Ensino de Estruturas de Dados e seus Algoritmos através de Implementação com Animações

Pedro J. de Rezende
Islene C. Garcia

{rezende|islene}@dcc.unicamp.br

Relatório Técnico IC–96-14

Novembro de 1996

Ensino de Estruturas de Dados e seus Algoritmos através de Implementação com Animações*

Pedro J. de Rezende
Islene C. Garcia

{rezende|islene}@dcc.unicamp.br

Sumário

Descrevemos as abordagens *passiva* e *ativa* utilizadas para ensino de algoritmos e estruturas de dados com recursos de animações gráficas. Introduzimos uma nova abordagem *construtiva* e descrevemos o ambiente Astral projetado para explorar seus benefícios. Nesta abordagem, além do estudante ter acesso a aplicativos exemplos providos de animações gráficas que ilustram o funcionamento de algoritmos, ele também realiza suas próprias animações durante o processo de implementação da estrutura e dos algoritmos de maneira simplificada mas com visualização gráfica notável e resultados sensivelmente positivos na aprendizagem. A experiência de uso deste ambiente tem mostrado grandes vantagens no ensino de disciplinas de graduação na área de computação.

1 Introdução

Diversos sistemas para implementação de animações de algoritmos e de estruturas de dados foram produzidos deste o trabalho pioneiro de Brown [3, 4], como por exemplo, [7, 15, 1]. Vários destes sistemas exploram muito bem a pontencialidade do uso de visualizações gráficas das operações realizadas nas estruturas de dados como ferramenta de ensino. O uso de movimento em tempo real, cores e sons enriquecem ainda mais o poder de comunicação [6].

De fato, algumas experiências positivas já foram relatadas na literatura [10] e [13], principalmente com o uso de sistemas que utilizam a abordagem *ativa* (veja seção 2) na qual o aprendiz interage com as animações na criação de instâncias testes.

Por outro lado, a ênfase na qualidade da animação gráfica para enriquecer a abordagem ativa da maioria dos sistemas levou-os a um grau de complexidade que impede seu uso para fins de *criação* de animações pelos próprios aprendizes. Deste modo, sua utilização se limita à interação, ainda que ativa, com as animações pré-implementadas.

*Pesquisa desenvolvida com suporte financeiro parcial do CNPq, Faep, Fapesp, e RHAE.

Do mesmo modo que a experimentação durante a operação das animações enriquece o aprendizado mais do que a mera observação passiva delas, é de se esperar que, com um sistema onde a própria implementação das animações gráficas é facilitada a ponto de poder ser realizada pelo estudante, a absorção do funcionamento dos algoritmos seja ainda mais intensa.

Com o objetivo de prover um tal sistema, desenvolvemos um ambiente para a arquitetura Macintosh utilizando a plataforma de programação THINK Pascal produzida pela Symantec Corp. <http://www.symantec.com/>.

Este artigo traz na seção 2 uma breve descrição das principais abordagens ao uso de animações de algoritmos no ensino. Na seção 3 são citados vários dos projetos mais relevantes na área. Descrevemos na seção 4 o ambiente que desenvolvemos, detalhando sua arquitetura e apresentando alguns exemplos. Finalmente, na seção 5 são colocadas algumas conclusões sobre os benefícios deste ambiente, tendo em vista a nossa experiência com seu uso no ensino de disciplinas de estruturas de dados.

2 Abordagens

Em livros textos e aulas convencionais, de modo a faciliar o aprendizado de algoritmos utilizam-se mecanismos que vão além da descrição em (pseudo) linguagem de programação, como ilustrações.

Por outro lado, o excesso destas mesmas ilustrações quando vistas estaticamente contribuem muito pouco para aumentar a compreensão dos algoritmos dado que além das informações depictadas há grande comunicação implícita na correlação entre elas. Esta correlação pode ser mostrada através da animação gráfica das imagens, dando ganho considerável de comunicação de informação.

Em [3], M. Brown introduziu a idéia de ambientes dedicados à construção de animação gráfica computadorizada como ferramenta para ensino em computação como já é também aplicado com sucesso em outras áreas, como química, anatomia e simulação de circuitos.

Um primeiro uso de animação de algoritmos é dado quando se assiste a um filme [11] ou à execução de uma seqüência pré-programada de eventos [3]. Essa abordagem é denominada *passiva*, pois o usuário se comporta como mero espectador e ela é útil para a complementação de livros-texto e aulas.

