma agregação de um grande conjunto categórico de dados, reduzindo o montante de dados a serem exibidos, e fornecendo uma imagem dos dados que mais se aproxima de uma representação de grandes conjuntos de dados.

Interação

Há várias formas de interação disponíveis no *Parallel Sets*: seleção e destacamento, consulta interativa, filtragem, e reordenação de dimensões e categorias.

A exploração de dados interativa começa com uma investigação não-direcionada das variáveis de dados disponíveis. O usuário escolhe as variáveis de dados mais interessantes, adiciona-as ao painel de visualização, e explora seus relacionamentos.

Uma característica fundamental é a composição de dimensões (Figura 3.12). O uso desta característica serve para reduzir a dimensionalidade da visualização, construindo categorizações mais práticas e significativas.

Figura 3.11: Em (a), as categorias são representadas por pontos nos eixos em Coordenadas Paralelas. Em (b), a técnica de *Parallel sets* mostram as frequências das categorias e suas relações. Em (c) a exibição de mosaicos permitem uma visão comprimida dos dados (as partes incubadas representam o mesmo sub-conjunto) [Kosara, Bendix e Hauser 2006].

Figura 3.12: Um exemplo de composição de dimensões: o usuário está interessado em uma classificação de 4 categorias: $C_{a1} \wedge C_{b1}$, $C_{a2} \wedge C_{b1}$ e $C_{a3} \wedge C_{b1}$. A interação em (a) ilustra uma sequência de seleções, a qual resulta em uma re-categorização personalizada pelo usuário em (b) [Kosara, Bendix e Hauser 2006].

3.3.5 InfoSky

O InfoSky [Kienreich et al. 2003] consiste de um sistema que permite aos seus usuários explorar grandes coleções de documentos, organizados hierarquicamente. Utiliza uma metáfora de galáxia, exibindo documentos como estrelas, e formando agrupamentos com formas distintas (Figura 3.13).

Figura 3.13: *InfoSky Visual Explorer*, exibindo a galáxia com várias partições e documentos, representados por estrelas [Kienreich et al. 2003].

Interação

A exploração do espaço, possibilitada pelo sistema, se resume a uma combinação de navegação e busca. Ou seja, é possível selecionar um ou vários documentos, aproximar uma região, selecionar os pais de um determinado documento, e variações destas funcionalidades.

3.3.6 InterRing

O *InterRing* [Yang, Ward e Rundensteiner 2002] é baseado em técnicas RSF (*Radial space-filling*¹⁴), que dá suporte a um sistema de visualização de dados hierárquicos e provê distorções multifocais, reconfiguração hierárquica interativa, e seleção manual e semi-automática.

Interação

Técnicas RSF (Figura 3.14) têm dificuldades para exibir grandes hierarquias, pois formam-se fatias muito difíceis de se distinguir. Algumas técnicas permitem uma visualização *Focus + Context*, em que duas janelas separadas são mantidas para representar uma visão geral e uma detalhada da árvore hierárquica.

O *InterRing* fornece várias formas de interagir com a visualização, como distorção,

¹⁴Preenchimento do espaço de forma radial

Figura 3.14: Exemplo de técnica RSF, com seus elementos rotulados [Yang, Ward e Rundensteiner 2002].

seleção, reconfiguração interativa da estrutura da hierarquia, *drill-down*¹⁵ e *roll-up*¹⁶, rotação e ampliação.

3.4 Considerações finais

A confecção de ferramentas de interação, e mesmo de visualização, não é uma tarefa trivial. Fatores humanos, como os discutidos na Seção 2.2.2, devem ser levados em consideração, a fim de contornar as limitações humanas e tirar proveito de características especiais, como o aprendizado de novas habilidades e melhor uso da memória quando se utilizam os apelos visuais adequados.

Considerando que o processo de exploração e entendimento do ser humano é comum à Visualização Computacional, e que cada vez mais possui-se, no mesmo ambiente, dados de natureza geométrica/espacial juntamente com dados abstratos, os sistemas passam a possuir mais flexibilidade e maior disponibilidade de técnicas integradas.

O próximo capítulo apresenta o *Super-Spider*, uma ferramenta de interação que permite codificar informações adicionais durante o processo interativo, mapeadas para a forma do cursor.

¹⁵Exibir detalhes

¹⁶Esconder detalhes

Super Spider

4.1 Considerações iniciais

Dentre as ferramentas apresentadas na Seção 3.3.1, a ferramenta de sondagem superficial - *Super Spider* - será o foco deste trabalho, a qual estendeu os recursos da ferramenta denominada *Spider Cursor* [Minghim et al. 2005].

Atualmente, existem 2 expansões do Cursor Aranha::

1. *Spider Cursor*, utilizando a linguagem C++ e as bibliotecas VTK e STK (Apêndice B.2);
2. *Super Spider*, em contínuo desenvolvimento, utilizando a linguagem Java e a biblioteca Java 3D (Apêndice B.3).

Embora as 2 versões da ferramenta compartilhem o mesmo componente de sondagem - o Cursor Aranha - neste momento tratam-se de ferramentas complementares, que possuem funcionalidades que a outra versão não possui. Tais recursos e diferenças serão discutidos a seguir, e mais detalhes podem ser obtidos no Apêndice B.

4.2 Descrição geral

O *Spider Cursor* [Minghim et al. 2005] consiste de uma ferramenta de interação capaz de navegar sobre malhas. Atualmente, esta técnica funciona apenas sobre triângulos, os quais devem ser definidos no arquivo de dados VTK [Schroeder, Martin e Lorensen 2002]. Esta ferramenta é classificada como sondagem bidimensional, adquirindo informações da

Figura 4.1: Exemplos do traçado do *Spider Cursor*, representando informações textuais [Salvador 2003]

investigação através do traçado de caminhos sobre uma superfície do conjunto de dados sob análise, e da investigação da vizinhança dos pontos.

Potencialmente, esta ferramenta pode funcionar sobre qualquer tipo de malha, desde que a estrutura de dados seja adaptada para suportar outros modelos de dados. No entanto, os mapas utilizados são gerados através da triangulação de Delaunay, a fim de se obter superfícies com triângulos menos obtusos, os quais são melhores para visualização. Portanto, atualmente, a estrutura de dados implementada suporta apenas malhas triangulares.

Nesse processo gráfico, o usuário movimenta um *cursor* sobre o volume de dados, e a cada vértice do modelo apontado pelo mouse, a ferramenta exibe todos os caminhos que podem ser escolhidos a partir daquele ponto. Graficamente, tem-se a impressão da formação de teias de aranha a cada ponto escolhido ou apontado, daí o nome *Spider Cursor* para a ferramenta (Figura 4.1).

Cada vértice da malha possui escalares associados, além de nomes descritivos ou rótulos. O formato de dados VTK permite que vários campos escalares sejam relacionados a um único vértice, exceto dados textuais, que são armazenados em arquivos separados.

Além do arquivo de dados VTK, contendo o modelo geométrico, e os arquivos com textos descritivos, a ferramenta também suporta a matriz de similaridades do modelo. Dado que a malha triangular - gerada em R^2 - é obtida a partir do espaço original dos dados, ou seja, de R^n , é possível associar estes 2 espaços distintos através da matriz de similaridades, que consiste no cálculo da distância entre os vértices. Desta forma, o *Spider Cursor* realiza a visualização da malha 2D, e exibe informações ou padrões apenas detectáveis no espaço original dos dados.

Portanto, com tantos atributos atrelados aos vértices, a visualização da malha pode ocorrer pela combinação de várias técnicas: iso-superfícies, iso-linhas, cores, *height fields*, som, variações das super-quádricas, *glyphs*, dimensão fractal, entre outras.

Assim, esta ferramenta exploratória fornece várias informações sobre os pontos sele-

cionados e sua vizinhança, através do mapeamento de atributos para várias características visuais. Uma extensão sonora para o *Super Spider* faz parte de outro trabalho de mestrado em andamento.

4.3 Funcionalidades

Mesmo compartilhando a mesma ferramenta-base de sondagem, o *Spider Cursor* e o *Super Spider* possuem recursos de mapeamento complementares, ou seja, um sistema possui funcionalidades que o outro não possui, e vice-versa. Isto reflete a diferença entre as bibliotecas gráficas - VTK e Java 3D - que fornecem suporte a mais ou menos primitivas e atributos, como mapeamento de cores, sombreamento, interação, etc.

Nas Seções seguintes, serão apresentadas as funcionalidades de cada um dos sistemas, bem como sua interface gráfica.

4.3.1 *Spider Cursor*

Derivado do DSVol [Salvador e Minghim 2003], a primeira versão do *Spider Cursor* foi desenvolvida no *Microsoft Visual C++ 6.0*, sem interface gráfica. A partir de então, o *Spider Cursor* começou a ser aperfeiçoado, com a adição da GUI¹ (Figura 4.2) e de novos recursos, tornando a ferramenta mais apropriada para lidar com uma quantidade maior de aplicações.

A utilização da ferramenta inicia ao se carregar um arquivo de dados no formato VTK com células triangulares. Adicionalmente, pode-se também carregar um arquivo de rótulos - arquivo NAMES - que descreve os dados. A partir de então, são disponibilizados recursos visuais e sonoros para interação com os dados.

Realizado o carregamento dos dados, faz-se a configuração sonora da interação. O usuário pode escolher um entre 12 tipos de instrumentos musicais, a variação de frequência, e o campo escalar - do arquivo VTK - para o qual será mapeado o som.

Terminada a configuração sonora, o usuário decide as características do mapeamento visual, podendo aplicar várias técnicas sobre os dados: iso-superfícies com transparência, *scatterplot*, iso-linhas (coloridas, brancas ou pretas) e filtro de conectividade, que exhibe apenas dados que estejam em uma faixa de valor pré-definida.

A técnica de *scatterplots* possui um recurso-extra, que estende a representação de pontos para *glyphs*². Assim, com este recurso, pode-se mapear um campo escalar para variadas formas, como cubos, esferas, cilindros, discos e cones.

Também é possível realizar o mapeamento da altura - coordenada z dos pontos - e da distribuição das cores. Para a altura, é realizado um pré-processamento, e calculado

¹Graphical User Interface

²Ícones.

Figura 4.2: Interface gráfica do *Spider Cursor*.

um fator de escala para este campo escalar, a fim de manter uma boa razão de aspecto da visualização. A cor, por sua vez, possibilita o usuário a escolher uma entre 4 tabelas de cores, que se adequam a diversos tipos de dados (como discutido em [Tutida 1998]). Pode-se definir os valores de fronteira para cada faixa de cor.

Iniciada a interação, o usuário pode definir a cor de fundo, e com um mouse de 3 botões, definir translação dos dados, ângulo de rotação e fator de ampliação. Para iniciar a utilização do *Spider Cursor*, o usuário deve pressionar a tecla "U", que é disponibilizada pelo próprio VTK para *User Interactions*³. Com a sondagem do cursor, o rótulo atribuído a cada ponto é exibido na parte superior da visualização, sendo possível realizar a busca de um ponto por seu rótulo. O recurso do corte é iniciado com cliques do *mouse*, em pontos sequenciais, até retornar ao primeiro. Outros detalhes do *Spider Cursor* são fornecidos no Apêndice B.2.

³Interações do usuário.

4.3.2 *Super Spider*

O *Super Spider* também possui uma GUI (Figura 4.3) que possibilita a ativação de várias técnicas de visualização, inclusive com o novo recurso de sondagem utilizando super-quádricas e cálculo da dimensão fractal. A forma de utilização da interface é semelhante à versão original do *Spider Cursor*, no entanto, há diferença de recursos disponíveis.

Figura 4.3: Interface gráfica do *Super Spider*.

De forma semelhante, o usuário deve carregar um arquivo no formato VTK, e adicionalmente carregar arquivos de rótulos dos dados. Por convenção, foi definido para o *Super Spider* que o arquivo NAMES contém o nome do arquivo que os dados estão atrelados, e o arquivo LABELS contém trechos de conteúdo dos arquivos.

A parte de sonificação consiste de um outro trabalho de mestrado, que se encontra em andamento, e portanto não foi implementada nesta versão inicial do *Super Spider*.

Para realizar o mapeamento visual dos dados, são disponibilizadas as seguintes técnicas: iso-superfícies com transparência, *scatterplot*, iso-linhas (coloridas, brancas ou pretas) e *wireframe*⁴.

⁴Contorno dos polígonos.

O mapeamento da altura, assim como da cor dos dados e da cor de fundo, também é associada a qualquer campo escalar presente no arquivo de dados VTK.

Assim como o *Spider Cursor*, a interação com o *mouse* é semelhante ao transladar os dados, rotacioná-los ou ampliá-los. Com o cursor ativado, os dados têm seus respectivos rótulos - NAMES e LABEL - exibidos, permitindo ao usuário buscar qualquer documento por seu nome ou conteúdo.

O recurso do corte, aprimorado nesta versão (Seção 4.3.3), é iniciado com a tecla “E” e desativado pela tecla “D”. O usuário pode reiniciar o traçado da linha de corte com a tecla “R”, e também desfazer as últimas ações com “U”.

As super-quádricas, por sua vez, consistem de um cursor auto-representativo, ou seja, capaz de exibir simultaneamente diversas características dos dados. Desta forma, através de atributos como forma, tamanho, cor e rotação das super-quádricas, consegue representar até 9 atributos ou dimensões de dados (ver atributos de forma no Apêndice A).

