EPROM
ENTREPRENEURIAL PROGRAMMING AND RESEARCH ON MOBILES

http://mit.edu/eprom

Mobile Application Prototyping with Python

A 3-Day Crash Course for the University of Nairobi

DAY 2

Nathan Eagle, PhD
Research Scientist
MIT Design Laboratory
Massachusetts Institute of Technology

November 16-18 2006
SCI, University of Nairobi

- Tutorial:
  - http://www.mobilenin.com/pys60/menu.htm
- Path:
  - C:\Symbian\8.1a\S60_2nd_FP3\Epoc32\release\winscw\udeb\z\system\libs
- DOWNLOAD:
  - http://www.mobilenin.com/pys60/ex_sms_sending.htm

Day 2 : Application Development

- Creating Our Own Modules
- Sending Text Messages / Making Phone Calls
- Building Applications
  - Title, Screen Size, Tabs, Threads, Body, Menus …
- GUI Design
  - Customizing Your Own Graphical User Interfaces
- Keyboard Keys
- Graphics and Drawing
- XML
- Contacts and Calendar Databases

Get More User Input

Example from: Larry Rudolph's Intro to Python slides

```
import appuifw
planets = [ u'Mars', u'Earth', u'Venus' ]
prompt = u'Enter your home planet'
index = appuifw.menu(planets, prompt)
appuifw.note(u'Hello '+planets[index] , u'info')
```

- The 'menu' method pops up the list of items in first param
- It returns with an index into the list of menu items
- Note that the prompt param must also be a unicode string

Our Own Interface

Example from: Larry Rudolph's Intro to Python slides

NAIROBI.py

- There are a bunch of annoyances in the current UI
- Let's put wrappers around basic calls
- We should go back and do this for location

Hiding the Unicode

Example from: Larry Rudolph's Intro to Python slides

```
# this is file nairobi.py
# wrappers to appuifw

def note( str , type = 'info'):
        appuifw.note( unicode(str),  type )

def query( str , type = 'text' ):
        return appuifw.query( unicode(str),  type )

def menu( list, prompt = 'select one' ):
   ulist = [ unicode(u) : for u in list ]
        return appuifw.menu( ulist , unicode( prompt ) )
```