Uma limitação clara, no entanto, está no fato de o usuário não poder testar novos conjuntos de dados e ter que ficar limitado aos conjuntos a ele fornecidos. Para suprir esta restrição, alguns sistemas [7, 15, 1] introduziram facilidades para que seus usuários pudessem fazer as suas próprias experiências, direcionando o andamento do algoritmo. Tal abordagem é denominada *ativa*.

Resultados positivos para o uso desta abordagem foram relatados por Lawrence *et al* em [13] em experiência realizada com o ambiente POLKA. Estes resultados confirmam que a abordagem ativa é mais poderosa, pois tem-se uma participação maior por parte do usuário.

Diante da oportunidade de associar o ensino teórico de algoritmos e estruturas de dados com uma disciplina de laboratório de software, decidimos verificar se o acréscimo de um nível adicional de participação do aprendiz no processo de animação resultaria em ganho na aprendizagem. Nossa tentativa visou incorporar à fase de implementação realizada pelos

estudantes algumas chamadas a rotinas gráficas de alto nível para visualização tanto das estruturas de dados quanto das atuações dos algoritmos implementados para manipulação delas. Para viabilizar esta tarefa fez-se necessária a implementação de um ambiente de programação de animações em uma plataforma adequada e amigável. Criamos, para implementar esta nova abordagem, denominada *construtiva*, o ambiente Astral descrito na seção 4 com o qual mesmo o estudante iniciante pode realizar animações gráficas sofisticadas com um mínimo de esforço.

Este ambiente foi utilizado em laboratório de software durante três semestres (2/94, 1/95, 2/95) com resultados muito positivos à aprendizagem, como descrevemos na seção 5.

3 Projetos Relacionados

Nesta seção são citados alguns trabalhos que influenciaram positivamente o projeto e desenvolvimento do ambiente Astral. Não houve pretensão alguma aqui de fazer uma resenha exaustiva de todos os trabalhos relacionados.

Balsa (Brown ALgorithm Simulator and Animator) [3] foi desenvolvido na Brown University no início dos anos 80 e seu intuito era o de servir como um laboratório de experimentação com representações dinâmicas de algoritmos em tempo real. Seu sucessor, Balsa-II introduziu uma interface mais amigável e uma maneira homogênea para a execução e interações com animações, trabalhando sobre a plataforma Macintosh e com a linguagem Pascal.

No início dos anos 90, surgiu o projeto Zeus <`http://www.research.digital.com/SRC/zeus/home.html`> de M. Brown [7, 5, 8, 9] cujas contribuições incluiram uma proposta de separação entre algoritmo e visualizadores, colocando interfaces bem definidas entre eles; o suporte à construção de programas de grande porte, orientados a objetos e a utilização da linguagem Modula 3. Em Zeus múltiplos algoritmos podem ser executados concorrentemente, facilitando a análise de diferenças e similaridades entre vários algoritmos.

O projeto ZADA <`http://ls4-www.informatik.uni-dortmund.de/RVS/zada.html`> desenvolvido na University of Dortmund implementou algoritmos distribuídos utilizando o ambiente Zeus.

XTango <`http://www.cc.gatech.edu/gvu/softviz/algoanim/xtango.html`> [15] é um sistema de animação de algoritmos de propósito genérico que permite a construção de animações em tempo real. O objetivo principal é prover facilidades para os usuários que irão construir as suas animações, fornecendo uma interface de alto nível. Utiliza a linguagem C e opera sobre plataformas Unix e X11 Window Systems. Polka <`http://www.cc.gatech.edu/gvu/softviz/parviz /polka.html`> é sucessor de Xtango e foi projetado para dar suporte ao desenvolvimento de animações concorrentes, de maneira a facilitar a exibição de algoritmos paralelos.

O projeto AnimA [1] foi desenvolvido na Unicamp por R. Amorim e P. de Rezende e seu principal objetivo é a produção sistemática de animações em C++ dentro de um ambiente de estações de trabalho Unix. As experiências obtidas através da implementação desse projeto foram de grande valia para a implementação do Astral.

4 Astral

O projeto do ambiente Astral <http://www.dcc.unicamp.br/~rezende/astral> visou atender as necessidades de preparação de diversos exercícios de implementação de estruturas de dados utilizando animações gráficas, de modo homogêneo sob uma interface consistente. A partir das características básicas do ambiente e de sua biblioteca de suporte, foram produzidos diversos destes exercícios que deram origem à primeira experiência de que temos conhecimento da aplicação da abordagem *construtiva* no ensino de estruturas de dados — o uso de mecanismos de animação para auxiliar na implementação de algoritmos pelos próprios estudantes.