Para a utilização no *Super Spider*, há 4 super-quádricas disponíveis para exploração de dados: super-elipsóide, super-hiperbolóide de 1 lâmina, super-hiperbolóide de 2 lâminas e super-toróide. Todas as variáveis podem ser mapeadas para um campo escalar qualquer, ou ter seus valores fixos, configurados manualmente pelo usuário através da barra de deslizamento ao lado das caixas de seleção. Cada uma das super-quádricas possui de 7 até 9 variáveis configuráveis:

- *e1* e *e1* configuram a forma da super-quádrica, que adota uma série de formas, como pode ser observado no Apêndice A;
- *a1*, *a2* e *a3* determinam a escala de cada um dos eixos *x*, *y* e *z*, respectivamente, e no caso do super-toróide, *a4* e *a5* determinam os raios do modelo;
- *Rotation speed* varia a velocidade de rotação da super-quádrica de acordo com o escalar especificado;
- *Color* define qual atributo controla a variação da escala de cores.

O usuário ainda possui a opção de analisar a dimensão fractal do conjunto de dados ou a taxa de acerto da projeção⁵. Para isso, é necessário carregar a matriz de distâncias entre os pontos da malha no espaço original dos dados.

Definido qual cálculo será realizado, define-se a forma de se obter a vizinhança: KNN (*K-Nearest Neighbors*⁶) ou por variação de raio. Definidos os parâmetros, a ferramenta realiza o pré-processamento dos valores e cria um campo escalar para o cálculo realizado. Assim, o usuário pode mapear a dimensão fractal para qualquer atributo visual, inclusive para as super-quádricas.

⁵Esta taxa responde à seguinte pergunta: os vizinhos mais próximos de um ponto no espaço original são os mesmos da projeção? Quantos são?

⁶K-vizinhos mais próximos

A seguir destacamos a forma como o corte da malha foi realizado, por se tratar de um algoritmo com particularidades dignas de menção.

4.3.3 Corte no *Super Spider*

Na Seção 3.2.2, foram discutidos alguns aspectos de como realizar um corte em um objeto. No caso do *Super Spider*, utiliza-se o mecanismo de corte *breakaway*. O motivo desta decisão é o fato de as células da superfície possuírem valor semântico imprescindível à sondagem, portanto as demais células da superfície não podem ser descartadas da visualização.

O objeto cortado sempre será o mais externo, desde que esteja dentro do campo de visão do observador. No entanto, o corte enfatiza a região cortada, pois essa possui mais valor que a simples exibição do conteúdo. Portanto, o requisito R7 não é atendido no *Super Spider*.

Figura 4.4: Em (a) a região destacada contém grande incidência do termo “aftosa”, a qual é marcada para corte. Em (b), a região destacada é explorada isoladamente, com o super-elipsóide buscando notícias sobre “economia” e “governo”, no contexto da “aftosa”. A notícia que consta no cabeçalho é: *Produtores do Estado participam de reuniões em Brasília*.

Para a realização do corte, o seguinte procedimento foi adotado:

1. Através de uma *hotkey*⁷, o corte é ativado;
2. Com o movimento do *mouse* sobre as arestas, o *path*⁸ vai sendo desenhado;

⁷tecla de atalho

⁸caminho

3. Assim que o *path* é fechado, ou seja, retorna à origem, forma-se um polígono;
4. Determina-se quais pontos de dados são internos e quais são externos ao polígono;
5. Ambos os polígonos (interno e externo) são gravados em disco, com seus respectivos arquivos de NAMES e LABEL.

Com o corte, deparamos com um problema simples, mas crucial: identificar se uma região selecionada é interna ou externa. A abordagem anterior do corte do *Spider Cursor* assumia que a região cortada é sempre menor - em número de pontos - que a região externa. Este critério é facilmente violável, portanto, ineficaz. Há várias formas de se determinar, em um plano 2D, se um ponto está interno, na borda, ou externo ao polígono [Bourke 1987]. As mais difundidas na literatura são:

Solução 1

Considere um polígono de n vértices (x_i, y_i) , onde i varia de 0 a $n - 1$. Assume-se que o último vértice (x_n, y_n) é o mesmo que o primeiro (x_0, y_0) , ou seja, o polígono é fechado. Para determinar o estado de um ponto (x_p, y_p) considere um raio horizontal partindo de (x_p, y_p) para a direita. Se o número de vezes que este raio intersecta as arestas do polígono for par, então o ponto está externo ao polígono. Entretanto se o número de intersecções for ímpar, o ponto (x_p, y_p) é interno ao polígono. A Figura 4.5 exemplifica a resolução deste problema, com a emissão de raios a partir de alguns pontos.

Figura 4.5: Solução 1: Lançamento de raios para determinar se um ponto é interno ou externo. Figura adaptada de [Bourke 1987].

Há casos especiais que merecem atenção, onde arestas ou vértices do polígono são justamente sobrepostos pelo raio de sondagem que sai de (x_p, y_p) . As situações possíveis são expostas na Figura 4.6.

As linhas grossas da Figura 4.6 não são consideradas como intersecções válidas, apenas as linhas finas. Ignorando os casos em que uma aresta está totalmente sobreposta ao raio, e

Figura 4.6: Solução 1: Casos especiais de intersecção de arestas com o raio. Figura adaptada de [Bourke 1987].

que uma aresta termina justamente sobre o raio, assegura que os pontos finais são contados apenas uma vez. Assim, são contabilizadas apenas os casos normais de intersecção (em 1 ponto) e no início da aresta.

Este algoritmo também funciona para polígonos com buracos internos, como ilustrado na Figura 4.7.

Figura 4.7: Solução 1: Polígono com uma região interna vazia: o algoritmo também funciona para este caso. Figura adaptada de [Bourke 1987].

Solução 2

Outra solução possível é determinar a convexidade do polígono. Se um polígono é convexo, assume-se que durante o seu traçado, foi percorrido um *path* a partir do primeiro vértice. Um ponto está no interior desse polígono se ele estiver sempre do mesmo lado das arestas que compõem o *path*.

No entanto, nota-se que esta condição apenas poderá ser verdadeira se o polígono for convexo.

Dado um segmento de linha $P_0(x_0, y_0)$ e $P_1(x_1, y_1)$, outro ponto $P(x, y)$ tem o seguinte relacionamento com o segmento de linha:

$$(y - y_0)(x_1 - x_0) - (x - x_0)(y_1 - y_0) \quad (4.1)$$

se o resultado for menor que zero, então P está à direita do segmento de linha; se for maior que zero, está à esquerda, mas se for igual a zero, está sobre o próprio segmento.

Figura 4.8: Solução 2: Exemplo de análise de convexidade de polígonos. Em um dos casos, os pontos internos estão à esquerda da orientação dos polígonos, no outro caso, à direita. Figura adaptada de [Bourke 1987].

Solução 3

A última abordagem encontrada, e que possui aplicação tanto em 2D como em 3D, consiste na soma dos ângulos obtidos entre o ponto de teste e cada par de pontos que formam as arestas do polígono.

Em 2D, basta verificar se esta soma é 2π : em caso positivo, o ponto está no interior do polígono; se a soma for zero, o ponto é externo. Este algoritmo também se aplica a polígonos com buracos, dado que o polígono é definido como um caminho de arestas coincidentes, havendo continuidade para dentro e para fora do polígono.

Solução para 3D

A priori, todas as abordagens citadas são aplicáveis para o campo 3D, ou mesmo R^N . No entanto, no caso 3D, primeiro verifica-se se o ponto está no mesmo plano que o polígono. Caso não esteja, ele está fora do polígono. Em caso positivo, os procedimentos a serem abordados são os mesmos apresentados para os casos 2D.

Implementação do *Super Spider*

Para o *Super Spider*, foi adotada a Solução 3, devido ao baixo número de iterações - igual ao número de segmentos de reta do *path* de corte - e devido à forma do polígono de corte, o qual não deve possuir cruzamentos.

Figura 4.9: Solução 3: Exemplo em 3D da soma dos ângulos de um ponto q com os vértices do polígono. Neste caso, a somatória se iguala a 2π , pois o ponto é interno. Figura adaptada de [Bourke 1987].

4.3.4 Navegação e manipulação

A interação 3D (manipulação e corte) oferece um grande potencial para aplicações médicas, mecânica dos fluídos, CAD/CAM, e outras. Devido à rápida evolução da indústria de *hardware*, particularmente de *hardware* gráfico, uma maior resolução pôde ser alcançada, assim como cores mais reais, e vídeos com maior taxa de atualização. Todos esses fatores contribuem para o processo cognitivo humano, que permite ao usuário interagir de forma mais confortável com o sistema, diminuindo a fadiga e facilitando a interpretação dos dados.

Foram observados alguns princípios desejáveis em uma interação [Krüger et al. 2005, Bade, Ritter e Preim 2005], os quais foram considerados para este trabalho:

1. Ações similares devem provocar reações similares, pois o comportamento deve ser previsível e reprodutível;
2. A direção da rotação deve coincidir com a direção do movimento do dispositivo 2D, o que está em conformidade com o sistema humano de resposta de estímulos e o cinestésico;
3. A interação deve ser transitiva, isto é, se o dispositivo de movimento aponta uma posição A, depois a posição B e finalmente a posição C, obtem-se o mesmo resultado movimentando o dispositivo de A para C;
4. A taxa de *control-to-display* [CHapanis e Kinkade 1972] deve ser ajustada à

aplicação. Assim, obtém-se a melhor combinação entre velocidade de navegação e precisão do cursor durante a interação.

5. Confortabilidade da interação;
6. Intuitividade dos comandos.

4.4 Detalhes de desenvolvimento

A extensão do *Spider Cursor*, denominada *Super Spider*, foi desenvolvida no *NetBeans 4.1* e totalmente reprogramada em Java, inclusive sobre outra biblioteca gráfica que utiliza OpenGL ou DirectX: o Java 3D. Desta forma, para utilizar o *Super Spider*, basta ter os *plugins* Java e Java 3D instalados, e executar a ferramenta a partir do navegador de Internet.

Trata-se de uma funcionalidade valiosa para ferramentas de Visualização Computacional - tanto Visualização Científica quanto Visualização de Informação - que ganham mais utilidade por causa da portabilidade. Portanto, é possível explorar e manipular volumes de dados a partir de qualquer navegador Web, sob qualquer Sistema Operacional.

O Java 3D consiste, portanto, de uma biblioteca gráfica de alto nível que implementa várias primitivas de *ray tracing* (Seção 4.4.1) de forma eficiente e possibilita que outras formas geométricas sejam implementadas, como as super-quádricas (Seção 4.4.2).

4.4.1 Primitivas de *ray tracing*

Para se entender a implementação da ferramenta Super-Quádrica é importante entender o tratamento de primitivas de *ray tracing* em Java 3D.

Uma primitiva consiste de um modelo, para o qual uma rotina de intersecção de raios em uma forma já tenha sido implementada. A partir da composição das primitivas, formas mais complexas podem ser construídas, sendo que vários *ray tracers* já implementam vários tipos de primitivas:

- *Plano*: Um plano infinito é um objeto simples que faz intersecção com um raio. Pode representar objetos de fronteira como chão, céu, mares, ou mesmo uma parede infinita, usando esta primitiva;
- *Polígono*: Para encontrar a intersecção de um raio com um polígono, deve-se primeiramente encontrar a intersecção do raio com o plano infinito sobre o qual o polígono se encontra. Então, deve-se certificar se a intersecção ocorre do lado interno ou externo: trata-se de uma operação gráfica bidimensional;

- *Caixas*: Uma caixa é composta por 6 polígonos e pode ser desenhada pelo *ray tracer* como tal. Entretanto, a intersecção com uma caixa de eixos alinhados pode ser otimizada. Qualquer caixa pode ter seus eixos alinhados por transformações apropriadas. Pode-se escrever uma rotina para cruzar um raio arbitrário com uma caixa de eixos alinhados, e então transformar o raio em consideração da mesma forma que a caixa seria transformada. Este tipo de idéia generaliza de modo organizado o conceito de especificar qualquer objeto: define-se o objeto num sistema de coordenadas conveniente, e então aplica-se as transformações para todo o objeto, para posicioná-lo no ponto apropriado no sistema de coordenadas global;
- *Esferas*: A esfera é a forma finita mais simples com a qual um raio pode cruzar. Praticamente qualquer *ray tracer* inclui as esferas como primitivas. Alterar a escala dos eixos de uma esfera, de forma arbitrária, pode produzir um elipsóide: uma esfera comprimida ou esticada. *Ray tracers* geralmente não apresentam elipsóides como primitivas, mas fornecem mecanismos para obtê-los a partir das esferas;
- *Cilindros*: Os cilindros são obtidos através do cruzamento dos raios com um cilindro longo apropriado, e depois certifica-se onde ocorre a intersecção. Se a intersecção ocorre ao longo do cilindro, esta é armazenada. Caso contrário, a intersecção é ignorada. Desta forma, o *ray tracer* utilizado não desenhara o cilindro com as tampas, o que exige cálculos extras;
- *Cones*: Cones são similares a cilindros. Da mesma forma que os cilindros infinitos, há uma definição matemática simples de um cone infinito, o qual facilita a escrita de um algoritmo de intersecção de raios sobre cones;
- *Discos*: O disco não é uma primitiva usual, mas é necessário nos *ray tracers* que implementam cones e cilindros sem tampas. Este objeto é manipulado de forma semelhante a um polígono. A rotina que checa se a intersecção com o plano ocorre interna ou externamente ao disco é mais simples do que sua equivalente para os polígonos. O disco foi posicionado de forma a refletir luz, o que ilustra como a reflexão especular se comporta em uma superfície chata. As imagens da esfera, do cone e do toro ilustram como a reflexão especular varia com as superfícies curvas;
- *Toros*: Toróides são razoavelmente raros na vida real, sendo mais raros de se encontrar como primitivas de *ray tracers*. Representar roscas e câmaras de pneus são 2 aplicações para esta forma, que por possuírem uma definição matemática razoável, são fáceis de serem implementados.

