GUILHERME STUTZ TÖWS

ANIMAÇÃO DE ALGORITMOS

Trabalho de graduação do Curso de Ciência da Computação do Setor de Ciências Exatas da Universidade Federal do Paraná.

Professor: André Luiz Pires Guedes

CURITIBA
2004

1 INTRODUÇÃO

Representações gráficas são superiores à representações textuais, em várias maneiras. Formas abstratas são, de certa forma, mais compreensíveis e universais do que texto escrito; e para iniciantes, figuras e animações são mais interessantes e motivadoras.

Sistemas de animação de algoritmos criam recursos para usuários verem e interagirem com implementações visuais de um algoritmo, e para programadores criarem estas animações de forma mais simples possível.

Neste trabalho, descrevemos a criação de um sistema deste tipo, capaz de mostrar de forma visual o funcionamento dos algoritmos fundamentais da ciência da computação, processos que lidam com estruturas de dados simples como vetores, árvores e grafos.

2 HISTÓRICO

Várias linguagens de programação tentaram criar um ambiente que facilitasse a interação homem-máquina, de várias maneiras.

2.1 LOGO

A linguagem de programação Logo foi criada nos anos 60 por um grupo da empresa BBN, buscando uma linguagem mais acessível às pessoas do que as linguagens predominantemente matemáticas da época. Inicialmente ela era usada para manipular palavras, de maneira semelhante ao Lisp mas menos complexa; A seguir, comandos foram adicionados para mover um robô (a primeira ‘tartaruga’) no espaço físico, e finalmente a linguagem tomou a forma conhecida hoje, com uma tartaruga ‘virtual’ que se move na tela do computador de acordo com o seu programa.

Logo é util para ensinar os fundamentos das linguagens de programação a crianças; Mesmo crianças pequenas podem compreender e refletir os movimentos da tartaruga. Atualmente existem várias variantes do Logo, concentrando-se em aspectos diferentes da linguagem, como as possibilidades de interação entre múltiplas tartarugas e a programação de robôs simples.

2.2 TOONTALK

ToonTalk busca ser uma implementação totalmente visual de uma linguagem de programação. Todas as entidades são representadas visualmente – caixas representam tuplas ou vetores de armazenamento, balanças de pesagem representam testes comparativos, pássaros e ninhos tratam de comunicação entre canais. ToonTalk utiliza um modelo de computação concorrente, em que várias sequências de eventos podem estar sendo executadas ao mesmo tempo.

ToonTalk tem atrativos para crianças, mas pode confundí-las: a linguagem de programação passa a ser um jogo em si, ao invés de um meio para atingir uma meta, e como jogo, ela não tem atrativos suficientes para manter a atenção.

2.3 MULTIMEDIA FUSION

Multimedia Fusion é o último em uma linha de produtos criados pela empresa Europress Software, destinados à programação de jogos de computador. A programação baseia-se em entidades – geralmente, cada objeto capaz de movimento independente na tela é uma entidade, assim como os contadores e temporizadores – e eventos: sequências de comandos ativadas por uma certa condição das entidades, como a colisão entre duas entidades ‘bola’ ativando o comando ‘ricochetear’.

2.4 ZEUS

Zeus é um ambiente criado especificamente para a criação de algoritmos animados. É baseado na discretização do algoritmo por meio da inserção de anotações chamadas ‘eventos interessantes’, e na exibição de diferentes aspectos do algoritmo simultaneamente.

3 IMPLEMENTAÇÃO

Buscamos uma implementação combinando estruturas animadas e descrições textuais simples onde a representação gráfica falha.

Foi escolhida a linguagem Java, pela sua portabilidade; o código compilado pode ser acessado por máquinas de todas as arquiteturas comuns, devido à sua implementação de máquina virtual, ou colocado em páginas web para fácil acesso.