Using nairobi.py

```
import nairobi.py

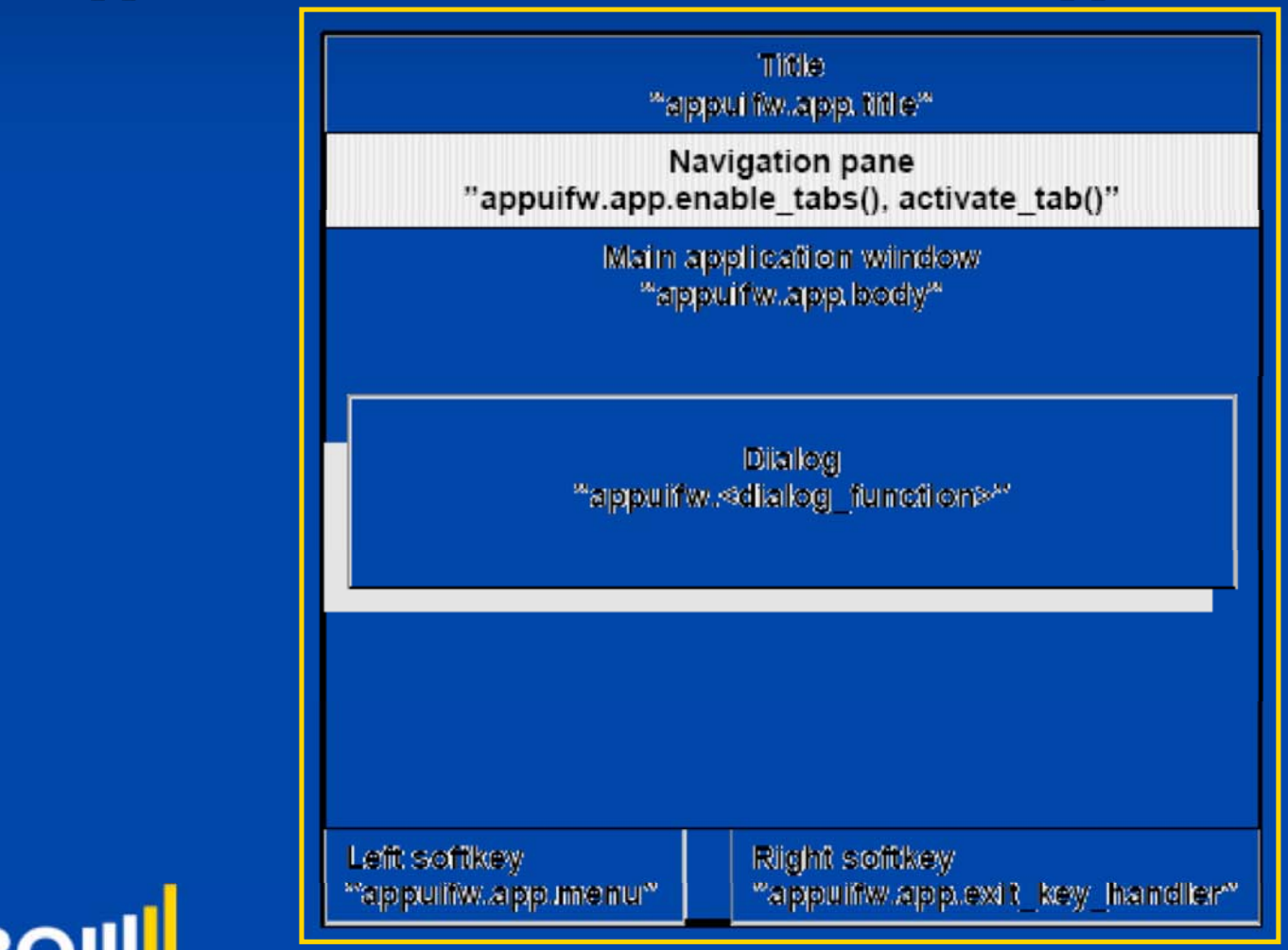
planets = [ 'Mars', 'Earth', 'Venus' ]
prompt = 'Enter your home planet'
index = nairobi.menu(planets, prompt)
nairobi.note('Hello '+planets[index] )
```

Sending Text Messages / Making Phone Calls

- Text / SMS Module: messaging
  - messaging.sms_send(number, txt)
- Telephone module: telephone
  - telephone.dial(number)
  - hang_up()
- Play with ex_sms_sendind_descr.py
  - http://www.mobilenin.com/pys60/ex_sms_sending.htm

appuifw.app

- Appuifw contains an instance of the class application, called app



The 9-Steps to Application Development

- 1. import all modules needed
- 2. set the screen size (normal, large, full)
- 3. create your application logic ...
- 4. create an application menu (if necessary)
- 5. set an exit key handler
- 6. set the application title
- 7. deal with active objects if necessary
- 8. set the application body (text or canvas or listbox or none)
- 9. create a main loop (e.g. while loop) if suitable

1. Importing Modules

```
import appuifw
import e32
```

2. Setting Screen Size

```
# screen has 3 different values:

#(a normal screen with title pane and softkeys
appuifw.app.screen='normal'

#(only softkeys visible)
appuifw.app.screen='large'

#(a full screen)
appuifw.app.screen='full'
```

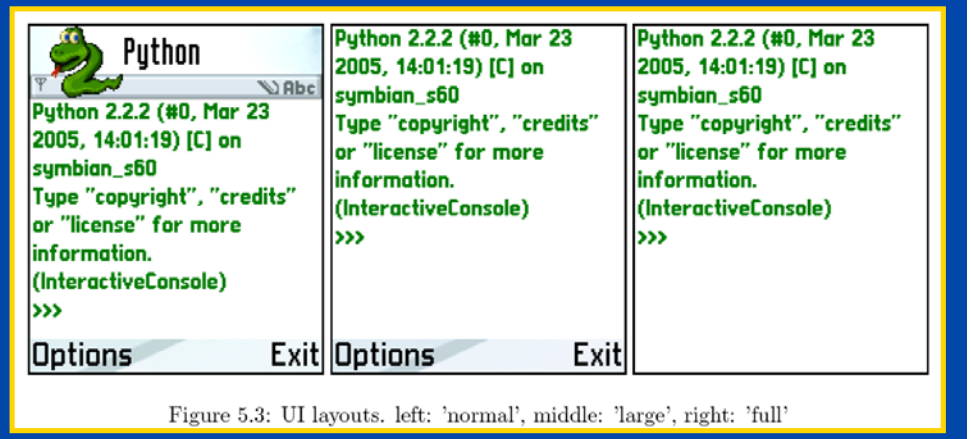


Figure 5.3: UI layouts. left: 'normal', middle: 'large', right: 'full'

Example script: app_screen.py

3. Application Logic

- This is where the heart of your application lies.

