# UBIK

(Brocom Project: Terminal Software Architecture)

**Helsinki University of Technology**

Thursday, 20 March 2003


| | |
|---|---|
| Author: | **Pablo Cesar** |
| Address: | **Telecommunications Software and Multimedia Laboratory** |
| Room: | **A251** |
| Telephone: | **+358 9 451 5128** |
| | **+358 40 384 9091** |
| E-mail: | **pcesar@tml.hut.fi** |

# Table of contents

# 1 Introduction

Ubik is being developed at the Helsinki University of Technology. It is part of the ongoing Brocom project. More specifically, it is part of the Terminal Software Architecture subproject (by Telecommunications Software and Multimedia Laboratory). The purpose is to create a universal multimedia platform. The key concept behind the project is interoperability. For this reason, the project is following the guidelines:

- Development of Linux and Java based embedded operating system
- Development of XML support
- Integration of applications from mobile and digital television world

# 2 System overview

Figure 1 depicts the architecture of the system. The proposed system consists, from bottom to top, of the following layers: Linux operating system, Native graphical system, Java Virtual Machine and core Java APIs, and Java APIs.

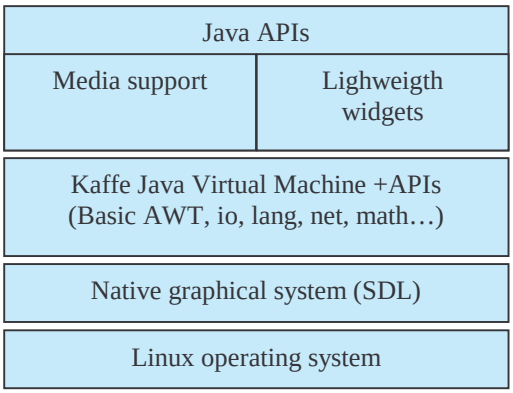



**Figure 1. Architecture of Ubik.**

## 2.1 Operating System

Linux was selected as operating system. Even though is not a real time operating system, it is a promising alternative. It is affordable, scalable and configurable. Hence, it can be modified as needed. In addition, is stable and the number of contributors is growing day by day.

## 2.2 Native graphical system

The native graphical system provides the basic graphic functions. To render graphics, the system uses directly the framebuffer, so there is no necessity of a windowing system like X-Windows. In this level, the features provided are the following:

- Font (selection, rendering)
- FontMetrics (characteristics of the font)
- Rectangle (position, size, rendering)
- Image (load, render)
- Colour (selection, including transparencies)

- Event (catching user inputs)
- Audio (load, render)  [Not yet]
- Video (load, render)  [Not yet]
- Screen (abstraction of the actual framebuffer)

In Ubik, the native functionality is performed by Simple DirectMedia Layer library (www.libsdl.org). SDL is a multimedia library API, which provides low-level access to the system's video framebuffer, sound output and input devices. In addition, libraries like SDL_image (image handler), SDL_ttf (font handler), SMPEG and TrueType Fonts 2 (fonts definition) are used.

### 2.3 Java Interface (JVM + APIs)

The next layer in the system is the Java Virtual Machine. The JVM is provided in order to achieve interoperability, so Java applications can be run on the system. The JVM used in the system is kaffe (www.kaffe.org). Its graphic part (java.awt and java.awt.event) was modified to allow the use of SDL libraries. This layer is depicted in figure 2, and it provides:

- Component
- Container
- Colour
- Event Dispatch
- Events
- Font
- Font Metrics
- Graphics
- Image
- Layout
- Toolkit

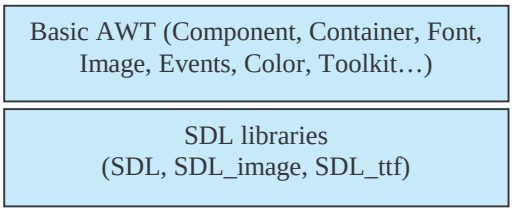



**Figure 2. Java Interface.**

### 2.4 Set of widgets

Even though in the previous layer a Component was provided, that is not enough to effectively develop applications. A set of widgets with defined functionality and representation are needed. Figure 3 depicts the relation between these widgets and AWT. Figure 4 is an UML diagram of these widgets. The widgets were developed following the Home Audio/Video Interoperability (HAVi: www.havi.org) standard for digital television. They are developed in Java, extending java.awt.Component class. They are

lightweight (rendered in Java) and separate the look (representation) and the feel (behaviour). The basic functionality of the widgets is the following:

- Selection (list)
- Text User Input (single entry)
- Action (set of buttons)
- Navigation (focus usage)
- Visible (static widgets)

In the system, a default look is provided for each component. Developers can create a more appropriate look, depending on their needs, and assigned to a given widget.