Figura 4.10: (a) polígono; (b) cubo; (c) esfera; (d) cilindro; (e) cone; (f) disco; (g) toro.

4.4.2 Cônicas, quádricas e super-quádricas

As primitivas de *ray tracing* possuem definições matemáticas relativamente simples. Justamente esta característica as torna atrativas para o uso em visualizações: a definição matemática simplificada permite a implementação de um código de intersecção de raios eficiente e robusto, motivo pelo qual foram adicionadas como primitivas. Partindo deste conceito, torna-se interessante a busca por outras formas ou modelos de representação matemática simplificada.

Assim, foram investigadas outras formas geométricas não-convencionais, que possibilitassem registrar as variações dos dados através de variações das suas formas:

- *Cônicas*: Trata-se de uma curva 2D, cujos principais representantes são a elipse (do qual o círculo é um caso especial), a hipérbole e a parábola;
- *Quádricas*: Consistem dos correspondentes 3D das cônicas. São classificadas em

4 categorias: elipsóides, hiperbolóides de 1 lâmina, hiperbolóides de 2 lâminas e toróides. Fazem parte desta família de superfícies paramétricas: esferas, cones e cilindros;

- *Super-quádricas*: As super-quádricas (Figura 4.11) consistem da extensão das quádricas, e possuem uma maior variedade de formas e parâmetros configuráveis. Pertencem a uma família de objetos paramétricos 3D, assim classificadas: super-elipsóides, super-hiperbolóides de 1 lâmina, super-hiperbolóides de 2 lâminas e super-toróides.

Figura 4.11: Em a), a notícia que consta é “Dirceu vai recorrer ao STF para reaver mandato”. A busca é realizada sobre os termos “crimes” e “advogado”, onde a forma arredondada do cursor indica a presença de um “ministro” ou “deputado” no texto. Em b), o super-elipsóide foi mapeado para o termo “dinheiro”. Neste caso, há alta incidência do termo na notícia “União fiscalizará a movimentação financeira de autoridades”.

As equações paramétricas, assim como as tabelas das variações de forma de cada um destes objetos geométricos, são encontradas no Apêndice A.

4.4.3 Dimensão Fractal

Fractais são formas geométricas auto-semelhantes, que são geradas a partir de funções matemáticas ou transformações geométricas. Curiosamente, por possuírem certos atributos como repetição de padrões, são úteis para descrever fenômenos naturais complexos, como linhas costeiras e folhas [Mandelbrot 1982].

Entretanto, tal complexidade também é observada em outras áreas. Por exemplo, muitas bases de dados possuem uma grande quantidade de características e atributos que são altamente co-relacionados. Tais atributos contribuem para o aumento da complexidade de qualquer tratamento que venha a ser aplicado ao conjunto de dados, como

indexação espacial, estimativa de densidade e descoberta de conhecimento através de um processo de *data mining*⁹.

Além disso, estas co-relações podem revelar padrões relevantes e informações úteis até então ocultas nos dados. As tarefas do processo de *visual data mining*¹⁰, como redução de dimensionalidade, classificação, *clustering* (agrupamento), aprendizagem, visualização e exploração auxiliam o usuário na descoberta desses padrões. Tais padrões são indicadores da forma em que os pontos de dados estão distribuídos no espaço.

Assim, o estudo de fractais configura um recurso importante na investigação dos dados, bem como no estudo de sua complexidade e da base como um todo.

Os fractais possuem 3 características básicas que os definem: dimensão fractal, complexidade infinita [Backes e Bruno 2005] e auto-semelhança em diferentes níveis de escala.

A dimensão fractal consiste da característica mais relevante dos fractais, e diferentemente da Geometria Euclidiana, em que a dimensão representa a dimensionalidade do espaço onde o objeto está inserido, a dimensão fractal representa o nível de irregularidade dos fractais.

Outra característica particular da dimensão fractal é a possibilidade de assumir valores fracionários, em contrapartida à Geometria Euclidiana. Dessa forma, a dimensão fractal informa o nível de ocupação do espaço pela forma, e não o espaço onde o fractal está inserido. Quanto maior a irregularidade do fractal, maior sua dimensão fractal.

Para o cálculo da dimensão fractal, há vários métodos disponíveis na literatura. Em [Backes e Bruno 2005], há um estudo comparativo de várias técnicas baseadas na *Dimensão de Hausdorff*, como Massa-Raio, análise da Intersecção Acumulativa, *Box Counting* (contagem de caixas), *Dividers* ou *Compass* e *Bouligand-Minkowski*.

Método de Massa-Raio

Este método consiste em sobrepor círculos sobre uma forma e contabilizar a quantidade de pontos desta forma presentes no interior do círculo traçado. Esse método se baseia na relação:

$$M_r(A) = \mu \cdot r^D \quad (4.2)$$

onde $M_r(A)$ é o número de pontos de uma forma A presentes em um círculo de raio r , sendo μ uma constante.

Portanto, de acordo com a relação dada, tem-se a dimensão fractal (D):

$$D = \lim_{r \rightarrow 0} \frac{\ln(M_r(A))}{\ln(r)} \quad (4.3)$$

⁹Processo de varrer grandes bases de dados à procura de padrões desconhecidos.

¹⁰Assim como o processo de *data mining*, visa a descoberta de padrões. No entanto, as técnicas utilizadas utilizam apenas visuais.

Quanto ao tamanho do raio, é interessante que o r_{max} (raio máximo) utilizado na expansão dos círculos seja igual ao raio de giro da imagem.

Frente aos outros métodos, para a exploração dos dados no espaço original, ou seja, utilizando a matriz de distâncias, o método mais apropriado é o Massa-Raio. Este método foi escolhido baseado nos seguintes critérios:

1. A base de dados, projetada no espaço R^n , não configura um contorno - condição ruim para este método - e sim uma nuvem de pontos esparsos;
2. A matriz de similaridade do espaço original - espaço R^n - já está ou pode ser facilmente calculada. Assim, é trivial o cálculo de massa das esferas;
3. O centro de massa é facilmente calculado através de uma matriz de similaridades (distâncias) entre os vértices do modelo.

Para o traçado dos círculos, há 2 opções:

1. Traçar vários círculos, em diferentes pontos aleatórios do espaço, e calcular a massa média dos mesmos;
2. Traçar apenas um círculo, preferencialmente sobre o centro de massa do volume de dados.

Devido à necessidade de uma resposta rápida no cálculo da dimensão fractal, a opção 2 foi escolhida, com o círculo sendo traçado no centro de massa de acordo com a matriz de distâncias. Por se tratar de uma medida global, cabe ao usuário definir o tamanho do raio sobre a região investigada, sendo o tamanho máximo limitado ao raio de giro r_{max} .

Para este tipo de aplicação, em que não se avalia a complexidade geométrica de imagens, e sim a similaridade de documentos, quanto maior a dimensão fractal, maior a homogeneidade do conjunto. Isso se deve à relação *massa/raio* para o cálculo da dimensão fractal. Para um mesmo raio, quanto mais pontos estiverem inseridos na hiper-esfera, maior é a semelhança entre eles, maior é a massa global acumulada, e portanto, maior é a dimensão fractal.

Dado que quanto maior a distância entre 2 pontos no hiper-espaço, maior a diferença entre eles, dados com pouca similaridade tendem a ficar isolados do conjunto, obtendo baixa massa acumulada em sua hiper-esfera, e assim, baixa dimensão fractal.

4.5 Considerações finais

O *Spider Cursor* (VTK e C++) e o *Super Spider* (Java e Java3D) são sistemas de exploração visual de dados representados por campos escalares associados a malhas de triângulos. A forma de apresentação é independente da área de aplicação, ou seja, é

possível investigar mapas de dados advindos de Visualização de Informação bem como objetos de Visualização Científica.

Técnicas difundidas como isosuperfícies, isolinhas, *height fields* e *scatterplots* são úteis para uma boa ferramenta de mineiração visual ao aparato de técnicas já utilizada pela comunidade científica. O mesmo vale para as metáforas de interação, que, preferencialmente, devem ser intuitivas e confortáveis ao usuário.

Na prática, quádricas compreendem uma classe de objetos simples de serem usados, pois possuem vetores normais e tangentes bem-definidos. Os vetores normais são usados nos cálculos de intensidade da luz, durante o processo de *rendering*. Para o cálculo da curvatura da superfície, normais e tangentes são utilizadas. A amostra de pontos, usada para representar um objeto, é geralmente definida de acordo com a curvatura: em regiões de grande curvatura, mais pontos são usados para alcançar uma melhor representação da região da superfície.

Por outro lado, o cálculo da dimensão fractal, através do método de Massa-Raio, é um recurso diferenciado para a análise do volume de dados. Por se tratar de uma medida global, é possível extrair características do conjunto, como alta co-relatividade de documentos ou alta variabilidade. Com essas ferramentas acredita-se poder adicionar dimensionalidade aos *displays* clássicos por superfícies.

Testes e conclusões

5.1 Considerações iniciais

Este capítulo possui 2 objetivos distintos: o primeiro envolve comparar os *frameworks*¹ de desenvolvimento para a ferramenta de exploração, neste caso, VTK em C++ ou Java3D. Apesar de ambos *frameworks* serem implementados sobre *OpenGL*, há diferenças quanto ao desempenho e portabilidade.

O segundo objetivo consiste em validar as novas técnicas de exploração de dados: super-quádricas e dimensão fractal. Tais recursos são incomuns em mineração visual de documentos, no entanto, foram largamente utilizados neste trabalho, cujos resultados podem ser observados na Seção 5.3.

5.2 Testes comparativos entre VTK e Java3D

A análise de desempenho entre o *Spider Cursor* - implementado em C++ utilizando a biblioteca gráfica VTK - e o *Super Spider* - desenvolvido em Java e utilizando a biblioteca gráfica Java3D - foi realizada considerando-se 2 parâmetros: FPS² (*Frames per second*) e memória utilizada.

Foi utilizada uma máquina para a realização do teste, cujas características relevantes para este caso são:

- Processador: AMD Sempron 2600+ (1.84GHz);
- Memória: DDR 400MHz 512MB;

¹Sistema ou ambiente de desenvolvimento.

²Quadros por segundo.

- Placa de vídeo: AGP GeForce4 MX 440 64MB;
- Sistema operacional: Windows XP SP2.

O arquivo utilizado para os testes de desempenho consiste num modelo geométrico construído a partir da tomografia de um mamão (Figura 5.1), com 28.934 células. Trata-se de um objeto de tamanho razoável e de forma complexa, pois não se resume a uma superfície, mas um objeto volumétrico. Neste caso, a ferramenta se comporta como um sistema de Visualização Científica. Os dados de entrada estão no formato VTK. Assim, o objetivo de se manter coerência entre as 2 versões torna possível uma comparação em iguais condições.

5.2.1 Testes de FPS (quadros por segundo)

Na Tabela 5.1 nota-se que o controle do objeto (interações básicas) possuem uma alta taxa de quadros por segundo, entretanto a sondagem pontual no Java3D se mostrou mais lenta. Isto ocorre pois a classe de objetos *Picking*, que realiza operações de seleção de pontos em Java3D, ainda é ineficiente, causando atrasos durante a interação do usuário.

A Tabela 5.2 mostra o desempenho utilizando a técnica de isolinhas, que funcionam como curvas de nível delineando a superfície. Para o teste, foram traçadas 50 curvas de nível, e relativamente os 2 sistemas mantiveram desempenho semelhante ao da técnica de isosuperfícies, onde o Java3D é menos eficiente na sondagem pontual.

Com a técnica de *scatterplots* (Tabela 5.3), o Java3D obteve desempenho bem superior em relação às operações de rotação, translação e ampliação, mantendo o baixo desempenho na sondagem pontual.

Utilizando super-quádricas, o desempenho da sondagem pontual no Java3D foi praticamente o mesmo, como pode ser observado nas Tabelas 5.1, 5.2 e 5.3, o que demonstra que os cálculos realizados para desenhá-las não são caros computacionalmente.

Portanto, o sistema desenvolvido em C++ utilizando a biblioteca gráfica VTK apresentou melhor desempenho que o sistema em Java com Java 3D, e a utilização das super-quádricas não acarretaram custo computacional notável.