O principal objetivo da implementação é tornar os algoritmos mais fáceis de se compreender, por meio de linguagem visual. Para atingir este objetivo, foram definidos os seguintes sub-objetivos:

- Implementar modelos gráficos das estruturas de dados mais comuns (variável, vetor, árvore);

- Definir operações entre objetos em termos visuais;

- Discretizar o algoritmo, dividindo-o em intervalos, de forma que o programa possa ser acompanhado passo-a-passo;

3.1 ESTRUTURA

A estrutura de animação é similar a do Zeus, com o algoritmo a ser animado dividido em passos entre os quais o usuário pode visualizar os elementos sendo manipulados. Mensagens curtas podem ser adicionadas a cada passo, facilitando a compreensão de elementos menos passíveis de implementação visual.

3.2 CÓDIGO

O código da implementação cria algumas classes para facilitar a exibição do algoritmo, substituindo estruturas de dados comuns por elementos visuais.

3.2.1 Interface de elemento visual

Interface definida por objetos que terão uma representação visual. Define dois métodos, movimentação e atualização visual. O método de movimentação é chamado 50 vezes por segundo e é utilizado para animações do objeto; o método de atualização visual deve conter instruções para desenhar a representação visual do objeto na interface gráfica, e é chamado sempre que o *buffer* gráfico puder ser atualizado.

3.2.2 Objeto Thread

VThread é a classe básica da *thread* que implementa o programa. Para o programador, ela contém dois métodos de interesse: O de inicialização (chamado *init*) e o de execução do algoritmo (chamado *Main*).

O método de inicialização é chamado quando a interface gráfica é ativada e sempre que o usuário pede o reinício do algoritmo. Neste método, certas opções devem ser incluídas. A variável *name* deve ser atualizada com o título do programa. E, mais importante, o vetor *program_items* deve ser atualizado com a lista de todos os elementos que terão uma representação visual.

O método de execução contém o algoritmo principal. Ele é demarcado por funções de passo, que denotam momentos importantes do processo. Quando o algoritmo passa por uma função de passo, ele para até o usuário requisitar o próximo passo, ou no caso de funcionamento contínuo, aguarda por um segundo. Em ambos os casos, a animação dos objetos continua sendo atualizada, possibilitando ao usuário a visão mais ponderada dos acontecimentos.

Esta classe contém ainda uma função que automatiza a comparação entre dois itens, automaticamente sublinhando os respectivos itens e encerrando um passo de animação.

3.2.3 Objeto Item

O elemento básico das estruturas de dados, *VItem* representa uma variável capaz de conter um único número. A sua representação gráfica é um círculo, com o valor da variável inscrito nele.

Ao criar um novo objeto de item, as suas coordenadas na tela devem ser fornecidas. Visualmente, o novo objeto aparece deslizando para a sua posição. O objeto item contém funções animadas de alteração de valor, cópia do valor de outro item e troca de valores com outro item, bem como duas *flags* para colocar o item em evidência.

3.2.4 Objeto Vetor

Define um vetor capaz de conter um conjunto de números. A representação gráfica é uma sequência de círculos similares aos de objetos de item, em formação linear.

Da mesma forma que o objeto de item, deve-se fornecer a posição de um objeto de vetor recém-criado, assim como o espaçamento entre seus itens (que pode ser horizontal, vertical ou diagonal). O objeto de vetor contém métodos para acessar seus itens componentes.

3.2.5 Objeto Árvore Binária

Representa uma árvore binária. A representação visual é estruturada de forma a garantir a visualização de todos os itens em camadas superiores. O objeto contém métodos para adicionar novos elementos e convertê-los em objetos de árvore binária.

3.2.6 Objeto de Grafo

Um vetor contendo um conjunto de vértices e arestas de um grafo, bem como sua implementação visual. As conexões podem ser selecionadas da mesma forma que itens.

3.2.7 Objeto de Interface Gráfica

Este objeto implementa uma *applet* Java capaz de exibir o algoritmo. Ele contém um único método de importância, que precisa ser atualizado com a classe utilizada para a definição do algoritmo.

4 UTILIZAÇÃO

4.1 Programação

A implementação de um novo algoritmo a ser animado segue um processo simples.

1. Conversão das variáveis para elementos gráficos. Todas as variáveis que terão um componente visual devem ser substituídas por seus similares gráficos. Similarmente, operações de atualização e comparação de valores devem ser atualizadas.

