MacPRO
o suplemento dos power users
Ano 4 - N°33
Parte integrante da revista Macmania
Não pode ser vendido separadamente

Cadê o programa que estava aqui?

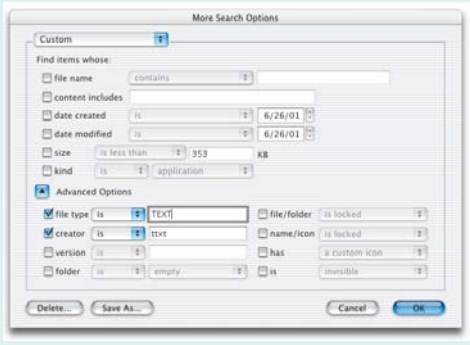
Com o Mac OS X, os arquivos passam a ter as famigeradas extensões .algumacoisa. Terá a Apple dado um passo para trás?

por Rainer Brockerhoff

Pouca gente conhece os sórdidos detalhes, mas muita coisa complexa acontece quando você clica duas vezes no ícone de um arquivo. No Mac OS 9.1 e anteriores, o Finder fazia uso de vários recursos do sistema para descobrir qual aplicativo deveria processar aquele arquivo. O Desktop Manager, o File Manager, o Translation Manager, o Internet Config, o Process Manager e outros mais obscuros conspiravam para fazer o documento abrir no programa certo. Aliás, boa parte desse processo já era desencadeado só para determinar o ícone a ser mostrado para cada arquivo!

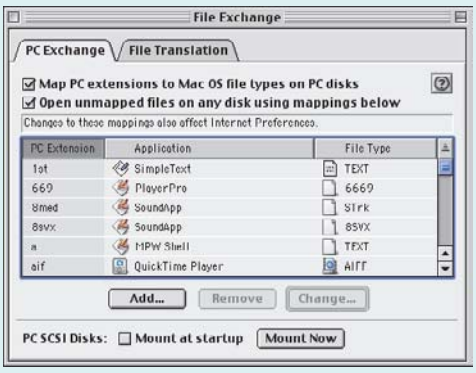
No Mac OS X, as extensões ficaram praticamente obrigatórias

Esta complexidade foi se agregando, historicamente, em várias camadas. Na aurora dos tempos, no System 1.0, havia uma base de dados global – mantida pelo sistema – que listava os vários aplicativos instalados, cada um com seu código único de quatro caracteres (o "Creator code"). Por sua vez, cada arquivo de dados tinha dois atributos – o "Creator" do aplicativo que o gerou e o tipo de arquivo, outro código de quatro caracteres (o "Type code").



Assim, quando o usuário clicava duas vezes num arquivo com Type/Creator igual a TEXT/ttxt consultava sua base de dados, via que havia um aplicativo chamado "TeachText" (que depois virou "SimpleText") cujo Creator era ttxt, e pronto - executava o TeachText e passava-lhe o arquivo para abertura. Tudo muito simples, certo? A grande jogada é que o arquivo poderia ter qualquer nome, e esse nome nunca interferia no tipo e ícone do arquivo.

Depois, com o advento da Internet e a influência de outros sistemas operacionais menos iluminados, foram se estabelecendo outros métodos de determinar o tipo de um arquivo. Como o Type/Creator geralmente não sobrevivia a um passeio pela Internet, nem ao armazenamento de arquivos em servidores Unix e Windows, esses métodos foram gradualmente sendo incorporados ao Mac OS. Por exemplo: no painel de controle File Exchange pode-se associar arquivos com extensões do Windows a certos aplicativos e tipos de arquivos. O painel Internet, similarmente, associa arquivos baixados pela Internet com determinado "MIME type" ao sistema Type/Creator.



Assim, no que agora se chama "Mac OS Clássico", tínhamos meios de enquadrar arquivos alienígenas no sistema Type/Creator do Mac OS. No entanto, o caminho inverso era ainda cheio de espinhos, uma vez que para mandar arquivos de volta para aqueles ambientes precisávamos nos informar sobre qual extensão colocar no nome.

Agora, no Mac OS X, a salada anterior de responsabilidades foi concentrada num único local: um recurso do sistema chamado Launch Services Manager. Isso simplificou a coisa para o desenvolvedor, uma vez que tem à sua disposição o mesmo recurso que o Finder usa para abrir um arquivo.

