Python for Series 60

Osvaldo Santana Neto
osantana@triveos.com

Original de Elvis Pfützenreuter

O que são celulares "Série 60"

- Smartphones com velocidade de um PC@1998
- Sistema operacional Symbian (consórcio mantido por diversos fabricantes de dispositivos móveis)
- Plataforma Série 60: Symbian + UI + r adicionais

O que são celulares "Série 60"

- Quase todos os celulares S60 são da Nokia
  - http://www.s60.com
  - http://www.symbian.com
- Concorrentes diretos da Série 60:
  - Baseados em Symbian: Série 80, Série 90, UIQ
  - Não baseados em Symbian: Série plus, maioria dos smartphones não Nokia

Desenvolvimento C++ para S60

- SDK completo (inclusive IDE e compilador) gratuito em http://forum.nokia.com
- O SDK inclui um emulador para desenvolvimento no PC, bem como *debug* no dispositivo (ODD)
- Alguns recursos mais avançados são pagos (UI designer, *debug* via JTAG etc.)
- API C++ do Symbian é completamente diferente de qualquer outra (POSIX, Win32, etc.)
  - A nova API OpenC tenta mitigar este problema

Por que desenvolver para S60

- Mais de 100 milhões de celulares vendidos
- Previsões apontam como plataforma dominante para os próximos 5-10 anos
- Várias opções de linguagem: J2ME, C++ e Python
- Plataforma aberta e SDK gratuito
- Existe uma comunidade de desenvolvedores
- Dispositivos são computacionalmente poderosos (evita ter de “escovar bit”)

Versões da Série 60

- 1st Edition – Symbian 6.1 (3600, N-Gage)
- 2nd Edition
  - Original – Symbian 7.0 (6600)
  - 2nd ed. Feature Pack 1 (7610)
  - 2nd ed. Feature Pack 2 – Symbian 8.0 (6681)
  - 2nd ed. Feature Pack 3 (N70, N90)
- 3rd Edition – quebra de compatibilidade binária
  - Original - Symbian 9.0 (3250, N93)
  - 3rd ed. Feature Pack 1 (N95)
- `http://wiki.forum.nokia.com/index.php/Aparelhos_S60`

Por que usar Python para S60

- Pelas virtudes naturais do Python :)
- Desenvolvimento C++ para dispositivos móveis é ainda mais árido que para desktop
  - Desenvolvimento no PC, uso real no celular; computadores completamente diferentes!
  - Emulador Symbian no PC diverge do celular
  - Recursos avançados não são gratuitos
  - Comunidade menos amigável do que deveria
  - Pouco software livre e/ou aberto para consulta
  - Muitas diferenças entre versões do S60, mesmo entre *minor releases* (“*feature packs*”)

Por que usar PyS60 (cont.)

- API mais estável, pois as diferenças entre versões são resolvidas no pacote Python
- Boa parte das bibliotecas padrão do Python está disponível no PyS60
- Permite adiar a depuração no dispositivo, de várias formas
- A depuração no dispositivo é mais fácil
- Torna muito mais fácil o acesso a recursos como câmera ou SMS, em relação a C++
- Comunidade muito mais atuante
- `http://pys60.sourceforge.net`

Desvantagens do PyS60

- Não faz coleta de lixo de referências circulares por motivos de performance
- Desempenho pode ser um problema
  - mas pode-se escrever bibliotecas C para as partes de desempenho criticamente importante
- Nem toda a API C++ está coberta
  - mas pode-se escrever módulos adicionais

PyS60 versus 3rd Edition

- O Symbian 9 (S60 3rd Edition) implementa um modelo muito mais rígido de segurança
  - Aplicativos assinados com certificado digital
    Aplicativos têm capabilities e escopo de distribuição embutidos na assinatura
- `http://forum.nokia.com/main/platforms/s60/capability_descriptions.html`
- Certificado do desenvolvedor assinado pela Symbian Ltd. custa dinheiro $$$
  - Certificado auto-assinado impede o uso de algumas capabilities mais avançadas
- Aplicativos para distribuição em massa têm de ser homologados pela Symbian Ltd. ($$$$$$$$)

PyS60 x 3rd Edition (cont.)

