Especificação de layout abstrato por manipulação direta

RAQUEL OLIVEIRA PRATES
LUIZ HENRIQUE DE FIGUEIREDO
MARCELO GATTASS

TeCGraf – Grupo de Tecnologia em Computação Gráfica
Departamento de Informática, PUC–Rio
Rua Marquês de São Vicente, 225
22451–041 Rio de Janeiro, RJ, Brasil
{raquel,lhf,gattass}@icad.puc-rio.br

Abstract. We present Visual LED, a three-view editor for creating user interfaces by direct manipulation. This editor is an interactive alternative to LED, a textual language for specifying the abstract layout of dialogs. We discuss strategies for specifying abstract layout using direct manipulation and their use in the design of Visual LED. We also suggest new methods for editing hierarchies, both for generic graphical editors and for interface editors that use abstract layout.

Introdução

Um sistema pode ser separado em duas partes fundamentais: a interface com usuário e a tecnologia da aplicação (Celes–Figueiredo–Gattass–Ierusalimschy 1994). A interface é responsável pela comunicação entre o usuário e a aplicação, enquanto que a tecnologia da aplicação é a parte que resolve o problema proposto. A princípio, a tecnologia independe da interface.

Embora a interface não faça parte do problema a ser resolvido, ela é parte fundamental de qualquer solução, pois é a ligação entre o usuário e a aplicação. Uma interface mal projetada pode dificultar o entendimento e o uso da aplicação, fazendo, assim, com que o usuário tenha que despender tempo e energia na utilização do sistema, ao invés de concentrá-los na tarefa a ser executada. Isto é frustrante e pode levar o usuário a abandonar o sistema (Neelamkavil–Mullarney 1990). Assim, uma boa tecnologia pode deixar de ser útil se a interface do programa que a contém não for boa.

Visando simplificar a criação de boas interfaces portáteis, o TeCGraf desenvolveu o IUP/LED, um sistema portátil de interface com o usuário, composto por um *toolkit* virtual (IUP) e por uma linguagem de especificação de diálogos (LED) (Levy 1993; Figueiredo–Gattass–Levy 1993).

O sistema IUP/LED vem sendo utilizado pelo TeCGraf em produção com sucesso. Os elementos de interface oferecidos são suficientes, permitindo que boas interfaces sejam rapidamente projetadas. Para criar estas interfaces, o programador deve aprender LED, que é uma linguagem simples e concisa, de fácil aprendizado. No entanto, novas dificuldades foram encontradas: ao projetar uma interface, deve-se primeiro visualizá-la mentalmente e depois traduzir esta visão em expressões LED, o que pode ser uma tarefa bastante trabalhosa. Reciprocamente, ler a descrição LED de uma interface e visualizá-la pode ser igualmente difícil, principalmente se o diálogo sendo descrito é grande ou se a interface é composta por mais de um diálogo, o que é freqüente. Um editor gráfico de interfaces LED por manipulação direta diminuiria consideravelmente o trabalho do projetista.

Neste trabalho, apresentamos Visual LED, um editor gráfico de interfaces por manipulação direta que segue o modelo de *layout* abstrato usado em LED, e edita descrições em LED (Prates 1994). Este sistema apresenta três vistas: uma contendo a descrição textual do diálogo em LED; outra contendo o *layout* abstrato; e uma terceira contendo o *layout* concreto. Como motivação para as soluções adotadas em Visual LED, discutimos inicialmente uma dificuldade presente em editores gráficos genéricos e de interfaces na edição de hierarquias, e apresentamos uma solução para ela.

Especificação de *layout* em IUP/LED

O sistema portátil de interface IUP/LED foi projetado para facilitar a criação de interfaces e para solucionar algumas dificuldades experimentadas no trabalho de produção do TeCGraf, sendo a principal delas a dificuldade de se especificar boas interfaces portáteis.

O sistema IUP/LED é composto pelo *toolkit* virtual IUP e pela linguagem de especificação de diálogos LED.

O IUP é um *toolkit* virtual, com aproximadamente cinqüenta funções, para construção e manipulação de diálogos em programas. Ele tem a vantagem de possibilitar a geração de interfaces com um *look-and-feel* (aparência final) tanto nativo, quanto fixo. Isto é desejável, pois, se um usuário trabalha com diversas aplicações em uma mesma máquina, então será mais fácil para ele usar uma nova aplicação se ela tiver o *look-and-feel* nativo, ou seja, semelhante ao das outras aplicações. Por outro lado, um usuário que trabalha com uma mesma aplicação em diversas máquinas vai preferir um *look-and-feel* fixo, ao qual ele já está acostumado e com o qual ele é produtivo e eficiente.