Antigamente, tinha-se que ou duplicar a complexa lógica de examinar extensões, Type e Creator, e consultar os vários "Managers", ou depender das boas graças do Finder para abrir um arquivo.

Para o usuário, algumas coisas também mudaram... e sob certo ponto de vista, infelizmente para pior. A partir de agora é obrigatório usar extensões em certos nomes – o sistema não coloca Type/Creator em várias circunstâncias! E não se pode alegar que isto é requisito da base Unix do Mac OS X – no Unix, o tipo de arquivo depende muito mais de onde ele está do que do seu nome. "Não", dizem os tradicionalistas, "o espírito do Windows nos invadiu!" Os aldeões com suas tochas estão subindo a estrada para o castelo do Dr. Frankenjobs, e clamando por sangue...

A justificativa oficial, citando o livro "Mac OS X: System Overview", é a seguinte (pág. 224 – "Why even have extensions"):

"...Usuários de Macintosh não vivem mais num limitado mundo Macintosh. Na era da Internet, documentos trafegam por redes heterogêneas, saindo de um Macintosh caseiro, passando por um servidor de rede Linux e chegando num PC com Windows numa rede corporativa. Cada computador nesse trajeto pode ter idéias diferentes sobre o que define o tipo de um documento e também sobre o que constitui um arquivo. Eles poderão não saber o que fazer com um arquivo cujo nome não tem extensão e tratá-lo erroneamente. Também ignorariam meta-dados como Type/Creator, perdendo-os irremediavelmente."

O problema aqui é que a maioria do que a Apple chama de meta-dados – Type/Creator, ícones personalizados, os antigos "Labels", etc. – não é suportada pelo Unix e outros sistemas. Assim, para armazená-los e tratá-los no Mac OS X, tem-se que recorrer a arquivos auxiliares invisíveis e outras gambiarras.

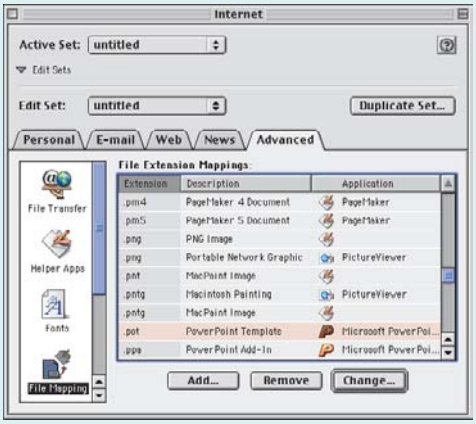
Para a criação de arquivos, aplicativos bem-comportados no Mac OS X devem sempre:

- Inserir seus valores apropriados no Type/Creator;
- Adicionar uma extensão condizente com o Type/Creator, se o usuário já não o fizer;

• Corrigir a extensão, se o usuário especificar uma extensão errada.

Notem que, na verdade, as extensões ficaram praticamente obrigatórias. Infelizmente, alguns aplicativos que vêm com o Mac OS X, como o TextEdit, não seguem esses mandamentos e criam arquivos com extensão, mas sem Type/Creator!

Neste período de transição, infelizmente convivemos com uma confusão: há arquivos com Type/Creator, com extensões, alguns com ambos, alguns com nenhum; e há aplicativos nativos do Mac OS X, aplicativos carbonizados e aplicativos clássicos. Socorro!



Como é, então, que o Finder (ou melhor, o "Launch Services Manager") do Mac OS X determina o que fazer quando o usuário clica duas vezes num arquivo? Ele supostamente segue os seguintes passos:

1 O usuário especificou (no "Show Info") que este arquivo deve ser aberto com determinado aplicativo? Se sim, usar esse aplicativo. (Note que especificar o aplicativo não muda o Type/Creator e nem afeta outros arquivos.)

2 O arquivo tem Creator? Se sim, procurar aplicativos que têm o mesmo Creator.

3 O nome tem extensão? Se sim, procurar aplicativos que dizem abrir arquivos com essa mesma extensão.

4 O arquivo tem Type? Se sim, procurar aplicativos que dizem abrir arquivos com esse Type.

5 Selecionar o primeiro aplicativo nativo ao Mac OS X da lista, se não houver, selecionar o primeiro aplicativo clássico.