- O pacote PyS60 1.4.4 é assinado pela Nokia
  - `All -TCB -AllFiles -DRM`
  - Os scripts “herdam” estas *capabilities* e não precisam ser eles mesmos assinados
- Instalar scripts via Bluetooth ou SMS não funciona na 3rd Edition; é preciso usar outro meio
  - Celulares 3rd Edition dão acesso via USB ao cartão de memória do telefone como se fosse um pen drive
  - Um truque: ZIPar script e mandar via Bluetooth, pois o celular tem como descompactar

Comprando um celular Série 60 para fins de desenvolvimento

- Modelos que já utilizei com PyS60:
  N-Gage QD, 6600, 6681, N93 e N95
- Se puder investir, compre um celular 3rd Edition
  - PyS60 funciona bem
  - Hardware muito mais rápido
  - Todos os novos celulares são 3rd Edition
- Um fone 2nd Edition também é interessante
  - Base de usuários 2nd Edition
  - É infinitamente melhor ter um celular lento do que nenhum celular para testar

Por onde começamos?

- Em primeiro lugar, provando que funciona!
- Instalar o pacote PyS60 no celular
- Executar um comando qualquer
- Descobrir que o teclado do celular é uma péssima interface
- Acessar a linha de comando Python via Bluetooth

Módulos específicos para S60

Módulo e32

- Serviços do Symbian, mais notadamente os Active Objects (AO)
  - objetos de sincronização que permitem multiplexar várias atividades “simultâneas”
  - Tem o papel que o `select()`/`poll()` tem no UNIX
  - Apenas a UI *thread* pode manipular UI, os Active Objects permitem “disparar” funções UI a partir de outras *threads*
  - `ao = e32.ao_callgate(função_da_ui)`
  - aí outra *thread* chama... `ao.signal()`
  - A UI do Symbian não tem nada equivalente a `gtk.gtk_main()`

Módulo e32 (continuação)

```
import appuifw

   trava = e32.Ao_lock()

def callback_sair():
   global trava
   trava.signal() # acorda

def main():
   global trava
   appuifw.app.exit_key_handler = callback_sair
   trava.wait() # dorme
```

Módulo sysinfo

- Informações diversas sobre o dispositivo:
  - `sysinfo.battery()`
  - `sysinfo.imei()`
  - `sysinfo.signal_dbm()`