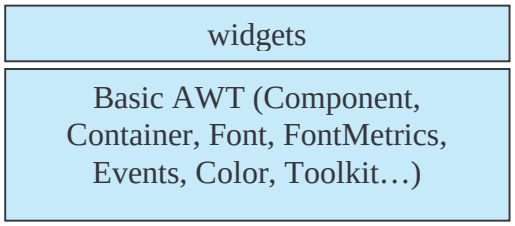



**Figure 3. Set of Widgets.**

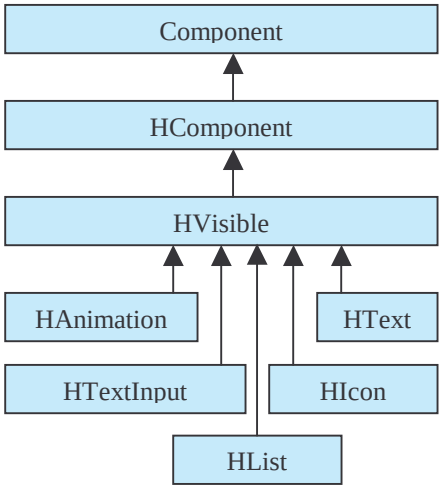



**Figure 4. UML Diagram of the widgets.**

The list of widgets is:

- Static Text
- Static Icon
- Static Animation
- Text
- Icon
- Animation
- Graphic Button
- Text Button
- Toggle Button
- Single Line Entry
- List

In addition to the widgets, the system provides a HScene (similar to java.awt.Window), it is the abstraction of the framebuffer. It provides a special Container (HContainer) to

allow Z-ordeting. Finally, it also provides the necessary events and listeners to develop multimedia applications.

### 2.5 Video Player

The system will include a video player (MPlayer), which functionality is depicted in figure 5. (More information: http://mp.dev.hu/homepage/design5/news.html). Since SDL provides an abstract framebuffer, it can be manipulated before the rendering is done in the actual framebufer. For that reason the video player can be implemented at any level of the system (hardware, native or layer). Then, the graphics done in Java are mixed. Finally, the final output is rendered in the actual framebuffer.

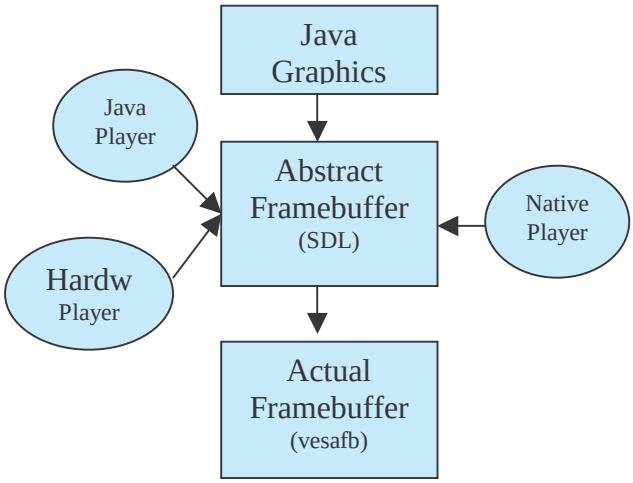



**Figure 5. Video render.**

### 2.6 Applications

In the system, applications from both the digital television and mobile world will be working. These include:

- SMIL player
- Navigator
- Teletext
- SIP client

## 3 Status

Ubik is still under development. Right now, the AWT support and a set of widgets are included. Still, some modifications are needed for the perfect performance of the system. Some demos and applications (navigator, teletext and ice hockey demo) are working properly. In relation to the video support some test have been performed, leading to the decision of using MPlayer as the native player, but the implementation is still under way. The SIP client, because of its video and audio needs, is not yet integrated in the system. But his developer is creating a cross platform version, in which it can run over different Java graphic frameworks like Swing, AWT and ubik set of widgets. The SMIL player is, as well, under way, but only minimal modifications are needed.