A linguagem LED foi proposta como uma solução para o problema de especificação de *layout* de diálogos (Levy 1993). Por "diálogo", entenda-se um grupo de objetos de interface que estão em um contexto espacial limitado (Marcus 1992).

Especificar o *layout* de um diálogo significa descrever a composição visual deste diálogo, isto é, como os elementos de interface que o compõem são criados, agrupados e dispostos geometricamente. Esta especificação pode ser feita de várias formas diferentes; não existe um consenso sobre qual delas é a melhor. Como conseqüência, o modo de se especificar o *layout* varia de um sistema de interface para outro. Um dos objetivos principais no projeto de IUP/LED é a uniformização da forma de especificação de *layout*.

Os dois principais métodos de especificação de *layout* são: *layout* abstrato e *layout* concreto (Levy 1993). Descrever concretamente o *layout* de um diálogo significa fornecer explicitamente a posição geométrica e o tamanho de cada elemento de interface; descrevê-lo abstratamente significa especificar a posição relativa dos elementos. O uso de *layout* abstrato tem a vantagem de permitir o recálculo automático da posição dos elementos de interface quando alguma modificação é feita no diálogo. Assim, especificações abstratas permitem manter a *intenção* de *layout*.

A linguagem LED especifica *layout* abstrato baseada no modelo *boxes-and-glue* do processador de texto $\TeX$ (Knuth 1984). Neste modelo, um diálogo é composto por elementos primitivos (botões, áreas de trabalho, bordas, teclas de funções, menus, submenus, listas, botões de dois estados e áreas de captura de texto e valor numérico). A especificação de seu *layout* utiliza elementos de composição e de preenchimento.

Seguindo $\TeX$, os elementos de composição oferecidos em LED são `hbox` e `vbox`, que permitem a exibição horizontal ou vertical dos elementos neles contidos. No `hbox`, os elementos são alinhados horizontalmente pelos seus lados superiores; no `vbox`, eles são alinhados verticalmente pelos seus lados esquerdos. Em $\TeX$, os elementos de preenchimentos são `vfill` e `hfill`, que ocupam proporcionalmente os espaços vazios em `vbox`'s e `hbox`'s, respectivamente. Como o conceito de `hfill` e `vfill` é o mesmo—a única diferença é a direção do grupo ao qual eles pertencem—optou-se em LED (e logo em Visual LED) por ter apenas o elemento `fill`, que ocupa proporcional e dinamicamente os espaços vazios, tanto em `hbox`'s quanto em `vbox`'s.

A adoção de *layout* abstrato simplifica a especificação de diálogos, mas a descrição textual de um *layout* pode ser de difícil compreensão. Um dos principais objetivos de Visual LED é permitir a visualização de *layouts* descritos em LED. Espera-se que esta visualização, aliada à manipulação direta, simplifique a edição de descrições LED.

Editores gráficos genéricos

Na fase inicial da implementação de Visual LED, estudamos as formas de interação usadas em editores gráficos genéricos e editores gráficos de interface. Estudamos a criação e seleção de primitivas, assim como as operações permitidas sobre elas. Analisando suas vantagens e desvantagens, tentamos implementar em Visual LED uma interação que fosse simples e intuitiva para o usuário.

As operações básicas necessárias em um editor gráfico de interfaces não diferem muito das operações necessárias em um editor gráfico genérico. Algumas das operações sobre primitivas comuns a ambos são: criar, selecionar, copiar, apagar, mover, agrupar, desagrupar, trocar os atributos. Estudamos então a interação com o usuário adotada para estas operações em editores gráficos genéricos. Enfocamos, especialmente, a política de interação usada nas operações de agrupamento e desagrupamento, uma vez que esta operação seria amplamente usada em Visual LED, dado o modelo de *layout* abstrato de LED.

Política de interação

A política de interação adotada pelos editores gráficos que estudamos, principalmente *idraw* e *Corel DRAW*, é simples e de fácil aprendizado; a chamaremos de **política simples**. Para agrupar elementos nestes editores, basta selecionar os elementos desejados e depois selecionar a opção de agrupamento. Uma vez agrupados, estes elementos são tratados como se fossem um único elemento. É importante notar que um agrupamento não implica em nenhuma mudança na posição ou na aparência dos elementos agrupados.