Layout de uma aplicação S60

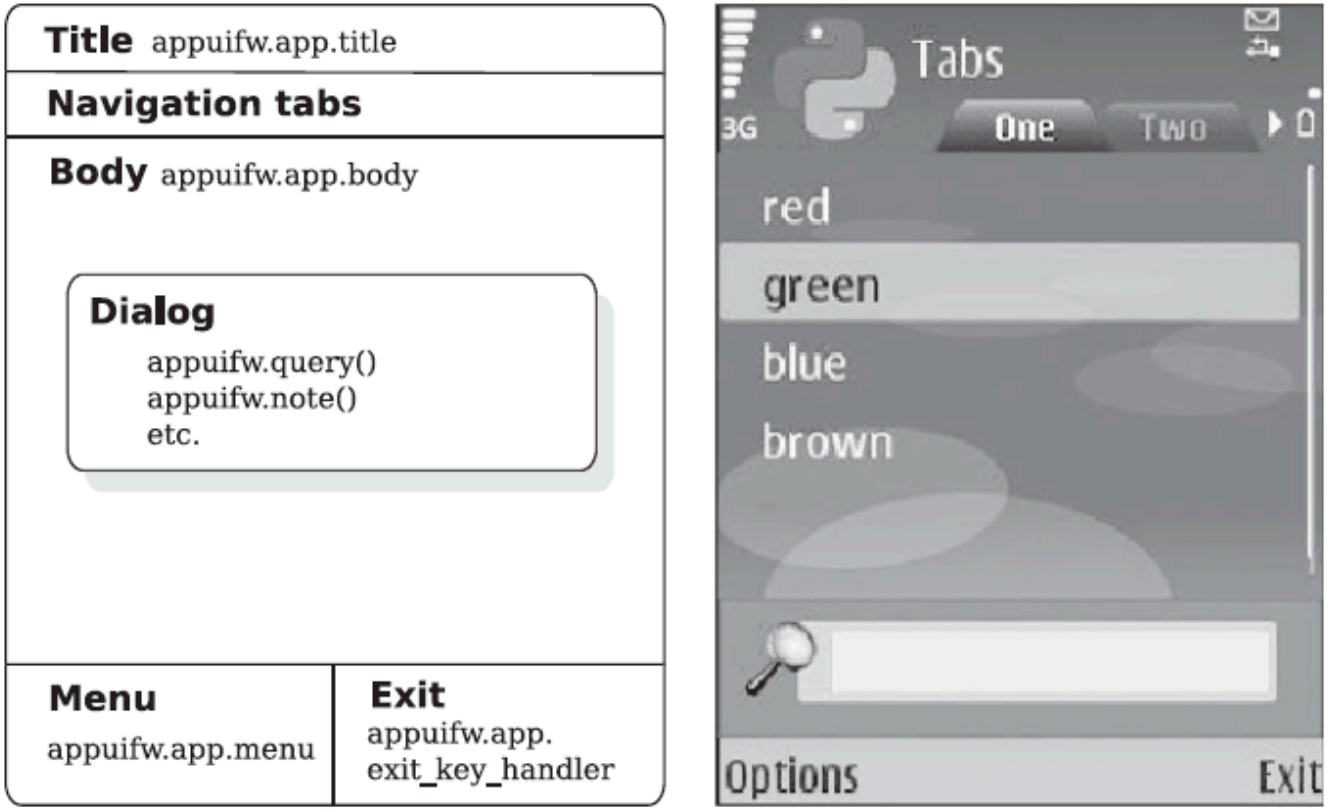

Figure 4.1 A typical user interface (a) structure and (b) screenshot based on the structure

Módulo appuifw

- Módulo da UI, o mais complexo dos específicos para Série 60
  - Quem já usou PyGTK+, vai achar familiar
  - Application (Reside em `appuifw.app`)
  - `appuifw.app.body`: Canvas, ListBox ou Text
  - `appuifw.app.menu`
  - *Dialogs* modais
    - Form (*widget* “composto”)
  - Binding de eventos e teclas a ações
- Exemplo de UI básica

Módulo graphics

- Módulo auxiliar de `appuifw`, lida com imagens
- Classe `Image`
- Métodos de desenho e texto
- Gravação e leitura de imagens: precisamos primeiro de uma fonte de imagens...

Módulo camera

- Permite manipular a(s) câmera(s)
- `camera.cameras_available()`
- `i = camera.take_photo() # Image`
- `camera.start_finder(função)`
- `...`
- `camera.stop_finder()`
- Não funciona no emulador padrão (mas há extensões para suprir esta função)
- Gravação e leitura de imagens

Módulos gles/glcanvas

- Módulo de acesso ao OpenGL/ES
- OpenGL/ES é **acelerado** por hardware nos celulares Nokia N93 e N95
- Exemplo: `gles.py` no pacote PyS60

Módulo audio

- Lida com recursos de áudio
- Funciona no emulador, inclusive grava som
- Classe `audio.Sound`
  - `s = Sound.open(u"c:\\arquivo.wav")`
  - `s.record()`
    - Grava inclusive ligações
  - `s.stop()`
  - `s.play()`

Módulo telephone

- `telephone.dial("5551234567")`
- `telephone.hang_up()`

Módulo messaging

- `messaging.sms_send(”99845555”, u”Oi”)`
- `messaging.mms_send(...)`
- Permite passar um *callback* para receber o *status* do envio

Módulos inbox e location

- Inbox: Manipulação da caixa de entrada
- `location.gsm_location()`

Módulo position

- API de localização
- Depende de dispositivo GPS, apenas o Nokia N95 tem GPS/AGPS por ora.
  - AGPS: GPS + dados das torres para maior precisão e rapidez

Módulos contacts/calendar

- Informações de contatos e calendário
- Módulos razoavelmente complexos
- Sugestão: explorar via terminal do Python