# 4 Structure of ubik

The structure of Ubik is as follows:

```
ubik
|
|--bin
|   |
|   |--data
|   |--lib (jar files needed for the compilation)
|
|--build (where the builds are done)
|         |
|         |--fonts (fonts of the system)
|         |--data (data for the applications)
|         |--configuration (configuration for the DTV applications)
|         |--runDigitv (to run DTV applications)
|         |--runGraphicButton (to test the HToggleButtons)
|         |--runIcon (to test HIcon)
|         |--runListGroup (to test HListGroup)
|         |--runSingleEntry (to test HSingleEntry)
|         |--runSmil (the SMIL player)
|         |--runStaticText (to test HStaticIcon)
|         |--runTestContainer (to test the new possibilities of ubik)
|
|--dist
|
|--README
|
|--src
|   |
|   |--bin (ANT needed files)
|   |--build.sh (the builder command)
|   |--digitv
|   |     |
|   |     |--hockey (ice hockey)
|   |     |--hockey2 (ice hockey, digital tv screen version)
|   |     |--navigator
|   |     |--teletext
|   |
|   |--havi (the widgets)
|   |--portal
|   |--sdlawt (java virtual machine modified)
|   |    |
|   |    |--java (all the needed java)
|   |    |--javax (part of kaffe)
|   |    |--kaffe (part of kaffe)
|   |    |--sdl (native graphics)
```

```
|   |
|   |--sip (SIP client)
|   |--smil (SMIL player)
|   |   |
|   |   |--fi (the actual classes of the smil player)
|   |   |--org (XML classes)
|   |   |--univnew (smil demo)
|   |
|   |--test (different Java tests of the system)
|
|
|--screenshots
|
|
|--ubik.pdf (this file)
|
|
|--util
    |
    |-sdl.tar.gz (SDL-1.2.5 modified)
    |-sdl_ttf.tar.gz (SDL_ttf-2.0.5 modified)
    |-sdl_image.tar.gz (SDL_image-1.2.2)
    |-smpeg.tar.gz (smpeg-0.4.4)
    |-ttf2.tar.gz (freetype-2.0.4)
    |-kaffe (The JM in use)
```

# 5 Dependencies

This project has several dependencies, which has to be taken into account in order to be able to run the demos.

## 5.1 Framebuffer

The first thing the user has to do, apart from installing Linux, is to install the framebuffer. Please check the following web page (http://en.tldp.org/HOWTO/Framebuffer-HOWTO.html).

## 5.2 Linux utilities

The system needs several Linux utilities, which normally are included in the installation process of Linux. For ubik, the versions of these utilities are at least:

- make: 3.79
- libtool: 1.4
- gcc: 2.96
- automake: 1.4
- autoconfig: 2.13

### 5.3 Native libraries

Then, the following native libraries have to been included:

- Trutype2 (http://www.freetype.org)
- SDL (www.libsdl.org)
- SDL_ttf (http://www.libsdl.org/projects/SDL_ttf)
- SDL_image (http://www.libsdl.org/projects/SDL_image)
- SMPEG (http://www.lokigames.com/development/smpeg.php3)

**Note:** even though the version of each of the libraries used in the system is provided in *"ubik2/util"*, the user can download them from Internet. Except SDL and SDL_ttf, which was modified specifically for the system.

They configuration of the system should look like:

SDL_DIR=/usr/local/include/SDL
SMPEG_DIR=/usr/local/include/smpeg
FREETYPE_DIR=/usr/local/include/freetype2
SDL_LIB=/usr/local/lib
SMPEG_LIB=/usr/local/lib
FREETYPE_LIB=/usr/local/lib

**Note:** If this is not the case, the user can modify the configuration of the *makefile* file in *"ubik2/src/sdlawt/"*

### 5.4 Fonts

The system uses the fonts provided in the directory *"ubik2/util/fonts"*. If the user wants to use other fonts, please include them in this directory. Other option is to modify the call method in *java.awt.Font* to the desired directory. (**Note:** This feature is going to be modified to make use of an environment variable called UBIK_FONTS).

### 5.5 Environment variables

In order to compile the source code provided in the CD-ROM, ANT has to be installed. In addition, the Java path has to be set properly.

e.g., *export JAVA_HOME="/usr/local/j2sdk1.3.1"*

**Note:** jdk 1.3 is only needed for the usage of ANT, this feature should be improved in the future. Ubik system only needs jdk 1.1.x.

In addition, the system places the linkage library of Java core APIs (libkaffe_sdlaet.so) in the directory *"/usr/local/lib"*. Hence, the environment variable LD_LIBRARY_PATH should be set correctly.

*export LD_LIBRARY_PATH="/usr/local/lib:$LD_LIBRARY_PATH"*

### 5.6 Compiler

The system uses jikes compiler version 1.14, it can be obtained at IBM jikes CVS (http://oss.software.ibm.com/developerworks/oss/jikes/). If the user wants to use other compiler, she/he can modify the build.xml file as desired.

### 5.7 JVM

The JVM is provided in the "*ubik/util/*" directory. The user can copy the kaffe file to *"/usr/local/bin"* and be sure that PATH is set correctly.

i.e., *export PATH="/usr/local/bin:$PATH"*

If there is ant problem running kaffe, please install the Linux library, libgmp-3.0 or higher.

