



Treinamento Eagle

Eagle — Desenvolvimento Ágil de Interfaces

Gustavo Sverzut Barbieri



2 de Junho de 2006



Tópicos

- 1 Introdução
- 2 Entendendo o Eagle
- 3 Mãos à Obra!
- 4 Referências e Materiais de Apoio
- 5 Agradecimento



O que é o Eagle

Introdução::O que é o Eagle

Camada de abstração em cima de Toolkits Gráficos

O que é o Eagle

Introdução::O que é o Eagle



Camada de **abstração** em cima de Toolkits Gráficos

- **Abstração:** manter o usuário longe das complexidades

O que é o Eagle

Introdução::O que é o Eagle



Camada de abstração em cima de **Toolkits Gráficos**

- **Toolkits Gráficos:** atualmente GTK e GTK/Maemo, QT a caminho

O que o Eagle não é

Introdução::O que é o Eagle



- Não é concorrente de GTK, QT ou MFC



O que o Eagle não é

Introdução::O que é o Eagle

- Não é concorrente de GTK, QT ou MFC
- Não é a solução para todos os problemas!

Objetivos principais

Introdução::O que é o Eagle



- Agilizar o desenvolvimento de GUI simples
- Expor componentes em altíssimo nível
- Manter a consistência
- Ajudar na “usabilidade”

Meios para atingir os objetivos

Introdução::O que é o Eagle



- Focar no casos mais usados
- Expor a interface para o programador mais simples possível
- Limites...

Meios para atingir os objetivos

Introdução::O que é o Eagle



- Focar no casos mais usados
- Expor a interface para o programador mais simples possível
- Limites...
- Limites!

Meios para atingir os objetivos

Introdução::O que é o Eagle



- Focar no casos mais usados
- Expor a interface para o programador mais simples possível
- Limites...
- Limites!
- Limites!



Procedural ou Orientado a Objetos?

Entendendo o Eagle::Organização

- Como o Python, é **feito** usando OO
- Porém existe API procedural

```
app = App( title="bla",  
           center=Label( id="label" ) )  
  
wid = get_widget_by_id( "label", app )  
set_inactive( wid )
```

Procedural

```
app = App( title="bla",  
           center=Label( id="label" ) )  
  
wid = app.get_widget_by_id( "label" )  
wid.set_inactive()
```

Orientado a Objetos



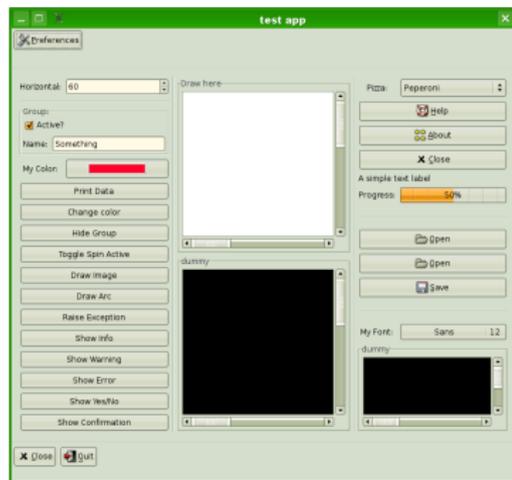
Sistema de Execução

Entendendo o Eagle::Organização

- Laço principal tratador de eventos
- Despacho baseado em chamadas de funções cadastradas (*"callbacks"*)
- Eventos do usuário: clique de botão
- Eventos do sistema: tempo expirado



- Trata cada janela como uma aplicação
- 6 áreas para componentes filhos:
 - Topo (organização horizontal)
 - Baixo (organização horizontal)
 - Esquerda
 - Centro
 - Direita
 - Preferências (em separado)
- Chama-se `App`
- Acesso aos filhos pelo nome:
`app["lab1"]` ou
`app.get_widget_by_id("lab1")`

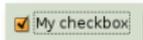


Componentes com Dados

Entendendo o Eagle::Componentes Gráficos



- Podem ser persistidos
- Têm métodos `set_value()` e `get_value()`
- Se acessados via `app["name"]`, já acessa o conteúdo
 - evita uso de “set” e “get”
 - acesso ao elemento ainda pode ser obtido com `app.get_widget_by_id()`
- Avisa quando dados foram modificados
- Componentes: `CheckBox`, `Label` e outros...





Componentes com Dados e Rótulo

Entendendo o Eagle::Componentes Gráficos

- Estendem os “Componentes de Dados”, adicionando um rótulo
- Agiliza o desenvolvimento
- Melhora a usabilidade
- Rótulo à esquerda em organizações verticais, em cima em organizações horizontais
- Componentes: [Entry](#), [Password](#), [Spin](#), [IntSpin](#), [UIntSpin](#), [Color](#), [Font](#), [Selection](#), [Progress](#)



Outros Componentes

Entendendo o Eagle::Componentes Gráficos



- **Agrupamento:** `Group`
- **Botão:** `Button` e especializações como `PreferencesButton`
- **Separadores:** `HSeparator` e `VSeparator`
- **Diálogos:** `information()`, `yesno()`, `warning()`, `error()`, `confirm()`



Superfície de Desenho

Entendendo o Eagle::Componentes Gráficos Avançados

- Funções básicas de desenho
- Funcionamento simplificado
- Facilidade para salvar imagens
- Avisa quando posição do mouse mudou
- Avisa quando botões do mouse foram pressionados
- Integração com tipo “imagem” do Eagle ([Image](#))
- Chama-se [Canvas](#)



Tabelas

Entendendo o Eagle::Componentes Gráficos Avançados



- Apresentação de dados em formato de tabela
- Compatível com Python-API para listas!
- Formatação simplificada por meio de função cadastrada
- Funcionalidade para edição
- Funcionalidade para remanejamento de itens
- Avisa quando dados foram modificados
- Chama-se `Table`

A screenshot of a window titled "Table" with a green header bar. The window contains a table with two columns: "Name" and "Phone". The table has two rows of data: "Gustavo" with phone number "12345" and "Bla" with phone number "3344-55". Below the table are three buttons: "Add" (with a plus icon), "Remove" (with a minus icon), and "Edit" (with a pencil icon).

