DESENVOLVENDO GTK COM JAVA

Hudson Simões Ribeiro - Jefferson Grijó Ferreira Pereira
Associação Educacional Dom Bosco
Estrada Resende-Riachuelo, 2535 - Resende - RJ
hudson_ribeiro@yahoo.com.br - jeffersongrijo@yahoo.com.br

RESUMO

Este artigo tem como objetivo apresentar idéias sobre desenvolvimento multiplataforma isolando interfaces gráficas do código. Estaremos apresentando o uso do conjunto de bibliotecas do GTK para desenvolvimento da interface, a portabilidade da linguagem C e JAVA para programação do código.

Apresentaremos também ferramentas que auxiliam no desenvolvimento conjunto, o GLADE para interface (GTK) e ECLIPSE para programação do código (JAVA) e um estudo de caso de uma ferramenta que deu origem ao GTK, o GIMP.

INTRODUÇÃO

Desde o surgimento do computador, o grande desafio dos programadores é construir um software e utilizá-lo em qualquer plataforma sem a necessidade de reescrever o código.

Há um grande número de variáveis a serem resolvidas para que esta portabilidade seja alcançada, como diferença entre sistemas operacionais (gerenciamento de processos, arquivos, memória e etc.) e até mesmo de hardware.

Neste artigo iremos abordar algumas soluções existentes, tanto para portabilidade de código (linguagem C e Java) quanto para portabilidade de interface (utilização do GTK).

DESCRIÇÃO

Os Sistemas operacionais definem a interface entre aplicações e recursos físicos. Infelizmente, esta interface pode significar limite de performance e liberdade de implementação de aplicações. Tradicionalmente, os sistemas operacionais escondem informações dos recursos da máquina utilizando abstrações de alto nível como os processos, arquivos, espaço de endereçamento e comunicação interprocessos. Estas abstrações restringem a flexibilidade de desenvolvimento de aplicações.

O problema se agravou ainda mais quando houve a substituição da interface em linha de comando por interface gráfica, que facilitou em muito o aprendizado de programas por parte dos usuários. Porém, essa transformação acarretou em mais um problema para os programadores ao desenvolverem uma aplicação, uma vez que agora além de se preocupar com o objetivo do programa (como os cálculos matemáticos, por exemplo) é necessário que ele desenvolva uma interface com objetos que interajam com o usuário (como caixas de texto ou botões) e que essa interface possa ser usada em qualquer plataforma.

Um dos itens mais importantes para tornar o programa multiplataforma é a portabilidade. Pode-se ter portabilidade de código (amplamente utilizada) e portabilidade de interface (mais difícil de ser praticada). A seguir serão abordados esses tipos de portabilidade.

PORTABILIDADE DE CÓDIGO

A portabilidade de código consiste em permitir que um determinado código fonte seja levado de uma plataforma para outra e compilado ou interpretado gerando assim um programa executável na plataforma destino. Abaixo são apresentados dois casos de portabilidade de código: Linguagem C e Java.

Linguagem C

De acordo com Trindade (2004), a linguagem C parece complicada e assustadora para alguns, mas na verdade possui as mesmas estruturas de controle da maioria das linguagens de programação estruturadas. O que a diferencia das demais (e talvez seja também o que assuste alguns desenvolvedores) é o seu status de "linguagem de baixo nível". O uso de ponteiros (pointers) e algumas outras estruturas um pouco mais complexas podem confundir o raciocínio do programador menos experiente no início de sua convivência com a linguagem C, mas com o tempo notam-se os benefícios do uso desse tipo de estrutura no dia-a-dia da engenharia de software.

A linguagem C foi criada na década de 70 por Dennis Ritchie, que trabalhava nos laboratórios Bell. Ele utilizou para isso o sistema operacional Unix e a linguagem BCPL. A linguagem BCPL havia servido de base para que Ken Thompson (que também participou do desenvolvimento da linguagem C) criasse a linguagem B. Como a linguagem de Ritchie sucedeu a linguagem B, recebeu o nome de "linguagem C".