## 6 How to compile

In the project Apache ANT is used for compilation purposes. To compile, the user should go to the *"/ubik2/src"* directory and run one of the following commands:

sh build-sh -> provides help

sh build.sh compile-sdlawt ->compiles sdlawt
sh build.sh compile-test ->compiles tests
sh build.sh compile-havi -> compiles set of widgets
sh build.sh compile-digitv -> compiles igital television application
sh build.sh compile-smil -> compiles SMIL player (not in function yet)
sh build.sh compile-sip -> compiles SIP client (not in function yet)
sh build.sh compile -> compiles all

sh build.sh clean-sdlawt -> cleans sdlawt
sh build.sh clean-test -> cleans tests
sh build.sh clean-havi -> cleans the widgets
sh build.sh clean-digitv -> cleans digital television applications
sh build.sh clean-smil -> cleans SMIL player (not in function yet)
sh build.sh clean-sip -> cleans SIP client (not in function yet)
sh build.sh clean -> cleans all

## 7 Running the applications

The system provides several examples. The demos provided are divided into four categories: system demos, digital television applications, SMIL player and SIP client. The following subsections will explain each of them.

**Note:** to start the demos, first start Linux with framebuffer option, choosing mode *317*. Then go to the framebuffer console (*crt+alt+F1*). Go to the *build* directory *"/ubik2/build"*. Proceed as explained in the following subsections. To quit an application, press ESCAPE key. If this does not work, press *crt+alt+F1* and then *crtl+c*. (Obviously, this is not a desired feature, which has to be improved in the future).

**Note 2:** It is important to remark that within the system are provided the Java core APIs. To run the applications, the user can use the kaffe VM provided in the system or any JVM 1.1 version (e.g., blackdown: www.blackdown.org).

## 7.1 System demos

These demos are studying particular features of the system such as Container, Graphic Buttons, Icons, Lists, Single Entry and Static Text.

### *7.1.1 Container*

This example shows functionalities as creating a Container, creating a component with text, defining a background image, transparencies and Z-ordering.
*To start:* run *./runTestContainer*
*Keys:* Q: pop to front component 1
W: pop to front component 2
E: push to back component 1
R: push to back component 2
T: change alpha to 125 component 3
Y: set opaque component 3
U: set completely transparent component 3
ESCAPE quit demo

### *7.1.2 Graphic Buttons*

This example shows how to use a graphic button. There are three buttons on the screen; the user can navigate among them and launch an action on the top-left corner one.
To Start: run ./runGraphicButton
Keys: LEFT, RIGHT, UP, DOWN: navigate
ENTER: action (only for the left-top component)
ESCAPE: exit

### *7.1.3 Icon*

This example shows how to use icons. There are four icons on the screen; the user can navigate among them, changing the image shown.
To start: run ./runIcon
Keys: LEFT, RIGHT, UP, DOWN: navigate
ESCAPE: exit

### *7.1.4 List*

This example shows the usage of a list. On the screen there is a list and an icon. The user can navigate among them. When the focus is on the list, the user can change its mode and select/deselect one or more items.
To start: run ./runListGroup
Keys: LEFT, RIGHT: navigate
ENTER: if list focused: change the mode of the list (active/normal), so items can be selected/deselected.
if list mode active: change to normal mode, so user can navigate to the icon
P: select/deselect an item
ESCAPE: exit

### *7.1.5 Single Entry*

This example shows the usage of a single line entry. On the screen there are two single line entries. The user can navigate among them. When the focus is on one single entry, the user can change the mode, so she can type characters, remove characters and move the cursor.
To start: run ./runSingleEntry
Keys: LEFT, RIGHT: if not single entry active: navigate
if single entry active: move the cater
ENTER: change the mode of a single entry (normal/active)
Character: inserts the character
ESCAPE: exit

### *7.1.6 Static text*

This example shows the usage of a static text. The font, size and background colour of the text can be modified.
To start: run ./runStaticText
Keys: Q: change the content of the text
W: change the transparency
A: centre alignment of the text
ESCAPE: exit

## 7.2 DigiTV applications

These are the digital television applications. The user starts the application manager. Then she can select one of the applications (ice hockey demo, teletext or navigator)
To start: run ./runDigitv
Keys:

- Application manager:
  - I: launches the ice hockey
  - T: launches the teletext
  - N: launches the navigator
- Ice hockey
  - UP, DOWN: navigate the buttons
  - ENTER: launches the action of a focused button
  - ESCAPE: exit
- Navigator
  - B: launches the info bar
  - N: launches the menu
  - ENTER: launches the action of a focused button
  - UP, DOWN: navigate the buttons, navigate the lists
  - ESPACE: selects an item on a list
  - B: launches the info bar
  - X: launches red button
  - C: launches green button
  - V: launches yellow button
  - Z: goes to previous screen
  - ESCAPE: exit

- Teletext
  - UP, DOWN: navigate the buttons, navigate the lists, navigate the pages.
  - ENTER: launches the action of a focused button
  - X: go back
  - ESCAPE: exit

### 7.3 SMIL player

Not completely functional yet.

