by Hilaire Fernandes
<hilaire/at/ofset.org>

About the author:

O Hilaire Fernandes é vice−presidente da OFSET, uma organização para promover e desenvolver software livre para educação no projecto Gnome. Ele também escreveu o Dr. Geo um software para geometria dinâmica. Presentemente está a trabalhar no Dr. Genius um outro software matemático com propósitos educacionais no projecto Gnome.

Translated to English by:
Guido Socher
<guido/at/linuxfocus.org>

Desenvolvendo aplicações Gnome com o Python (Parte 2)

Abstract:

Esta série de artigos é principalmente para programadores iniciantes na área do Gnome e GNU/Linux. O Python foi escolhida como a linguagem de programação porque os principiantes entram mais cedo nesta linguagem do que em linguagens compiladas como o C. Para entender este artigo precisa de algumas bases de programação em Python.

Utilitários necessários

O software preciso para executar o programa descrito foi listado no primeiro artigo desta série.

Precisa também de:

- O ficheiro original .glade [drill.glade] ;
- O código Python fonte do drill Python [drill.py].

O procedimento de instalação e utilização do Python−Gnome com o LibGlade está também descrito na primeira parte da série destes artigos.

Drill, o nosso suporte

O objectivo do primeiro artigo foi demonstrar o mecanismo e os modos de interacção entre os diferentes componentes de um programa escrito para um ambiente Gnome, Glade, LibGlade e Python.

O exemplo utilizava a widget `GnomeCanvas`. Este exemplo fornecia−nos uma apresentação colorida e mostrava−nos a facilidade de desenvolvimento com esta configuração.

Para as próximas secções, sugiro trabalhar com uma estrutura na qual explicaremos as widgets diferentes do Gnome. Este artigo concentra−se na configuração da estrutura. . Artigos posteriores utilizaram esta estrutura adicionando−lhe mais características para ilustrar as imensas widgets do Gnome.

A nossa estrutura modelo chama−se ***Drill***. Isto é uma plataforma para propósitos educacionais que será utilizada nos nossos exemplos e exercícios. Estes exemplos são, somente para propósitos educacionais para demonstrar a utilização das widgets.

Criando uma interface com o Glade

As widgets

A janela da aplicação é criada com o Glade. Como no artigo anterior primeiro cria uma janela para a aplicação Gnome. Para esta janela precisa de apagar os ícones e menus desnecessários.

A parte principal do ***Drill*** foi dividida em dois espaços de trabalho utilizando a widgets `GtkPaned`.

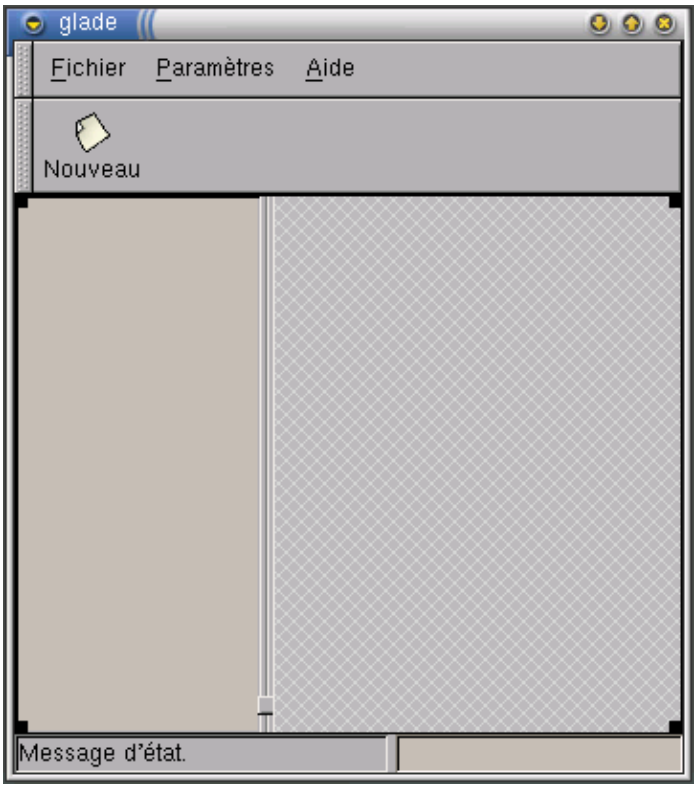

Fig. 1 – janela principal do Drill

Ambos os espaços de trabalho estão separados verticalmente com um manuseador para ajustar o tamanho de cada um. O espaço de trabalho esquerdo contem a arvore widget (`GtkTree`), na qual as diferentes partes do exercício serão armazenadas por categoria. O espaço de trabalho direito está vazio. É aqui que adicionaremos os exercícios segundo a escolha do utilizador.