Durante muito tempo à linguagem C foi distribuída junto à versão 5 do sistema Unix. A distribuição da linguagem junto ao sistema operacional mais importante da época, aliada ao então novíssimo conceito de portabilidade (o código em C produzido em uma máquina era facilmente recompilado em outra) causou a popularização do C e tornou necessária sua padronização. O padrão ANSI (American National Standard Institute) foi criado em 1983 para organizar a distribuição e o desenvolvimento de aplicações com base na linguagem C.

Muito se discute atualmente sobre a classificação (em nível) da linguagem C. Muitos autores a consideram uma linguagem de nível médio, outros a considera uma linguagem de baixo nível (como o ASSEMBLY). Particularmente, considero C uma linguagem de nível médio, pois apesar da facilidade de acesso a recursos avançados de hardware ela possui uma sintaxe relativamente simples e muitas vezes intuitiva (principalmente se compararmos com a sintaxe do ASSEMBLY).

C é uma linguagem portável. É claro que não estaremos comparando o padrão de portabilidade da linguagem C com o da linguagem Java, que foi desenvolvida pela Sun Microsystems tendo como objetivo principal alcançar o maior nível de portabilidade possível. Antes que houvesse uma padronização de hardware como há atualmente, as incompatibilidades entre linguagens de programação eram comuns. Um dos elementos de destaque de C quando de seu lançamento foi à substituição dos comandos por funções de entrada e saída de dados. A linguagem C também é capaz de esconder o código usado para uma determinada função do resto do programa, através de variáveis locais e outros conceitos relacionados. Por essa característica chamamos a linguagem C de linguagem estruturada. Porém, C não é uma linguagem estruturada em blocos (como Pascal), razão pela qual não se podem criar procedimentos e funções dentro de outras funções.

Apesar de sua idade, a linguagem C continua atual e segue sendo utilizada em diferentes plataformas. A linguagem C++, implementada a partir de C, é hoje a linguagem mais utilizada em aplicações comerciais. Resumidamente, podemos dizer que C++ é uma linguagem C bastante otimizada e com o advento da orientação a objetos. O domínio da linguagem C é pré-requisito para uma operação consistente com a linguagem C++. Além disso, a popularização do ambiente Windows criou um novo nicho de aplicação de C: o desenvolvimento de DLL's, as famosas bibliotecas de vínculo dinâmico presentes em praticamente todo aplicativo Windows.

Linguagem Java

Segundo Alecrim (2003), Java é uma linguagem de programação orientada a objetos, desenvolvida por uma pequena equipe de pessoas na Sun Microsystems. Inicialmente elaborada para ser a linguagem-base de projetos de software para produtos eletrônicos, Java teve seu grande boom em 1995, devido ao sucesso mundial da World Wide Web. Mas o que tem Java a ver com a WWW?

Como foi dito, o propósito inicial do desenvolvimento de Java foi para funcionar em processadores de eletrodomésticos. Os projetistas de sistemas de controle desses processadores, descontentes com linguagens convencionais de programação, como C, propuseram a criação de uma linguagem específica para uso em processadores de aparelhos domésticos, como geladeiras e torradeiras. Todo o descontentamento dos projetistas residia no fato de que programas escritos e compilados em C são fortemente dependentes da plataforma para a qual foram desenvolvidos. Como o ramo de eletro-eletrônicos está em constante evolução, a cada novo liquidificador lançado no mercado com um novo processador embutido, um novo programa deveria ser escrito e compilado para funcionar no novo compilador, ou então, na melhor das hipóteses, para reaproveitar o antigo programa, no mínimo ele teria de ser re-compilado para o novo processador.

Um outro problema no uso de linguagens de programação tradicionais em produtos eletro-eletrônicos reside no fato de que o consumidor em geral quer uma longa vida útil para seu aparelho. De outro lado, é ponto pacífico que a maioria dos softwares são projetados já se prevendo sua breve obsolescência, ou seja, escreva um programa hoje e em poucos meses ele estará ultrapassado. Desta forma, sempre que novos processadores para eletrodomésticos fossem desenvolvidos, eles teriam de apresentar uma espécie de compatibilidade retroativa.

Por fim, os projetistas de software de eletrodomésticos desejavam que o software por eles fabricado fosse seguro e robusto, capaz de funcionar em um ambiente tão adverso quanto uma cozinha. E que fosse confiável também, mais ainda que um software normal, pois quando ocorre alguma falha em um aparelho eletro-eletrônico, peças mecânicas são trocadas, gerando um custo a mais pro fabricante.