### 7.4 SIP client

Not provided yet.

## 8 Known problems

There are several problems, which has to be solved as soon as possible. The two more important are the handle of images (is not so well performed as could be seen in the code) and the animations rendering (the repaint method of a component is not still perfectly done). Improving these features will be done as soon as possible (in the following 2 or 3 weeks).

## 9 Future features

The research group is working already in the following topics:

- Improvement of the system
- Development of video/audio support
- Integration of XForms
- Integration of SIP client
- Integration of digital television portal
- Integration of a new layer under SDL, called DirectFB www.directfb.org, to improve the rendering performance of the system and include the possibility of 3D graphics

## 10 FAQ

Please, before looking at this section take a look at sections 4, 5 and 6.

1. *Where to get the source code?*
   The source code is distributed in this CD-ROM.
2. *Which Linux distribution should I use?*
   It should not matter, but the system has been tested only under RedHat distribution.
3. *How to install the framebuffer?*
   Please follow the instructions of the web page:
   http://en.tldp.org/HOWTO/Framebuffer-HOWTO.html
4. *Which framebuffer mode should I use?*
   317
5. *Error configuring or making SDL?*

Please check out section 5.2. If the problem persists, please refer to SDL instructions (www.libsdl.org)

6. *Error compiling SDL_image?*
   Please check out section 5.2. If the problem persists, please refer to SDL_image instructions (http://www.libsdl.org/projects/SDL_image)
7. *Error compiling SDL_ttf?*
   Please check out section 5.2. If the problem persists, please refer to SDL_ttf instructions (http://www.libsdl.org/projects/SDL_ttf)
8. *Error compiling smpeg?*
   Please refer to smpeg instructions (http://www.lokigames.com/development/smpeg.php3)
9. *Error compiling Truetype2?*
   Please refer to True Type font instructions (www.ttf.org)
10. *Error compiling, JAVA_HOME not found?*
    Set JAVA_HOME variable to your Java distribution. Take into account the dependencies of ANT for this case.
    e.g., *export JAVA_HOME="/usr/local/j2sdk1.3.1"*
11. *Error running, Kaffe softCall?*
    It seems that you are not using the correct version of Kaffe, please take it from the CD-ROM (*ubik2/util/kaffe*) and copy it in */usr/local/bin*. As well, be sure the PATH is correctly set:
    i.e., *export PATH="/usr/local/bin/:$PATH"*
12. *Error running. It says Hscene not found?*
13. *Error running, It says java.lang.UnsatisfiedLinkError?*
    Supposing you have compile the code, the problem could be that the environment variable LD_LIBRARY_PATH is not correctly stated. Please set is as follows:
    *export LD_LIBRARY_PATH="/usr/local/lib:$LD_LIBRARY_PATH"*
    If the problem persists, it could be that you have a previous installation of Kaffe, please remove it for the use of the ubik system. (Obviously this is not a desired feature, which has to be improved in the future)
14. *The fonts are not found?*
    Please, be sure that the fonts are included in *ubik2/build/fonts*

# 11 Contributors

Professor Petri Vuorimaa: head of the research group
D. SC. Cheguang Peng: development of digital television applications
M. SC. Kari Pihkala: development of SMIL player
M. SC. Ganes Sivaraman: provide a JVM (kaffe) running on framebuffer console
Mr. Jukka Rauhala: development of the SIP client
Mr. Jukka Santala: provide a stable JVM (kaffe) with XML parsers support
M. SC. Pablo Cesar: definition and development of ubik system, develop of ubik widgets, provide the set of core Java APIs to render using SDL, integration of applications
Open source projects: development of helpful libraries