O ambiente Astral foi desenvolvido para arquitetura *Macintosh* [2], sobre Mac/OS v. 7.5. Um ponto positivo desta escolha é relacionado às facilidades oferecidas por esta plataforma para a implementação de aplicativos gráficos, com mecanismos simples para gerenciamento de janelas, *menus*, assim como para o desenho e manipulação de objetos gráficos. Além disso, o fato de a interface oferecida ao usuário final ser extremamente amigável (e portanto excelente para pessoas com pouca experiência) faz o Astral ter grande apelo a esse público.

A linguagem escolhida para implementação foi Pascal que é adequada ao ensino de programação básica devido à sua simplicidade, verificação forte de tipos e mecanismos para estruturação e modularização. O processo de animação foi feito utilizando-se funções e procedimentos, sem alterações à linguagem ou utilização de pré-processadores.

Em um nível acima do sistema operacional, temos uma camada que agrupa rotinas de interesse geral para a construção de animações que obedeçam ao nosso modelo, denominada *camada básica*. Fazem parte desta camada o encapsulamento de tópicos como manipulação de janelas de texto, a visualização e a comunicação com o usuário da aplicação assim como o tratamento de eventos e o gerenciamento de objetos gráficos.

Para cada animação é construída uma *camada específica* que contém o código referente à visualização da estrutura de dados que se pretende animar. Ao usuário do ambiente caberá escrever o seu algoritmo fazendo uso das interfaces oferecidas pelo conjunto de camadas que estão em níveis inferiores, como ilustra a figura 1.

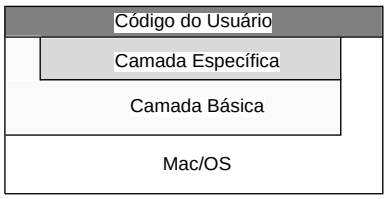


Figura 1: Arquitetura básica do ambiente

As interfaces oferecidas ao usuário do ambiente são escolhidas de maneira a formar, para cada animação, um conjunto minimal e padronizado, não sobrecarregando a tarefa do aprendizado de sua utilização. É conseqüência deste requisito que a qualidade da animação final, em termos de detalhes dos passos exibidos, é função da criatividade do programador da camada específica que deve maximizar o conteúdo semântico da informação mostrada graficamente, sem estender o número de chamadas gráficas. Para ilustrar o poder que pode ter um tal conjunto minimal de chamadas gráficas, a figura 4, na seção 4.2, mostra que uma

única chamada à função `DrawTree` é suficiente para a animação das alterações de uma árvore AVL devidas a uma inserção, incluindo os efeitos de rotação.

