MEDIS: A Methodology for the Formation of Highly Qualified Engineers at Masters Level in the Design and Development of Advanced Industrial Informatics Systems

WP2.1 Chapter 6.2: Graphic User Interface (GUI) Part II

Co-funded by the Tempus Programme of the European Union

544490-TEMPUS-1-2013-1-ES-TEMPUS-JPCR

Authors: Capella, J.V. Perles, A., Albaladejo, J

MEDIS: A Methodology for the Formation of Highly Qualified Engineers at Masters Level in the Design and Development of Advanced Industrial Informatics Systems

WP2.1 Chapter 6.2: Graphic User Interface (GUI) Part II

Contract Number: 544490-TEMPUS-1-2013-1-ES-TEMPUS-JPCR

Starting date: 01/12/2013 **Ending date:** 30/11/2016

Deliverable Number: 2.1

Title of the Deliverable: AIISM teaching resources - Industrial Computers

Task/WP related to the Deliverable: Development of the AIISM teaching resources - Industrial Computers

Type (Internal or Restricted or Public): Public

Author(s): Capella, J.V. Perles, A., Albaladejo, J

Contractual Date of Delivery to the CEC: 30/09/2014

Actual Date of Delivery to the CEC: 30/09/2014

Project Co-ordinator

Company name :	Universitat Politecnica de Valencia (UPV)
Name of representative :	Houcine Hassan
Address :	Camino de Vera, s/n. 46022-Valencia (Spain)
Phone number :	+34 96 387 7578
Fax number :	+34 963877579
E-mail :	husein@upv.es
Project WEB site address :	https://www.medis-tempus.eu

Context

WP 2	Design of the AIISM-PBL methodology
WPLeader	Universitat Politècnica deValència (UPV)
Task 2.1	Development of the AIISM teaching resources - Industrial Computers
Task Leader	UPV
Dependencies	MDU, TUSofia, USTUTT, UP

Author(s)	Capella, J.V. Perles, A., Albaladejo, J
Reviewer(s)	Hassan, H., Martínez, J.M., Domínguez, C.

History

Version	Date	Author	Comments
0.1	01/03/2014	UPV Team	Initial draft
1.0	19/09/2014	UPV Team	Final version

Table of Contents

1 Executive summary

WP 2.1 details the learning materials of the Advanced Industrial Informatics Specialization Modules (AIISM) related to the Industrial Computers Module.

The contents of this package follow the guidelines presented in the UPV's documentation of the WP 1 (Industrial Computers Module)

- The PBL methodology was presented in WP 1.1
- The list of the module's chapters and the temporal scheduling in WP 1.2
- The required human and material resources in WP 1.3
- The evaluation in WP 1.4

During the development of this WP a separate document has been created for each of the chapters of the Industrial Computers Module (list of chapters in WP1.2).

In each of these documents, section 2 introduces the chapter; sections 3, 4, 5 and 6 details the Lecture, Laboratory, Seminar and Mini-project of the chapter; section 7 lists the bibliography and the references.

2 Introduction

The chapter #6 is dedicated to the graphical user interface (GUI). Effective and usable user interface and interaction design is essential for desktop and mobile devices. And it's critical for a growing number of industrial platforms and applications. In recent years, we have seen the rapid proliferation of products with integrated digital user interfaces. A new product or application without an attractive and usable interface seems behind the times.

3 Lecture

3.1 Objectives

- To know the possibilities of a GUI design environment and how to use it.
- To be able to design an intuitive and accurate graphical user interface.

3.2 Some graphical user interfaces

3.2.1 LabView

LabView of National Instruments is an application generator for user interface. It was originally conceived as a tool for simulation of electronic circuits, which is why we offer plenty of potential profits

to represent the cover (or front panel) devices, such as oscilloscopes, etc. You can display many types of buttons, measuring instruments, signal windows, and others. See Figure 6.2.

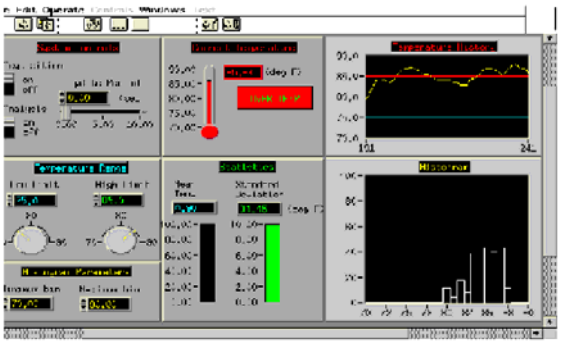


Figure 6-2: Panel LabView.

Simply select with the mouse the object that you want the set that we are offered and properties we want (as the maximum or minimum, spacing, color, object name, etc.) are established.