M. SC. Mikko Honkola: development of XForms (future)
Mr. Juha Vierinen: new ideas to the system (future)

M. SC. Jean Luc Lamadon: portal development (future)

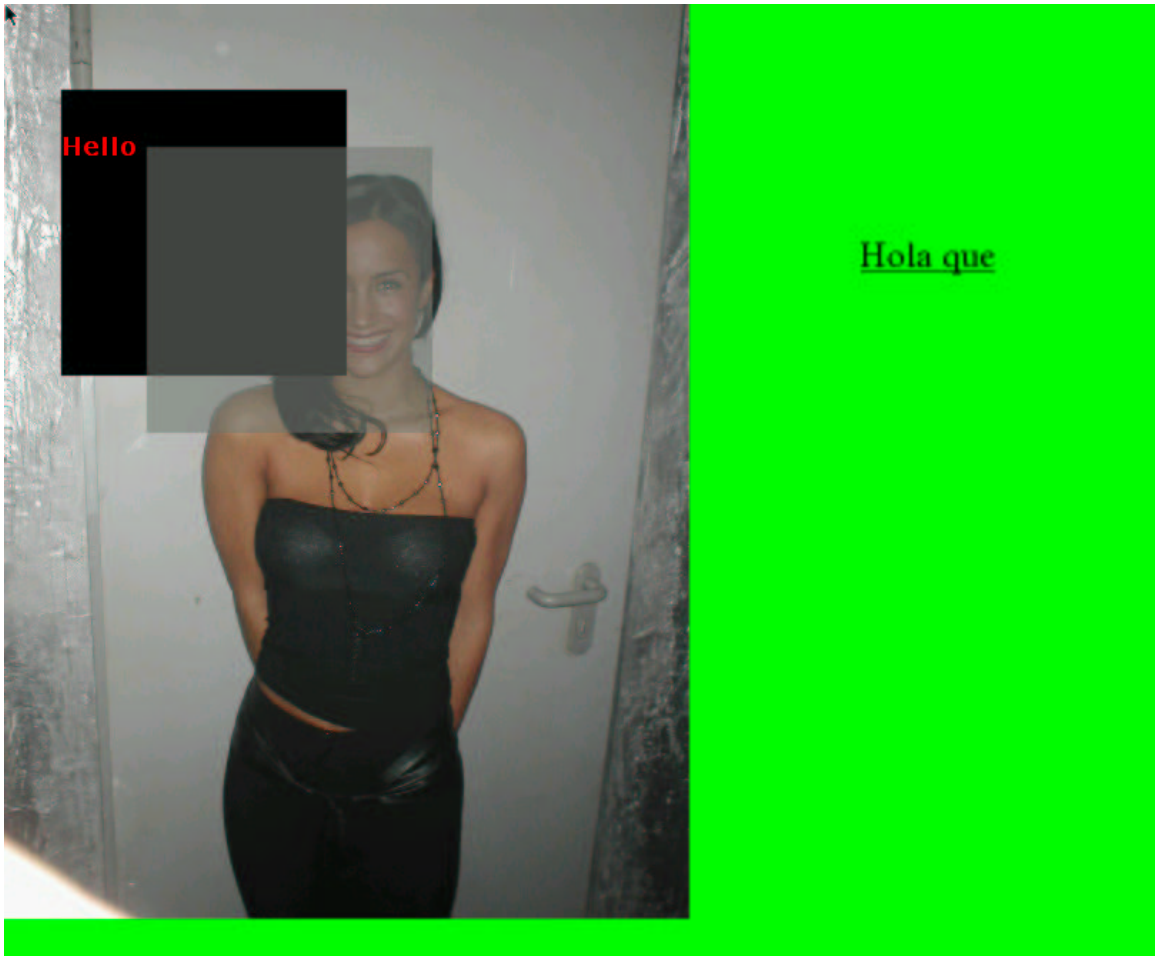
## 12 Thanks

The author, Pablo Cesar, would like to thank all the people doing open source projects such as Kaffe, SDL, SDL_ttf, SDL_image, JSDL, SMPEG and Truetype. Without their ideas and help (and code) Ubik would have been impossible to do. Thanks to Nokia Oyj Foundation for its support during the research. Thanks to the Brocom project because of its financial support. He would also like to express his thanks to D. SC. Cheguang Peng because of the development of the digital television applications, M. SC. Kari Pihkala because of the development of the SMIL player and Mr. Jukka Rauhala for the development of the SIP client. M. SC. Ganesh Sivaraman was an essential person in the beginning of my research life, he was helpful and essential to understand what is a restricted environment. In addition, thanks to professor Petri Vuorimaa because of all the teaching, support and trust on me. Finally, his most deep thank to Mr. Juha Vierinen because his friendship, support, help, teaching...

