Archived:Mobile Design Pattern - multi screens application approach

Aquivado: Este artigo foi arquivado, pois o conteúdo não é mais considerado relevante para se criar soluções comerciais atuais. Se você achar que este artigo ainda é importante, inclua o template {{ForArchiveReview|*escreva a sua justificativa*}}.

All PySymbian articles have been archived. PySymbian is no longer maintained by Nokia and is not guaranteed to work on more recent Symbian devices. It is not possible to submit apps to Nokia Store.

First case: one main screen plus one configuration screen

Some kind of applications may need a 'complex' configuration and just a small amount of interaction from the user for the normal use. One approach could be to split the application in 2 different screens:

1. Main screen: as clean as possible and with the minimum amount of controls

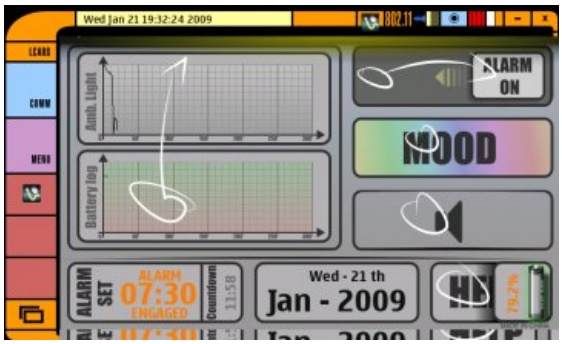
the main screen's task must be clear and focused on the goal (show the time and be pretty) while the configuration screen can be as complex as the application need.

2. Configuration screen: with all the controls

Dividing in 2 the application the 'interactions style' could be a bit more free from the 'consistency rule' (let the user use the fingers for the main screen and give some stylus licences on the config screen).

Overlayed help

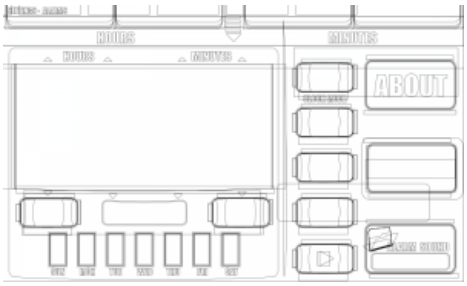
If the application need more gestures is possible to give hints to the user with an overlay graphics:



(C) Copyright Nokia 2014. All rights reserved.

Wire structure

Since python facilitates the port between different platforms is recommended to create the graphic elements using a vector base program [1] (Adobe Illustrator, Inkscape etc.)



Second case: Many different screen with equal importance

Different case when the application need more interaction from the user like, for example, in an audio player. It is important to always leave a clue of where the application is and to keep always avaible the main controls

Language and librarie used for the examples

Python is the ideal choice for rapid prototyping and have 'binds' with almost every graphics library on any possible device. Since is an interpreted language[2] (more specific compiled at runtime) it works without any modify to the source on every device. Since the interface is the application part that really 'talk' with the user is important to have a library for the graphics that can react and be responsive as much as possible like a videogame. That's the reason behind the pygame choice but any library can do the trick (Qt, GTK, CAIRO, etc)

- Official Python page
- Python on Symbian
- Python on Maemo: pymaemo.garage.maemo.org
- Official Pygame site
- **S60**: Pygame
- **Maemo** : Maemo Pygame page (direct link to the installation .deb)

External links

- Project page with sources of the example application
- Clip (version 0.5)

(C) Copyright Nokia 2014. All rights reserved.

 (C) Copyright Nokia 2014. All rights reserved.