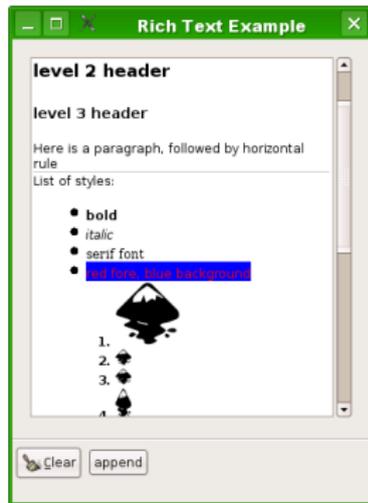
Name	Phone
Gustavo	12345
Bla	3344-55

Texto Formatado

Entendendo o Eagle::Componentes Gráficos Avançados



- Apresentação de texto formatado com subconjunto de HTML
- Avisa quando um “link” foi pressionado
- Possibilidade de inclusão de imagens do tipo `Image` em memória no Eagle
- Possibilidade de inclusão de imagens em qualquer outro protocolo, usando função cadastrada para obtê-las
- Chama-se `RichText`





Obtenha a cópia do SVN

Mãos à Obra!::Obtendo e Configurando

Usaremos a versão em desenvolvimento (SVN):

```
svn co http://code.gustavobarbieri.com.br/eagle/  
export PYTHONPATH=$PYTHONPATH:$PWD/eagle/gtk/
```



Aplicativo 1: Hello World!

Mãos à Obra!::Hello World!

```
#!/usr/bin/env python

from eagle import *

App( title="Hello World!",
      center=Button( id="btn",
                    label="Hello World!" ) )

run()
```



Aplicativo 1.1: Hello World! Revisto

Mãos à Obra!::Hello World!

```
#!/usr/bin/env python

from eagle import *

def my_func( app, button ):
    print "app=%s, button=%s" % ( app, button )

App( title="Hello World!",
      center=Button( id="btn",
                    label="Hello World!",
                    callback=my_func ) )

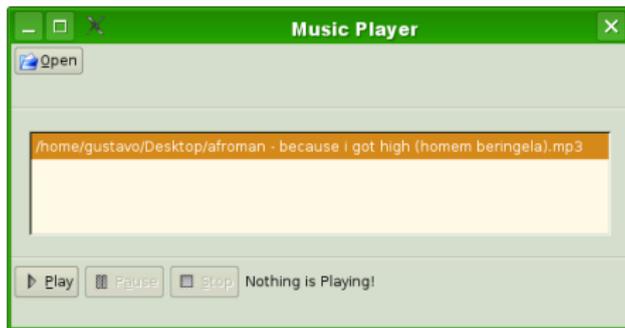
run()
```

Aplicativo 2: Music Player

Mãos à Obra!::Music Player



Obtenha o código de `player.py` e implemente uma interface gráfica para a classe `Player`.

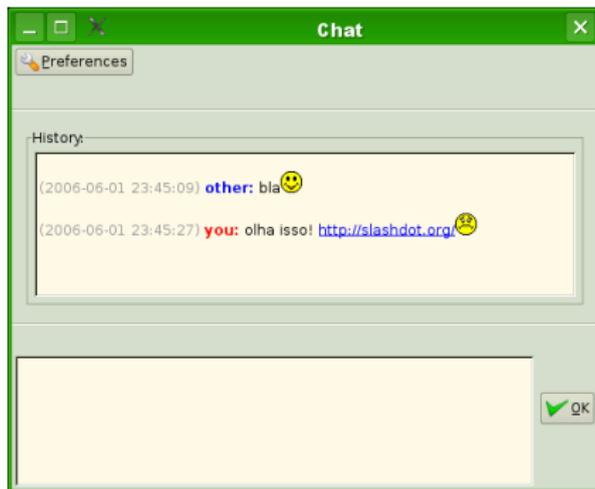


Aplicativo 3: Chat

Mãos à Obra!::Chat



Obtenha o código de `chat.py` e implemente uma interface gráfica para a classe `Chat`.



Referências

Referências e Materiais de Apoio::Referências



- Eagle: <http://www.gustavobarbieri.com.br/eagle/>
- Docs:
<http://www.gustavobarbieri.com.br/eagle/docs/>
- API:
<http://www.gustavobarbieri.com.br/eagle/docs/api/>
- Downloads:
<http://www.gustavobarbieri.com.br/eagle/packages/>



Agradecimento

Obrigado **INdT** por ter ajudado com a participação na
PyConBrasil 2006!

Contato

Agradecimento::Contato



Gustavo Sverzut Barbieri

Email: gustavo.barbieri@indt.org.br

Website: <http://www.gustavobarbieri.com.br>

ICQ: 17249123

MSN, Jabber: barbieri@gmail.com

Obtenha esta palestra em:

http://palestras.gustavobarbieri.com.br/treinamento_eagle/