6 Se houver várias versões do mesmo aplicativo na lista, selecionar a versão mais recente.

7 Se a lista estiver vazia, abre-se um diálogo pedindo ao usuário para especificar qual aplicativo deve ser usado.

Há uma ambiguidade aí sobre a ordem em que os aplicativos candidatos são colocados na lista. No Mac OS clássico, aplicativos que já estão executando têm preferência. Ainda não se tem a palavra oficial se isso também acontece no Mac OS X.

Antigamente, se havia vários aplicativos similares instalados, a ordem era bastante arbitrária – muitas vezes o mais recentemente instalado tinha prioridade, mas nem sempre. No Mac OS X a preferência é bem definida *(veja artigo na Macmania 84 para detalhes):*

• Aplicativos no domínio do usuário (~/Applications/)

• Aplicativos no domínio local (/Applications/)

• Aplicativos no domínio de rede (/Network/Applications/)

• Aplicativos em outros locais – em certos casos (por exemplo, para alguns utilitários), aplicativos instalados fora dos três primeiros locais podem não ser localizados corretamente.

Mas esses locais todos são pesquisados cada vez que se tenta abrir um arquivo? Não, seria muito demorado – e lembre-se de que algo muito semelhante é feito para decidir sobre o ícone de um arquivo.

Na prática, o Finder monta uma base de dados que indexa os arquivos, juntamente com os respectivos Type/Creator/extensões, e os ícones que cada um define para seus arquivos. No Mac OS 9, essa base de dados era individual de cada volume, mas global para todos os usuários – ela era remontada com o famigerado "Desktop Rebuild". No Mac OS X, a base é específica para cada usuário e é montada durante a inicialização do sistema, seguindo os domínios na ordem descrita há pouco. O interessante é que aplicativos "em outros locais" só são acrescentados à base na primeira vez que aquele usuário abre, no Finder, aquelas pastas.

Há muita controvérsia, nos comentários que se vê pela Internet, sobre o rumo que a Apple está seguindo no tratamento de tipos de arquivo. Alguns sugerem que se use apenas uma lista de MIME-types versus extensões, como faz o BeOS. Outros sugerem colocar extensões automaticamente no instante em que os arquivos saem para a rede e suprimi-las nos arquivos que chegam. Alguns até ameaçam retornar aos braços da Microsoft!

Alguns sugerem colocar extensões, mas escondê-las; isso levaria à confusão que há no Windows – que permite isso – onde se pode ter vários arquivos com (aparentemente) o mesmo nome e ícones diferentes. Na verdade, uma única extensão é escondida pelo Finder; é a extensão .app, que denota aplicativos no formato nativo do Mac OS X.

Alguns lembram, com razão, que usuários pokaprátika acham que basta mudar a extensão de um arquivo para mudar seu formato... Meu favorito é um programador iniciante que me escreveu reclamando que o primeiro programa dele (para PC) não funcionava. Quando fui investigar como estava compilando o programa, ele dizia que só mudava o nome de "programa.c" para "programa.exe" ("Ué, não é só fazer isso?"). Aliás, o Windows, neste caso, muda o ícone do arquivo imediatamente, reforçando a idéia errada!

Ainda estão para sair ferramentas realmente práticas para que o usuário do Mac OS X possa ajustar Type/Creator e extensões de acordo com sua necessidade. Enquanto isso, o remédio é estar bem informado sobre o mecanismo. Nas próximas atualizações do sistema, sem dúvida o mecanismo e as ferramentas disponíveis devem ser refinados, e acho que no ano que vem mal estaremos lembrando dessa controvérsia toda. **M**

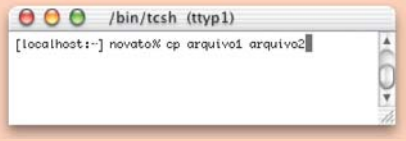
Metendo

Parte 2: copiando,

Na Macmania 85 você viu como se movimentar pelos diversos diretórios (ou pastas) do Mac OS X pelo Terminal, como saber sua localização e como visualizar a relação de arquivos em uma pasta. A Lição 2 irá um passo adiante, mostrando como copiar, renomear, mover e apagar arquivos, além de criar e apagar diretórios.

Passo 1

Existem duas maneiras básicas de copiar arquivos no shell, e ambas envolvem o comando `cp` (copy). A primeira maneira é, simplesmente, indicar o nome do documento que você deseja copiar, e o novo nome que gostaria de dar a esse documento.