A partir do, a vista da interface do ***Drill*** mostra–nos a estrutura dos seus componentes.

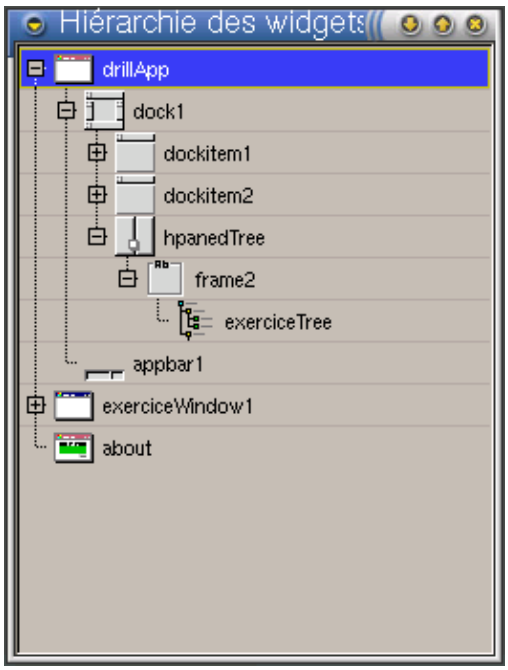

Fig. 2 – vista de árvore do Drill

Pode ver na Fig. 2 que a widget `hpanedTree` (do tipo `GtkPaned`) só contém uma widget, a `frame2` (do tipo `GtkFrame`), a lado esquerdo. A `frame2` contém a widget `exerciceTree`. É preferível inserir primeiro uma wdiget `GtkFrame` com uma sombra do tipo `GTK_SHADOW_IN` numa widget `GtkPaned`. Isto evita mascarar o manuseador.

Por último, a caixa de diálogo do Gnome "About ***Drill***" pode−se parecer com esta.

Fig. 3 – Caixa de Diálogo "About" Drill

Os seus diferentes itens são editados a partir do Glade na folha do `Widget` da janela `Properties`.

As widgets e o processamento das funções de nome

Utilize os seguintes nomes para estas widgets para as manipular com esses nomes no Python.

Janela de aplicação Gnome:
`drillApp`
Separador da árvore dos exercícios :
`hpanedTree`
Árvore de exercícios:
`exerciceTree`
Caixa de diálogo Gnome acerca de :
`about`

Pode ver os nomes destas widgets na Fig. 2

Listamos aqui, rapidamente os nomes das funções de processamento. Se precisar de mais informação acerca desta matéria, então pode ler a parte I da série deste artigo.

Nome da widget	Sinal	Processamento
acerca de	clicked	gtk_widget_destroy
acerca de	fechar	gtk_widget_destroy
acerca de	destruir	gtk_widget_destroy

botão1 (Ícone New na tool bar)	clicked	on_new_activate
novo	activar	on_new_activate
drillApp	destruir	on_exit_activate
sair	activar	on_exit_activate
acerca de	activar	on_about_activate

Ajustamentos Finais

A partir do Glade é possível especificar a geometria das widgets. No seu cado pode definir o tamanho da `drillApp` para 400 e 300 a partir do tab `Common` na painel de `properties`. Pode, também definir a posição do divisor horizontal para 100 em vez de 1.

Agora a widget `exerciceTree` precisa de ser ajustada para só permitir uma selecção num tempo. De facto, só pode ser seleccionada um exercício de cada vez. A partir do painel `properties`, seleccione `Selection->Single`. As outras opções para este widget são menos importantes.

Voilà! E está tudo, o que diz respeito do ***Drill***. Começaremos a desenvolver os exercícios no próximo artigo. Por agora, vejamos como podemos utilizar a interface a partir to Python e como manipular a widget `GtkTree`.