As standard devices, there is a distinction between the front panel of the object and logic circuit or electronic internal circuit. In LV the logic circuit is the program, and is called diagram and displayed in a separate window.

The LabView works as a icons language where once the front panel created the set of input and output variables that connects to it via wired function that has been set. In Figure 6.3, it is seen as calculating a multiplication of two vectors X (i) and Y (i) actual in LV is introduced:

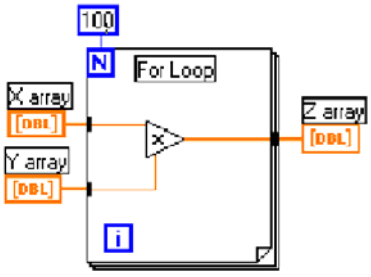


Figure 6-3: Diagram in LabView.

The same example code written in "C":

```
for (i=0, i<100, i++)
{ Z[i] = X[i] * Y[i];}
```

Figure 6-4: Example in C ++ ..

LabView create virtual instruments (VI), this is because a virtual instrument as a program and SubIV is equivalent to a subroutine. However each SubIV is more independent than a process in the sense that one can run a SubIV and not a procedure.

As in most graphics applications, the display is a problem in terms of speed or size. However LV transforms a complicated diagram on a small icon that the user can draw. So overall a program in LV is a small drawing that can be quite profound.

The simulation with LV replaces then purification with other programs, where a kind of colored bubbles and different intensities appear, intermittently flashing where the road running through the data shown, allowing the designer to see exactly what happens in the implementation of program.

Other features of the LV are parallelism, which allows the execution of two parts of the program simultaneously. Generating the application can be executed without LV or any runtime. Connection by TCP/IP is allowed. There is also a network of user groups that facilitate response to the various questions we have.

Bookshops LV offers, besides the basic instructions for any programming language (such as loops, arithmetic relations, etc.), are sophisticated functions such as string functions, conversion functions, functions of matrices and sets (sub-matrices, inverse of a vector), the dialogue and temporal functions and many other (as division number, circular shift with carry, etc.). There are also some analysis for signal generation (Gaussian white noise, ...) processing of signals (the inverse fast Fourier transform, Hilbert transform fast, ...), filters (Tchebyshev, Bessel, ...). In most projects all possibilities are not used, but what can be achieved with LV is enough functionality for complete acquisition: Show the result data and processing in real time.

LV works with programs in ANSI C and linked programs that allow run are generated.

Some of the shortcomings of LV, can be found in some confusing diagrams, some hidden objects, unable to find them, and that one can not group some objects when a diagram is done. Also, the generated source code to link it with a program in LV is not easily accessible. The biggest problem is the automatic search may often offers some surprises. But overall the three stages of data acquisition (results, treatment, and user interface) are usually quite acceptable with this product.

3.2.2 Genie

Other software for data acquisition and control is the Genie of Advantech. It is designed to work on Microsoft Windows environment. It is a fairly intuitive GUI that attempts to simplify the steps of visualization and control strategies. Icons of tool box selected, are connected together, and dynamic visualization (Dynamic Display) is drawn. These objects are found in the library of icons functional blocks that represent the industry standard for data acquisition, control, math, and display functions.

The characteristics of batch processing, networking and icons dynamic link libraries or DLL created by the user, allowing with Genie solve an application virtually.

The object-oriented graphics let you quickly design and implement the designs themselves of an application without the hassle of learning a new language, simply placing the editor to the point of working plane, allowing easier and less time made designs. The user interface is given by a variety of icons that allows us to make a user interface itself. Each selected object can be resized and color. And if the system supports audio output, the program can set even alarm messages (*.wav files).

Network utilities allow both work as single server (standalone) or network. The Genie 2.0 lets you share data with remote workstations when making any decisions. Processes can be evaluated continuously by messages Alarm/Event, which can even be sent to printers and display screens. The batch sequence Genie allows different strategies, so that applications can be modulated and simplified. User icon allows an open architecture being able to program DLLs that are integrated directly into the toolbox.