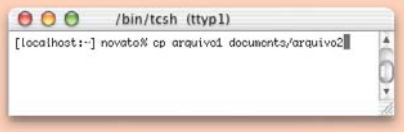
1 Se você não indicar um nome para o `arquivo2`, ele ficará com o mesmo do `arquivo1`.

2 Caso o documento `arquivo1` não exista, o comando acima não funcionará.

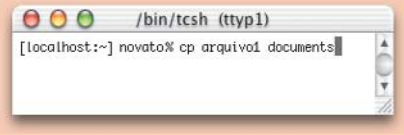
3 Se um documento com o nome `arquivo2` não existir, será criada uma cópia exata do `arquivo1`.

4 Se o documento `arquivo2` já existir, então este será apagado sem nenhum aviso e o novo documento `arquivo2` será criado.

Se você deseja copiar o documento `arquivo1` em outro diretório, precisa apenas especificar a localização desse novo diretório. Por exemplo: se você deseja copiar o documento `arquivo1` para um diretório chamado `documentos`, o comando será o seguinte:



Para manter o nome original do documento, especifique o diretório sem especificar um novo nome.



a mão no Unix

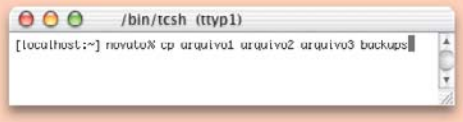
renomeando e apagando

por Alberto Mendonça

Epa! Como é que esse Unix sabe que eu não quero criar na mesma pasta uma cópia do `arquivo1`, chamada `documentos`? Isso aconteceria caso já não existisse um diretório chamado `documentos`.

Para copiar arquivos em outros diretórios, fora daquele onde está localizado o arquivo original, você deve rever alguns tópicos da Lição 1.

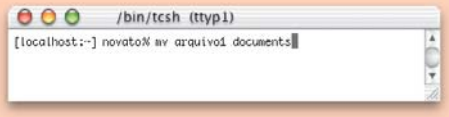
A segunda maneira de copiarmos documentos utilizando o comando `cp` é especificar mais de um documento a ser copiado e o diretório destino desses documentos. Para fazer um backup dos documentos `arquivo1`, `arquivo2` e `arquivo3` em um diretório já existente chamado `backups` você precisa executar o seguinte comando:



Essa operação pode ser entediante quando você precisa mover uma grande quantidade de documentos. Na Lição 3 examinaremos alguns comandos que podem nos ajudar nesses casos e que fazem da linha de comando uma maneira muito prática de manipular arquivos.

Passo 2

Um documento pode ser simplesmente movido, de forma semelhante a que fizemos para copiar arquivos, utilizando o comando `mv` (*move*). Especifique o nome do arquivo a ser movido e seu diretório de destino. Para mover o arquivo `arquivo1` para um diretório já existente chamado `documentos`, você precisa executar o seguinte comando:

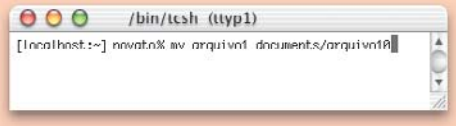


Utilizando o comando `ls` você pode verificar que o documento `arquivo1` não se encontra mais na pasta onde estava localizado, mas pode ser encontrado no diretório chamado `documentos`.

Passo 3

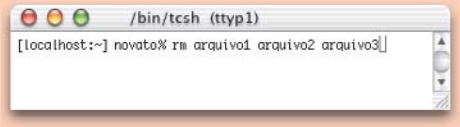
O comando `mv` também pode ser utilizado para renomear arquivos. Se isso parece confuso, pense nisso como mover um arquivo de diretório alterando seu nome original. Veja os exemplos a seguir:

1 O primeiro simplesmente renomeia o arquivo. O segundo renomeia o arquivo, movendo-o para outro diretório.



Passo 4

Para apagar um arquivo, utilizamos o comando `rm` (*remove*). Funciona de maneira semelhante aos comandos apresentados acima. Especificamos uma relação de arquivos a serem apagados.

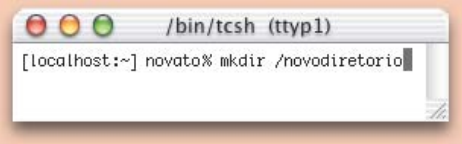


Lembre-se de que você apenas poderá apagar os arquivos que você possui permissão de editar. (Deixaremos o tópico "Permissões de Acesso" para uma próxima lição.)