No início de 1990, Naughton, Gosling e Sheridan começaram a definir as bases para o projeto de uma nova linguagem de programação, apropriada para eletrodomésticos, sem os problemas já tão conhecidos de linguagens tradicionais como C e C++. O consumidor era o centro do projeto, e o objetivo era construir um ambiente de pequeno porte e integrar esse ambiente em uma nova geração de máquinas para "pessoas comuns". A especificação da linguagem terminou em agosto de 1991, e a ela deu-se o nome de "Oak". Por problemas de copyright (já existia uma linguagem chamada Oak) o nome foi mudado em 1995 para Java, em homenagem à ilha de Java, de onde vinha o café consumido pela equipe da Sun.

Em 1992, Oak foi utilizada pela primeira vez em um projeto chamado Projeto Green, que tinha por propósito desenvolver uma nova interface de usuário para controlar os aparelhos de uma casa. Tal interface consistia em uma representação animada da casa, que era exibida em um computador manual [chamado star seven, bisavô dos palmtops de hoje], e que tinha uma tela sensível ao toque que permitia a manipulação dos eletrodomésticos. Essa interface era totalmente escrita em Oak, e evoluiu para um projeto de interface para redes de televisão pay-per-view. Contudo, o padrão proposto por esses dois projetos não vingou, e outros padrões, pelo menos em sistemas de TV pay-per-view vêm tomando conta do mercado. Um personagem animado desses projetos, Duke, tornou-se um dos símbolos de Java.

Em meados de 1993, pode-se dizer que Oak ia "mal das pernas". Os projetos propostos não eram economicamente viáveis, e não se via um grande futuro no desenvolvimento de aparelhos que suportassem essa nova linguagem. Justamente nessa época, a World Wide Web estava em seu nascimento, trazendo um novo horizonte para a Internet. (É importante lembrar que a Internet já existia muito antes do surgimento da WWW).

A WWW nada mais é que um conjunto de protocolos que permite um acesso mais amigável aos recursos disponíveis na Internet. Dentre esses protocolos, por exemplo, o mais conhecido em geral é o de transferência de hipertexto [http]. Com o lançamento do primeiro browser do mercado, o Mosaic, ocorreu à equipe de desenvolvimento da Sun que uma linguagem independente de plataforma, segura e robusta como a que estava sendo desenvolvida para eletrodomésticos caberia como uma luva para uso na Internet, uma vez que um aplicativo gerado nessa linguagem poderia rodar nos diversos tipos de computadores ligados na Internet, rodando qualquer sistema operacional, de PCs rodando OS/2 a estações

RISC rodando AIX Unix, ou SparcStations rodando Solaris, os programas escritos nessa linguagem que viria a ser conhecida por Java seriam o modelo para qualquer aplicativo Web.

Com o novo ânimo trazido pelo advento da WWW, a equipe da Sun desenvolveu um browser totalmente escrito em Java, tendo-o terminado no início de 1995 e denominado-o HotJava. O grande diferencial de HotJava para outros browsers da época (como o Mosaic, o Netscape Navigator e o Lynx) é que ele permitia a inserção de programas escritos em Java dentro de páginas HTML comuns. HotJava como browser foi um fiasco comercial, mas abriu os olhos dos desenvolvedores para um fato muito importante: as páginas HTML estariam fadadas a serem estáticas e sem ações embutidas em si, não houvesse uma linguagem padrão na qual fossem escritos programas que pudessem ser embutidos nas páginas Web. HotJava demonstrou que isso era possível (ou seja, incluir um programa, no caso escrito em Java, em uma página HTML rodando em um browser preparado para dar suporte à execução do programa, no caso o próprio HotJava). O grande "pulo do gato" de Java veio logo a seguir, quando a Netscape anunciou que sua próxima versão do browser Navigator iria dar suporte a aplicativos Java embutidos em documentos HTML. Em seguida, a Microsoft anunciou o mesmo para o seu Internet Explorer. E Java estourou no mundo, a Sun contabilizava inúmeros downloads de seu JDK, diversas empresas desenvolveram IDEs para a programação em Java, e vieram JavaScript, JavaBeans, a briga deste com ActiveX, e...