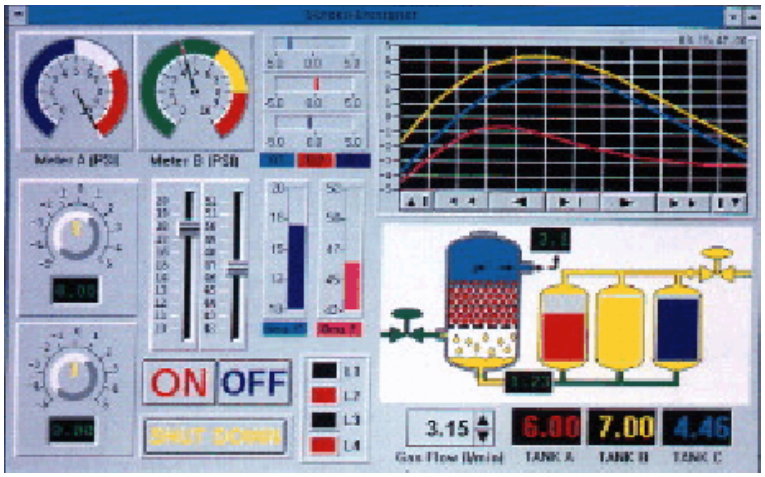


Figure 6-5: Panel Genie.

The most frequent application domains are:

- Automation industries.
- Laboratory automation.
- Human- Machine Interface.
- Monitoring processes.
- Test and measurement.
- Explotation of data.

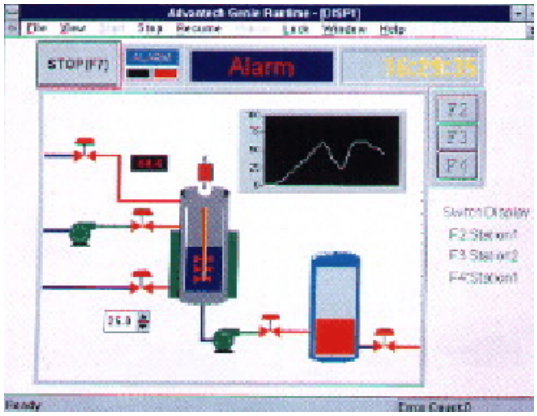


Figure 6-6: Running a program in Genie.

3.2.3 Ptolemy

Ptolemy is a project of the University of Berkeley in California (USA) trying range from modeling to the design of heterogeneous and concurrent systems. This oriented toward embedded systems, and particularly those where both hybrid technologies are used analog electronics and digital, hardware and software, and (including MEMS and micro-electromechanical systems) mechanical devices. The objective is a complex systems using different operations, such as signal processing, feedback control, making sequential decisions and graphical interface.

Perhaps the most important Ptolemy objective is to define and generate code for embedded software while producing own devices embedded systems.

The visual representation of electronic circuits has been given by its schematic in the past, but today are replaced by text to be simulated using a hardware description language (Netlist, VHDL or Verilog). The plot, in general, has not had a good result for the execution of software where it is trying to capture the behavior of flow diagrams. While the visual language for modeling objects ULM is having more attention and is being widely used. That is why Ptolemy II uses it.

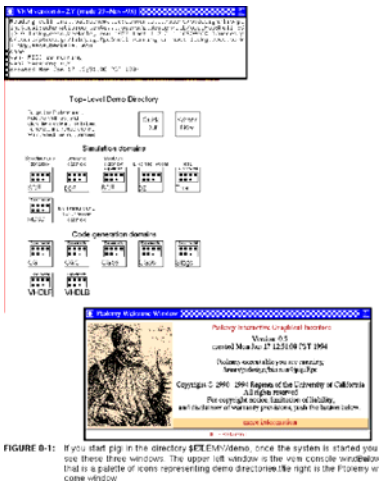


Figure 6-7: Ptolemy Program.

Some features of Ptolemy are:

- Concurrent Design at high level in Java.

- Modularization packet that can be used independently, or distributed across the network on demand from a server.

- Complete separation of abstract syntax semantics. Is structured by separating graph groups and infrastructure in different packets corresponding semantics (as a data stream, analog circuits, finite state machines, etc.) of the graph itself.

- Models of heterogeneous computing. Differential equations, difference equations, finite state machines, synchronous reactive models, discrete event models, synchronous step/asynchronous messages, Petri nets (PN) are timed and timed sequential processes communicating (CSP).

- Concurrent execution protected thread of execution.

- Architecture-based software modeling object.

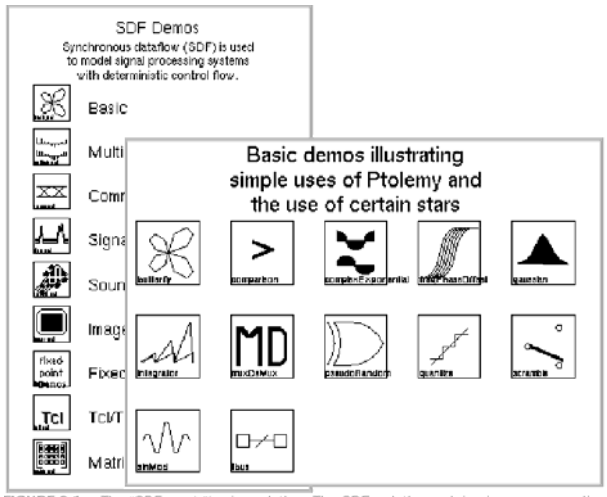


FIGURE 0-1: The "SDF" and "basic" palettes. The SDF palette contains icons representing palettes containing a variety of demos in the synchronous dataflow domain. "basic" palette is one such palette of demos. The icons here represent unive These palettes are explained in more detail in "An overview of SDF demonstrat on page5-36

Figure 6-8: Examples in Ptolemy.