Agora que já temos os conhecimentos básicos necessários para copiar, mover, renomear e apagar arquivos, vamos trabalhar com diretórios.

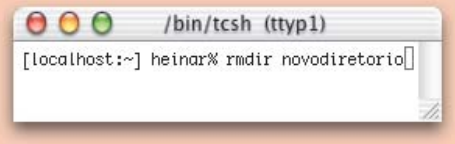
Passo 5

Para criarmos um diretório, utilizamos o comando `mkdir` (*make directory*) seguido do nome que desejamos dar ao diretório. Veja abaixo uma variedade de exemplos e tente descobrir onde cada um deles está sendo criado. Se você não for capaz de saber onde todos eles estão sendo criados, indicamos que revise a Lição 1.

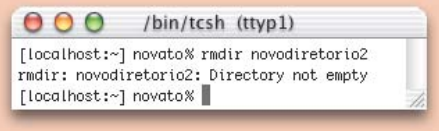


Passo 6

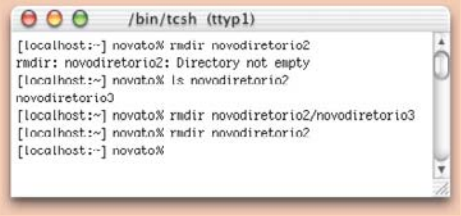
Remover um diretório é ainda mais simples, e inicialmente utilizaremos o comando `rmdir` (*remove directory*).



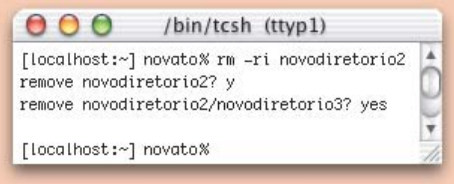
Um detalhe muito importante é que você não será capaz de eliminar um diretório que contenha algo.



Acima você foi notificado de que não é possível eliminar o diretório `novodiretorio2`, pois ele não está vazio. Observe que ele contém um outro diretório chamado `novodiretorio3`. Para continuar, precisamos eliminar todo o conteúdo do diretório que desejamos remover.



Essa tarefa pode se tornar inviável, mas temos um atalho, utilizando o comando `rm` que utilizamos anteriormente com arquivos. Se especificarmos `-r` (recursive), seremos capazes de eliminar o diretório e todo seu conteúdo. Seja cuidadoso ao utilizar esse comando!! Você pode acabar eliminando arquivos que não deveria! Para fazer essa operação de forma segura, indico que você utilize `-ri` (*recursive, interactive*). Assim você será consultado antes de cada arquivo ser apagado, pedindo sua aprovação.



Agora que já temos os conhecimentos básicos necessários para copiar, renomear e apagar arquivos, assim como criar e apagar diretórios, digite na janela de terminal `man cp`, `man mv`, `man rm`, `man mkdir` ou `manrmdir` para visualizar o manual de cada um desses comandos. M

ALBERTO MENDONÇA
É desenvolvedor Mac (adorado por alguns, odiado por outros).