4. Application Menus

- An application menu uses the left softkey and can always be accessed while your application is running. An application menu can contain also a submenu

```
# create the callback functions that shall be executed
   when selecting an item in
   # the menu:

   def item1():
       print "item one"
def subitem1():
       print "subitem one"
def subitem2():
       print "subitem two"
# create the menu using appuifw.app.menu[(title,
   callback1), (title, (subtitle, callback2))]
appuifw.app.menu = [(u"item 1", item1), (u"Submenu 1",
   ((u"sub item 1", subitem1), (u"sub item 2", subitem2)))]
```

- Example script: app_menu.py

5. The Exit Key Handler

- The exitkey handler gets activated when you press the right (exit) softkey. By assigning an extra function to the .exit_key_handler you can define what shall happen when it is pressed.

```
def quit():
      appuifw.app.set_exit()
app.exit_key_handler=quit
```

6. The Application Title

```
appuifw.app.title = u"SMS sending"
```

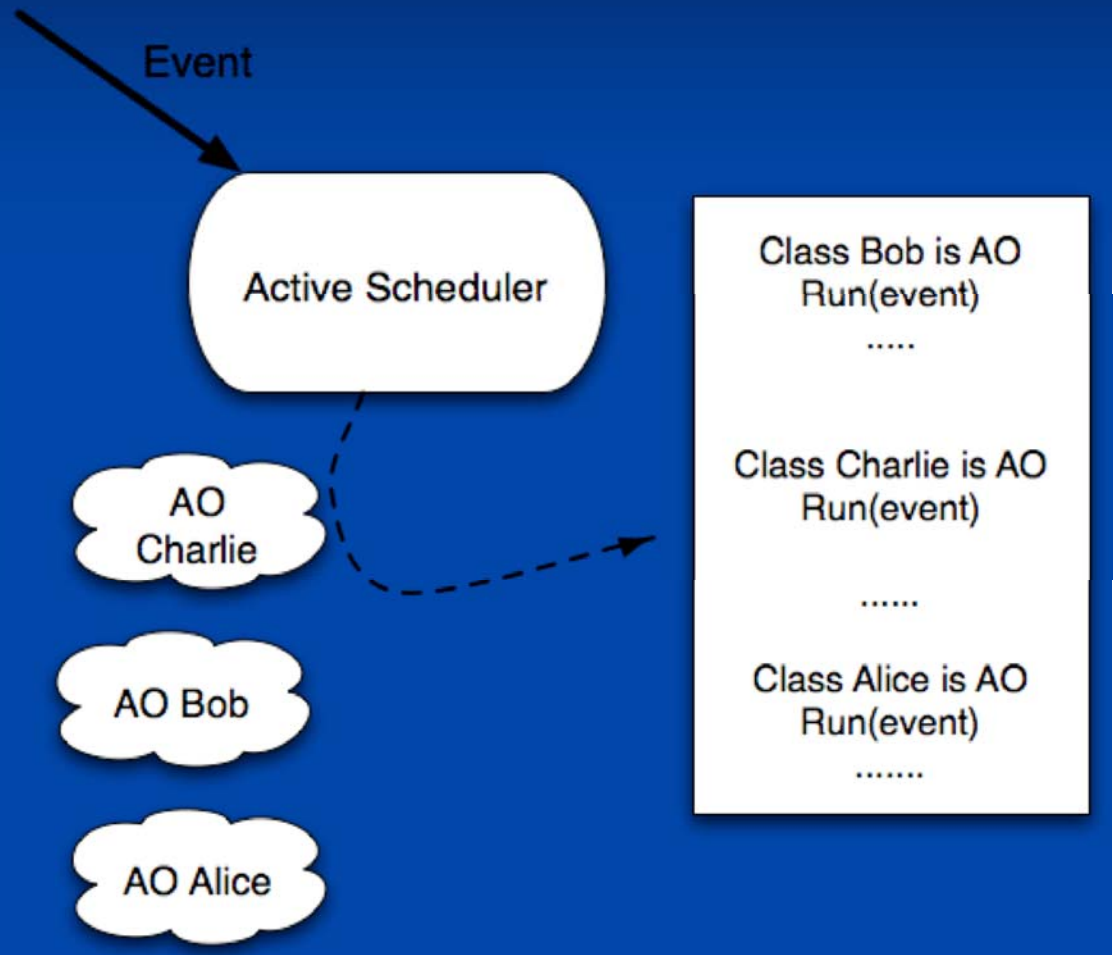
7.0 UI Threads

- places objects on screen
- registers callbacks procedures associated with screen & keyboard events
- when event occurs, want to pass control to the callback procedure.
- what if thread is executing something else?
- Callbacks should execute quickly
- UI thread should spend most of the time idle

7.1 e32 module: Coordination

Graphic from: Larry Rudolph's Intro to Python slides

- Don't use normal thread locks:
  - import thread
  - lock = thread.allocate_lock()
- Whole application gets blocked, since no UI actions would be handled
- Use e32.Ao_lock instead



7.2 Active Objects

- If Symbian written today, AO's would be called "listeners"
- Get called by a thread scheduler (have a little bit of state)
- Run to completion then return to scheduler
- They preserve the responsiveness of the UI and sockets

```
# You need to import the e32 module
import e32
# create an instance of the active object
app_lock = e32.Ao_lock()
# starts a scheduler -> the script processes
#events (e.g. from the UI) until lock.signal() is
# callled.
app_lock.wait()