Java é muito parecida com C++, mas muito mais simples. Java não possui sobrecarga de operadores, structs, unions, aritmética de ponteiros, herança múltipla, arquivos .h, diretivas de pré-processamento e a memória alocada dinamicamente é gerenciada pela própria linguagem, que usa algoritmos de garbage collection para desalocar regiões de memória que não estão mais em uso.

O processo de compilação de um programa Java é feito de acordo com os seguintes passos: o código fonte (extensão .java) é compilado e armazenado em um arquivo de extensão .class. De cara, percebe-se a impossibilidade de utilizar-se de DOS como sistema operacional para a elaboração de aplicativos Java, uma vez que o mesmo tem um suporte limitado a nomes de arquivos. Mas essa limitação quanto ao nome dos arquivos é somente a razão aparente da não-portabilidade de Java para DOS. A grande razão reside no fato de que Java foi projetada para sistemas de 32 bits, e só foram escritas Máquinas Virtuais Java para ambientes de 32 bits.

A portabilidade de Java depende fortemente da existência de JVMs que rodem em diversas plataformas. Um programa Java rodará em um computador se existir uma JVM que nele rode. Ao contrário de programas Java, as JVMs devem ser programas feitos e compilados para máquinas específicas, de forma que serão as JVMs as responsáveis pela tradução de bytecodes Java para as linguagens nativas das máquinas.

Bytecode é um formato de código intermediário entre o código fonte, o texto que o programador consegue manipular, e o código de máquina, que o computador consegue executar. Na plataforma Java, o bytecode é interpretado por uma máquina virtual Java (JVM). A portabilidade do código Java é obtida à medida que máquinas virtuais Java estão disponíveis para diferentes plataformas. Assim, o código Java que foi compilado em uma máquina pode ser executado em qualquer máquina virtual Java, independentemente de qual seja o sistema operacional ou o processador que executa o código (Ricarte, 2000).

PORTABILIDADE DE INTERFACE

Conforme Gomes (2004), a substituição da interface em linha de comando por interface gráfica facilitou em muito o aprendizado de programas por parte dos usuários. Porém, essa transformação acarretou em mais um problema para os programadores ao desenvolverem uma aplicação, uma vez que agora além de se preocupar com o objetivo do programa (como os cálculos matemáticos, por exemplo) é necessário que ele desenvolva uma interface com objetos que interajam com o usuário (como caixas de texto ou botões).

Além disso, o mercado de sistemas operacionais também está sofrendo mudanças, uma vez que novos sistemas operacionais estão aumentando significativamente a concorrência contra o Windows (atual padrão de mercado), como é ocaso do Linux©, por exemplo. Logo, surge um novo desafio ao programador: portar seus programas de modo que possam ser compilados em várias plataformas.

Para resolver o primeiro problema apresentado podem ser utilizadas ferramentas apropriadas para o desenvolvimento de interfaces, conhecidas como IDEs (Integrated Development Enviroment), as quais são de ótimo desempenho e qualidade. Porém essas ferramentas geralmente destinam-se a uma única plataforma e, às vezes, podem custar bastante caro.

Como o mercado atual requer de um programa que este seja portável e a principal dificuldade encontrada para dar essa característica de portabilidade a um código de programa é justamente no desenvolvimento de interfaces gráficas, surgíramos kits PIGUI (Plataform-Independent Grafical UserInterface) como um meio de se esquivar de tal problema.

Um kit PIGUI é uma biblioteca de software que um programador usa para produzir códigos de interfaces gráficas para diferentes sistemas computacionais. O kit apresenta funções e/ou objetos que são independentes de qual interface gráfica o programador tem como objetivo. O kit não necessariamente fornece quaisquer características adicionais de portabilidade (McKay, 1997).

A idéia lançada é que debates sobre onde o Macintosh é mais fácil de usar que o Windows, ou os méritos do Unix sobre o Windows NT, ou qual será o futuro do OS/2, irão continuar. Mas se um programador usa um kit multiplataforma para construir suas aplicações, ele não terá que ficar apostando sua experiência nos resultados desses debates (Apiki, 1994).