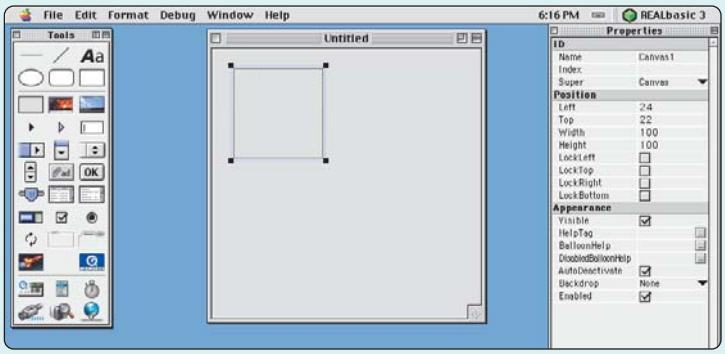
Construa um programa de desenho

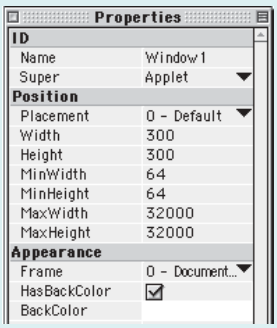
Curso de REALbasic 3, parte 4

por Gilbert Canaan

Depois de criar um jogo e um tocador de MP3, chegou a hora de colocar o REALbasic à prova. Vamos construir um programa de desenho. Não, esse software não tem Gaussian Blur nem curvas Bézier e não aceita plug-ins. Para dizer a verdade, nem mesmo irá salvar um arquivo. Ele simplesmente mostra como um programa de desenho pode ser construido no REALbasic. Afinal, um bom artista só precisa das cores primárias para pintar um bom quadro. O resto é luxo!
Por causa do volume de código que esse programa requer, este tutorial continuará na próxima edição. Vamos começar, como de costume, com o mais fácil: a interface.

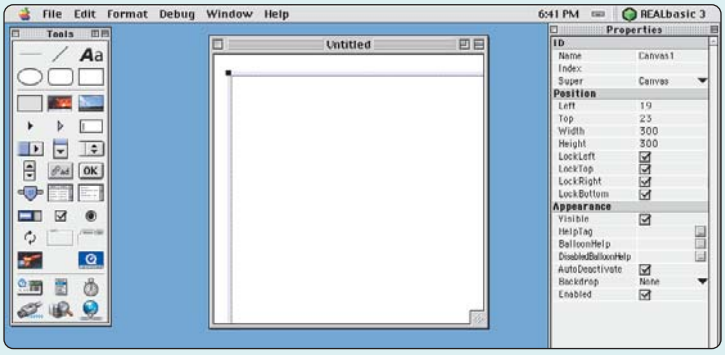
1 Depois de abrir o REALbasic, arraste o controle Canvas da Palete de Ferramentas (o botão com uma figura de paisagem, como um quadro) para dentro dela. Este Canvas servirá para você desenhar nele. Pense no controle Canvas como sendo uma tela de pintura. A janela funcionará simplesmente como a moldura do quadro.



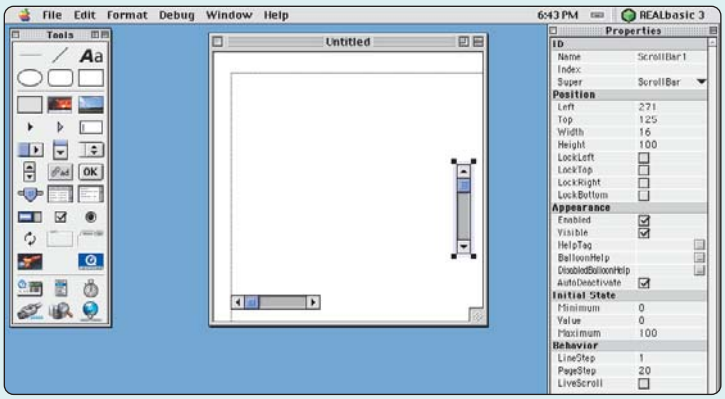


2 Não é agradável desenhar numa tela cinza, por isso vamos mudar a cor para branco. Clique dentro da janela (fora do Canvas) e depois marque a propriedade HasBackColor na janela de Propriedades (Properties).

3 É preciso ajustar o tamanho do Canvas de maneira que ele fique do mesmo tamanho da janela. Também temos que "trancar" o Canvas para que, quando o usuário aumentar o tamanho da janela, o Canvas também aumente junto com ela. Na janela de Propriedades, coloque em Altura (Height) e Largura (Width) o valor 300 e marque os itens em LockLeft, LockTop, LockRight e LockBottom.

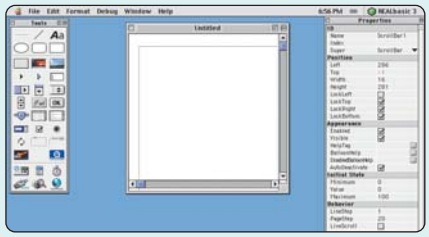


4 Todo programa gráfico que se preze tem que ter barras de rolagem *(scrollbars)* para que o usuário possa desenhar numa área maior dentro da janela. Para implementar essa função, arraste os controles Scrollbar horizontal e vertical da Palete de Ferramentas.