O código Python

O código completo pode ser encontrado no fim deste documento. Precisa de guardá–lo no mesmo directório que o ficheiro `drill.glade`.

Os módulos requeridos

```
from gtk import *
from gnome.ui import *
from GDK import *
from libglade import *
```

A interface gráfica com o LibGlade

A criação da interface gráfica e a ligação das funções de processamento com o LibGlade é feita do mesmo modo que no exemplo anterior. Não voltaremos atrás neste particular aspecto.

No programa Python definimos as variáveis globais:

- `currentExercice`: Ponteiro para a widget que representa o exercício corrente. Este último é colocado na parte direita da janela de aplicação do ***Drill***. Os exercícios serão criados a partir do

Glade.
- `exerciceTree` : Ponteiro para a árvore widget do lado direito da janela de aplicação do ***Drill***.
- `label` : Ponteiro para uma label (`GtkLabel`). Esta label é paliotiva do facto que não temos nenhum exercício no momento. Será colocada na parte esquerda da árvore −− onde os exercícios serão colocados −− e apresentará aqui os identificadores dos exercícios seleccionados.

A árvore é criada a partir do LibGlade, o seu ponteiro é obtido através da seguinte chamada:

```
exerciceTree = wTree.get_widget ("exerciceTree")
```

Precisamos também, do ponteiro dos painéis horizontais, de facto, o contentor de referência (`GtkPaned`) dos dois painéis horizontais separados por um separador. O da parte direita contém a árvore; o da parte esquerda contém os exercícios; por agora colocaremos a label aqui :

```
  paned = wTree.get_widget ("hpanedTree")
  label = GtkLabel ("No exercise selected")
  label.show ()
  paned.pack2 (label)
```

Novamente, a utilização de ambos, o **GTK+ Reference manual** −− nos objectos `GtkLabel` e no `GtkPaned` −− e o código fonte do Python `/usr/lib/python1.5/site-packages/gtk.py` fornecem−lhe a necessário compreensão para a utilização correcta dos objectos. .

A widget `GtkTree`

Esta é agora, a parte mais importante do artigo : como utilizar a árvore do tipo `GtkTree`.

A árvore é preenchida com chamadas consecutivas para as funções `addMathExercices()`, `addFrenchExercices()`, `addHistoryExercices()` e `addGeographyExercices()`. Estas funções são todas muito semelhantes. Cada uma destas funções adiciona uma nova sub−categoria (uma sub−árvore) bem como os títulos dos exercícios (itens) :

```
def addMathExercices ():
  subtree = addSubtree ("Mathematics")
  addExercice (subtree, "Exercise 1", "Math. Ex1")
  addExercice (subtree, "Exercise 2", "Math. Ex2")
```

A sub−árvore

```
def addSubtree (name):
  global exerciceTree
  subTree = GtkTree ()
  item = GtkTreeItem (name)
  exerciceTree.append (item)
  item.set_subtree (subTree)
  item.show ()
  item.connect ("select", selectSubtree)
  return subTree
```

Para criar uma sub−árvore numa árvore existente precisa de fazer duas coisas: Gerar uma árovre `GtkTree` e um item `GtkTreeItem`, com o nome da sub−árvore. De seguida o item é adicionado à àrvore raiz −− a árvore contendo todas as categorias −− e adicionamos a sub−árvore ao item utilizando o método `set_subtree()`. Finalmente o evento `select` é ligado ao item, assim, quando a categoria é seleccionada a função `selectSubtree()` é chamada.

GtkTreeItem

```
def addExercice (category, title, idValue):
  item = GtkTreeItem (title)
  item.set_data ("id", idValue)
  category.append (item)
  item.show ()
  item.connect ("select", selectTreeItem)
  item.connect ("deselect", deselectTreeItem)
```

Os itens têm os nomes dos exercícios como seu título, aqui somente `Exercice 1`, `Exercice 2`, ... A cada item associamos um atributo adicional `id`. O GTK+ tem a possibilidade de adicionar a qualquer objecto to tipo `GtkObject` −− todas as widgets GTK+ descendem dele −− alguns atributo. Para fazer isto, existem dois métodos, `set_data (key, value)` e `get_data (key)` para inicializar e obter o valor de um atributo. O item é então adicionado à sua categoria −− uma sub−árvore. O seu método `show()` é chamado pois é requerido para forçar a apresentação. Por último os eventos `select` e `deselect` são ligados. O evento `deselect` torna−se activo quando o item perde a sua selecção. Cronologicamente, o método `deselectTreeItem()` é chamado quando o item perde a sua selecção, de seguida é chamado o `selectTreeItem()` quando o item está a tomar a sua selecção.