It also allows real polymorphism in the type of system and polymorphs actors in each domain.

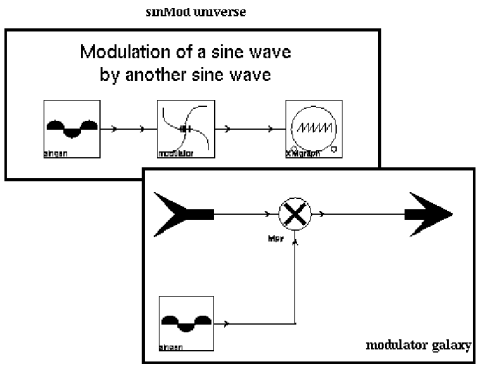


FIGURE 0-1: One of the synchronous dataflow demos. This Ptolemy application modulates a wave with another sine wave. The upper diagram is the top level. The lower contents of the "modulator" subsystem.

Figure 6-9: Simulation of a program icons on Ptolemy

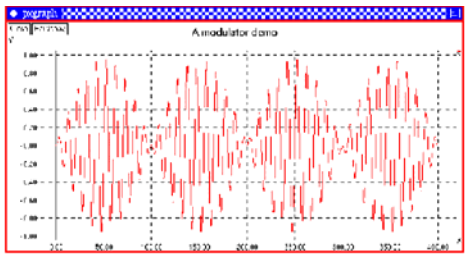


FIGURE 0-1: The graph generated by the "sinMod" application in figure 2-4. The graph is disp by a program called "pxgraph," based on xgraph by David Harrison.

Figure 6-10: Results of the simulation of the previous example.

It is a public domain software and can be obtained from the following URL: http://ptolemy.eecs.berkeley.edu

3.3 Design of a user interface

There are different ways to design a user interface, which is slated for known good one. The most effective will make a good mix of all of them. The first thing to ask yourself is whether it should be oriented interface to the application or user. Much information about building models, where there is speculation about what goes through the user's head when you use a program, that is, what are the concepts that are extracted looking as working a program. These studies are based on the observation of users who often think that a program internally performs work in a certain way, when in fact it does differently. As a result the user expects certain responses (for example) when you press the mouse and surprisingly unpleasant perform another operation did not expect.

The idea is that by understanding how users think about how programs work, and what is the behavior of a program and what is the same (that is, user interface), which suggests users that the program works in a way, and it should be possible to create user interfaces that would help users create models of how a program, which would be closer to reality works. Once done, users should be surprised what happens less often. For this reason it is sometimes said that the user interface should be intuitive.

The literature on the modeling of knowledge of users is closer to the psychoanalysis of software engineering, so most of them engineers believe that contributes little to the designers of a user interface. Most likely this is not correct. For example, graphical interfaces today are

considerable improvements over the command line interfaces of the past. The concepts of the graphical user interface (GUI) have not been invented by programmers, were obtained after many years of research on the human factor at the center Xerox Palo Alto. The results are "borrowed" and put into practice by Steve Jobs of Apple, incorporating them into your Macintosh, so it was the first personal computer you were using a mouse and a GUI. Microsoft is "auto lent" the ideas of Mac and developed.

Today, the leading manufacturer in user interface designs is Microsoft, investigating human factors for cognitive models, responses to color, readability of different character fonts, factors affecting the retina, all in order to improve their environments user. They are the ones who posted the best guides to build a good user interface.

The way to solve the design performing interface application oriented done by analyzing the modules in the problem domain, with which the user must interact. These are the parts that the user must connect as part of the problem you are trying to solve. For example, in a system library, entities of the problem include things like books, borrowed books and whom they have served. Or chips, or linked lists or tables of standardized data should not be included, this is part of the solution. The way to solve the user interface, considering the application, which considers scenarios (or different cases of utility) which involve entities. Interaction diagrams produced by these scenarios often serve as starting points for designing the user interface.

That is why the user interface designer considers the best way to represent user-relevant modules, and how best to enable the required interaction between these modules.