2. Divisão do algoritmo em inicialização e processo principal. A inicialização posiciona os elementos que aparecerão no início e as adiciona ao vetor de elementos visuais.

3. Adição de código próprio. A implementação requer a adição de algumas linhas de código para criar a *applet* com o algoritmo específico.

4. Inserção de elementos visuais específicos e pontos de passo. Pontos de passo são geralmente adicionados após mudanças de valor das variáveis ou alterações na estrutura de dados. Elementos específicos que devem ser colocados em evidência requerem a ativação e subsequente desligamento de suas *flags* de seleção.

A classe resultante pode ser usada como uma applet em páginas HTML, bem como executada individualmente.

4.2 Uso

A interface gráfica traz quatro botões para manipular a execução do algoritmo. Ao iniciar, o método de inicialização do algoritmo é ativado, e o programa fica em estado de espera, aguardando o acionamento de um botão.

O botão de reinício (*Reset*) termina a *thread* de programa corrente e cria uma nova, chamando novamente o método de inicialização. O botão de passo (*Step*) executa o algoritmo, esperando que este seja interrompido por uma função de passo, quando volta ao estado de espera. O botão de funcionamento contínuo (*Play*) executa o algoritmo da mesma forma que o de passo, com a diferença que ao invés de ser interrompido numa função de passo, o algoritmo aguarda um segundo e continua seu funcionamento, até o término ou até o comando de parada (*Stop*) ser acionado.

5 EXEMPLOS

5.1 Ordenação por Seleção

O algoritmo de Ordenação por seleção é um dos mais simples e mais demorados algoritmos para ordenação de vetores. Localiza-se o menor elemento do vetor, que é trocado com o primeiro. O próximo menor elemento é trocado com o segundo elemento, e assim por diante.

A implementação visual do algoritmo substituiu as trocas por funções animadas, e adicionou código de forma que o menor elemento encontrado na busca esteja sempre em evidência.

5.2 Inserção em Árvore

A inserção de elementos em árvore envolve uma busca binária. Comparamos o elemento a ser inserido com a raiz: Se for menor, procedemos à direita, se for maior, procedemos à esquerda. Prosseguimos com o novo elemento raiz até encontrar um nó vazio, onde o elemento é inserido.

O algoritmo de inserção é traduzido com facilidade para a forma visual. Foi adicionado código para colocar os itens atuais da árvore e do vetor em evidência, assim como funções de passo com mensagens compreensíveis.

5.3 Árvore Mínima em Grafo

O problema de encontrar a árvore de menor peso em um grafo tem várias soluções. O algoritmo usado aqui itera pelas arestas adjacentes às partes já escolhidas do grafo, escolhendo aquela que tiver o menor peso.

A implementação visual faz uso extensivo do objeto de grafo, colocando em evidência as arestas conforme elas são selecionadas e destacando a árvore em formação.

6 CONCLUSÃO

Neste trabalho foram abordadas as técnicas de animação de algoritmos e descrita a criação de um sistema capaz de criá-las. Procurou-se apresentar uma descrição detalhada do processo usado, bem como ele pode ser adaptado para o uso de outros programas com o intuito de torná-los mais compreensíveis.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] MORAES, C. *Estruturas de Dados e Algoritmos.* Berkeley, 2001.

[2] BARNETT, M. e BARNETT, S. *Animated Algorithms*. McGraw-Hill, 1986.

[3] FANTA, R. et al. *Java Unleashed*. Sams, 1996.

[4] MENZIES, T. *Evaluation Issues for Visual Programming Languages*. 1998.
Em: http://ksi.cpsc.ucalgary.ca/KAW/KAW98/menzies4/.

[5] Comp.lang.logo FAQ. 1995.
Em: http://ekstern.alu.hist.no/~dennisgl/logo/logo-faq.html.

[6] ToonTalk – Making programming child's play. Em: http://www.toontalk.com.

[7] CHAN, S. *The Unofficial Multimedia Fusion FAQ*. 2000.
Em: http://sprite.firastudios.com/old/tech/mmffaq/.

[8] BROWN, M. *Zeus: A System for Algorithm Animation and Multi-View Editing*. 1992.