Módulos e32db/e32dbm

- `e32db`: mini-banco de dados do Symbian, com interface baseada em comandos SQL
  - Para persistência de configurações e pequenas quantidades de dados
  - Mesma idéia do SQLite
  - Será mesmo o SQLite na 3rd Ed. FP2
- `e32dbm`: Interface DBM para `e32db`
  - A API DBM é mais conhecida dos Pythonistas
  - `anydbm` é apelido de `e32dbm` em PyS60

Ciclo de desenvolvimento

- Editar no PC, testar somente no celular
  - auto-suficiente porém moroso
- Testar (partes do) aplicativo no Python PC
  - muitas partes exigirão teste moroso no celular
- Testar no emulador Symbian
  - algumas partes ainda exigirão teste no celular
  - A priori, somente Windows (GNUPOC?)
  - Emulador é lento e diferente do celular
- Será necessário usar uma combinação das três técnicas para minimizar o tempo de desenvolvimento

Testando scripts PyS60 no PC

- Desacople os módulos dependentes do celular – tire proveito da orientação a objetos!
- Crie módulos ”emuladores” para o PC
- Use camadas de abstração que já foram criadas por terceiros:
  - PyS60-compat (UI e gráficos)
  - Lightblue (Bluetooth)
- Muita atenção com o Unicode
  - Saiba quais APIs demandam ou retornam *strings* Unicode

PyS60 no emulador Symbian

- É necessário instalar Java, Perl, Carbide e o SDK de uma versão Séries 60 do nosso interesse (no nosso caso, 3rd edition)
  - O processo de instalação avisa se falta algum software auxiliar como o Perl
- Obter e instalar o pacote PyS60 SDK para a versão escolhida (pois o emulador não simula o processador ARM)
- `EPOC32=C:\Symbian\9.2\S60_3rd_FP1\Epoc32`
- `$EPOC32\winscw\c\python` (scripts)
- `$EPOC32\winscw\c\resource` (módulos)
- Cygwin como linha de comando estilo UNIX/Linux

Criando um pacote SIS

- Pacote SIS é um formato aberto de distribuição de aplicativos Symbian
- O PyS60 é distribuído como SIS, nossos scripts também podem sê-lo
- 2nd Edition: usar py2sis (fácil)
- Na 3rd Edition, pacote SIS é assinado digitalmente; é preciso usar o Ensymble
  - http://www.nbl.fi/~nbl928/ensymble.html

Criando um pacote SIS (cont.)

```
ensymble py2sis
    -n LucaMobile
    -r 1.0.1
    -l EN
    -s "Luca"
    -c "Luca Mobile"
    --caps=NetworkServices+
          LocalServices+
          ReadUserData+WriteUserData+
          UserEnvironment
    LucaMobile/
    LucaMobile
```

Criando um pacote SIS (cont.)

- Ensymble usa o OpenSSL para assinatura
  - no Windows é preciso instalar um binário do OpenSSL (usar o OpenSSL fornecido pelo projeto Stunnel)
- O pacote SIS do script pode conter também o pacote do Python (facilita distribuição de aplicativos)
- As *capabilities* Symbian do pacote precisam ser suficientes para as necessidades do aplicativo!
  - *Capabilities* avançadas dependem de certificado do desenvolvedor assinado pela Symbian ($$$)
  - Mas scripts “soltos” herdam as capabilites do PyS60

Desenvolvendo PyS60 no Linux ou no Mac OS X

- É perfeitamente possível
- Emulador: a princípio é Windows-only, mas o GNUPOC pode funcionar
- Compilador C++: GNUPOC

Criando extensões C++ PyS60

- 2nd Edition: diversos exemplos na Internet
- 3rd Edition: exemplo modificado para este curso
- Python 1.4.0 assinado pela Nokia criou problemas para escrever extensões:
  - No emulador, funciona
  - No dispositivo, DLL tem de ser assinada com as mesmas capabilities do Python
  - ... e isso exige ter um certificado de desenvolvedor $$$
  - ... e exigiria certificar a aplicação para distribuição em massa $$$ $$$$$
- Solução temporária: usar versão mais antiga do PyS60

Links sobre PyS60

- http://www.mobilenin.com/pys60/menu.htm
- http://sourceforge.net/projects/pys60-compat/
- http://pdis.hiit.fi/pdis/download/miso/