Quando um programador julgar melhor utilizar um kit PIGUI para desenvolver seu programa para que este possa ser compilado em várias plataformas, ele não terá (teoricamente) que mudar qualquer linha de seu código. Com tal kit, quando ele decidir colocar um menu na tela, ele chamará a função 'PIGUI_menu' do kit. Quando ele compilar seu código em um Macintosh, a biblioteca PIGUI colocará um menu de Macintosh na tela em resposta à chamada PIGUI_menu.

O Gtk (Gimp Tool Kit) é um conjunto de bibliotecas para desenvolver interfaces GUI (Graphics User Interface). O Gtk é um conjunto bastante completo, e possui uma grande quantidade de extensões. É prática comum que os projetos de software livre baseados em Gtk liberem seus componentes mais genéricos através de bibliotecas.

Os primeiros idealizadores do GTK foram Peter Mattis, Spencer Kimball, Josh MacDonald no início da década de 90, e tem esse nome, pois foi originalmente concebido para servir de ferramenta no desenvolvimento do GIMP (GIMP Tool Kit). O GIMP (GNU Image Manipulation Program) é um poderoso editor gráfico, ao estilo do Adobe Photoshop, com licença GPL, o que significa que ele é gratuito, e pode ser distribuído livremente.

Devido à versatilidade das funções do GTK, hoje este é utilizado na produção de diversos outros programas além do GIMP (GNU Image Manipulation Program), que variam desde pequenos utilitários, como o GTK-ICQ, até grandes projetos, como o gerenciador de Desktop GNOME.

O GTK é na verdade um conjunto de widgets (você verá o significado desta palavra mais adiante), e para facilitar o seu porte para outras plataformas, o GTK divide- se em 4 camadas de bibliotecas:

As duas primeiras camadas, GLib (GNU Library - biblioteca de funções gerais) e GDK (GIMP Drawing Kit - biblioteca responsável pelas funcionalidades gráficas), são dependentes da plataforma , e chamam outras funções de baixo nível do ambiente gráfico em que o programa é compilado (XLib), já a Atk (biblioteca que suporta acessibilidade) e a Pango (biblioteca que suporta internacionalização do Gtk) são independentes de plataforma, aumentando assim a portabilidade.

INTERFACE DE DESENVOLVIMENTO GRÁFICO

O GTK oferece suporte para todos os principais tipos de objetos gráficos, como por exemplo, botões, menus e barras de rolagem e etc. Objetos do tipo caixas de posicionamento são utilizadas para posicionar os objetos na tela, de maneira relativa, de forma que a janela pode ser livremente redimensionada pelo usuário (como em JAVA).

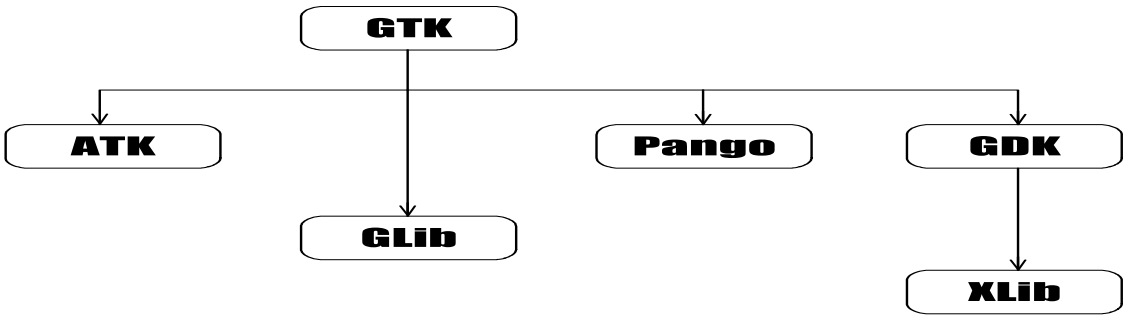

Figura 1 – Arquitetura do GTK

Quem estiver acostumado com linguagens orientadas a objetos (Delphi, Visual Basic, Dot Net, Visual C, etc), terá grande facilidade em migrar para GTK, devido os conceitos ser os mesmos.

INTEGRAÇÃO DAS FERRAMENTAS GLADE E ECLIPSE

Para facilitar a integração do GTK com o Java, utilizamos duas ferramentas: Eclipse (código) e Glade (interface).