Another issue that arises in view of the above is what should be chosen, dialogs to edit or direct manipulation?. This is the case we want to modify an entity (for example, the size of a circle), at least there are two ways to do it. You can choose one that is an edit box to fill in where the user enters the new values that are requested (the new center and the radius of the circle), or, you can directly manipulate visual representation to achieve the desired effect (pick up from the edge of the circle and stretch or shrink to the desired size).

Which one should be used?. Depends, sometimes one approach could be better than other. When does each?; direct manipulation is preferable because it is often faster and more intuitive. Although a drawing editor, the user may want to specify the size and/or shape position accurately (for example, the radius of 4 cm or 125 pixels). To do the directly manipulated should have feedback, perhaps in a current status label or better in the cursor directly giving the current size and position. Even the user when it finds some difficulty in handling the mouse on accuracy, it should be possible from the keyboard, using the cursor keys move pixel by pixel, or whether it is more appropriate to move with a horizontal and vertical spacing to come given in tenths of millimeter, or any other spacing chosen by the user. Then it should work out the details of the user interface to set the gain. Perhaps a dialog box will be as simple in these circumstances, especially if there are other attributes to be established at the same time (for example, a fill color, border color, width of an edge, ...) should be set to one dialog box rather than via various menu operations. Chances are that you use both methods. Sometimes the direct manipulation is not an option (for example, name a file).

Perhaps the most important consideration is the implementation platform. Each platform has its own set of conventions that can be changed risking our own. For example, all applications platforms of Windows follow certain rules to observe menus. Have at least one file menu (FILE). And the choice of the menu bar when used on the left. Many others have the edit menu (EDIT) and are usually the side menu files. The help menu (HELP) is usually on the part of rightmost, etc. There other conventions to establish that items in each menu keys related to them, etc. Like other concepts appear, such as the position where you need the items in a user interface (this is the case of the position of the edit box that appears when you want to save or open a file). Here is the API ("Application Programming Interface") of Windows who deals with dialog boxes to provide them to the user from any point, while we are using (instead of doing it yourself), even if Microsoft should be automatically updated Windows conventions changed in later versions.

Where does the information on the conventions is removed? There are two main sources of information on these conventions:

The guidance document for the design by the seller of the platform. What if Microsoft is:

"The Microsoft Visual Design Guide" that is bought with Visual Basic. It contains general principles and a section of icon design

The Visual Basic 5.0, in the "Books Online" from Microsoft. You can search the program guide, in section 1 of Part 2, "Creating a User Interface". This section mainly indicates the general rules. Also you should read the section "Designing with the user in mind."

You can also extract information from examples of written applications for the platform in question, which in the case of Microsoft, there are countless.

What about portability?. Since Java has appeared, which is portable to any platform a binary level, most application designers have tried writing applications to run on any machine. What agreements must be applied to a user interface in these applications?. Two aspects should be considered:

First, note that Java libraries called packages include advanced tools for windows ("Advanced Windows Toolkits" or AWT) and Swing, which contains some classes that implement most of the user interface elements that are found in most GUI part of today (for example buttons, drop down menus folders, sets of boxes, list boxes, scrollbars, edit boxes, etc.). The written in AWT and Swing code produces a code of user interface with user interface elements, which look and behave like any other platform that can run. For example, a menu on a PC should remain stable once it has stopped pressing the mouse button while on a "Mac" disappears if the user stops pressing the mouse. Written in Java AWT and Swing packs the code should run on any machine. AWT and Swing is necessarily restricted themselves to the elements of user interface supported by all platforms that support a graphical user interface or GUI. That is, these produce a "lowest common denominator" of GUI tools. Therefore, if a user interface using AWT and/or Swing should work on all GUI platforms, but not be able to take advantage of the special features that are not supported by all platforms built, and this is the price portability.

Second, although the appearance and behavior of the individual elements of a user interface is taken into account by AWT/Swing, appearance and overall performance of the user interface is our responsibility. Then, for example, if a box of external dialogue at the individual elements of the user interface is built, it will have to build the structure of the box, the

interaction between elements within the dialogue and interaction between dialogue and other application. If is designed according to the user interface of an O.S then be some unexpected behaviors should have conflicts with the guidelines of the Macintosh user interface or Motif, or both. Sometimes the best thing to do is to design the interface as neutral as possible for any user operating platform, but it is certain that the risk to disturb everyone.

3.4 Designing graphical user interfaces with Qt

Qt Creator is a set of tools based on the Qt Project Digia. Use the C ++ language, JavaScript and QML platform for developing applications with possibilities multiplatform.

This development environment has a specialized development of graphical interfaces, QML, which allows the developer to maintain separated by a side program logic (using C ++ or JavaScript) and on the other side of the graphic interface (QML). This allows a much more readable code that development aid applications.