5.2.2 Análise do consumo de memória

A segunda parte dos testes também utilizou a tomografia do mamão como base para a análise da memória ocupada pelo sistema. Comparando os 2 sistemas (5.4), observa-se uma grande diferença na utilização de memória: em C++, o sistema é compilado para linguagem de máquina, em um sistema operacional específico; em Java, o sistema é compilado para *bytecodes*⁵, que necessitam de uma JVM (*Java Virtual Machine*) para executar.

⁵Representação binária de um programa, executável apenas pela JVM (*Java Virtual Machine*).

Técnica: isosuperfícies		
	VTK	Java3D
Rotação / Translação / Ampliação	+60 FPS	+60 FPS
Sondagem com Spider Cursor	12 FPS	8 FPS
Sondagem com super-quádricas	*	7 FPS

Tabela 5.1: (*) Recurso não disponível na versão C++/VTK.

Técnica: 50 isolinhas		
	VTK	Java3D
Rotação / Translação / Ampliação	9 FPS	9 FPS
Sondagem com Spider Cursor	7 FPS	5 FPS
Sondagem com super-quádricas	*	5 FPS

Tabela 5.2: (*) Recurso não disponível na versão C++/VTK.

Técnica: <i>scatterplots</i>		
	VTK	Java3D
Rotação / Translação / Ampliação	30 FPS	58 FPS
Sondagem com Spider Cursor	12 FPS	6 FPS
Sondagem com super-quádricas	*	6 FPS

Tabela 5.3: (*) Recurso não disponível na versão C++/VTK.

Consumo de memória		
	VTK	Java3D
Isosuperfícies	29.4MB	81.8MB
Isolinhas	46.3MB	98.0MB
<i>Scatterplots</i>	33.2MB	68.0MB

Tabela 5.4: Arquivo de dados: mamão.vtk, com 28.934 células.

Figura 5.1: Foto da tomografia de um mamão, sendo visualizada por diversas técnicas. Em (a), isosuperfícies no *Spider Cursor*. Em (b), isosuperfícies no *Super Spider*. Em (c), visualização em *wireframe*⁴ no *Super Spider*. Em (d), isolinhas no *Spider Cursor*. Em (e), a sondagem com o *Spider Cursor*.

Portanto, ao executar uma aplicação em Java, além do programa em execução, há também a JVM executando em paralelo, agindo como uma integradora entre aplicação e sistema operacional. Eis o motivo pelo qual um programa em Java é totalmente portátil a outros sistemas operacionais, pois independem deste.

A JVM passa a maior parte do tempo realizando *garbage collection*⁶, ou seja, desalocando memória não-utilizada. Ainda assim a aplicação pode ocupar mais memória que o esperado. Desta forma, há como limitar o *heap size*⁷ através dos seguintes parâmetros na linha de comando:

```
java -Xms64m -Xmx256m programa_exemplo.class
```

onde a memória mínima e a máxima são limitadas a 64MB e 256MB, respectivamente. O sufixo *m* pode ser substituído por *g*, que representa GigaByte.

5.3 Análise de dados

Para a validação dos novos recursos do *Super Spider*, foram utilizados dados já explorados com uso de outra ferramenta - *Projection Explorer* [Paulovich e Minghim 2006]. Assim, o comportamento destes dados é conhecido de antemão, e assim servem para comprovar a aplicabilidade das novas funcionalidades do *Super Spider*.

5.3.1 Notícias internacionais

Entre os dias 5 e 7 de abril de 2006 foram coletadas diversas notícias dos sites *ABC News* (<http://abcnews.go.com/>), *BBC News* (<http://news.bbc.co.uk/>), *CNN.com* (<http://www.cnn.com/>) e *RedOrbit* (<http://www.redorbit.com/news/>), totalizando 2684 notícias. Baseado em seu conteúdo, foi gerado um mapa com a ferramenta *Projection Explorer*, aproximando notícias de conteúdo semelhante (Figura 5.2).

Desta forma, com o mapa gerado pode-se iniciar a exploração com o *Super Spider*, e novas inferências podem ser realizadas. Pelo contexto internacional dessa época, diversos assuntos receberam destaque na mídia internacional, como: a aparição de uma ave morta no Reino Unido, gerando alarde em torno da gripe do frango; o processo da autoria em torno do lançamento do polêmico filme *Da Vinci Code*; o final da temporada de tornados nos Estados Unidos; e principalmente, a Guerra do Iraque.

Este último tópico foi o mais discutido dentre os demais, e consequentemente, há muitas notícias com conteúdo similar ou redundante. Desta forma, quanto maior a similaridade, maior a dimensão fractal dos textos.

Tal característica fica visível com a utilização das super-quádricas. Como pode ser observado na Figura 5.3, as cores indicam regiões em que foi discutido o termo "Iraq". Quanto mais próximo ao anil, mais vezes o termo foi discutido. Quanto mais próximo ao vermelho, menos o termo foi citado. Assim, a super-quádrica em sua forma de cruz 3D (conforme Tabela A.1) indica alta dimensão fractal para as notícias em destaque sobre a Guerra do Iraque. Ou seja, são notícias altamente similares.

⁶Recolhimento de lixo

⁷O espaço de memória em que a aplicação está alocada.

Figura 5.2: A visualização gerada pelo *Projection Explorer* apresenta o mapa de notícias internacionais, colhidas de diversos *sites*, cuja distribuição é baseada em similaridade de conteúdo. Grupos são formados com notícias altamente similares.

Na Figura 5.4, altura e cor também foram mapeados para representar a dimensão fractal: quanto mais alto e próximo ao anil, maior a dimensão fractal. Regiões baixas e vermelhas, indicam notícias isoladas em seu contexto. Observa-se, então, uma região concentrada e altamente co-relacionada em (a), uma vez que possui maior altura, cor anil e a Super-Quádrlica em sua forma pontiaguda. Em (b), temos a situação oposta: uma notícia afastada da região de alta similaridade, de baixa altura, cor vermelha e a Super-Quádrlica em sua forma de cubo.

5.3.2 Artigos científicos

Foram coletados do *CiteSeer.IST* (<http://citeseer.ist.psu.edu/>) vários dados de artigos, como título, autor, resumo e referências. No total, 682 artigos, distribuídos nas seguintes categorias: *Case-Based Reasoning* (CBR), *Inductive Logic-Programming* (ILP), *Information Retrieval* (RI) e *Sonification* (SON).

Dado que alguns artigos foram inseridos manualmente, sabe-se que estes são altamente correlatos, exceto o artigo *Morse Operators for Digital Planar Surfaces and Their Application to Image Segmentation*, que não possui relação com os demais "intrusos".

As cores da Figura 5.5 e 5.6 identificam cada uma das classes: CBR (vermelho), ILP (amarelo), SON (anil), RI (azul), e os intrusos (verde).

Na Figura 5.5, pode ser observado que em (a) e (b) ocorre alta dimensão fractal, pela forma do cursor. Portanto, são artigos fortemente relacionados. Em (c), o outro

Figura 5.3: As notícias em destaque são: a) *Mosque explosion kills 40 in Iraq*, b) *Bombing near Najaf shrine kills 9*, c) *43 killed in Baghdad mosque attack*, d) *Mosque explosion kills 52 in Iraq*. A super-quádrica, em sua forma pontiaguda, demonstra uma alta dimensão fractal para as notícias investigadas, ou seja, alta similaridade.

“intruso” possui baixa dimensão fractal (cursor está em forma de cubo), o que confirma a não-similaridade deste com os demais.

Na Figura 5.6, os artigos destacados em (a) e (b) possuem fraca relação com os demais de suas respectivas áreas, pois a baixa dimensão fractal apresentada pela super-quádrica - em forma de cubo - sugere um alto nível de inovação e originalidade nos documentos.

Desta forma, a dimensão fractal fornece uma informação de conjunto, muito útil em uma mineração visual de dados.

Figura 5.4: As manchetes que constam nas imagens: a) *Hussein in cross-examined* e b) *Bob Woordruff sends message to co-workers*. Em a), nota-se uma região de alta dimensão fractal, que trata de dados diários da Guerra do Iraque. Em b), um assunto também relacionado à guerra, mas que discute sobre um jornalista, seus colegas de trabalho e família. Ou seja, fracamente relacionado ao cotidiano da guerra, e por este motivo apresenta um certo isolamento das demais.

5.4 Conclusão

O primeiro objetivo mencionado, sobre a escolha da melhor arquitetura de desenvolvimento gráfico, foi atendido. Para uma ferramenta de exploração, há 3 características fundamentais que a tornam útil: portabilidade, disponibilidade e desempenho.

A plataforma Java possibilita duas destas características de forma poderosa: é totalmente portátil, sendo seus aplicativos independentes de sistema operacional, demandando apenas a instalação de *plugins*; e por ser uma linguagem voltada para a *Web*, possui alta disponibilidade, pelo fato de seus aplicativos executarem diretamente a partir do *browser*.

No quesito desempenho, o conjunto VTK e C++ mostrou-se mais eficiente. No entanto, é possível contornar esse problema utilizando Java, delegando parte ou todo processamento para servidores dedicados.

Assim, a biblioteca gráfica Java3D tem maior aceitabilidade por parte dos seus usuários, que desejam apenas usar a ferramenta, sem ter o trabalho de instalá-la, e com o conforto de utilizá-la pelo navegador *Web*.

Por outro lado, atendendo o segundo objetivo, os novos recursos implementados obtiveram bons resultados. O grande desafio era de extrair informações intrínsecas dos dados, os quais as técnicas usuais como isosuperfícies, *height fields*, isolinhas, mapeamentos de cor e outras não são capazes de exibir. Com as super-quádricas, várias dimensões

Figura 5.5: Os títulos dos artigos são: a) *DSVOL - Implementation of a Distributed Architecture Combining General-purpose Visualization and Sonification*, b) *DSVOL II - A Distributed Visualization and Sonification Application Communicating via an XML-Based Protocol*, c) *Morse Operators for Digital Planar Surfaces and Their Application to Image Segmentation*. Pela forma da super-quádrica, observa-se que a) e b) possuem alta dimensão fractal, pois são documentos fortemente ligados. Em c) observamos que o intruso não tem relação alguma com os demais.

adicionais puderam ser visualizadas ao mesmo tempo, e com o cálculo da dimensão fractal, informações sobre os dados originais de conjunto foram reveladas.

Ainda há muito o que aprimorar nestas novas técnicas, a fim de torná-las mais adaptadas às reais necessidades do usuário. Mas sinto que este trabalho atendeu as expectativas de apresentar e avaliar novos métodos de exploração, sempre com a meta de contribuir junto à comunidade, despertando os sentimentos de empenho e insaciabilidade, inerentes à pesquisa científica.

A ferramenta, assim como os mapas gerados e utilizados nos testes desta dissertação,

Figura 5.6: Os títulos dos artigos são: a) *Applying Case-Based Reasoning to Automated Deduction*, b) *Tetravalent Miniantibodies with High Avidity Assembling in Escherichia coli*. Pela forma da super-quádrica, observamos que a) e b) também são artigos com grande teor de inovação, pois possuem baixa dimensão fractal.

podem ser encontrados no site <http://www.icmc.usp.br/~lionis>.

Referências Bibliográficas

- [Andrews 1972]ANDREWS, D. F. Plots of high-dimensional data. *Biometrics*, v. 29, p. 125–136, 1972.
- [Backes e Bruno 2005]BACKES, A. R.; BRUNO, O. M. Técnicas de estimativa da dimensão fractal: Um estudo comparativo. *INFOCOMP Journal of Computer Science*, v. 4, n. 3, p. 50–58, 2005.
- [Bade, Ritter e Preim 2005]BADE, R.; RITTER, F.; PREIM, B. Usability comparison of mouse-based interaction techniques for predictable 3d rotation. In: *Smart Graphics*. [S.l.: s.n.], 2005. p. 138–150.
- [Barr 1981]BARR, A. H. Superquadrics and angle-preserving transformations. *IEEE Computer Graphics and Applications*, v. 1, n. 1, p. 11–23, 1981.
- [Bartram 1997]BARTRAM, L. Perceptual and interpretative properties of motion for information visualization. In: *NPIV '97: Proceedings of the 1997 workshop on New paradigms in information visualization and manipulation*. [S.l.]: ACM Press, 1997. p. 3–7. ISBN 1-58113-051-1.
- [Bartram et al. 1994]BARTRAM, L. et al. Contextual assistance in user interfaces to complex, time-critical systems: The intelligent zoom. In: *Graphics Interface'94*. [S.l.: s.n.], 1994. p. 216–224.
- [Becker 1997]BECKER, B. G. Research report: Volume rendering for relational data. In: *Proceedings of the 1997 IEEE Symposium on Information Visualization (InfoVis '97)*. [S.l.]: IEEE Computer Society, 1997. p. 87. ISBN 0-8186-8189-6.
- [Beddow 1990]BEDDOW, J. Shape coding of multidimensional data on a microcomputer display. In: *Proceedings of the 1st conference on Visualization '90*. San Francisco, CA, USA: [s.n.], 1990. p. 23–26.
- [Bier et al. 1993]BIER, E. A. et al. Toolglass and magic lenses: the see-through interface. In: *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer Graphics and interactive techniques*. New York, NY, USA: ACM Press, 1993. p. 73–80. ISBN 0-89791-601-8.
- [Bourke 1987]BOURKE, P. Determining if a point lies on the interior of a polygon. Disponível online em <http://astronomy.swin.edu.au/pbourke/geometry/insidepoly/>, 1987.