Os editores gráficos estudados permitem que a seleção seja feita de duas formas. Na primeira, seleciona-se os elementos desejados um a um, em qualquer ordem. Na segunda, especifica-se uma região da tela; todos os elementos que estiverem totalmente contidos nesta região serão selecionados (esta forma é conhecida como *fence*). Dependendo do editor gráfico, a região é determinada por um retângulo ou uma curva fechada (chamada **laço**).

Nos editores gráficos genéricos, a ordem em que os elementos são selecionados não tem nenhuma relevância para o agrupamento, dado que o agrupamento não afeta a geometria dos elementos do grupo. Nos editores gráficos de interface, a ordem de seleção pode ser fundamental, pois a ordem dos elementos dentro do grupo pode definir o *layout* do diálogo. Nestes editores, os elementos aparecem no grupo na ordem selecionada; no caso de seleção com *fence*, a ordem de seleção é arbitrária, normalmente a mesma da criação.

Problema: edição hierárquica

Se, por um lado, a simplicidade é a principal vantagem da política simples, por outro lado, ignorar a hierarquia de grupos é uma grande desvantagem. Como um grupo passa a ser tratado como uma primitiva simples, não se tem mais acesso direto aos seus elementos internos. Assim sendo, para se fazer qualquer operação em um elemento interno ao grupo, deve-se primeiro desfazer todo o grupo até chegar ao elemento desejado, fazer a operação desejada e refazer todo o grupo. No pior caso, pode ser necessário desfazer todos os níveis de agrupamento e refazer tudo após a alteração. Este problema é claro no exemplo a seguir.

O usuário deseja fazer o desenho da Figura 1. Inicialmente, ele desenha um segmento de reta e um triângulo, e os agrupa, formando uma seta (Figura 2). A seguir, ele cria dois retângulos e agrupa-os junto com a seta (Figura 3). Finalmente, ele desenha um retângulo em volta do grupo existente e cria mais um grupo (Figura 1), o que facilita operações como mover e duplicar todo o desenho. Feito isso, se o usuário resolve trocar a cor do triângulo da seta, então todos os agrupamentos terão que ser desfeitos; caso contrário, a troca de cor afetará todos os elementos do grupo, incluindo os retângulos e a reta. Uma vez desfeitos todos os grupos, o usuário consegue selecionar o triângulo e mudar sua cor, mas é necessário reagrupar todos os elementos após a mudança. Na prática, este problema é mais grave pois, mesmo em diagramas com poucas primitivas, é possível que a hierarquia de grupos seja complexa.

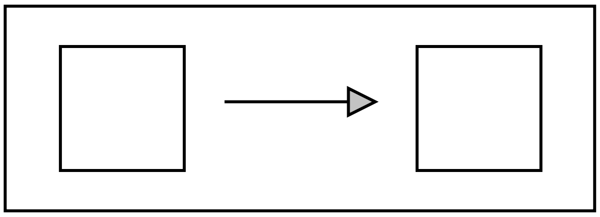

Figura 1: Grupo total.

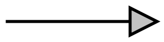

Figura 2: Grupo contendo seta.

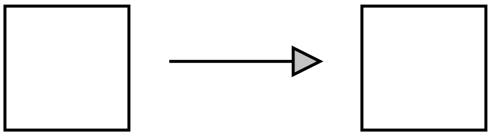

Figura 3: Grupo contendo seta e retângulos.

Uma solução

Uma melhoria a ser introduzida nos editores gráficos que usam a política simples seria então a **edição em profundidade**, isto é, permitir percorrer a hierarquia de um grupo e editar seus componentes. No entanto, editar uma hierarquia diretamente sobre o desenho pode ser difícil, pois vários elementos podem estar sobrepostos em uma mesma posição. Conseqüentemente, para poder editar uma hierarquia, é necessário representá-la de forma que qualquer elemento possa ser acessado, sem ambigüidade, independentemente da sua profundidade na hierarquia.

Como a hierarquia de um grupo é abstratamente uma árvore, uma boa solução seria apresentar esta árvore graficamente. Através desta representação gráfica da hierarquia, o usuário poderia visualizar simultaneamente todos os elementos do grupo, sem ambigüidade. Conseqüentemente, ele poderia selecionar e editar todos os elementos individualmente, sem necessidade de desfazer agrupamentos. Para facilitar a visualização da relação entre os nós da árvore e os elementos que eles representam, a árvore seria apresentada em uma vista auxiliar.