ECLIPSE

O eclipse é uma comunidade aberta da fonte cujos projetos sejam focalizados em fornecer uma plataforma extensível do desenvolvimento e estruturas da aplicação para o software do edifício. O eclipse fornece as ferramentas e as estruturas que medem o ciclo de vida do desenvolvimento do software, incluindo a sustentação para modelar, os ambientes do desenvolvimento da língua para Java, o C/C++ e o outro, testar e desempenho, inteligência de negócio extensível, aplicações ricas do cliente e desenvolvimento encaixado. Um eco sistema grande, vibrante de vendedores principais da tecnologia, umas partidas inovativas, umas universidades e umas instituições e uns indivíduos de pesquisa estendem, complementam e suportam a plataforma do eclipse.

O eclipse é uma plataforma aberta para a integração da ferramenta construída por uma comunidade aberta de provedores de ferramenta. Operando-se sob um paradigma de fonte aberto, com uma licença pública comum que forneça o código de fonte livre, a plataforma de eclipse proporciona para o programador da ferramenta, com a flexibilidade e o controle finais sobre sua tecnologia do software.

Eclipse formou um sistema aberto independente em torno da tecnologia livre e de uma plataforma universal para a integração das ferramentas. Eclipse fundou ferramentas baseadas em código livre e dão a colaboradores a liberdade da escolha em uma multi-língua, multiplataforma,. Com isso faz com que ele seja mais fácil de criar, interagir e utilizem ferramentas do software, o tempo conservando e o dinheiro. Colaborando e explorando a tecnologia de integração do núcleo, os produtores da ferramenta enlatam reusar da plataforma da força de alavanca e concentram-no em competências do núcleo para criar a tecnologia nova do desenvolvimento.

A plataforma do eclipse é escrita na língua de Java e vem com os toolkits de encaixe extensivos e os exemplos da construção. Tem sido desdobrada já em uma escala de estações de trabalho do desenvolvimento including Linux, Cavalo-força-ux, AIX, Solaris, QNX, OS X do mac e sistemas baseados Windows. Uma descrição cheia da comunidade do eclipse e dos

papéis brancos que documentam o projeto e o uso da plataforma do eclipse está disponível em http://www.eclipse.org .

A fundação do eclipse é um corporation non-profit dado forma para avançar a criação, a evolução, o promotion, e a sustentação da plataforma do eclipse e para cultivar uma comunidade aberta da fonte e um ecosystem de produtos complementares, de potencialidades, e de serviços. Você pode aprender mais sobre a estrutura e a missão da fundação do eclipse lendo os originais formais que estabelecem como a fundação se opera, e lendo a liberação de imprensa que anuncia a criação da organização independente.

GLADE

O Glade é um programa que lhe permite desenhar rapidamente interfaces gráficas multiplataforma, o Glade torna-se uma ferramenta interessante para a construção desse tipo de aplicativos, despontando, com o auxílio do ambiente de desenvolvimento MonoDevelop, como uma forte alternativa livre ao Visual Studio.NET.

Esta ferramenta que permite a criação de Interfaces sem necessidade de programação direta (no estilo Delphi e Visual C), facilitando drasticamente a manutenção dos objetos gráficos (Ex.: Alterar Texto do Botão ou Acrescentar novo item ao Menu Principal). A descrição da interface é armazenada em um arquivo XML, o que facilita que outros programas utilizem esta descrição. Este é o caso da biblioteca libGlade, que permite que a interface de um programa seja criada em tempo de execução, a partir da descrição em XML, possibilitando assim a alteração sem necessidade de recompilação (bastando apenas editar o arquivo XML, utilizando uma ferramenta da sua escolha).

O que o Glade pode fazer:

Desenvolver a interface e associar código a ela; Criar chamadas vazias e manipuladores de sinais para ligar a interface com o código da aplicação;

O que o Glade não pode fazer:

O Glade não pode ajudá-lo em todo o processo de codificação de sua aplicação, pois não incluir um compilador, editor ou debugger. Ele pode ser utilizado em conjunto com essas ferramentas.

GIMP, UM EXEMPLO DE PORTABILIDADE

O GNU Image Manipulation Program ou GIMP é um editor de imagens bitmap, um programa para criação e edição de imagens de bitmap que utiliza as bibliotecas GTK. Também tem suporte a formatos de imagem vetorial. O projeto foi criado em 1995 por Spencer Kimball e Peter Mattis e hoje é mantido por um grupo de voluntários; é licenciado sob a GNU General Public License.