- [Bowman e Hodges 1999]BOWMAN, D. A.; HODGES, L. F. Formalizing the design, evaluation, and application of interaction techniques for immersive virtual environments. *Journal of Visual Languages and Computing*, v. 10, n. 1, p. 37–53, 1999.
- [Brodlie et al. 1992]BRODLIE, K. W. et al. (Ed.). *Scientific visualization: techniques and applications*. [S.l.]: Springer-Verlag New York, Inc., 1992. ISBN 0-387-54565-4.
- [Card, Pirolli e Mackinlay 1994]CARD, S. K.; PIROLI, P.; MACKINLAY, J. D. The cost-of-knowledge characteristic function: Display evaluation of direct-walk dynamic information visualizations. *CHI'94*, ACM, p. 238–244, 1994.
- [CHapanis e Kinkade 1972]CHAPANIS, A.; KINKADE, R. *Design of controls*. [S.l.], 1972.
- [Chernoff 1973]CHERNOFF, H. The use of faces to represent points in k-dimensional space graphically. *Journal Amer. Statistical Association*, v. 68, p. 361–368, 1973.
- [Cleveland 1985]CLEVELAND, W. S. *The elements of graphing data*. [S.l.]: Wadsworth Publ. Co., 1985. ISBN 0-534-03730-5.
- [Cleveland 1993]CLEVELAND, W. S. *Visualizing Data*. Summit, New Jersey, U.S.A.: Hobart Press, 1993. ISBN 0963488406.
- [Colby e Scholl 1991]COLBY, G.; SCHOLL, L. Transparency and blur as selective cues for complex visual information. In: BENDER, W. R.; PLOUFFE, W. (Ed.). *SPIE '91: Proceedings of image handling and reproduction systems integration*. [S.l.: s.n.], 1991. p. 114–125.
- [Cook e Scavone 1999]COOK, P.; SCAVONE, G. *The Synthesis ToolKit (STK)*. 1999. Disponível em: <citeseer.ist.psu.edu/article/cook99synthesis.html>.
- [Diepstraten, Weiskopf e Ertl 2003]DIEPSTRATEN, J.; WEISKOPF, D.; ERTL, T. Interactive cutaway illustrations. In: *Proceedings of Eurographics Conference '03 (to appear in Computer Graphics Forum)*. [S.l.: s.n.], 2003. p. 523–532.
- [Elcacho, Dingel e Klein 2001]ELCACHO, C.; DINGEL, T.; KLEIN, R. Object-centered navigation in virtual construction applications. In: SKALA, V. (Ed.). *WSCG 2001 Conference Proceedings*. [S.l.: s.n.], 2001. p. 33–40.
- [Feiner e Beshers 1990]FEINER, S. K.; BESHES, C. Visualizing n-dimensional virtual worlds with n-vision. In: *SI3D '90: Proceedings of the 1990 symposium on Interactive 3D graphics*. Snowbird, Utah, United States: ACM Press, 1990. p. 37–38. ISBN 0-89791-351-5.
- [Furnas 1986]FURNAS, G. W. Generalized fisheye views. In: *CHI '86: Proceedings of the SIGCHI conference on Human Factors in computing systems*. New York, NY, USA: ACM Press, 1986. p. 16–23. ISBN 0-89791-180-6.
- [Furnas e Buja 1994]FURNAS, G. W.; BUJA, A. Prosection views: Dimensional inference through sections and projections. *Journal of Computational and Graphical Statistics*, p. 323–325, december 1994.
- [Greene et al. 1990]GREENE, S. L. et al. No ifs, ands, or ors: A study of database querying. *International Journal of Man-Machine Studies*, v. 32, n. 3, p. 303–326, 1990.

- [Hearst 1995]HEARST, M. A. Tilebars: Visualization of term distribution information in full text information access. In: *Proceedings of the Conference on Human Factors in Computing Systems, CHI'95*. [s.n.], 1995. Disponível em: <<http://citeseer.ist.psu.edu/hearst95tilebars.html>>.
- [Huber 1985]HUBER, P. J. Projection pursuit. *The Annals of Statistics*, v. 13, n. 2, p. 435–475, 1985.
- [Hughes 2005]HUGHES, S. B. *Assisted Viewpoint interaction for 3D visualizations*. Dissertação (Mestrado) — University of Pittsburgh, School of Information Sciences, Pittsburgh, Pennsylvania, USA, 2005.
- [Inselberg 1985]INSELBERG, A. The plane with parallel coordinates. *The Visual Computer*, v. 1, n. 2, p. 69–91, 1985.
- [Inselberg e Dimsdale 1990]INSELBERG, A.; DIMSDALE, B. Parallel coordinates: a tool for visualizing multi-dimensional geometry. In: *VIS '90: Proceedings of the 1st conference on Visualization '90*. San Francisco, California, USA: IEEE Computer Society Press, 1990. p. 361–378. ISBN 0-8186-2083-8 (PAPER).
- [Interrante e Grosch 1997]INTERRANTE, V.; GROSCH, C. Strategies for effectively visualizing 3D flow with volume LIC. In: *IEEE Visualization*. [s.n.], 1997. p. 421–424. Disponível em: <<http://citeseer.ist.psu.edu/interrante97strategie.html>>.
- [Johnson 1993]JOHNSON, B. S. *Treemaps: visualizing hierarchical and categorical data*. Tese (Doutorado) — University of Maryland at College Park, 1993.
- [Keim 1997]KEIM, D. A. *Visual techniques for exploring databases*. [S.l.], agosto 1997. 3rd Int. Conf. Knowledge Discovery and Data Mining.
- [Keim, Ankerst e Kriegel 1995]KEIM, D. A.; ANKERST, M.; KRIEGEL, H.-P. Recursive pattern: A technique for visualizing very large amounts of data. In: *VIS '95: Proceedings of the 6th conference on Visualization '95*. Atlanta, GA, USA: IEEE Computer Society, 1995. p. 279–286. ISBN 0-8186-7187-4.
- [Keim e Kriegel 1994]KEIM, D. A.; KRIEGEL, H. VisDB: Database exploration using multidimensional visualization. *Computer Graphics and Applications*, p. 40–49, setembro 1994. Disponível em: <<http://citeseer.ist.psu.edu/keim94visdb.html>>.
- [Keim, Kriegel e Seidl 1994]KEIM, D. A.; KRIEGEL, H.-P.; SEIDL, T. Supporting data mining of large databases by visual feedback queries. In: *Proceedings of the Tenth International Conference on Data Engineering*. [S.l.]: IEEE Computer Society, 1994. p. 302–313. ISBN 0-8186-5400-7.
- [Kienreich et al. 2003]KIENREICH, W. et al. Infosky: A system for visual exploration of very large, hierarchically structured knowledge spaces. In: *Proceedings of FGWM'03 - Workshop Wissens*. Karlsruhe, Germany: [s.n.], 2003.
- [Kosara, Bendix e Hauser 2006]KOSARA, R.; BENDIX, F.; HAUSER, H. Parallel sets: Interactive exploration and visual analysis of categorical data. *IEEE Transactions on Visualization and Computer Graphics*, v. 12, n. 4, p. 558–568, 2006.

- [Kosara, Miksch e Hauser 2002]KOSARA, R.; MIKSCH, S.; HAUSER, H. Focus+context taken literally. *IEEE Computer Graphics and Applications*, v. 22, n. 1, p. 22–29, janeiro 2002.
- [Kosara, Sahling e Hauser 2004]KOSARA, R.; SAHLING, G. N.; HAUSER, H. Linking scientific and information visualization with interactive 3D scatterplots. *WSCG 2004*, p. 2–6, february 2004.
- [Kreuseler, Lopez e Schumann 2000]KREUSELER, M.; LOPEZ, N.; SCHUMANN, H. A scalable framework for information visualization. In: *INFOVIS '00: Proceedings of the IEEE Symposium on Information Visualization 2000*. Washington, DC, USA: IEEE Computer Society, 2000. p. 27. ISBN 0-7695-0804-9.
- [Krüger et al. 2005]KRÜGER, A. et al. Interactive visualization for neck dissection planning. In: *Proceedings of IEEE EuroVis'05*. [S.l.: s.n.], 2005. p. 295–302.
- [LeBlanc, Ward e Wittels 1990]LEBLANC, J.; WARD, M. O.; WITTELS, N. Exploring n-dimensional databases. In: *VIS '90: Proceedings of the 1st conference on Visualization '90*. San Francisco, CA, USA: IEEE Computer Society Press, 1990. p. 230–237. ISBN 0-8186-2083-8 (PAPER).
- [Leung e Apperley 1994]LEUNG, Y. K.; APPERLEY, M. D. A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction*, ACM Press, New York, NY, USA, v. 1, n. 2, p. 126–160, 1994. ISSN 1073-0516.
- [Levkowitz 1991]LEVKOWITZ, H. Color icons: Merging color and texture perception for integrated visualization of multiple parameters. In: *IEEE Visualization*. [S.l.: s.n.], 1991. p. 164–170.
- [Lokuge e Ishizaki 1995]LOKUGE, I.; ISHIZAKI, S. Geospace: an interactive visualization system for exploring complex information spaces. In: *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1995. p. 409–414. ISBN 0-201-84705-1.
- [Mandelbrot 1982]MANDELBROT, B. B. *The Fractal Geometry of Nature*. [S.l.]: W. H. Freeman, 1982. ISBN 0716711869.
- [McDonald e Erbacher 2005]MCDONALD, D.; ERBACHER, R. F. Hardware accelerated graphics in java. In: *GMVAG 2005: Proceedings of the International Conference on Geometric Modeling, Visualization & Graphics 2005*. Salt Lake City, UT: [s.n.], 2005. p. 1708–1712.
- [Michard 1982]MICHARD, A. A new database query language for non-professional users: Design principles and ergonomic evaluation. *Behavioral and Information Technology*, v. 1, n. 3, p. 279–288, julho-setembro 1982.
- [Minghim et al. 2005]MINGHIM, R. et al. Spider cursor: a simple versatile interaction tool for data visualization and exploration. In: *SIGGRAPH: ACM SPECIAL INTEREST GROUP ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES. GRAPHITE '05: Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*. Dunedin, New Zealand: ACM Press, New York, NY, USA, 2005. p. 307–313.

- [Minghim e Oliveira 1997] *Uma introdução à Visualização Computacional*. Brasília: XVII Congresso da SBC, 1997.
- [Mroz e Hauser 2001] MROZ, L.; HAUSER, H. RTVR: a flexible java library for interactive volume rendering. In: *Proceedings of the conference on Visualization '01*. [S.l.]: IEEE Computer Society, 2001. p. 279–286. ISBN 0-7803-7200-X.
- [Parker, Franck e Ware 1998] PARKER, G.; FRANCK, G.; WARE, C. Visualizing of large nested graphs in 3D: Navigation and interaction. *Journal of Visual Languages* 9, p. 299–317, 1998.
- [Paulovich e Minghim 2006] PAULOVICH, F. V.; MINGHIM, R. Text map explorer: a tool to create and explore document maps. In: *IV '06: Proceedings of the conference on Information Visualization*. Washington, DC, USA: IEEE Computer Society, 2006. p. 245–251. ISBN 0-7695-2602-0.
- [Pickett 1970] PICKETT, R. M. Visual analyses of texture in the detection and recognition of objects. In: LIPKIN, B. S.; ROSENFELD, A. (Ed.). *Picture Processing and Psycho-Pictorics*. New York, NY, USA: Academic Press, 1970.
- [Pickett e Grinstein 1988] PICKETT, R. M.; GRINSTEIN, G. G. Iconographic displays for visualizing multidimensional data. In: *Proc. IEEE Conf. on Systems, Man and Cybernetics*. Piscataway, NJ, USA: IEEE Press, 1988. p. 514–519.
- [Piringer, Kosara e Hauser 2004] PIRINGER, H.; KOSARA, R.; HAUSER, H. Interactive focus+context visualization with linked 2D/3D scatterplots. *Coordinated and Multiple Views in Exploratory Visualization, 2004. Proceedings. Second International Conference on*, p. 49–60, july 2004. Disponível em: <<http://citeseer.ist.psu.edu/694129.html>>.
- [Reina, G. e Ertl 2004] REINA, G.; ERTL, T. Volume Visualization and Visual Queries for Large High-Dimensional Datasets. In: O. Deussen and C. Hansen and D.A. Keim and D. Saupe (Ed.). *Proceedings of EG/IEEE TCVG Symposium on Visualization VisSym '04*. [S.l.: s.n.], 2004.
- [Rekimoto e Green 1993] REKIMOTO, J.; GREEN, M. The information cube: Using transparency in 3d information visualization. In: *Proceedings of the Third Ann. Workshop Information Technologies & Systems (WITS '93)*. [S.l.: s.n.], 1993. p. 125–132.
- [Robertson e Mackinlay 1993] ROBERTSON, G. G.; MACKINLAY, J. D. The document lens. In: *UIST '93: Proceedings of the 6th annual ACM symposium on User interface software and technology*. New York, NY, USA: ACM Press, 1993. p. 101–108. ISBN 0-89791-628-X.
- [Robertson, Mackinlay e Card 1991] ROBERTSON, G. G.; MACKINLAY, J. D.; CARD, S. K. Cone trees: Animated 3d visualizations of hierarchical information. In: *Proc. Human Factors in Computing Systems CHI 91 Conf.* New Orleans, LA, USA: [s.n.], 1991. p. 189–194.
- [Salvador 2003] SALVADOR, V. C. L. *Desenvolvimento de um modelo para exploração em visualização científica distribuída usando som*. Tese (Doutorado) — Instituto de Ciências Matemáticas e Computação - USP, São Carlos - SP, Brazil, agosto 2003.