Para efetuar uma operação em um elemento qualquer da hierarquia, o usuário selecionaria na árvore o objeto a ser modificado e aplicaria nele a operação desejada. Os objetos primitivos seriam representados na árvore pelo seu próprio desenho em escala reduzida. A seleção na árvore seria simultaneamente indicada no desenho, na forma usual. Uma vez selecionado o objeto, a operação seria aplicada normalmente na vista do desenho. A vista contendo a árvore seria portanto apenas um mecanismo auxiliar de seleção. Uma idéia semelhante foi implementada no modelador de sólidos *Genesys*, onde a hierarquia refletia construções CSG (Fischer 1991).

Outra solução

Ao contrário das versões anteriores, o *Corel DRAW* 4 já possibilita a edição em profundidade, diretamente no original, ou numa representação *wireframe* dos objetos. Este modo *wireframe* é importante, pois nele é possível visualizar todos os objetos simultaneamente, o que não ocorre no desenho original quando há sobreposição de objetos opacos. Entretanto, selecionar o objeto desejado no modo *wireframe* pode ser confuso quando há interseção de vários objetos.

No *Corel DRAW* 4, a hierarquia é percorrida da raiz para as folhas. Se nenhum objeto está selecionado, seleciona-se o objeto mais alto na hierarquia na posição indicada. Se nesta posição já existe um objeto selecionado, e este objeto é um grupo que tem um filho nesta posição, então seu filho é selecionado.

Comparação

A solução apresentada pelo *Corel DRAW* permite que o usuário trabalhe ou no desenho original ou na representação *wireframe*. Ambos os modos possuem dificuldades de seleção, devidas à sobreposição de elementos ou a interseções entre linhas numa área da tela. Existe ainda uma dificuldade que é comum aos dois modos: quando um grupo tem a mesma *bounding box* que um de seus filhos, não é possível diferenciar a indicação gráfica da seleção do grupo da indicação gráfica da seleção deste filho. Decidindo qual o melhor modo a cada momento, o usuário consegue superar as dificuldades apresentadas por cada um deles. No entanto, estes modos são mutuamente exclusivos: apenas um deles pode ser visto de cada vez, o que desencoraja a troca freqüente de modos.

Na solução da árvore, o usuário trabalha na vista do desenho. A representação gráfica da hierarquia, exibida simultaneamente, é utilizada como ferramenta auxiliar de seleção em profundidade. Nesta vista auxiliar, não há dificuldades de seleção, pois cada nó da árvore representa um elemento distinto.

Acreditamos que a solução da árvore seja mais simples que a solução do *Corel DRAW*, pois a vista contendo o desenho—o objetivo final do usuário—está sempre presente; a vista da árvore é auxiliar e pode estar ou não presente, conforme a preferência do usuário. Além disso, uma seleção feita na árvore também é representada no desenho.

Visual LED

Em LED, os diálogos são construídos agrupando elementos primitivos. No Visual LED, isto implica numa especificação *bottom-up*, pois parte-se dos elementos primitivos e alcança-se o diálogo desejado através de agrupamentos. Portanto, a operação de agrupamento é extremamente importante na composição de diálogos em Visual LED. Se tratarmos os grupos (`hbox`'s e `vbox`'s) como elementos simples, teremos as dificuldades de edição detectadas nos editores gráficos genéricos e discutidas na seção anterior.

Uma vez que as dificuldades são as mesmas, põe-se a questão: a solução da seleção na árvore é adequada para o Visual LED? Apesar de acreditarmos que a solução da árvore é boa para editores gráficos genéricos, não a julgamos adequada para o Visual LED. Primeiramente, a árvore seria redundante, pois a informação contida na árvore já está contida na descrição textual LED. Além disso, a representação gráfica do agrupamento não facilitaria a visualização do *layout* concreto dos diálogos pelo usuário, pois é uma representação topológica e não geométrica.

Um editor gráfico de interfaces com *layout* abstrato, no entanto, apresenta algumas peculiaridades em relação a editores gráficos genéricos e editores de interfaces com *layout* concreto. Por exemplo, elementos de um mesmo diálogo nunca se sobrepõem. Mais precisamente, a hierarquia da árvore sempre corresponde à geometria do diálogo. Isso significa que, se um elemento é o irmão da direita de outro na árvore, então o elemento também está geometricamente à direita deste irmão; se um elemento é filho do outro, então geometricamente ele está contido no seu pai; e assim por diante. Observe na Figura 4 que `botão1` está à esquerda de `botão2`, pois `botão1` é o irmão da esquerda de `botão2` na árvore correspondente (Figura 5). Note também que ambos são filhos de um `hbox`, e logo se encontram dentro deste no diálogo, embora seja difícil ver isto no *layout* concreto, uma vez que o `hbox` não tem nenhuma representação visual na interface. Sendo assim, se representarmos geometricamente *todos* os elementos da árvore, e não somente os elementos primitivos, teremos automaticamente a hierarquia.