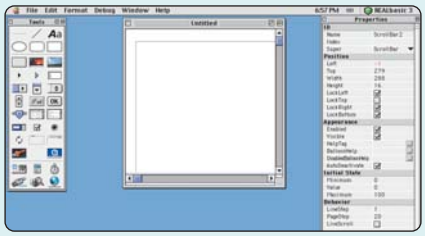


5 Agora você precisa posicioná-las e dimensioná-las de maneira que possam controlar o Canvas de maneira correta. Como regra, toda barra de rolagem tem uma largura de 16 pixels, ou seja, na horizontal, a propriedade Height é 16. Na barra vertical a propriedade Width é 16. Uma outra curiosidade é que na scrollbar horizontal a propriedade Left é -1, e na vertical a propriedade Vertical é -1. Isso acontece para que não apareça uma borda de dois pixels na lateral da barra. As scrollbars também devem ser "trancadas" para que possam ser aumentadas ou diminuídas junto com a janela. As propriedades de cada uma são:

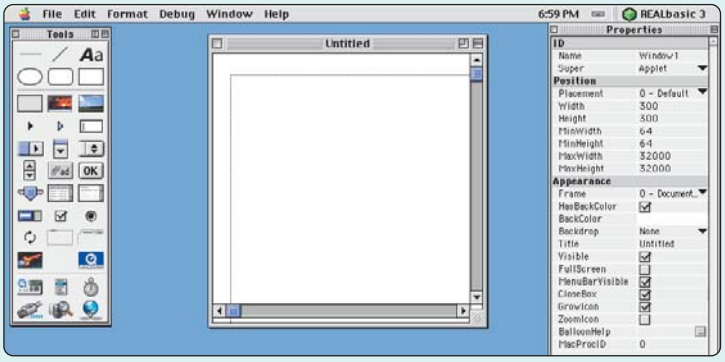
Horizontal:
Left: -1
Top: 279
Width: 288
Height: 16
LockLeft: ☑
LockRight: ☑
LockBottom: ☑



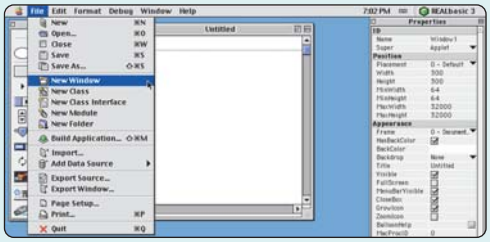
Vertical:
Left: 286
Top: -1
Width: 16
Height: 281
LockTop: ☑
LockRight: ☑
LockBottom: ☑

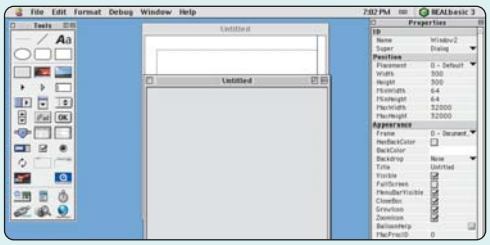


6 Se você executar o programa agora, verá que a janela não muda de



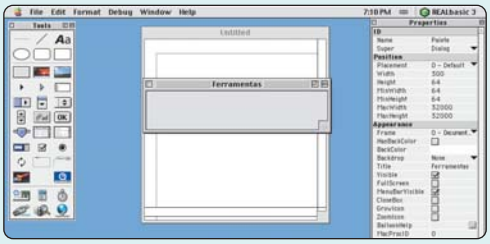
tamanho. Isso acontece porque a propriedade GrowIcon não foi ativada na Janela de Propriedades. Ative essa propriedade e depois execute o programa e verá que agora a janela poderá ser redimensionada.





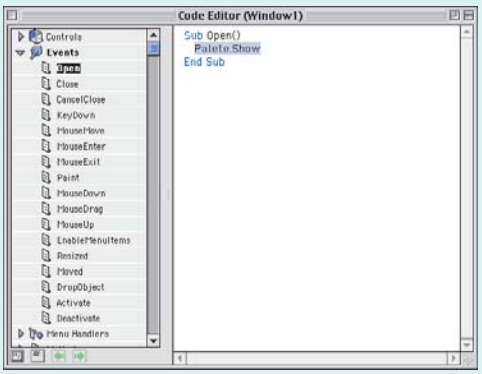
7 Agora é que vem a parte legal: a palete de ferramentas. Ela é interessante porque nunca é sobreposta por outra janela, está sempre visível. Isso acontece porque as paletes são janelas flutuantes. Para criar uma palete, você usa o mesmo procedimento de criar uma janela. Selecione o menu Arquivo ▸ Nova Janela (File ▸ New Window). Isso criará uma nova janela dentro da Janela de Projeto.