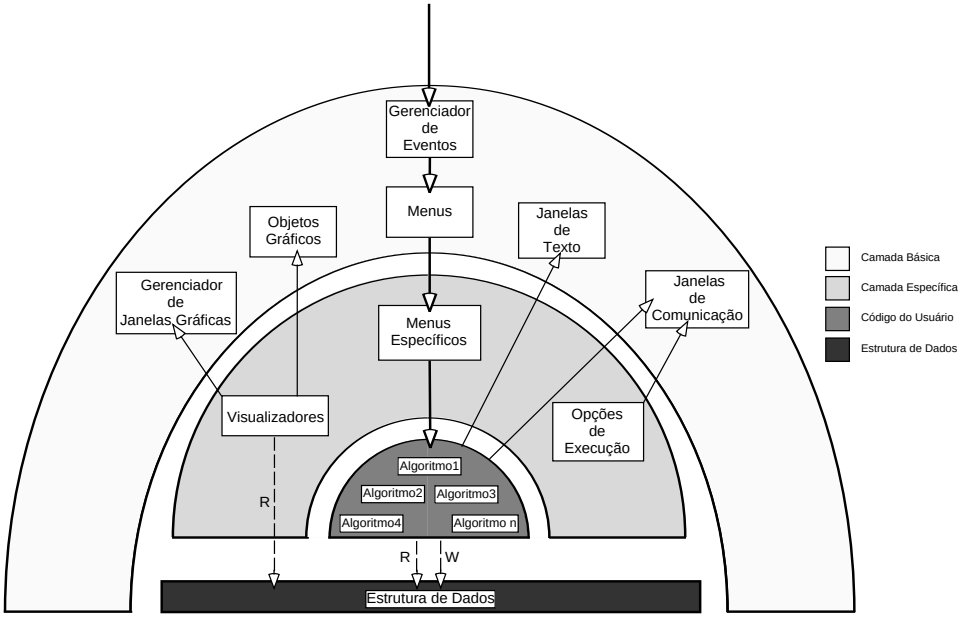


Figura 2: Arquitetura detalhada do ambiente

4.1 Arquitetura

A figura 2 ilustra de maneira mais detalhada o conteúdo de cada uma das camadas e os principais relacionamentos entre elas e mostra, de maneira resumida, o fluxo de execução de uma animação. O usuário que a executa gera eventos, como seleção de itens de *menu* ou entrada de dados, que são filtrados pelo gerenciador de eventos. A maior parte destes eventos são tratados na camada básica e são transparentes para os demais programadores.

Eventos específicos para uma animação, como a seleção de um item de *menu* indicando um pedido de execução de um algoritmo é passado para a camada específica, que irá ativar o código do usuário. É este quem irá se encarregar de fazer as alterações devidas na estrutura de dados e chamar as rotinas adequadas dos visualizadores. A cada vez que um visualizador é acionado, ele verifica o estado da estrutura de dados e reflete as alterações feitas de maneira suave (veja um exemplo na figura 4).

4.2 Utilização do Ambiente

A finalidade de cada uma das animações, ou exercícios, é que o usuário entenda o seu objetivo, consiga acompanhar os passos necessários para alcançá-lo e seja capaz de fazer sua própria implementação de cada algoritmo. Para isto, além dos elementos do ambiente propriamente dito contamos com uma camada de apoio, ilustrada na figura 3. Tal camada é fundamental para o sucesso do processo de aprendizado e pode complementar aulas expositórias.

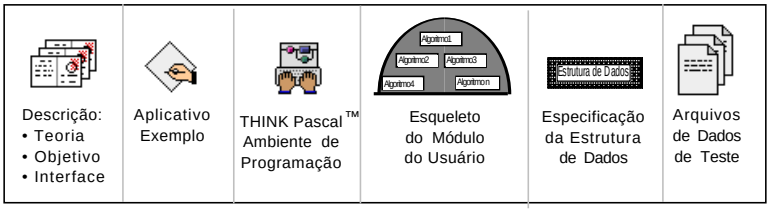


Figura 3: Camada de Apoio

Para a compreensão dos objetivos é fornecido um arquivo texto explicativo e um aplicativo-exemplo sobre o qual o usuário pode explorar o comportamento dos algoritmos com conjuntos de dados sugeridos ou próprios (participação ativa).

Para a fase implementação contamos com o ambiente de programação THINK Pascal [17], desenvolvido pela Symantec Corporation. Este contém um editor de texto que realiza verificação de erros sintáticos em tempo de digitação assim como indentação automática. Além disso, seu ambiente de depuração é bastante completo, permitindo a execução passo a passo, o estabelecimento de *breakpoints* (interrupção da execução em pontos específicos do código), de *watchpoints* (interrupção da execução quando determinada posição de memória é alterada) e depuração a nível de programa fonte, dentre outras facilidades.