Figura 4: Um diálogo.

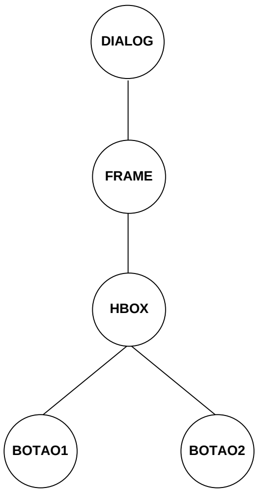

Figura 5: Árvore do diálogo.

A solução que adotamos no Visual LED foi criar uma vista contendo uma representação geométrica explícita do *layout* abstrato do diálogo. O usuário trabalha manipulando os elementos de interface diretamente nesta vista; a vista contendo o *layout* concreto (o resultado) é automaticamente atualizada. A vista do *layout* abstrato soluciona os problemas apresentados acima, pois permite que o usuário visualize a interface que ele tem em mente e simultaneamente representa a hierarquia do diálogo, permitindo sua edição.

É importante ressaltar que, nesta nova vista, é possível visualizar elementos de interface que não têm representação visual na interface final (`hbox`, `vbox`, `frame` e `radio`). A representação visual destes elementos é a base da representação da hierarquia do diálogo. Uma distinção importante a ser feita é que mesmo os elementos que têm tamanho natural zero no *layout* concreto devem ter um tamanho mínimo na representação do *layout* abstrato, para torná-los passíveis de manipulação direta.

Como exemplo, considere o diálogo da Figura 6. Uma descrição em LED para este diálogo é (Figueiredo–Gattass–Levy 1993):

```
DIALOG[TITLE="Attention"](
 VBOX(
  FILL(),
  HBOX(
   FILL(),
   LABEL("File already exists!"),
   FILL()),
  FILL(),
  HBOX(
   FILL(),
   BUTTON(\"Ok",do_ok),
   FILL(),
   BUTTON("Cancel",do_cancel),
   FILL()
  )))
```

A representação geométrica do *layout* abstrato deste diálogo em Visual LED é mostrada na Figura 7.

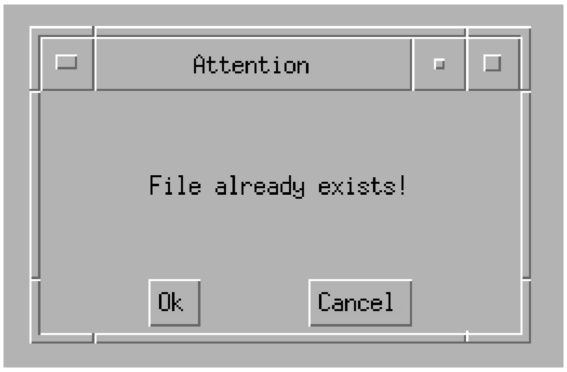

Figura 6: *Layout* concreto.

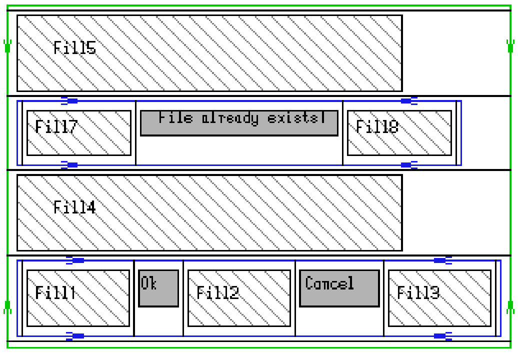

Figura 7: Representação de *layout* abstrato.

Política hierárquica

Para permitir a edição em profundidade, criamos uma política de interação que chamamos **política hierárquica**. Esta política permite que o usuário acesse qualquer elemento de um grupo (`hbox` ou `vbox`), independente do seu nível de profundidade, podendo assim modificar atributos de elementos internos, trocá-los de posição, retirá-los ou inserí-los, sem precisar desfazer e refazer agrupamentos.

Com esta política, não é necessário o uso da estratégia *bottom-up* sugerida por LED. Em Visual LED, é possível usar uma estratégia *top-down* na criação de diálogos: em vez de criar os elementos primitivos, e depois agrupá-los, um grupo pode ser criado como uma caixa vazia. O usuário então introduz nesta caixa os elementos, na ordem e posição desejadas.