8 Na Janela de Projeto, abra a Janela 2 com um duplo clique e digite as seguintes propriedades:



Name: `Palete`
Super: `Dialog`
Placement: `0`
Width: `300`
Height: `64`
MinWidth: `64`
MinHeight: `64`
MaxWidth: `32000`
MaxHeight: `32000`
Frame: `3 - Floating Window`
Backdrop: `Nenhum (None)`
Title: `Ferramentas`
Visible: ☑
MenuBarVisible: ☑

No Ambiente de Design você não vai notar nenhuma diferença na aparência da janela, mas quando executar o programa, verá que ela tem a aparência de uma Palete. Porém, mesmo que você execute o programa agora, a janela não aparecerá. Isso acontece porque o REALbasic só pode abrir automaticamente uma janela de cada vez. Se você quiser abrir mais de uma quando o seu aplicativo for iniciado, terá que dizer ao programa para fazer isso.



9 Essa será a única linha de código que vamos digitar nesta parte do tutorial. Clique novamente na janela 1 (a da interface) e dê um duplo clique dentro dela. No Editor de Código, digite no evento Open:

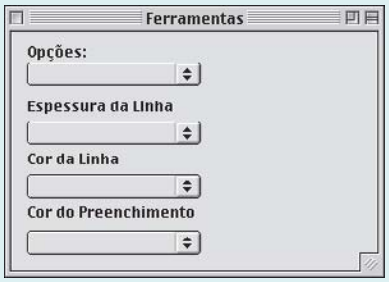
```
Palete.Show
```

Atenção: O código chamando a palete de Ferramentas, na verdade, deveria ser digitado em um objeto Application, que é usado como depósito de código global do programa. Mas para simplificarmos o tutorial, e como nosso programa só abrirá uma janela por vez, não tem problema.

10 Agora que a palete pode ser aberta, arraste quatro controles Static Text (o das letras "Aa") para ela e digite os seguintes nomes na propriedade Text para cada uma delas na Janela de Propriedades:


```
Opções:
Espessura da Linha:
Cor da Linha:
Cor do Preenchimento:
```

Arraste as janelas até ver o texto aparecendo completamente.



11 Para adicionar a funcionalidade de ferramenta na palete, arraste quatro controles Pop-up menus e posicione-os abaixo do texto.

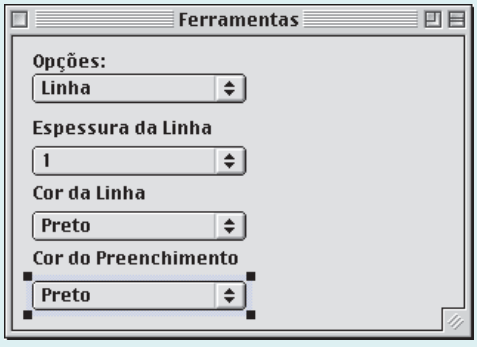
12 Agora você pode digitar os textos dentro de cada Pop-up menu. Selecione o primeiro controle Pop-up menu e na Janela de Propriedades, clique no botão Editar (Edit). Uma caixa de diálogo aparecerá e dentro dela você poderá digitar as funções de cada Pop-up menu. Siga o nosso exemplo:

Opções:
`Linha`
`Retângulo`
`Oval`

Espessura da Linha:
`1`
`2`
`3`
`4`
`5`
`6`

Cor da Linha:
`Preto`
`Vermelho`
`Verde`
`Azul`

Cor de Preenchimento:
`Preto`
`Vermelho`
`Verde`
`Azul`



13 Quando terminar de digitar esses valores, verá que eles aparecerão nos controles Pop-menu na palete de ferramentas.

Bem, é isso aí. Por hoje é só. Agora você pode tomar um cafezinho e executar o programa só para ver a palete de Ferramentas aparecer junto com a janela principal. Na segunda parte, nós digitaremos o código necessário para fazer tudo funcionar. E não se esqueça de sempre salvar o projeto a cada novo passo dado. **M**

GILBERT CANAAN
É fundador da Canvicz Software e trabalha com Mac desde 1988. Colaborou **Sérgio Miranda**