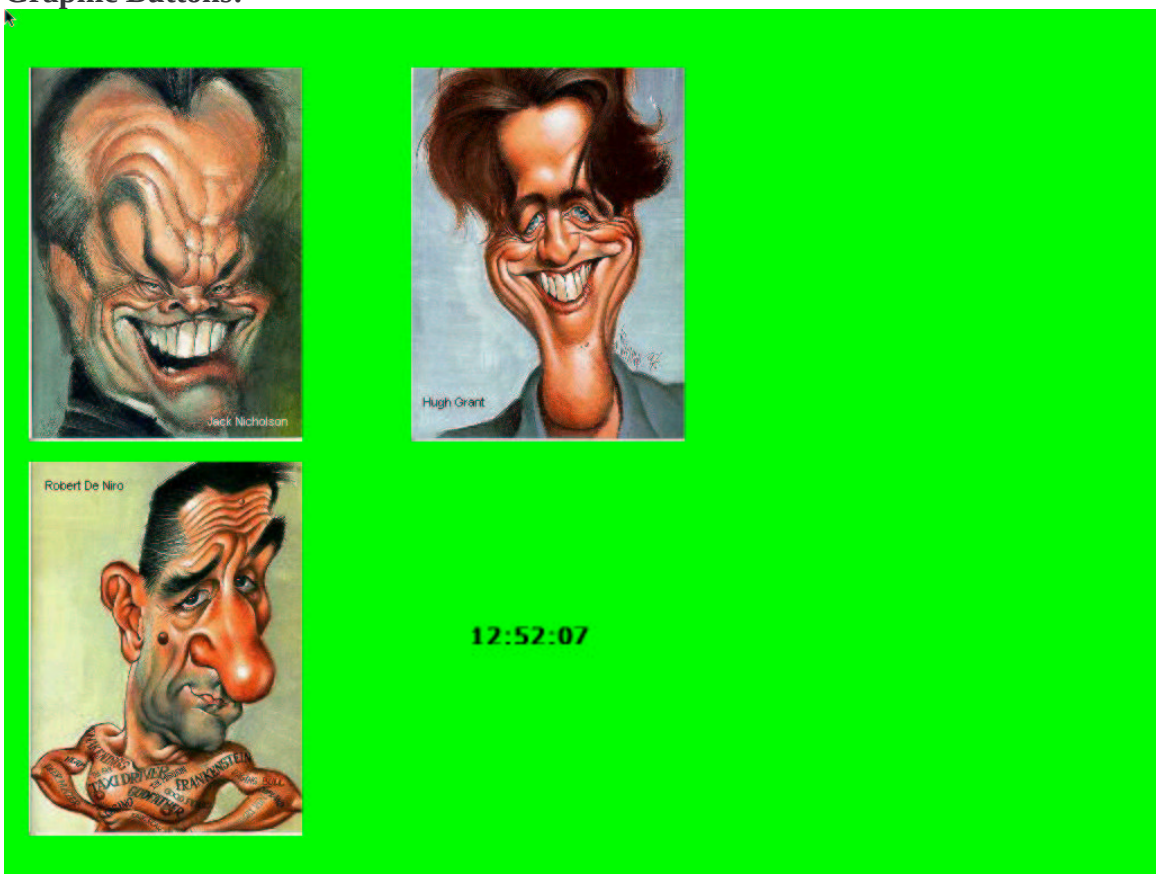
## 13 Appendix (Screenshots)

Here is included some screenshots from the different applications.