As funções de processamento

Definimos três funções de processamento selectTreeItem(), `deselectTreeItem()` e `selectSubtree()`. Elas actualizam a label de texto −− do lado esquerdo −− com o valor do atributo `id`. E é tudo por agora.

Uma palavra final

Definimos somente a infra−estrutura na qual adicionaremos os exercícios −− bem como novas widgets descobertas. Discutimos principalmente a widget `GtkTree` e como associar atributos às widgets. Este mecanismo é muitas vezes utilizado para obter informação adicional respeitante às funções de processamento, o que fizemos aqui. Até ao próximo artigo pode tentar transformar o jogo ***Couleur***, que utilizámos na parte I, como um exercício no ***Drill***.

Apêndice: código fonte completo

```
#!/usr/bin/python
# Drill − Teo Serie
# Copyright Hilaire Fernandes 2001
# Release under the terms of the GPL licence
```

```
# You can get a copy of the license at http://www.gnu.org from gtk import *
from gnome.ui import *
from GDK import *
from libglade import * exerciceTree = currentExercice = label = None

def on_about_activate(obj):
    "display the about dialog"
    about = GladeXML ("drill.glade", "about").get_widget ("about")
    about.show ()

def on_new_activate (obj):
    global exerciceTree, currentExercice

def selectTreeItem (item):
    global label
    label.set_text ("L'exercice " +
            item.get_data ("id") + "est sélectionné.")

def deselectTreeItem (item):
    global label
    label.set_text ("L'exercice " +
            item.get_data ("id") + "est désélectionné.")

def selectSubtree (subtree):
    global label
    label.set_text ("No selected exercise")

def addSubtree (name):
    global exerciceTree
    subTree = GtkTree ()
    item = GtkTreeItem (name)
    exerciceTree.append (item)
    item.set_subtree (subTree)
    item.show ()
    item.connect ("select", selectSubtree)
    return subTree

def addExercice (category, title, id):
    item = GtkTreeItem (title)
    item.set_data ("id", id)
    category.append (item)
    item.show ()
    item.connect ("select", selectTreeItem)
    item.connect ("deselect", deselectTreeItem)

def addMathExercices ():
    subtree = addSubtree ("Mathématiques")
    addExercice (subtree, "Exercice 1", "Math. Ex1")
    addExercice (subtree, "Exercice 2", "Math. Ex2")

def addFrenchExercices ():
    subtree = addSubtree ("Français")
```

```
    addExercice (subtree, "Exercice 1", "Français Ex1")
    addExercice (subtree, "Exercice 2", "Français Ex2")

def addHistoryExercices ():
    subtree = addSubtree ("Histoire")
    addExercice (subtree, "Exercice 1", "Histoire Ex1")
    addExercice (subtree, "Exercice 2", "Histoire Ex2")

def addGeographyExercices ():
    subtree = addSubtree ("Géographie")
    addExercice (subtree, "Exercice 1", "Géographie Ex1")
    addExercice (subtree, "Exercice 2", "Géographie Ex2")

def initDrill ():
    global exerciceTree, label
    wTree = GladeXML ("drill.glade", "drillApp")
    dic = {"on_about_activate": on_about_activate,
        "on_exit_activate": mainquit,
        "on_new_activate": on_new_activate}
    wTree.signal_autoconnect (dic)
    exerciceTree = wTree.get_widget ("exerciceTree")
    # Temporary until we implement real exercice
    paned = wTree.get_widget ("hpanedTree")
    label = GtkLabel ("No selected exercise")
    label.show ()
    paned.pack2 (label)
    # Free the GladeXML tree
    wTree.destroy ()
    # Add the exercices
    addMathExercices ()
    addFrenchExercices ()
    addHistoryExercices ()
    addGeographyExercices ()

initDrill ()
mainloop ()
```

2005−01−10, generated by lfparser_pdf version 2.51