O nome GIMP originalmente era sigla de General Image Manipulation Program; em 1997, ele foi mudado para GNU Image Manipulation Program. Ele é integrante oficial do Projeto GNU.

O GIMP é muito utilizado para processamento de imagens e fotografias exibidas na Internet. Seus usos incluem criar gráficos e logotipos, redimensionar fotos, alterar cores, combinar imagens utilizando o paradigma de camadas, remover partes indesejadas das imagens e converter arquivos entre diferentes formados de imagem digital.

O GIMP também tem a característica de talvez ser o primeiro grande projeto de código aberto para usuários finais. Outros trabalhos, como o GCC, o kernel Linux, entre outros, eram ferramentas desenvolvidas por programadores, e com público alvo, especialmente, de programadores. O GIMP foi a prova de que projetos de código aberto e livres poderiam desenvolver coisas para serem usadas por pessoas normais, abrindo as portas para a união de esforços que levaram ao desenvolvimento de outros grandes projetos como o GNOME, o KDE, o Mozilla e o OpenOffice.org, e vários outros aplicativos que os seguiram.

O GIMP não foi criado como uma alternativa livre ao Photoshop, foi um projeto universitário que amadureceu bastante e já é usado profissionalmente. Porém o GIMP ainda tem um fatia de mercado muito menor que a do Photoshop, talvez porque:

Não existia um bom suporte ao padrão CMYK color space até a versão 2.0, usado em produção gráfica impressa.

Photoshop inclui licenças para o padrão Pantone, contudo o GIMP tem ótimas paletas e pode contornar esse problema muito bem.

O número de Plugins do Photoshop é maior. Porém o gimp tem um conjunto de scripts que permite que se aumentem os efeitos indeterminadamente.

Até o início de 2004 a GTK (biblioteca para gerenciar janelas usada pelo GIMP) não se apresentava bem no MS Windows. A melhora da GTK facilitou o uso de outros programas como o Inkscape no Windows.

Assim como o uso interativo, o GIMP pode ser inserido em scripts e chamadas de sistemas em programas compilados. Para isso pode-se usar, Scheme (ou ScriptFu), Perl, Python, Tcl, Ruby, e programas capazes de executar comandos UNIX. Isso permite se escrever plugins e scripts que utilizem o vim sem interface com o usuário; é possível, por exemplo, produzir imagens para uma página web utilizando scripts CGI, ou realizar correção de cor ou redimensionamento de imagens em lote. Para usos não interativos, entretanto, é recomendável se utilizar outros programas, de poder superior, como o ImageMagick, por exemplo.

GIMP utiliza o GTK como base para construção de sua interface. Na verdade, GTK originalmente fazia parte do GIMP. GIMP e GTK foram desenvolvidos para o X Window System (servidor gráfico) rodando em UNIX ou GNU/Linux, mas depois foram portados para Microsoft Windows, OS/2 e MacOS X.

A versão atualmente em produção do GIMP é a 2.2.8 (lançada em 26/06/2005).

O GIMP usa uma nova biblioteca gráfica genérica chamada GEGL

A Interface

A interface do Gimp não muda, não importa em qual plataforma ele esteja instalado.