**Test Container:**



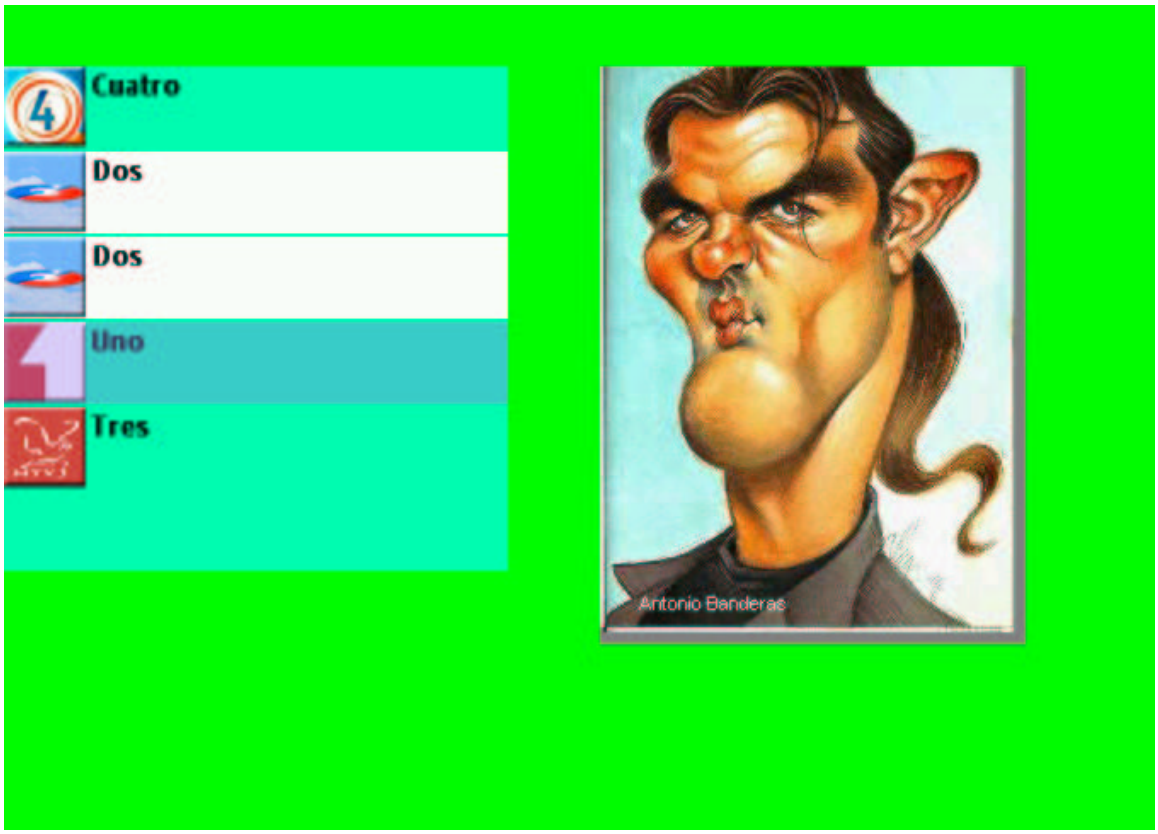
**Graphic Buttons:**



**Icons:**

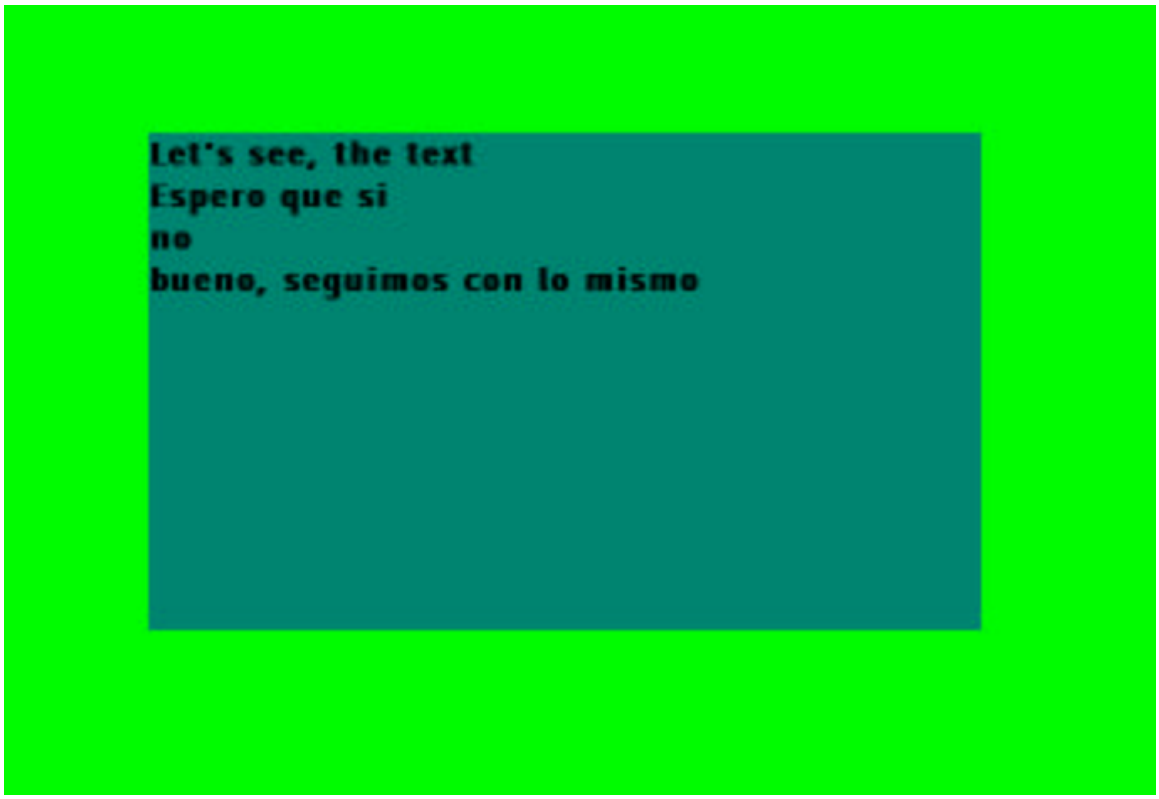

Antonio Banderas
Hugh Grant
Robert De Niro
TAXI DRIVER
FRANKENSTEIN
Bruce Willis

**List:**




**Single Entry:**

**Static Text:**


Let's see, the text
Espero que si
no
bueno, seguimos con lo mismo

**Digitv Ice Hockey:**

**Digitv Navigator:**

**Digitv Teletext:**