Para facilitar a escolha do lugar de cada elemento dentro de um grupo, ele possui "antenas". Estas "antenas" aparecem no início e fim do grupo, e entre os seus elementos, indicando as posições em que novos elementos podem ser inseridos. Para inserir um elemento, basta colocá-lo sobre a antena ativa correspondente à posição desejada. As Figuras 8 e 9 mostram um exemplo de inserção usando antenas: a Figura 8 mostra o elemento `Novo botão` fora do `hbox` e o `hbox` com suas antenas; a Figura 9 mostra o grupo após a inserção de `Novo botão` na posição correspondente à segunda antena.

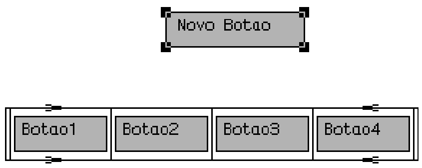

Figura 8: `Novo botão` antes da inserção no `hbox`.

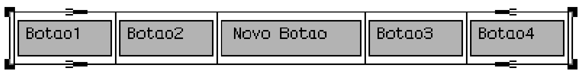

Figura 9: `hbox` após inserção de `Novo botão`.

A antena selecionada como ponto de inserção é aquela interceptada pelo elemento de interface a ser inserido. Se mais de uma antena ativa interceptar o elemento de interface, então a antena mais próxima do seu canto superior esquerdo será selecionada. No caso de haver mais de um elemento de interface a ser inserido, a seleção da antena de inserção será feita dentre as antenas que interceptarem o elemento de maior prioridade interceptado, onde a prioridade é definida pela ordem de seleção (o primeiro elemento selecionado é o de maior prioridade). Entretanto, o usuário não está normalmente consciente destes detalhes: ele simplesmente "passeia" com o elemento a ser inserido sobre as antenas até que a antena correspondente à posição desejada seja ativada (a antena ativada tem uma cor diferente das demais).

Seleção

O Visual LED possui duas formas de selecionar elementos internos a grupos: a seleção descendente e a seleção ascendente. Na seleção descendente, percorre-se a hierarquia a partir da raiz até chegar ao elemento desejado. Ao se pressionar o botão do *mouse* em uma posição, se nenhum elemento estiver selecionado, o elemento mais alto na hierarquia, naquela posição, é selecionado. Quando naquela posição já existe um elemento selecionado, se ele tiver um filho nesta posição, ele é de-selecionado e seu filho é selecionado. Para selecionar o irmão de um elemento já selecionado, não é necessário percorrer novamente a hierarquia.

Na seleção ascendente, ao se pressionar o botão do *mouse*, se nenhum elemento está selecionado, o elemento mais profundo na hierarquia naquela posição é selecionado. Se o elemento mais profundo naquela posição já está selecionado, seu pai é selecionado.

A seleção descendente tem a desvantagem de ser necessário pressionar o botão do *mouse* diversas vezes para atingir elementos muito profundos na hierarquia. Por outro lado, na seleção ascendente, quando a área de um elemento é pequena, pode ser difícil posicionar o cursor sobre ela e, conseqüentemente, selecionar o elemento desejado. O Visual LED permite as duas formas de seleção; o usuário usa a que for mais conveniente ou eficiente a cada momento. A passagem de um modo de seleção para outro pode ser feita por seleção explícita na interface ou por *hotkeys*, não criando desconforto para o usuário.

É possível selecionar mais de um elemento de uma só vez, mas somente se eles estiverem todos no primeiro nível de profundidade ou forem irmãos. Se não for feita em profundidade, a seleção pode ser feita usando *fence*, como em um editor gráfico genérico. Neste caso, a ordem de seleção será a ordem de criação.

Combinação de políticas

Nem sempre a política hierárquica é melhor que a política simples. Quando já se sabe exatamente que elementos agrupar, e em que ordem, é mais simples e rápido selecioná-los nesta ordem e requisitar o agrupamento desejado do que criar uma caixa vazia e colocá-los um a um dentro dela. O Visual LED usa as duas políticas. Conseqüentemente, o usuário pode criar o grupo da forma que lhe parecer mais conveniente, usando a política simples ou a política hierárquica. Uma vez criado, o grupo é tratado de acordo com a política hierárquica, pois esta é a única que permite edição de elementos internos.

As três vistas

Visando a geração rápida e eficiente de interfaces LED, o Visual LED oferece duas maneiras distintas e simultâneas de edição. Além disso, apresenta também o resultado da interface sendo gerada. O Visual LED é portanto composto por três vistas:

- uma vista textual, contendo a descrição LED;
- uma vista gráfica, contendo uma representação geométrica do *layout* abstrato;
- uma vista do *layout* concreto, contendo um protótipo da interface.