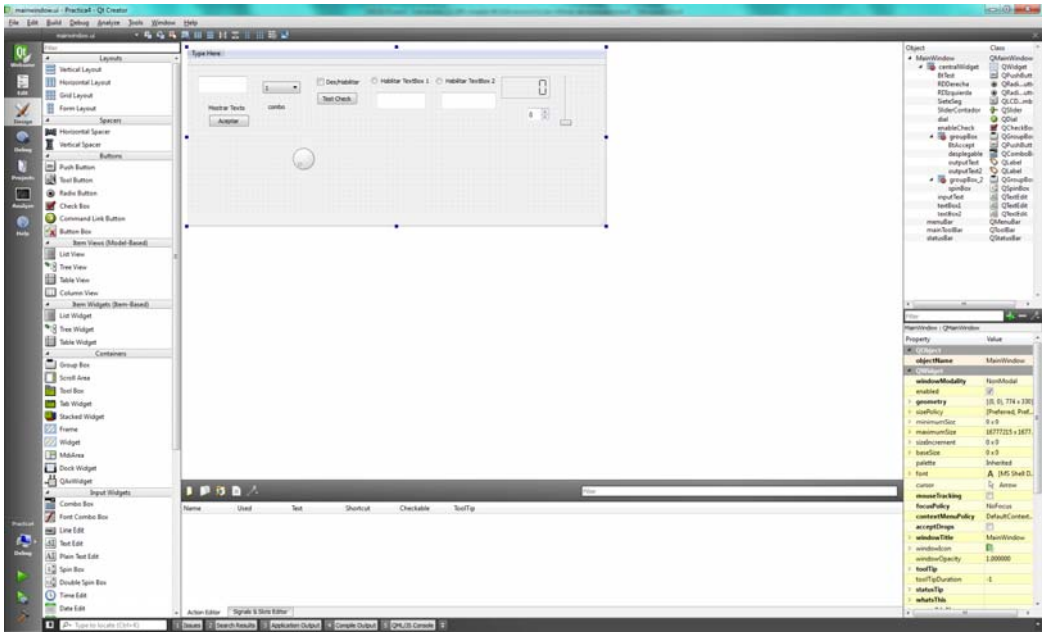


Figure 6-11: Qt design view.

In the laboratory practices the student will learn to develop IGUs using this environment.

4 Lab

4.1 Objective

The main goal is to develop a GUI programming the advanced Qt controls.

4.2 Equipment

- PC compatible computer with Microsoft Windows 7 operating system.
- Open source version of Qt framework for MWindows.

4.3 Departing point

The student should have completed the previous practices.

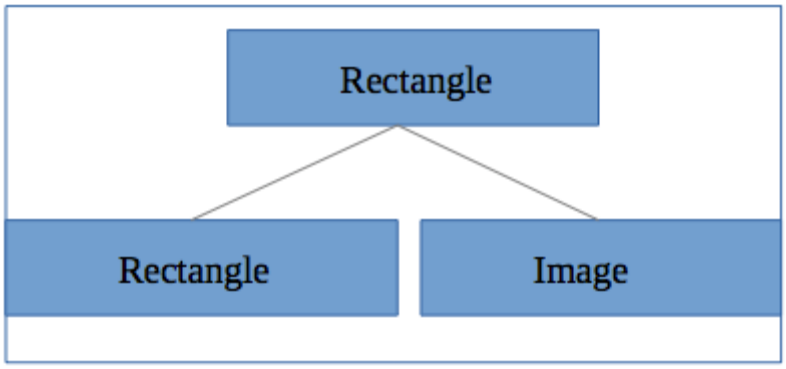
4.4 Activities

QML is a declarative language Meta-Objects based in JavaScript that allows designing user interfaces. It is part of the IDE Qt.

The language is based on as declaring QML elements. These are organized as a tree of objects to visualize blocks, text, graphics, images and interact with them through states, animations, transitions.

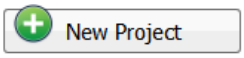
```
Rectangle{
    width: 100
    height: 100
    color: "yellow"

    Rectangle{
        anchors.fill: parent
        anchors.margins: 10
        color: "blue"
    }

    Image {
        id: test
        source: "test.png"
  }

}
```

The representation in tree form the previous example is as follows.

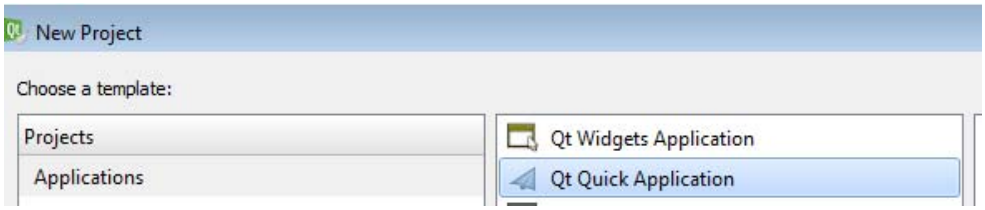


In this example you can see a few elements, 2 boxes, 1 image, which are formed by different values called "properties", which allow us to define relationships with other objects and other features.