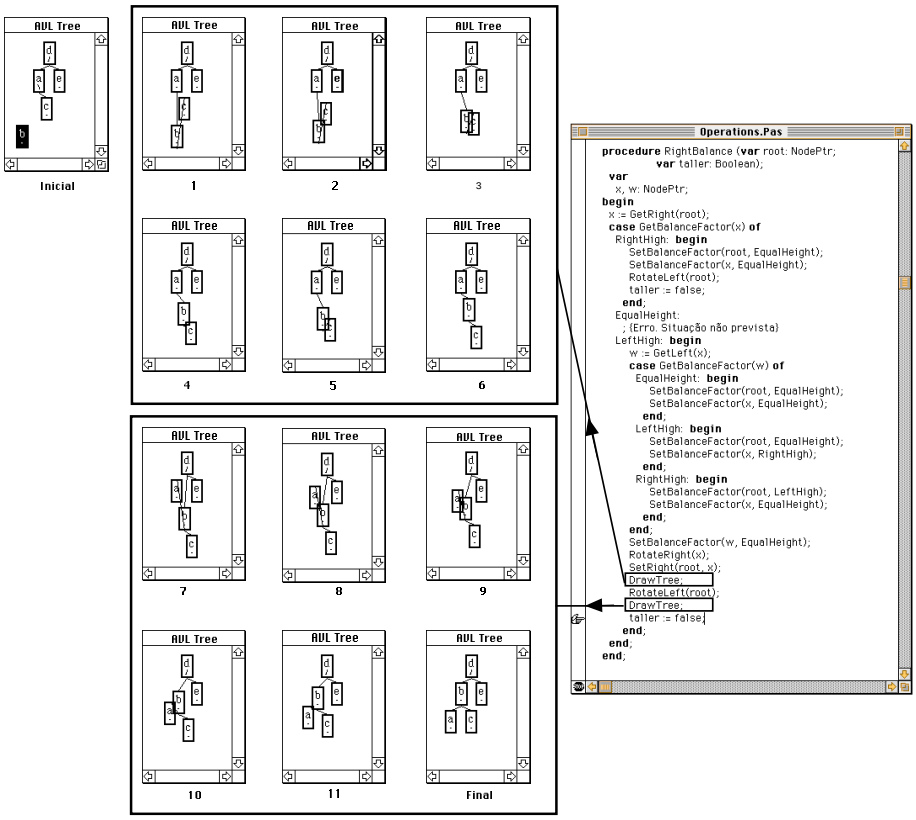


Figura 4: Animação de uma inserção em árvore AVL

Tal ambiente define uma aplicação como sendo um *project* formado por vários módulos. Desta forma, é entregue ao estudante um projeto que inclui um arquivo contendo a especificação da estrutura de dados e um modelo do módulo do usuário que consiste apenas de um esqueleto das operações que ele deve completar.

Além disso, a camada de apoio conta com um arquivo texto que contém as interfaces das rotinas de visualização que são usadas para a realização das animações pelo estudante. Podem ainda ser fornecidos arquivos de dados especiais para teste, como por exemplo um conjunto de elementos a serem inseridos em uma árvore AVL, causando rotações em série.

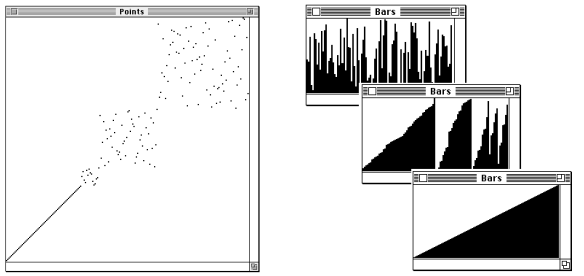
4.3 Vantagens da abordagem construtiva

Dentre as vantagens da abordagem construtiva estão mecanismos para facilitar o processo de abstração, o fato de a animação refletir o código implementado pelo aprendiz e as várias facilidades para a detecção visual de erros. Com isso, está-se incentivando o processo de compreensão e de auto-correção.

O uso de chamadas às rotinas gráficas pelo usuário desde as etapas iniciais de sua implementação se mostra uma excelente ferramenta de depuração. (Veja figura 4.) A informação visual acelera os processos de entendimento e desenvolvimento.

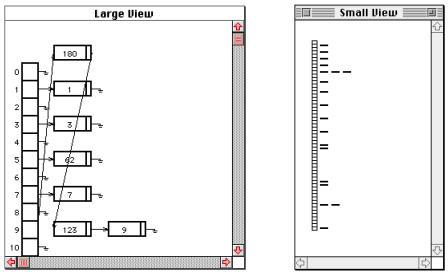
4.4 Exemplos de Animações Implementadas

Esta seção é dedicada à apresentação de figuras retiradas de algumas das animações implementadas por estudantes utilizando conjuntos minimais de funções gráficas providas pela biblioteca do ambiente Astral. Por limitações de espaço, serão colocadas apenas algumas imagens e uma breve explicação para cada um dos tópicos apresentados.