- [Salvador e Minghim 2003]SALVADOR, V. C. L.; MINGHIM, R. An interaction model for scientific visualization using sound. In: Instituto de Ciências Matemáticas e de Computação - USP. São Carlos - SP, Brazil: Proceedings of the XVI Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI'03), 2003. p. 132–139.
- [Salvador, Minghim e Pacheco 1998]SALVADOR, V. C. L.; MINGHIM, R.; PACHECO, M. L. Sonification to support visualization tasks. In: Instituto de Ciências Matemáticas e de Computação - USP. São Carlos - SP, Brazil: Proceedings of SIBGRAPI'98, 1998. p. 150–157.
- [Sarkar et al. 1993]SARKAR, M. et al. Stretching the rubber sheet: a metaphor for viewing large layouts on small screens. In: *UIST '93: Proceedings of the 6th annual ACM symposium on User interface software and technology*. New York, NY, USA: ACM Press, 1993. p. 81–91. ISBN 0-89791-628-X.
- [Schroeder, Martin e Lorensen 2002]SCHROEDER, W.; MARTIN, K.; LORENSEN, B. *The Visualization Toolkit: an object-oriented approach to 3D graphics*. 3. ed. [S.l.]: Kitware Inc., 2002.
- [Shneiderman 1992]SHNEIDERMAN, B. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. Graph.*, ACM Press, v. 11, n. 1, p. 92–99, 1992. ISSN 0730-0301.
- [Shneiderman 1996]SHNEIDERMAN, B. The eyes have it: A task by data type taxonomy for information visualizations. In: *VL '96: Proceedings of the 1996 IEEE Symposium on Visual Languages*. [S.l.]: IEEE Computer Society, 1996. p. 336. ISBN 0-8186-7508-X.
- [Spoerri 1993]SPOERRI, A. Infocrystal: A visual tool for information retrieval & management. In: *CIKM 93, Proceedings of the Second International Conference on Information and Knowledge Management, Washington, DC, USA, November 1-5, 1993*. [S.l.]: ACM, 1993. p. 11–20. ISBN 0-89791-626-3.
- [Stone, Fishkin e Bier 1994]STONE, M. C.; FISHKIN, K.; BIER, E. A. The movable filter as a user interface tool. In: *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA: ACM Press, 1994. p. 306–312. ISBN 0-89791-650-6.
- [Su et al. 1995]SU, H. et al. *An Interactive Visualization Tool for Tolerance Design*. London, UK, 1995.
- [Toutin 1997]TOUTIN, T. Qualitative aspects of chromo-stereoscopy for depth-perception. *Photogrammetric Engineering & Remote Sensing*, p. 193–203, february 1997.
- [Tufte 1983]TUFTE, E. R. *The Visual Display of Quantitative Information*. Cheshire, CT: Graphics Press, 1983.
- [Tutida 1998]TUTIDA, S. M. *Mapeamento por cores baseado em aspectos de percepção visual*. Dissertação (Mestrado) — Instituto de Ciências Matemáticas e Computação - USP, São Carlos, SP, BRAZIL, maio 1998.
- [Ward 2003]WARD, M. O. A taxonomy of glyph placement strategies for multidimensional data visualization. *Information Visualization*, Palgrave Macmillan, p. 194–210, janeiro 2003.

- [Ware 2000]WARE, C. *Information Visualization: perception for design*. 2. ed. [S.l.]: Morgan Kaufmann, 2000.
- [Wijk e Liere 1993]WIJK, J. J. van; LIERE, R. van. Hyperslice: visualization of scalar functions of many variables. In: *VIS '93: Proceedings of the 4th conference on Visualization '93*. San Jose, California, USA: [s.n.], 1993. p. 119–125. ISBN O-8186-3940-7 (PAPER).
- [Wright 1999]WRIGHT, W. Information animation applications in the capital markets. Morgan Kaufmann Publishers Inc., p. 83–91, 1999.
- [Yang, Ward e Rundensteiner 2002]YANG, J.; WARD, M. O.; RUNDENSTEINER, E. A. Interring: An interactive tool for visually navigating and manipulating hierarchical structures. In: *INFOVIS '02: Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*. Washington, DC, USA: IEEE Computer Society, 2002. p. 77. ISBN 0-7695-1751-X.
- [Young e Shneiderman 1993]YOUNG, D.; SHNEIDERMAN, B. A graphical filter/flow model for boolean queries: An implementation and experiment. *Journal of the American Society for Information Science*, v. 44, n. 6, p. 327–339, julho 1993.

Cônicas, quádricas e super-quádricas

A.1 Produto esférico

Para que seja possível compreender a representação paramétrica das super-quádricas - utilizada para *rendering* - é necessário explicar o conceito de *produto esférico*. O *produto esférico*, denotado por $\otimes$, é definido para operar duas curvas 2D:

$$\underline{h}(\omega) = \begin{bmatrix} h_1(\omega) \\ h_2(\omega) \end{bmatrix}, \omega_0 \leq \omega \leq \omega_1 \quad (\text{A.1})$$

$$\underline{m}(\eta) = \begin{bmatrix} m_1(\eta) \\ m_2(\eta) \end{bmatrix}, \eta_0 \leq \eta \leq \eta_1 \quad (\text{A.2})$$

o que resulta em uma superfície 3D.

$$\underline{x}(\eta, \omega) = \underline{m} \otimes \underline{h} = \begin{bmatrix} m_1(\eta) & h_1(\omega) \\ m_1(\eta) & h_2(\omega) \\ m_2(\eta) & \end{bmatrix}, \begin{matrix} \eta_0 \leq \eta \leq \eta_1 \\ \omega_0 \leq \omega \leq \omega_1 \end{matrix} \quad (\text{A.3})$$

Cada curva 2D tem um grau de liberdade, portanto a superfície resultante possui 2 graus de liberdade.

Pode-se abstrair que a função $\underline{h}$ seja uma curva horizontal, que é arrastada verticalmente, de acordo com a função $\underline{m}$. O termo $m_1(\eta)$ determina a escala de $\underline{h}$, enquanto $m_2(\eta)$ define o movimento de varredura vertical. Desta forma, tem-se que o parâmetro ω afeta a superfície horizontalmente, enquanto η afeta a superfície verticalmente.

Adicionando um termo escalar para cada uma das direções espaciais, obtém-se uma forma com 5 graus de liberdade:

$$\underline{x}(\eta, \omega) = \begin{bmatrix} \alpha_1 & m_1(\eta) & h_1(\omega) \\ \alpha_2 & m_1(\eta) & h_2(\omega) \\ \alpha_3 & m_2(\eta) & \end{bmatrix}, \begin{matrix} \eta_0 \leq \eta \leq \eta_1 \\ \omega_0 \leq \omega \leq \omega_1 \end{matrix} \quad (\text{A.4})$$

Um exemplo simples de um produto esférico é uma esfera unitária (da qual o termo

produto esférico herda seu nome). Nesta linha de raciocínio, uma esfera unitária consiste apenas do produto esférico de um círculo (horizontal) e um meio-círculo (vertical).

$$\begin{bmatrix} \cos \eta \\ \sin \eta \end{bmatrix} \otimes \begin{bmatrix} \cos \omega \\ \sin \omega \end{bmatrix} = \begin{bmatrix} \cos \eta & \cos \omega \\ \cos \eta & \sin \omega \\ \sin \eta & \end{bmatrix}, \quad \begin{matrix} -\pi/2 \leq \eta \leq \pi/2 \\ -\pi \leq \omega \leq \pi \end{matrix} \quad (\text{A.5})$$

A.1.1 Cônicas

Uma cônica é uma curva 2D, descrita pela equação geral:

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0 \quad (\text{A.6})$$

Esta forma geral pode ser rotacionada, escalada, e transladada a fim de que fique alinhada com os eixos do sistema de coordenadas. A equação ganha uma forma simplificada:

$$ax^2 + by^2 = k \quad \text{ou} \quad ax^2 + by = k \quad (\text{A.7})$$

As cônicas mais utilizadas são a elipse (da qual o círculo é um caso especial), a hipérbole, e a parábola.

A.1.2 Super-cônicas

Consistem da extensão das cônicas 2D, com a diferença que os expoentes das coordenadas é um valor n arbitrário. Variando seus parâmetros, obtém-se a super-elipse, que é definida pela forma paramétrica:

$$\begin{aligned} x(\omega) &= r_x \cos^n(\omega) & 0 \leq \omega \leq 2\pi \\ y(\omega) &= r_y \sin^n(\omega) & 0 < n < \infty \end{aligned} \quad (\text{A.8})$$

A curva intersecta o eixo x em r_x e $-r_x$, e intersecta o eixo y em r_y e $-r_y$, ou seja, a super-elipse é enquadrada por um retângulo de comprimento $2r_x$ e altura $2r_y$.

Esta é a forma mais empregada para desenhar uma super-elipse: varia-se o ângulo ω de 0 a 2π em poucos passos, e uma linha é desenhada para cada valor subsequente de ω .

Figura A.1: Super-elipse 2D de raio r .

Formar a curva usando a forma paramétrica abaixo e variando uniformemente o ângulo, a definição da curva é ajustada automaticamente de acordo com a curvatura. Ou seja,

a definição é melhor onde a curvatura é maior; trata-se de uma característica desejável. Pode-se escrever também:

$$f(x, y) = \left| \frac{x}{r_x} \right|^{2/n} + \left| \frac{y}{r_y} \right|^{2/n} \quad (\text{A.9})$$

- Se $f(x, y) = 1$, (x, y) está sobre a borda da super-elipse;
- Se $f(x, y) < 1$, (x, y) está dentro da super-elipse;
- Se $f(x, y) > 1$, (x, y) está fora da super-elipse.

A forma acima pode ser usada para formar a super-elipse (ainda que seja feito em quadrantes). Resolvendo para y :

$$y = r_y \left| 1 - \left| \frac{x}{r_x} \right|^{2/n} \right|^{n/2} \quad (\text{A.10})$$

Variando n , controla-se a forma da super-elipse, cujos casos podem ser vistos na Tabela A.1.

n	r_x, r_y	<i>forma resultante</i>
0	-	retângulo
0	$r_x = r_y$	quadrado
< 1	-	retângulo arredondado
$= 1$	-	elipse
$= 1$	$r_x = r_y$	círculo
$= 2$	-	diamante
> 2	-	diamante pontudo
$\Rightarrow \infty$	-	cruz

Tabela A.1: Possíveis formas de uma super-elipse 2D, de acordo com n .

Figura A.2: Super-elipse 2D variando sua forma.

Nota-se que a super-elipse sempre passa pelos mesmos pontos $\omega = 0, \pi/2, \pi, 3\pi/2$ e então é possível ter diferentes valores de n em cada quadrante.

A.1.3 Quádricas

Esferas, cones e cilindros fazem parte de uma família geral de superfícies paramétricas denominada quádricas. Tratam-se dos correspondentes tridimensionais das cônicas bidimensionais.

As quádricas são o elemento-base para a formação das super-quádricas. Há 4 formas básicas de superfícies quádricas: elipsóides, hiperbolóides de uma lâmina, hiperbolóides de duas lâminas e os toros. Em [Barr 1981] é realizada uma descrição das equações-padrão para estes objetos, e também são apresentadas as representações dos produtos esféricos.

As quádricas são os análogos 3D das cônicas. A equação geral é descrita:

$$Ax^2 + By^2 + Cz^2 + Dxy + Eyz + Fzy + Gx + Hy + Jz + K = 0 \quad (\text{A.11})$$

Se rotacionada, escalada, e transladada para que fique alinhada com os eixos do sistema de coordenadas, a equação adquire a forma:

$$ax^2 + by^2 + cz^2 = k \quad \text{ou} \quad ax^2 + by^2 + cz = k \quad (\text{A.12})$$

As quádricas mais usadas são os elipsóides (dos quais a esfera é um caso especial), o cilindro infinito, e o cone infinito. Vários hiperbolóides e parabolóides também são definidos por estas equações, e são utilizados para representar diversos objetos, como os pratos dos satélites, refletores de lâmpadas (parabolóides), e torres de ventilação de usinas de energia (hiperbolóides).