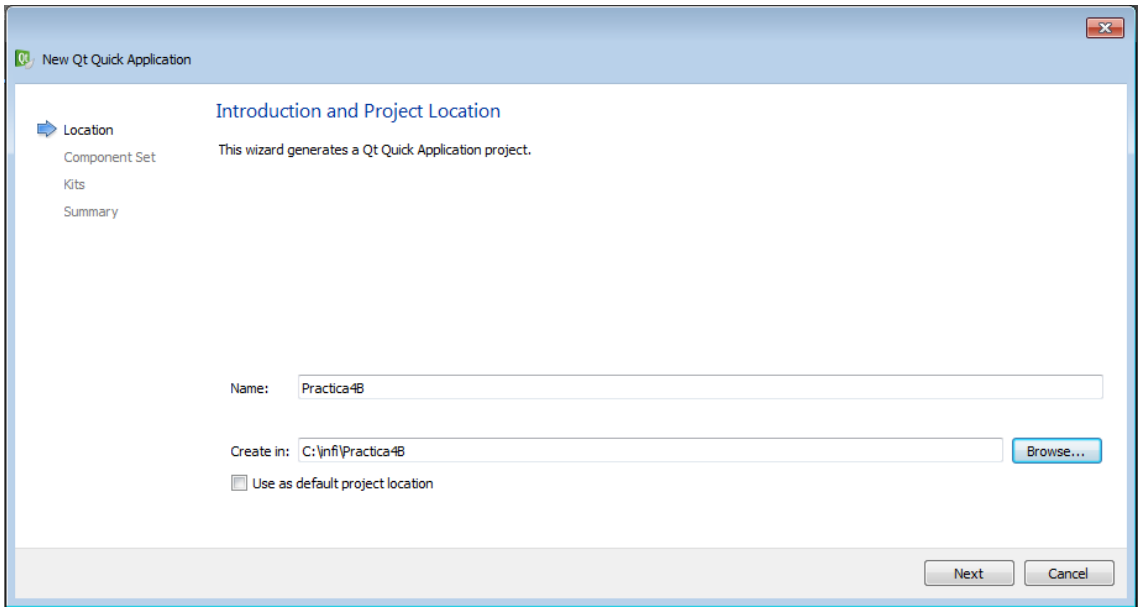
It then shows how to create a QML project, we will click on the button



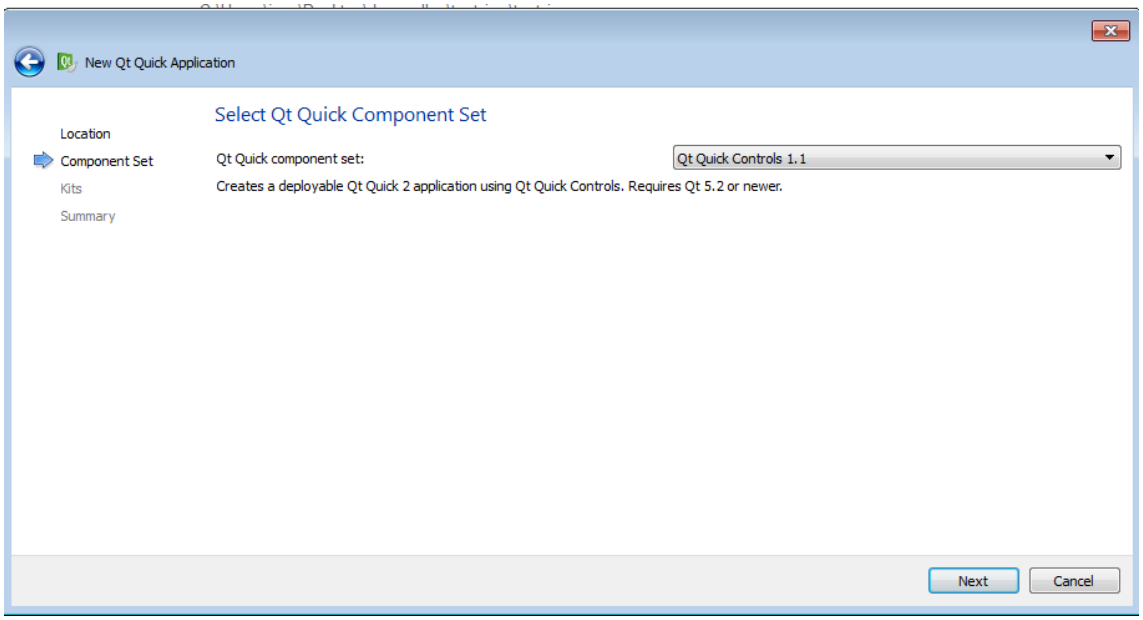
In the window that leave us then select Qt QuickApplication.



