

Scripting Environment for Pervasive Application Exploration on Mobile Phones

Jukka Laurila Jukka.P.Laurila@nokia.com
Ville Tuulos Ville.Tuulos@helsinki.fi
Ronan MacLavery Ronan.MacLavery@nokia.com

Software and Application Technologies Laboratory, Nokia Research Center

Abstract. Research into pervasive computing has many difficult challenges: user application design, interaction models, and system architectures. Furthermore, exploratory research is limited by the fact that developing for pervasive computing devices is complex, due to the embedded system nature of the devices. Here we describe a scripting platform, Python for S60, for rapid development of applications on smartphones. It provides access to a large set of functionality needed to explore pervasive applications, including local and remote communications, UI, graphics, camera, sound and device databases. The system is easily extensible with native C++ code to improve performance and provide access to additional APIs. This technology is currently being used inside Nokia to develop and explore experimental pervasive applications.

1 Introduction

It can be claimed that the mobile phone is the first practically successful pervasive computer, with almost two billion users worldwide. A modern smartphone has a number of unique features that make it a highly desirable platform for experimenting with pervasive computing applications. It is a rugged, easily portable device with built-in Internet access that is generally carried continuously by users. As a consequence, it has access to a wide variety of contextual and personal information by default:

- location and location history (from cell ID or GPS data)
- names of people the user knows (from the phonebook)
- appointments the user has (from the local calendar)
- pictures and sound clips the user has recorded
- ambient light and noise level
- devices that are nearby (from Bluetooth, Zigbee or WLAN scanning)

Mobile phones have traditionally been fairly difficult to program for, since instead of generic computers they have been thought of as embedded systems where optimizing memory footprint and battery life are critical goals. Combined with a desire to still have rich functionality, these drivers have led to operating systems that are quite different from traditional desktop environments.

*Tom Pfeifer et al. (Eds.): Advances in Pervasive Computing 2006
Adjunct Proceedings of Pervasive 2006, Dublin, 7-10 May
books@OCG.at Vol. 207, ISBN 3-85403-207-2*

This is now changing with improvements in processing power and RAM capacity. For example, a typical modern smartphone, Nokia N70, has a 220 MHz ARM processor and 34 MB of free RAM for applications. This development has enabled trading off some efficiency for programmer convenience, bringing mobile programming closer to the mainstream.

Scripting languages have been found to improve programmer productivity significantly[2], and now they are migrating into the mobile domain[5]. They are ideally suited for exploratory programming and research prototyping. Transparent compilation combined with a possibility for interactive development enables experimenting with many ideas very rapidly.

2 Python for S60

Python for S60 is a port of the popular Python language[7] to S60[8], a Symbian OS -based smartphone platform used by several vendors such as Nokia, Samsung and Panasonic. S60 1st and 2nd Edition devices are currently supported, and a port for S60 3rd Edition is being prepared. The source code of Python for S60 is available[6] under the Apache 2 and Python 2.2.2 licenses.

The recommended strategy for implementation of research prototypes in Python for S60 is to write the first version entirely in Python. If high computational performance is required for some parts, you can write a native extension module that performs these calculations. It is also possible to embed the interpreter in your own C++ application if e.g. you have an existing C++ application you would like to add scripting capabilities to.

2.1 Features

The interpreter core is a port of Python 2.2.2 into a Symbian DLL. The Python for S60 platform provides many of the modules in standard Python, and an extensive set of modules that interface to device APIs. Where possible, the APIs have been made similar to APIs on mainstream operating systems. Highlights include:

- File access: Unrestricted access to all files in devices running S60 1st and 2nd Edition, limited access in 3rd Edition.
- Standard socket API, including TCP and UDP over whatever transport is available (GPRS, EDGE, UMTS, WLAN), SSL, Bluetooth serial port (RFCOMM), object sending and receiving (OBEX) and discovery support.
- Most common Internet protocols such as HTTP, FTP and IMAP are available in the Python standard library as pure-Python modules.
- “appuifw”, a simple API to the S60 user interface toolkit
- Simple 2D graphics support, including a set of drawing primitives, image loading, saving, rotation and scaling, and text rendering.
- Access to camera, microphone and speaker
- Making telephone calls