Elipsóide

A forma paramétrica da elipsóide é:

$$\left(\frac{x_1}{\alpha_1}\right)^2 + \left(\frac{x_2}{\alpha_2}\right)^2 + \left(\frac{x_3}{\alpha_3}\right)^2 = 1 \quad (\text{A.13})$$

$$\underline{x}(\eta, \omega) = \begin{bmatrix} \alpha_1 & \cos \eta & \cos \omega \\ \alpha_2 & \cos \eta & \sin \omega \\ \alpha_3 & \sin \eta & \end{bmatrix}, \quad \begin{matrix} -\pi/2 \leq \eta \leq \pi/2 \\ -\pi \leq \omega < \pi \end{matrix} \quad (\text{A.14})$$

Hiperbolóide de uma lâmina

A forma paramétrica da hiperbolóide de uma lâmina é:

$$\left(\frac{x_1}{\alpha_1}\right)^2 + \left(\frac{x_2}{\alpha_2}\right)^2 - \left(\frac{x_3}{\alpha_3}\right)^2 = 1 \quad (\text{A.15})$$

$$\underline{x}(\eta, \omega) = \begin{bmatrix} \alpha_1 & \sec \eta & \cos \omega \\ \alpha_2 & \sec \eta & \sin \omega \\ \alpha_3 & \tan \eta & \end{bmatrix}, \quad \begin{matrix} -\pi/2 < \eta < \pi/2 \\ -\pi \leq \omega < \pi \end{matrix} \quad (\text{A.16})$$

Hiperbolóide de duas lâminas

A forma paramétrica da hiperbolóide de duas lâminas é:

$$\left(\frac{x_1}{\alpha_1}\right)^2 - \left(\frac{x_2}{\alpha_2}\right)^2 - \left(\frac{x_3}{\alpha_3}\right)^2 = 1 \quad (\text{A.17})$$

$$\underline{x}(\eta, \omega) = \begin{bmatrix} \alpha_1 & \sec \eta & \sec \omega \\ \alpha_2 & \sec \eta & \tan \omega \\ \alpha_3 & \tan \eta & \end{bmatrix}, \quad \begin{array}{ll} -\pi/2 < \eta < \pi/2 & \text{lâmina 1} \\ -\pi/2 < \omega < \pi/2 & \\ \pi/2 < \omega < 3\pi/2 & \text{lâmina 2} \end{array} \quad (\text{A.18})$$

Toróide

O toróide é, talvez, o objeto menos usado em aplicações reais de modelagem mas ainda aparece como um padrão em pacotes de modelagem e *rendering* à frente de outras primitivas geométricas. Em todo caso, dadas as convenções abaixo, ou seja, definidos o raio r_0 a partir do centro para o meio do anel do toro, e o raio r_1 da seção cruzada do anel do toro.

Os pontos do toro são definidos como:

Figura A.3: Toro de raio interno r_0 , e raio da seção cruzada r_1 .

$$\begin{aligned} x &= \cos(\theta) [r_0 + r_1 \cos(\phi)] \\ y &= \sin(\theta) [r_0 + r_1 \cos(\phi)] , \\ z &= r_1 \sin(\phi) \end{aligned} \quad \begin{array}{ll} 0 \leq \theta < 2\pi \\ 0 \leq \phi < 2\pi \end{array} \quad (\text{A.19})$$

Ou na forma implícita:

$$z^2 + (\sqrt{x^2 + y^2} - a)^2 - b^2 = 0 \quad (\text{A.20})$$

Usando a formulação acima, um toro centralizado com raio $r_0 = 1$ e raio externo $r_1 = 0,25$ será algo do tipo:

$$(r - \alpha)^2 = \left(\frac{x_3}{\alpha_3} \right)^2 = 1 \quad (\text{A.21})$$

$$r = \sqrt{\left(\frac{x_1}{\alpha_1} \right)^2 - \left(\frac{x_2}{\alpha_2} \right)^2} \quad (\text{A.22})$$

Figura A.4: Toro de raio interno $r_0 = 1$, e raio da seção cruzada $r_1 = 0,25$.

$$\underline{x}(\eta, \omega) = \begin{bmatrix} \alpha_1(\alpha + \cos \eta) & \cos \omega \\ \alpha_2(\alpha + \cos \eta) & \sin \omega \\ \alpha_3(\alpha + \sin \eta) & \end{bmatrix}, \quad \begin{matrix} -\pi \leq \eta < \pi \\ -\pi \leq \omega < \pi \end{matrix} \quad (\text{A.23})$$

A.1.4 Super-Quádricas

As super-quádricas pertencem a uma família de objetos paramétricos 3D, úteis para modelagem geométrica. A partir do ajuste de alguns poucos parâmetros, uma grande variedade de formas pode ser obtida.

Consistem de uma extensão das quádricas, em que o expoente das coordenadas não é necessariamente 2. A forma geral de uma super-quádrica, já alinhada com os eixos e transladada para a origem é:

$$(ax)^n + (by)^n + (cz)^n = k \quad (\text{A.24})$$

As super-quádricas podem ser divididas em 4 formas paramétricas: super-elipsóides, super-hiperbolóides de uma parte, super-hiperbolóides de duas partes, e super-toróides. Barr (1981) deriva um conjunto de fórmulas para as representações, vetores normais, vetores tangentes, e funções internas e externas para cada uma destas formas.

As super-quádricas são derivadas das superfícies quádricas, cuja flexibilidade é obtida pela variação de cada termo trigonométrico de acordo com um expoente. Estes expoentes controlam o arredondamento e o achatamento em ambas direções horizontal e vertical.

Assim como as quádricas, as super-quádricas possuem normais e tangentes bem-definidas. Barr (1981) também inclui uma função interna-externa para cada tipo de super-quádrica, particularmente útil na aplicação de técnicas geométricas de sólidos. As equações a seguir descrevem o modelo das super-quádricas. Outras equações podem ser encontradas em [Barr 1981].

As Figuras A.5, A.6, A.7 e A.8 exibem as super-quádricas, e uma variedade de formas obtidas pela variação dos expoentes, de 0.1 a 3.0. A componente horizontal aumenta da esquerda para a direita, e a componente vertical aumenta de cima para baixo.

Super-elipsóide

O super-elipsóide é o nome dado a uma família de formas obtidas através do produto esférico de duas curvas super-quadráticas. Estas formas podem ser usadas para modelar uma grande variedade de modelos, incluindo esferas, cilindros, e paralelepípedos.

O super-elipsóide também pode ser escrito como:

n_1	n_2	r_x, r_y, r_z	forma resultante
0	0	-	caixa
0	0	$r_x = r_y = r_z$	cubo
< 1	< 1	-	cubóide
< 1	1	-	forma de almofada
1	< 1	-	cilíndrica
1	1	-	super-elipsóide
1	1	$r_x = r_y = r_z$	esfera
2	2	$r_x = r_y = r_z$	octaedro
n_1 ou $n_2 > 2$	-	-	apertada
n_1 ou $n_2 = 2$	-	-	chanfrada
∞	∞	-	cruz 3D

Tabela A.2: Possíveis formas de uma super-elipse 3D, de acordo com n_1 e n_2 .

$$f(x, y, z) = \left[\left| \frac{x}{r_x} \right|^{2/n_2} + \left| \frac{y}{r_y} \right|^{2/n_2} \right]^{n_2/n_1} + \left| \frac{z}{r_z} \right|^{2/n_1} \quad (\text{A.25})$$

onde as mesmas relações, como no caso 2D, podem ser feitas:

- Se $f(x, y, z) = 1$, (x, y, z) está na superfície do super-elipsóide;
- Se $f(x, y, z) < 1$, (x, y, z) está no interior do super-elipsóide;
- Se $f(x, y, z) > 1$, (x, y, z) está fora do super-elipsóide;

Expressões fechadas podem ser derivadas da normal em qualquer ponto do super-elipsóide, fazendo vários cálculos de *rendering* com modelos de iluminação mais fáceis. A normal é dada por:

$$\begin{aligned} N_x(\eta, \omega) &= \frac{1}{r_x} \cos^{2-n_1}(\eta) \cos^{2-n_2}(\omega) \\ N_y(\eta, \omega) &= \frac{1}{r_y} \cos^{2-n_1}(\eta) \sin^{2-n_2}(\omega) \\ N_z(\eta, \omega) &= \frac{1}{r_z} \sin^{2-n_1}(\eta) \end{aligned} \quad (\text{A.26})$$

De forma interessante, se n_1 e n_2 são menores que 2, então a equação da normal do super-elipsóide é outra super-elipsóide (seu complemento) com $n_1 \Rightarrow 2 - n_1$ e $n_2 \Rightarrow 2 - n_2$. Por exemplo, as normais de uma super-elipsóide com $n_1 = 0,5$ e $n_2 = 0,2$ é outra super-elipsóide (escalada adequadamente) com $n_1 = 1,5$ e $n_2 = 1,8$.

A Equação A.27 descreve a forma paramétrica da super-elipsóide:

$$\underline{x}(\eta, \omega) = \begin{bmatrix} \alpha_1 & \cos^{\epsilon_1} \eta & \cos^{\epsilon_2} \omega \\ \alpha_2 & \cos^{\epsilon_1} \eta & \sin^{\epsilon_2} \omega \\ \alpha_3 & \sin^{\epsilon_1} \eta & \end{bmatrix}, \quad \begin{matrix} -\pi \leq \eta \leq \pi \\ -\pi \leq \omega < \pi \end{matrix} \quad (\text{A.27})$$

Super-hiperbolóide de uma lâmina

$$\underline{x}(\eta, \omega) = \begin{bmatrix} \alpha_1 & \sec^{\epsilon_1} \eta & \cos^{\epsilon_2} \omega \\ \alpha_2 & \sec^{\epsilon_1} \eta & \sin^{\epsilon_2} \omega \\ \alpha_3 & \tan^{\epsilon_1} \eta & \end{bmatrix}, \quad \begin{matrix} -\pi < \eta < \pi \\ -\pi \leq \omega < \pi \end{matrix} \quad (\text{A.28})$$

Figura A.5: Variações de super-elipsóides de acordo com os expoentes e_1 e e_2 .

Super-hiperbolóide de duas lâminas

$$\underline{x}(\eta, \omega) = \begin{bmatrix} \alpha_1 & \sec^{\epsilon_1} \eta & \sec^{\epsilon_2} \omega \\ \alpha_2 & \sec^{\epsilon_1} \eta & \tan^{\epsilon_2} \omega \\ \alpha_3 & \tan^{\epsilon_1} \eta & \end{bmatrix}, \quad \begin{array}{ll} -\pi/2 < \eta < \pi/2 & \text{lâmina1} \\ -\pi/2 < \omega < \pi/2 & \\ \pi/2 < \omega < 3\pi/2 & \text{lâmina2} \end{array} \quad (\text{A.29})$$

Super-toróide

O super-toróide é uma família de primitivas geométricas baseada nos toróides, onde r_0 é o raio do anel interno e r_1 é o raio do anel externo. Os ângulos θ e ϕ variam de 0 a 2π .

A equação para o super-elipsóide é a mesma para o toróide, exceto pelo fato de que os termos seno e cosseno são elevados a potências.

São diferentes valores desses expoentes que aumentam a variedade da família de toróides. O valor de n_1 determina a forma do anel do toróide, e n_2 determina a forma da

Figura A.6: Variações de super-hiperbolóides de 1 lâmina, de acordo com os expoentes e_1 e e_2 .

seção que atravessa o anel.

Exemplos de super-toróides, gerados a partir de diferentes valores de n_1 e de n_2 são mostrados na figura abaixo. Os valores possíveis para n_1 e n_2 formam um intervalo contínuo, de 0 ao ∞ (embora haja problemas de representação próximo a 0 e acima de 5).

$$\underline{x}(\eta, \omega) = \begin{bmatrix} \alpha_1(\alpha + \cos^{\epsilon_1} \eta) & \cos^{\epsilon_2} \omega \\ \alpha_2(\alpha + \cos^{\epsilon_1} \eta) & \sin^{\epsilon_2} \omega \\ \alpha_3(\alpha + \sin^{\epsilon_1} \eta) & \end{bmatrix}, \quad \begin{matrix} -\pi \leq \eta < \pi \\ -\pi \leq \omega < \pi \end{matrix} \quad (\text{A.30})$$

Figura A.7: Variações de super-hiperbolóides de 2 lâminas, de acordo com os expoentes e_1 e e_2 .

Figura A.8: Variações de super-toróides, de acordo com os expoentes e_1 e e_2 .

Implementação

B.1 Java 3D

Tradicionalmente, a programação gráfica 3D tem sido procedimental. Grande parte do código hoje é escrito em C, potencialmente com algumas instruções de montagem *inline*. Java 3D é uma tentativa de criar um novo paradigma, totalmente orientado a objetos, que visa criar um *framework* para elaborar um universo virtual 3D.

O Java 3D consiste de uma API ¹ da hierarquia de classes Java, que serve como uma interface de um sistema de *rendering* de gráficos 3D e de som. Trata-se de um *framework* em que o programador se depara com uma API com nível de abstração elevado. Esta API provê construtores de alto-nível para criar e manipular geometrias 3D, e construir estruturas usadas para *renderizar* estas geometrias.