(a) (b)

Figura 5: Ordenação

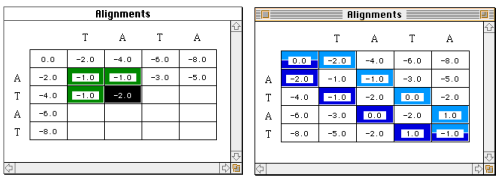


(a) (b)

Figura 6: Hashing

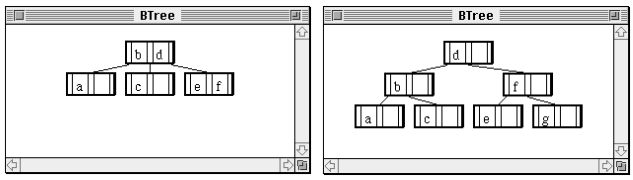
Iniciando com algoritmos de ordenação, podemos ver na figura 5(a) o *QuickSort*, cujo comportamento recursivo pode ser observado através da disposição dos elementos, que estão separados em grupos que vão se aproximando progressivamente da diagonal da janela. O segundo algoritmo, figura 5(b), ilustra o *MergeSort* cujo comportamento é exibido em três fases: antes da execução do algoritmo, com alguns grupos já ordenados e prontos para a realização de uma etapa de conquista do algoritmo e ao término da execução, com todos os elementos ordenados.

Na figura 6, temos uma amostra de um algoritmo de *hashing* utilizando encadeamento externo. Note que são utilizados dois tipos de visualizadores: um para mostrar em detalhes o conteúdo das entradas na tabela e o outro reduzido, destinado exibir o grau de espalhamento da tabela.



(a) (b)

Figura 7: Alinhamentos



(a) (b)

Figura 8: Árvores B

A figura 7 mostra uma aplicação do paradigma de programação dinâmica, cuja motivação veio da área de biologia computacional (determinação de alinhamento de cadeias de DNA). A estrutura de dados é, portanto, trivial, mas este problema ilustra o preenchimento da informação de modo dinâmico: o preenchimento da primeira linha e primeira coluna é imediato, enquanto o cálculo do valor dos demais elementos é baseado no valor já determinado de seus elementos vizinhos, como ressalta a figura 7(a). Dado que a matriz está completa, é possível determinar a partir dela vários alinhamentos. Na figura 7(b) estão marcados os alinhamentos superior e inferior.

Por fim, a figura 7(b) mostra uma árvore B antes e depois da inclusão de um elemento (g) que causa a explosão da raiz e conseqüentemente o aumento da altura da árvore.

Estes exemplos mostram a diversidade de exercícios que podem ser construídos baseando-se nas facilidades do ambiente Astral de modo que o estudante possa se beneficiar da sua *própria* implementação de animações gráficas como mecanismo de aprendizagem. A construção das camadas específicas a cada um destes exercícios é simplificada devido à camada básica do ambiente que fatora todo o gerenciamento de eventos e processamento de mais baixo nível para fora da camada específica.

5 Conclusões

Foi descrito o ambiente Astral para auxílio ao entendimento e implementação de algoritmos utilizando-se animações gráficas. Este ambiente permite a exploração dos três tipos de abordagem citadas: passiva — através da utilização de ferramentas de gravação de eventos; ativa — através dos aplicativos-exemplo, e construtiva — através de interfaces minimais e padronizadas para implementação de animações.

Embora sem a aplicação de um método sistemático de análise de resultados quantitativos de aprendizado, depois de utilizarmos o Astral dentro de disciplinas de laboratório de software para ensino de estruturas de dados ao longo de um ano e meio, ministradas por três docentes, tivemos oportunidade de verificar uma sensível melhoria de desempenho dos estudantes comparando-se com semestres anteriores, onde os laboratórios de software utilizavam programação tradicional (sem animação). Este desempenho destacado foi observado tanto

nas avaliações durante a própria disciplina, quanto também vem sendo observado neste momento (um ano mais tarde) quando os estudantes estão cursando uma disciplina de projeto e análise de algoritmos. A sua fixação das noções de algoritmos em estruturas de dados, de paradigmas de projeto de algoritmos e de análise de eficiência se percebe muito mais sólida que a de estudantes de semestres passados.

Agradecimentos

Nossos agradecimentos aos professores João Meidanis e Eliane Martins que se dispuseram a utilizar o Astral em suas disciplinas assim como aos estudantes envolvidos.

Referências

[1] R. V. Amorim and P. J. Rezende. Compreensão de Algoritmos através de Ambientes Dedicados a Animação. In *XX Semish*, 1993.

[2] Apple Computer. *Inside Macintosh*, 1994.

[3] M. H. Brown. *Algorithm Animation*. The MIT Press, 1987.

[4] M. H. Brown. Exploring Algorithms Using Balsa-II. *Computer*, 21(5):14–36, may 1988.

[5] M. H. Brown. An Anthology of Algorithm Animations using Zeus. Digital Systems Research Center, 1991. Video 76b.