# stops the scheduler
app_lock.signal()
```

- For more detail see the python_api.pdf.

8. Application Body

- text or canvas or listbox or none

- ```
  # body as Listbox:
  appuifw.app.body =
  appuifw.Listbox(entries,shout)
  # Example script: app_body_listbox.py
  ```

- ```
  # body as Text:
  appuifw.app.body = appuifw.Text(u'hello')
  # Example script: app_body_text.py
  ```

- ```
  # body as Canvas:
  appuifw.app.body=appuifw.Canvas(event_callbac
  k=None, redraw_callback=handle_redraw)
  # Example script: app_body_canvas.py
  ```
  ```

Get the code

- http://reality.media.mit.edu/code/nairobi_code.zip

9. The Main Loop

- # put in the main loop the things that need to be run through again and again in your script

```
running = 1
while running:
        # #e.g. redraw the screen:
        handle_redraw(())
```

Application Skeletons

- 1. no main loop because the application logic works without
  - Example script: app_skeleton.py
- 2. with mainloop (if suitable)
  - Example script: app_skeleton_with_mainloop.py

The Keyboard

- import keys
- **from key_codes import ***
- **Example:** ex_use_of_keys_descr.py

1. EKeyLeftSoftkey EScancodeLeftSoftkey
2. EKeyYes EScancodeYes
3. EKeyMenu EScancodeMenu
4. EKey1...9,0 EScancode1...9,0
5. EKeyStar EScancodeStar
6. EKeyLeftArrow EScancodeLeftArrow
7. EKeyUpArrow EScancodeUpArrow
8. EKeySelect EScancodeSelect

9. EKeyRightArrow EScancodeRightArrow
10. EKeyDownArrow EScancodeDownArrow
11. EKeyRightSoftkey EScancodeRightSoftkey
12. EKeyNo EScancodeNo
13. EKeyBackspace EScancodeBackspace
14. EKeyEdit EScancodeEdit
15. EKeyHash EScancodeHash

Keyboard Exercises

- Program an application that uses keyboard to trigger pop-up notes telling you what key has been pressed.

```
while running:
    if keyboard.pressed(EScancodeLeftArrow):
        appuifw.note(u"Arrow left", "info")
```

- Create "Easter Egg" Codes in Your Application that trigger events if the user presses the right code…
- Extra Credit: Modify an application to accept keyboard commands to trigger functions rather than menus.
- LOOK AT extended_use_of_keys.py!
- ex_graphics_drawing_descr.py