O conceito de *scene graph* é fundamental para a construção das cenas em Java 3D. Consiste de uma árvore de estrutura de dados, cuja principal função é anexar os objetos físicos do mundo 3D juntamente com seus comportamentos. Entretanto, há um conjunto de regras sobre como construir um *scene graph* válido, as quais devem ser seguidas rigorosamente segundo a documentação, para que se obtenha o comportamento esperado dos objetos. Os fragmentos de código a seguir são sub-rotinas de uma aplicação Java que usa o JOGL (*Java bindings for OpenGL*, ou seja, programação direta em OpenGL) e o Java3D para exibir um cubo rotacionando sobre o eixo Y. A diferença de estilos entre as API's Java 3D e JOGL podem ser observadas.

```
/* Código em JOGL */
```

```
public void display(GLDrawable glDrawable) {  
    GL gl = glDrawable.getGL();  
  
    gl.glRotatef(ANGLE, 0, 1, 0);
```

¹ *Application Program Interface*

```

gl.glClear(GL.GL_COLOR_BUFFER_BIT);

gl.glBegin(GL.GL_QUADS);
gl.glVertex3f(-70, 70, 0);
gl.glVertex3f(70, 70, 0);
gl.glVertex3f(70, -70, 0);
gl.glVertex3f(-70, -70, 0);
gl.glEnd();
}

/* Código em Java 3D */

private BranchGroup createScene() {
    BranchGroup scene = new BranchGroup();

    TransformGroup spin = new TransformGroup();
    spin.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
    scene.addChild(spin);

    spin.addChild(new ColorCube(CUBE_SIZE));

    Alpha rotation = new Alpha(-1, 4000);

    RotationInterpolator rotator = new RotationInterpolator(rotation, spin);

    BoundingSphere bounds = new BoundingSphere();
    rotator.setSchedulingBounds(bounds);
    spin.addChild(rotator);

    scene.compile();
    return scene;
}

```

Nota-se claramente a diferença entre os 2 estilos de programação: o primeiro - JOGL - consiste da programação básica em OpenGL, ou seja, em baixo nível. No entanto, no segundo fragmento de código, percebe-se nitidamente a orientação a objetos, que, inclusive, independe de qual API gráfica o sistema utiliza (OpenGL ou DirectX).

Rotacionar um cubo, como no exemplo mencionado, é uma tarefa simples. Entretanto, quando mais objetos estão *renderizados* na tela, cada qual com seus comportamentos, atributos e interações possíveis, o universo se torna mais complexo, demandando um maior poder de gerenciamento e eficiência sobre tudo o que ocorre na visualização. Desta forma, a abordagem orientada a objetos é a mais indicada.

Além dos testes realizados em [McDonald e Erbacher 2005], que comprovam melhor eficiência do Java3D frente ao JOGL para universos complexos, as características que tornam o Java 3D uma opção interessante são:

- Gerenciamento da cena simplificado, pois a construção de visualizações parte da definição do *scene graph*;

- Detalhes de *rendering* são gerenciados automaticamente, assim como cálculo de normais e iluminação de malhas;
- Utilização de *threads*, melhorando a eficiência do processo de *rendering* paralelo, quando vários objetos coexistirem na cena;
- O *renderer* do Java 3D otimiza automaticamente os objetos em cena, para melhorar a eficiência da exibição;
- Mobilidade e facilidade de instalação do *plugin* Java 3D, permitindo à aplicação executar diretamente do navegador de Internet, sob qualquer plataforma.

B.1.1 Elementos de uma aplicação Java3D

Como já foi discutido, uma aplicação em Java3D deve seguir uma estrutura rigorosa para que se obtenha o comportamento desejado dos elementos em cena. Um *scene graph* típico (Figura B.1) possui os seguintes itens:

Figura B.1: Exemplo de *scene graph* com controle da visão. Figura adaptada de [<http://download.java.net/media/java3d/javadoc/1.4.0/javafx/media/j3d/doc-files/intro.html>].

- *Virtual Universe*: Trata-se do *container*² de mais alto nível, que contém todos os *scene graphs*. Um universo virtual consiste de um conjunto de *Locale objects*, cada qual com uma posição de alta resolução dentro do universo virtual;
- *Locale object*: Este objeto define a origem das coordenadas do mundo virtual para todos os objetos anexados a ele;

²Recipiente

- *BranchGroup nodes*: Estes nós servem como ponteiros para a raiz de uma ramificação do *scene graph*, e são os únicos que podem ser inseridos em *Locales*. *BranchGroups* podem conter outros *BranchGroups*, e quando são compilados, todos seus filhos também são;
- *TransformGroup node*: Um *TransformGroup* especifica uma transformação espacial singular, através de um objeto *Transform3D*, que pode posicionar, orientar e escalar todos seus filhos;
- *Shape3D node*: O nó *Shape3D* contém uma lista de um ou mais objetos-componentes *Geometry*, e um objeto *Appearance*. O objeto *Geometry* define os dados geométricos do modelo, e o objeto *Appearance* especifica os atributos de aparência do modelo, incluindo cor, material, textura, entre outros;
- *Behavior node*: Provê um *framework* para adicionar ações, definidas pelo usuário, dentro do *scene graph*. Possui internamente métodos para processamento de estímulos, condições de disparo, sincronização de intervalos e regiões, etc.;
- *ViewPlatform object*: Este nó posiciona a visão dentro do *scene graph*. Assim, define um sistema de coordenadas interno, e forma uma estrutura de referência com seu ponto de referência, dentro do mundo virtual.
- *View*: O objeto *View* contém todos os parâmetros necessários para *renderizar* uma cena 3D por um ponto de vista. Uma visão contém uma lista de objetos *Canvas3D*, os quais fornecem a tela para ser desenhada.

B.2 *Spider Cursor* - C++ e VTK/STK

A primeira versão, desenvolvida no *Microsoft Visual C++ 6.0*, mantém a mesma base do DSVol, fazendo uso da biblioteca gráfica VTK e da biblioteca sonora STK. Desde então, o *Spider Cursor* tem sido aperfeiçoado, com a adição de novas características que o tornam mais apropriado para uma quantidade maior de aplicações. As melhorias implementadas foram:

- Criação de uma interface gráfica para o uso do *cursor*, utilizando o *Microsoft Visual C++ 6* (Figura 4.2);
- Criação de uma classe para gerenciar e configurar parâmetros da visualização e sonificação da ferramenta;
- Adição da visualização de linhas de contorno;
- Adição da visualização baseada em conectividade de células;
- Criação de um filtro VTK que transforma *PolyData Cells* [Schroeder, Martin e Lorensen 2002] em pontos desconexos, para obter um *scatterplot*;
- Adição da visualização de *glyphs* 3D, que utiliza o filtro de conversão para *scatterplot*. Os pontos são convertidos para objetos 3D: cubos, esferas, cilindros, discos e cones;

- Criação de outro filtro VTK, denominado `vtkHeightFields`, que mapeia a coordenada z ou um campo escalar qualquer para altura. Também é possível obter uma visualização planar, sem altura;
- Mapeamento de som configurável: o usuário pode escolher entre vários instrumentos, além de definir a faixa de variação da frequência dos sons em Hertz. Além disso, é possível mapear qualquer campo escalar para que seja mapeado para som;
- Mapeamento de cor configurável: o usuário pode determinar qual atributo dos dados deseja mapear para cor;
- Mapeamento de altura configurável: utilizando o `vtkHeightFields`, o usuário tem a possibilidade de remapear a altura, normalmente atribuída à coordenada z , para qualquer campo escalar de seu interesse;
- Busca por rótulo dos pontos, posicionando o *Spider Cursor* sobre o dado correspondente (Figura B.2).

Figura B.2: Exemplo de utilização do *Spider Cursor*, realizando busca pela cidade de Cardoso.

- Aperfeiçoamento do corte, que possibilita alterar o conjunto de dados sob análise, excluindo partes irrelevantes à investigação ou regiões de interesse;
- Adição de tabelas de cores variadas, para melhor observar a variação das características dos dados, de acordo com a semântica passada [Tutida 1998].

B.2.1 Estrutura interna

Esta versão mantém a mesma estrutura do *Spider Cursor* desenvolvido no trabalho de doutorado [Salvador 2003]. No entanto, a aplicação foi reorganizada de forma modular, separando a interface gráfica, gerenciamento de funcionalidades e o motor de execução. As principais classes adicionadas foram:

- *Classe VisManager*: Esta classe é responsável por gerenciar os recursos ativados e os parâmetros configurados para a visualização e para a sonificação. Também define as tabelas de cores para maior abrangência semântica dos campos escalares (como discutido em [Tutida 1998]);
- *Classe vtkHeightFieldsFilter*: Este filtro possibilita mapear um campo escalar do conjunto de dados para a altura, ou seja, a coordenada Z dos vértices é definida de acordo com o campo escalar atribuído ao filtro. Desta forma, padrões ou características dos dados podem ser facilmente identificadas;
- *Classe GUI*: A classe *GUI* consiste da interface gráfica desenvolvida para acessar todos os recursos disponíveis para a exploração dos dados.

B.3 Super Spider - Java e Java3D

A segunda versão do *Spider Cursor*, denominada *Super Spider*, foi desenvolvida no *Net-Beans 4.1* e totalmente reprogramada em Java, inclusive utilizando outra biblioteca gráfica que utiliza OpenGL ou DirectX: o Java 3D. Desta forma, para utilizar o *Super Spider*, basta ter os *plugins* Java e Java 3D instalados, e executar a ferramenta a partir do navegador de Internet.

Trata-se de uma funcionalidade indispensável para ferramentas de visualização computacional - tanto visualização científica quanto visualização de informação - que ganham mais utilidade por causa da portabilidade. Portanto, é possível explorar e manipular dados a partir de qualquer navegador Web, sob qualquer Sistema Operacional.

B.3.1 Estrutura interna

A estrutura interna, com as principais classes da ferramenta e suas trocas de mensagem, é apresentada na Figura B.3. Uma descrição mais detalhada das funcionalidades é discutida a seguir:

- *VTK File*: Representa o arquivo VTK com a projeção e os campos escalares dos dados a serem visualizados;
- *NAMES File*: Indica o arquivo NAMES de rótulos para os dados visualizados;
- *LABEL File*: Representa um arquivo de rótulos alternativos ao NAMES;
- *Classe CScalarField*: A classe *ScalarField* realiza o armazenamento dinâmico dos campos escalares do arquivo VTK, guardando os nomes dos campos, seus respectivos escalares e valores mínimos e máximos;

- *Classe CFileReader*: Esta classe realiza a leitura do arquivo VTK e do arquivo NAMES, estejam armazenados localmente ou acessíveis remotamente, via HTTP;
- *Classe CNeighborhood*: Esta classe é responsável pelo cálculo da *Rightness Rate* (taxa de acerto) da projeção. Esta taxa consiste na verificação dos vizinhos de um determinado ponto: se os vizinhos mais próximos no espaço original forem os mesmos da projeção, então a taxa de acerto é 100%. Caso esta taxa seja baixa, significa que a projeção possui uma alta taxa de Erro embutida;
- *Classe CDistanceMatrix*: A matriz de distâncias, por se tratar de um arquivo volumoso, necessita de um tratamento especial, para otimizar a busca e, principalmente, diminuir o espaço ocupado em memória. Esta classe realiza esta função, carregando para a memória apenas o triângulo superior da matriz, e disponibilizando métodos para acesso das posições da matriz, calculando distâncias;
- *Classe CFractalDimension*: O cálculo da Dimensão Fractal é realizado na região ao redor do ponto selecionado, ou seja, analisando sua vizinhança através do método Massa-Raio (Seção 4.4.3) e indicando o grau de complexidade da região;
- *Visual Techniques*: Este conjunto de classes contém as técnicas de visualização possíveis para os dados: iso-superfícies, isolinhas, *wireframe* e *scatterplot*;
- *Classe CDataStructure*: Esta classe fornece a estrutura de dados necessária ao *Super Spider* para gerenciar e consultar os dados geométricos e escalares.
 - Armazenamento e normalização das coordenadas, para que a projeção fique centralizada na tela;
 - Armazenamento das células e dos respectivos vértices;
 - Relação dos vizinhos de cada vértice;
 - Inserção dos escalares na classe *CScalarField*;
 - Ordenação dos rótulos do arquivo NAMES, para facilitar a busca na caixa de seleção *Point label*;
 - Indexação do arquivo NAMES ordenado com os vértices da projeção;
- *Classe CSpiderCursor*: Esta classe recebe a estrutura de dados e é disparada pela classe *CWatcher*. Realiza as seguintes funções:
 - Realiza o *rendering* do *Super Spider* e das Super-Quádricas - super-elipsóide, super-hiperbolóide de 1 lâmina, super-hiperbolóide de 2 lâminas e super-toróide - de acordo com a configuração dos parâmetros;
 - Realiza a normalização dos parâmetros, para que se adequem ao mapeamento dos campos escalares;
 - Inverte a cor do cursor em relação à superfície que esteja sondando, mantendo o cursor sempre visível;
- *Classe CCut*: Disparada pelo *Super Spider*, este objeto realiza e grava a linha de corte, verifica o término da seleção, e grava as regiões interna e externa ao corte, bem como seus respectivos rótulos (NAMES e LABEL). Mais detalhes são descritos na Seção 4.3.3;

- *Classe CWindow*: Esta janela serve como um *pop-up*³, que desloca a super-quádrica da janela principal para esta janela separada, liberando campo da visualização;
- *Classe JMainFrame*: Esta classe possui a tela para desenho propriamente dita. Portanto, ela é responsável pelas técnicas de iso-superfícies, *scatterplots* e linhas de contorno, assim como atribuir cores, coordenadas e normais aos vértices.
- *Classe CWatcher*: A classe *CWatcher* é responsável pelo tratamento de eventos, direcionando-os corretamente à classe *CSpiderCursor*, informando o vértice e a célula sob observação;

Figura B.3: Diagrama de componentes, enfatizando as trocas de mensagens entre as classes do *Super Spider*.

³Janela de destaque que se abre separadamente da principal.