Estas vistas podem estar ativas ao mesmo tempo, ou não, conforme a preferência do usuário. No caso de estarem ativas, o usuário pode editar um diálogo tanto na vista textual quanto na vista gráfica, sem que seja necessário explicitar a transferência de uma para outra. Durante a edição, o usuário pode acompanhar o *look-and-feel* do diálogo na vista de *layout* concreto; esta vista não é editável. Qualquer modificação feita em uma das vistas editáveis é imediatamente atualizada nas outras duas.

Na vista textual, o usuário edita a descrição LED da interface. Durante esta edição, a descrição LED passa por estados nos quais ela não é válida (por exemplo, quando os parênteses não estão corretamente casados). Por isso, as outras duas vistas só são atualizadas a pedido explícito do usuário.

Na vista da representação gráfica do *layout* abstrato, o usuário edita os diálogos por manipulação direta dos seus elementos. Como nesta vista o diálogo nunca passa por situações inválidas, as outras vistas são automaticamente atualizadas. Como o sistema IUP/LED é multi-plataforma, os elementos de interface são representados nesta vista por caixas sem decoração. O compromisso maior da aparência da representação do *layout* abstrato é transmitir ao usuário o arranjo dos objetos de interface, conforme ilustra o uso de setas para distinguir visualmente `hbox`'s e `vbox`'s (Figuras 8 e 9).

Os elementos primitivos possuem um título que, inicialmente, é o tipo do elemento seguido por um número. Estes elementos são preenchidos de cinza claro, com exceção do `canvas` e do `fill`, que são os únicos elementos expansíveis. O `canvas` é preenchido de cinza escuro e o `fill` é hachurado. Os elementos que possuem filhos são representados por bordas coloridas em torno dos seus filhos (azul para `hbox`'s, verde para `vbox`'s, preto para `frame`'s e preto pontilhado para `radio`'s).

Como dissemos, a vista do *layout* concreto não é editável. Sua função é apresentar ao usuário o *look-and-feel* real do diálogo (na plataforma em que a edição é feita), e permitir que o seu comportamento dinâmico seja testado durante a construção, isto é, prototipagem rápida da interface.

Outros editores de *layout* abstrato

Nesta seção, descrevemos dois outros editores de interfaces de *layout* abstrato: *ibuild* (Vlissides–Tang–Brauer 1991) e *FormsEdit* (Avrahami–Brooks–Brown 1989). Ambos usam *layout* abstrato baseados no modelo *boxes-and-glue* do TEX, e influenciaram o projeto de Visual LED. Em seguida, comparamos a edição em hierarquia e a representação do *layout* abstrato destes editores com o Visual LED.

O *ibuild* é o editor gráfico de interfaces do sistema InterViews (Linton–Vlissides–Calder 1989); ele possui uma única vista e permite criar interfaces gráficas por manipulação direta para aplicações em *workstations* sobre o sistema de janelas X. O *ibuild* não representa graficamente os `hbox`'s e `vbox`'s, mas apenas os seus filhos. Isto dificulta a visualização da hierarquia do diálogo sendo gerado. É possível editar níveis da hierarquia. No entanto, ao se selecionar um nível do diálogo nesta vista, só são visíveis os elementos pertencentes a este nível e a outros mais profundos no mesmo ramo da sub-árvore. Isto faz com que se perca a noção do diálogo como um todo. Para evitar isto, pode-se abrir múltiplas vistas, o que pode se tornar confuso.

Como o Visual LED, o *FormsEdit* também é um editor gráfico de três vistas simultâneas. No entanto, elementos que possuem filhos (`hbox`, `vbox`, `frame` e `radio`) têm a mesma representação geométrica na vista do *layout* abstrato. Sendo assim, não é possível diferenciar os elementos de composição, o que dificulta a visualização da hierarquia do diálogo. A seleção de objetos é feita pressionando o botão do *mouse* sobre o objeto desejado, o que pode ser difícil quando este tem área pequena.