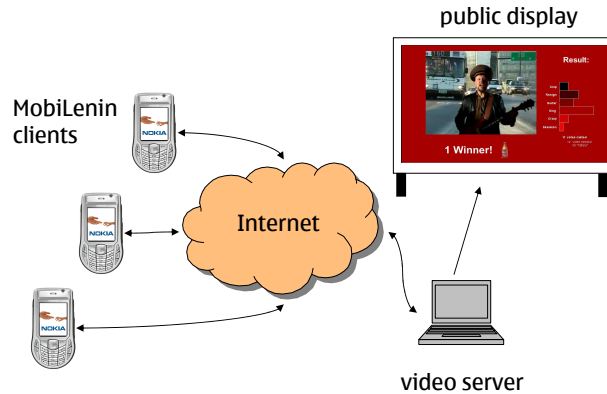


Fig. 1. MobiLenin system architecture

- Cell ID for obtaining coarse-grained location information
- Sending and handling received SMS messages
- Access to the contacts and calendar databases

3 Characteristic application examples

3.1 MobiLenin

The MobiLenin system[3] allows a group of people to interact with a music video running on a large public display using mobile phones running a voting client written in Python for S60 (Figure 1). There are several alternative styles of the same video available, and the audience can vote which one they prefer. The system periodically performs a lottery where some of the users can win a prize. The client communicates with the server in real time using a TCP/IP socket connection, queries the user for the voting response when instructed by the server, and displays a picture token of the prize for the users who win the lottery.

This is a good example of the kind of application that is especially easy to implement using Python for S60. All the necessary libraries are included in the standard distribution and the client does no heavy computation, so no custom C++ code was required. Python proved to be easy to learn and it made possible to implement the client in a much shorter timeframe and with significantly less lines of code than what it would have taken with C++ or Java code.[4]

3.2 Bluetooth scanning

A common function in many pervasive applications is the scanning of nearby Bluetooth devices (see e.g. [1]). This information may be used to reveal the user's location or her social context, for instance.

```
import bt, socket, e32

def normalize_addr(a):
    return ":".join(map(lambda x: "%.2x" % int(x, 16), a.split(":")))
def bluetooth_scan():
    while True:
        e32.ao_sleep(20)
        results = map(normalize_addr, bt.btinq())
        if not results: continue
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect(('server.example.com', 31337))
        print "Devices nearby:"
        for mac in results:
            print mac
            s.send(mac)
        s.close()
```

Fig. 2. Bluetooth scanning in Python for S60

Figure 2 is an example of a Bluetooth scanning script. A low-level Bluetooth scan is initiated scan every 20 seconds by the function `bluetooth_scan()` by calling the C++ extension module function `bt.btinq()`. The returned MAC addresses are cleaned up using the function `normalize_addr()`. If the scanning produced results, a standard TCP/IP socket is connected to the backend data collection server, and results are printed to the screen and sent to the socket.

This code was actually used in a small-scale empirical study with some 20 phones in Autumn 2005 and Spring 2006. This researched the various social patterns and time-location relationships which simple Bluetooth scanning might reveal. This information was crucial in planning some larger-scale experiments. In this case, Python allowed rapid prototyping and made it possible to base decisions on real data.

4 Future work

The near term target is to port the interpreter to the next generation of S60 smartphones. This will allow it to play a role in several projects inside Nokia Research Center, e.g. the SensorPlanet project that will create a research platform for experimenting with sensor networks[9]. The technology will also be made available to external parties through the release of source code for PyS60. This holds exciting prospects for the future since it allows the global open source community to collaborate in and adopt its development. This will also permit pervasive computing researchers to modify and extend it for their own purposes.

References

1. N. Eagle: Machine Perception and Learning of Complex Social Systems, Ph.D. Thesis, Massachusetts Institute of Technology 2005.
2. Prechelt, L.: An empirical comparison of seven programming languages, IEEE Computer 33(10):23-29, October 2000
3. Scheible, J., Ojala, T.: MobiLenin: Combining A Multi-Track Music Video, Personal Mobile Phones and A Public Display into Multi-User Interactive Entertainment, Proc. ACM Multimedia 2005, Singapore, 199-208.
4. Scheible, J., personal communication
5. Wood, D.: Smartphone programming goes mainstream, available online at <http://www.symbian.com/technology/insight4.html>
6. <http://opensource.nokia.com/>
7. <http://python.org/>
8. <http://s60.com/>
9. <http://www.sensorplanet.org/>