[6] M. H. Brown. Color and Sound in Algorithm Animations. In *Proc. IEEE Workshop on Visual Languages*, pages 10–17, 1991.

[7] M. H. Brown. Zeus: A System for Algorithm Animation and Multi-View Editing. In *Proc. IEEE Workshop on Visual Languages*, 1991.

[8] M. H. Brown. The 1992 SRC Algorithm Animation Festival. Technical Report 98, Digital Systems Research Center, march 1993.

[9] M. H. Brown. The 1993 SRC Algorithm Animation Festival. Technical Report 126, Digital Systems Research Center, july 1994.

[10] M. H. Brown and R. Sedgewich. Progress Report: Brown University Instructional Computing Laboratory. *ACM SIGCSE Bulletin*, 16(1):91–101, feb 1984.

[11] F. Hopgood. Computer Animation Used as a Tool in Teaching Computer Science. In *Proc. 1974 IFIP Congress*, pages 889–892, 1974.

[12] E. Horowitz and S. Sahni. *Fundamentals of Data Structures*. Rockville, 1984.

[13] A. W. Lawrence, A. N. Badre, and J. T. Stasko. Empirically Evaluating the Use of Animations to Teach Algorithms. Technical Report GIT-GVU-94-07, Computer Science Department, 1994.

[14] P. J. Rezende and I. C. Garcia. Astral: Animação Gráfica de Algoritmos e Estruturas de Dados — uma abordagem construtiva. In *VIII Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens*, outubro 1995.

[15] J. T. Stasko. Tango: A Framework and System for Algorithm Animation. *Computer*, 23(9):14–36, sep 1990.

[16] D. F. Stubbs and N. W. Webre. *Data Structures with Abstract Data Types and Pascal*. Pacific Grove: Brooks/Cole, second edition, 1988.

[17] Symantec Corporation. *THINK PASCAL User Manual*, 1993.

[18] J. Szwarcfiter and L. Markenson. *Estruturas de Dados e seus Algoritmos*. LTC, 1994.

Relatórios Técnicos – 1995

95-01 **Paradigmas de algoritmos na solução de problemas de busca multidimensional,** *Pedro J. de Rezende, Renato Fileto*

95-02 **Adaptive enumeration of implicit surfaces with affine arithmetic,** *Luiz Henrique de Figueiredo, Jorge Stolfi*

95-03 **W3 no Ensino de Graduação?,** *Hans Liesenberg*

95-04 **A greedy method for edge-colouring odd maximum degree doubly chordal graphs,** *Celina M. H. de Figueiredo, João Meidanis, Célia Picinin de Mello*

95-05 **Protocols for Maintaining Consistency of Replicated Data,** *Ricardo Anido, N. C. Mendonça*

95-06 **Guaranteeing Full Fault Coverage for UIO-Based Methods,** *Ricardo Anido and Ana Cavalli*

95-07 **Xchart-Based Complex Dialogue Development,** *Fábio Nogueira de Lucena, Hans K.E. Liesenberg*

95-08 **A Direct Manipulation User Interface for Querying Geographic Databases,** *Juliano Lopes de Oliveira, Claudia Bauzer Medeiros*

95-09 **Bases for the Matching Lattice of Matching Covered Graphs,** *Cláudio L. Lucchesi, Marcelo H. Carvalho*

95-10 **A Highly Reconfigurable Neighborhood Image Processor based on Functional Programming,** *Neucimar J. Leite, Marcelo A. de Barros*

95-11 **Processador de Vizinhança para Filtragem Morfológica,** *Ilka Marinho Barros, Roberto de Alencar Lotufo, Neucimar Jerônimo Leite*

95-12 **Modelos Computacionais para Processamento Digital de Imagens em Arquiteturas Paralelas,** *Neucimar Jerônimo Leite*

95-13 **Modelos de Computação Paralela e Projeto de Algoritmos,** *Ronaldo Parente de Menezes e João Carlos Setubal*

95-14 **Vertex Splitting and Tension-Free Layout,** *P. Eades, C. F. X. de Mendonça N.*

95-15 **NP-Hardness Results for Tension-Free Layout,** *C. F. X. de Mendonça N., P. Eades, C. L. Lucchesi, J. Meidanis*