Para implementarmos o Visual LED, utilizamos tanto idéias do *ibuild* quanto do *FormsEdit*. A maior contribuição do *ibuild* foi na maneira de agrupar elementos primitivos na política simples. No entanto, o *FormsEdit* teve uma influência maior no Visual LED. Baseado no *FormsEdit*, o Visual LED foi construído contendo três vistas. No entanto, no *FormsEdit* a vista textual é prioritária e o usuário sempre tem que utilizá-la. No Visual LED, as vistas têm a mesma prioridade e o usuário pode trabalhar em qualquer uma delas. A representação geométrica de hierarquias usando bordas ao redor dos elementos de composição e a política de seleção ascendente também foram baseadas no *FormsEdit*. No Visual LED, foi introduzida uma melhoria nesta representação de hierarquias, pois os diferentes elementos que contém filhos têm sua representação diferenciada pela cor, o que não acontece no *FormsEdit*.

Conclusão

Discutimos uma dificuldade no uso do sistema IUP/LED e a necessidade de se ter uma maneira mais intuitiva de criar descrições LED. Como solução, apresentamos Visual LED, um editor gráfico de interfaces de *layout* abstrato por manipulação direta. Discutimos alguns dos problemas de edição de hierarquias tanto em editores gráficos genéricos quanto em editores de interface de *layout* abstrato.

Mostramos a importância de permitir a edição em profundidade nos editores gráficos genéricos, mas principalmente em editores gráficos de interface de *layout* abstrato. Para os editores gráficos genéricos, apresentamos a solução da representação gráfica de hierarquias através de árvores, como mecanismos auxiliares de seleção. Para podermos avaliar esta solução, era necessário implementá-la. Entretanto, para isso, é necessário poder acessar e modificar a estrutura de dados do editor gráfico. Isso impede que esta solução seja simulada para editores gráficos comerciais, ou seja, implementada como uma camada acima destes editores, pois geralmente não é fornecido ao usuário acesso programável a essas estruturas de dados. Implementamos então a solução da árvore no *TeCdraw*, um editor gráfico genérico do TeCGraf. Conforme esperado, esta solução transformou algumas operações trabalhosas e demoradas em operações simples e imediatas.

Como a solução da árvore não nos pareceu adequada para editores gráficos de interface do *layout* abstrato, criamos e implementamos a política hierárquica no Visual LED, que possui dois métodos de seleção em profundidade.

Visual LED será a base para um sistema completo de programação visual para a linguagem Lua, desenvolvida pelo TeCGraf e já em uso em produção (Figueiredo–Ierusalimschy–Celes 1994). Este sistema será chamado Visual Lua.

Agradecimentos

Carlos Henrique Levy colaborou de perto, fornecendo acesso oficial às estruturas de dados do IUP. O sistema IUP/LED e Visual LED estão sendo desenvolvidos em parceria com o CENPES/PETROBRÁS. Os autores são parcialmente financiados com bolsas de formação e pesquisa do CNPq.

Referências

G. Avrahami, K. P. Brooks, M. H. Brown, A two-view approach to constructing user interfaces, *Computer Graphics* **23** (1989) 137–146 (Proceeding of SIGGRAPH '89).

W. Celes Filho, L. H. de Figueiredo, M. Gattass, R. Ierusalimschy, Estratégias de reuso de *software* no TeCGraf, *Monografias em Ciência da Computação* **20/94**, Departamento de Informática, PUC-Rio, 1994.

L. H. de Figueiredo, M. Gattass, C. H. Levy, Uma estratégia de portabilidade para aplicações gráficas interativas, *Anais do VI SIBGRAPI* (1993) 203–211.

L. H. de Figueiredo, R. Ierusalimschy, W. Celes Filho. The design and implementation of a language for extending applications, *Anais do XXI Semish* (1994) 273–283.

R. Fisher, *Genesys: sistema híbrido para modelagem de sólidos*, dissertação de mestrado, Departamento de Informática, PUC-Rio, 1991

D. E. Knuth, *The TEXbook*, Addison-Wesley, 1984.

C. H. Levy, *IUP/LED: uma ferramenta portátil de interface com usuário*, dissertação de mestrado, Departamento de Informática, PUC-Rio, 1993.

M. A. Linton, J. M. Vlissides, P. R. Calder, Composing user interfaces with InterViews, *IEEE Computer* **22** (1989) 8–22.

A. Marcus, *Graphic Design for Electronic Documents and User Interfaces*, ACM Press Tutorial Series, Addison Wesley, 1992.

F. Neelamkavil, O. Mullarney, Separating graphics from application in the design of user interfaces, *The Computer Journal* **33** (1990) 437–443.

R. O. Prates, *Visual LED: uma ferramenta interativa para criação de interfaces gráficas*, dissertação de mestrado, Departamento de Informática, PUC-Rio, 1994.

J. Vlissides, S. Tang, C. Brauer, *Ibuild User's Guide*, October 1991.