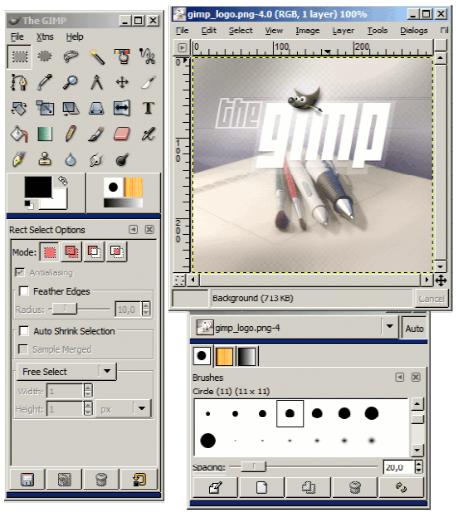

Figura A

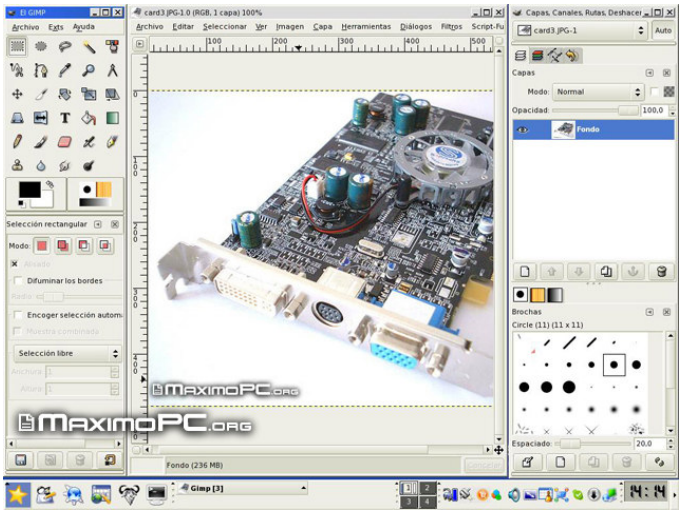

Figura B

Acima o Gimp no ambiente Windows (Figura A) e no ambiente Linux (Figura B).

Os usuários mais acostumados com o ambiente e os programas Windows terão um pouco de dificuldades em se acostumar com a interface do GIMP, já que esta não conta com o esquema padrão de uma janela principal, onde ficam todas as outras janelas do programa.

No GIMP, existe uma pequena janela, onde ficam o menu e os principais botões necessários para as tarefas de manipulação de imagem. Além dela, existem a janela de desenho e as outras janelas, que podem ficar visíveis de acordo com o gosto do usuário. Algumas janelas importantes são as de camadas, opções de ferramentas, pincéis e degrades.

Outro detalhe muito importante é com relação ao menu, outra coisa meio diferente no GIMP. Você precisará colocar na cabeça que para acessar qualquer menu principal do GIMP, basta um clique do botão direito do mouse sobre a janela de desenho. O mesmo vale para acessar menus específicos de outras janelas. Com o tempo, é bom ir se familiarizando com as teclas de atalho, que agilizam em muito o trabalho.

Tome cuidado para não se perder em meio a seus trabalhos. Devido à natureza da interface do GIMP, sem uma "janela-mãe", você pode acabar fazendo confusão ao ter muitas imagens abertas.

CONCLUSÃO

Este artigo apresentou idéias sobre desenvolvimento multiplataforma isolando interfaces gráficas do código, através das ferramentas de desenvolvimento GLADE para interface (GTK) e ECLIPSE para programação do código (JAVA) e um estudo de caso de uma ferramenta que deu origem ao GTK.

E notório que ganhasse muito em portabilidade, porém é necessária a preparação da plataforma utilizada, esta preparação para os iniciantes é bastante árdua devido à quantidade de pacotes necessários para a integração entre o JAVA e o GTK, mas o trabalho é recompensado com uma programação leve e eficiente.

REFERÊNCIAS

TRINDADE, Cristiano. (2004) Programação para plataforma Palm OS – Parte 01. Endereço eletrônico: http://www.imasters.com.br/artigo.php?cn=2647&cc=21

ALECRIM, Emerson. (2003) Linguagem Java. Endereço eletrônico: http://www.infowester.com/lingjava.php

RICARTE, Ivan Luis Marques. (2000) Bytecodes. Endereço eletrônico: http://www.dca.fee.unicamp.br/cursos/PooJava/javaenv/bytecode.html

GOMES ,Renato de Souza; SCHNEIDER, Bruno de Oliveira; UCHÔA, Joaquim Quinteiro. (2004) Desenvolvimento Multiplataforma de Interfaces Gráficas. Endereço eletrônico: http://www.dcc.ufla.br/infocomp/artigos/v2.1/rsgomes.pdf

MCKAY, Ross. (1997) Plataform Independent FAQ. Endereço eletrônico: http://www.zeta.org.au/rosko/pigui.htm.

APIKI, Steve. (1994) Paths to Plataform Independence. Endereço eletrônico: http://www.byte.com/art/9401/sec9/ art1.htm.

Endereço eletrônico: http://pt.wikipedia.org/wiki/The_GIMP - acessado em 10/09/2005.