95-16 **Agentes Replicantes e Algoritmos de Eco,** *Marcos J. C. Euzébio*

95-17 **Anais da II Oficina Nacional em Problemas Combinatórios: Teoria, Algoritmos e Aplicações,** *Editores: Marcus Vinicius S. Poggi de Aragão, Cid Carvalho de Souza [Not available.]*

95-18 **Asynchronous Teams: A Multi-Algorithm Approach for Solving Combinatorial Multiobjective Optimization Problems,** *Rosiane de Freitas Rodrigues, Pedro Sérgio de Souza*

95-19 **wxWindows: Uma Introdução,** *Carlos Neves Júnior, Tallys Hoover Yunes, Fábio Nogueira de Lucena, Hans Kurt E. Liesenberg*

95-20 **John von Neumann: Suas Contribuições à Computação,** *Tomasz Kowaltowski*

95-21 **A Linear Time Algorithm for Binary Phylogeny using PQ-Trees,** *J. Meidanis and E. G. Munuera*

95-22 **Text Structure Aiming at Machine Translation,** *Horacio Saggion and Ariadne Carvalho*

95-23 **Cálculo de la Estructura de un Texto en un Sistema de Procesamiento de Lenguaje Natural,** *Horacio Saggion and Ariadne Carvalho*

95-24 **ATIFS: Um Ambiente de Testes baseado em Inje,c ao de Falhas por Software,** *Eliane Martins*

95-25 **Multiware Plataform: Some Issues About the Middleware Layer,** *Edmundo Roberto Mauro Madeira*

95-26 **WorkFlow Systems: a few definitions and a few suggestions,** *Paulo Barthelmess and Jacques Wainer*

95-27 **Workflow Modeling,** *Paulo Barthelmess and Jacques Wainer*

Relatórios Técnicos – 1996

96-01 **Construção de Interfaces Homem-Computador: Uma Proposta Revisada de Disciplina de Graduação,** *Fábio Nogueira Lucena and Hans K.E. Liesenberg*

96Abs **DCC-IMECC-UNICAMP Technical Reports 1992–1996 Abstracts,** *C. L. Lucchesi and P. J. de Rezende and J.Stolfi*

96-02 **Automatic visualization of two-dimensional cellular complexes,** *Rober Marcone Rosi and Jorge Stolfi*

96-03 **Cartas Náuticas Eletrônicas: Operações e Estruturas de Dados,** *Cleomar M. Marques de Oliveira e Neucimar J. Leite*

96-04 **On the edge-colouring of split graphs,** *Celina M. H. de Figueiredo, João Meidanis and Célia Picinin de Mello*

96-05 **Estudo Comparativo de Métodos para Avaliação de Interfaces Homem-Computador,** *S'ılvio Chan e Heloisa Vieira da Rocha*

96-06 **User Interface Issues in Geographic Information Systems,** *Juliano Lopes de Oliveira and Claudia Bauzer Medeiros*

96-07 **Conjunto fonte máximo em grafos de comparabilidade,** *Marcos Fernando Andrielli e Célia Picinin de Mello*

96-08 **96-08 The Effectiveness of Multi-Level Policing Mechanisms in ATM Traffic Control,** *J.A. Silvester, N. L. S. Fonseca, G. S. Mayor e S. P. S. Sobral*

96-09 **Sequential and Parallel Experimental Results with Bipartite Matching Algorithms,** *João Carlos Setubal*

96-10 **96-10 A CPU for Educational Applications Designed with VHDL and FPGA,** *Nelson V. Augusto, Mario L. Côrtes and Paulo C. Centoducatte*

96-11 **Network Design for the Provision of Distributed Home Theatre Services,** *Nelson L. S. Fonseca, Cristiane M. R. Franco, Frank Schaffa*

96-12 **Modelling the Output Process of an ATM Multiplexer with Correlated Priorities,** *Nelson L. S. Fonseca e John A. silvester*

96-13 **Algoritmos de afinamento tridimensional: exemplos de técnicas de otimização,** *F. N. Bezerra and N. J. Leite*

96-14 **Ensino de Estruturas de Dados e seus Algoritmos através de Implementação com Animações,** *Pedro J. de Rezende e Islene C. Garcia*

96-15 **Sinergia em Desenho de Grafos Usando Springs e Pequenas Heurísticas,** *H. A. D. do Nascimento, C. F. X. de Mendonça N., P. S. de Souza*

Instituto de Computação
Universidade Estadual de Campinas
13081-970 – Campinas – SP
BRASIL

reltec@dcc.unicamp.br