We pressed the Choose button, and precede to the next window, select the directory where you create the project.



We spent the next window, we can choose several options, for example, we selected that comes by default and click on next.



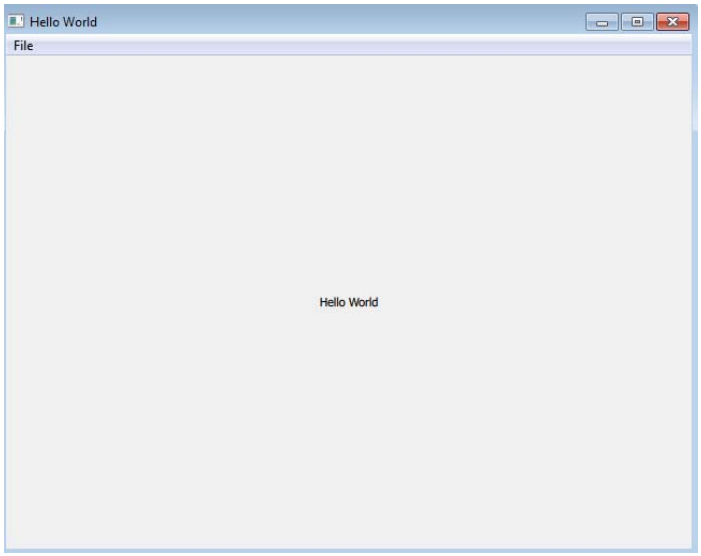
We will boot directly QML editor, and we sample code shows:

```
import QtQuick 2.2
import QtQuick.Controls 1.1

ApplicationWindow {
    visible: true
    width: 640
    height: 480
    title: qsTr("Hello World")

    menuBar: MenuBar {
        Menu {
            title: qsTr("File")
            MenuItem {
                text: qsTr("Exit")
                onTriggered: Qt.quit();
            }
        }
```

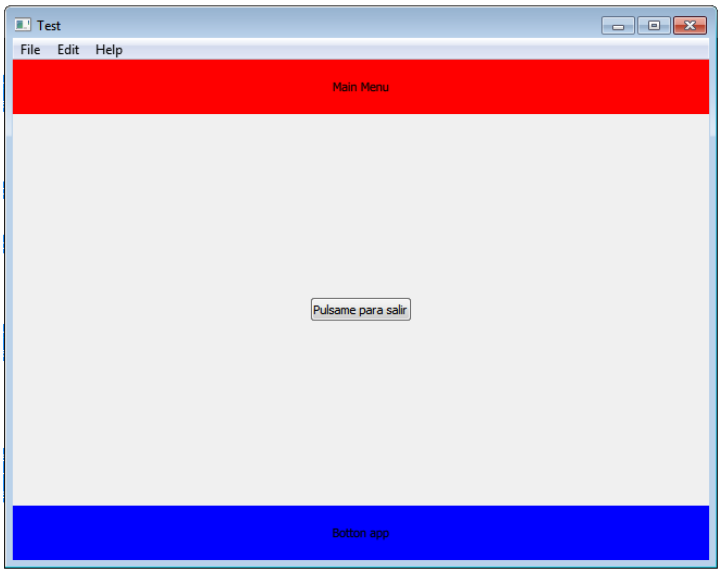
```
    }

    Text {
        text: qsTr("Hello World")
        anchors.centerIn: parent
    }
}
```

Archivo: main.qml //versión 1

If we give this code will compile, we ripped a window, which will interface example.



After displaying this we will show how to create a simple interface that consists of few elements as shown below.




```
import QtQuick 2.2
import QtQuick.Window 2.1
import QtQml.Models 2.1
import QtQuick.Controls 1.1

ApplicationWindow {
    visible: true  // with this property we make it visible in the execution
    width: 640     // application width
    height: 480    // application heith
    title: qsTr("Test")  // Window name


    menuBar: MenuBar {   // with this option the menu bar is created
        Menu {           // File button available in the bar to expand
            title: qsTr("File") // given name
            MenuItem {  // add an element the file menu
                text: qsTr("Print")
            }
            MenuItem {  // add an element the file menu
                text: qsTr("Exit")
                onTriggered: Qt.quit();  // application end action
```

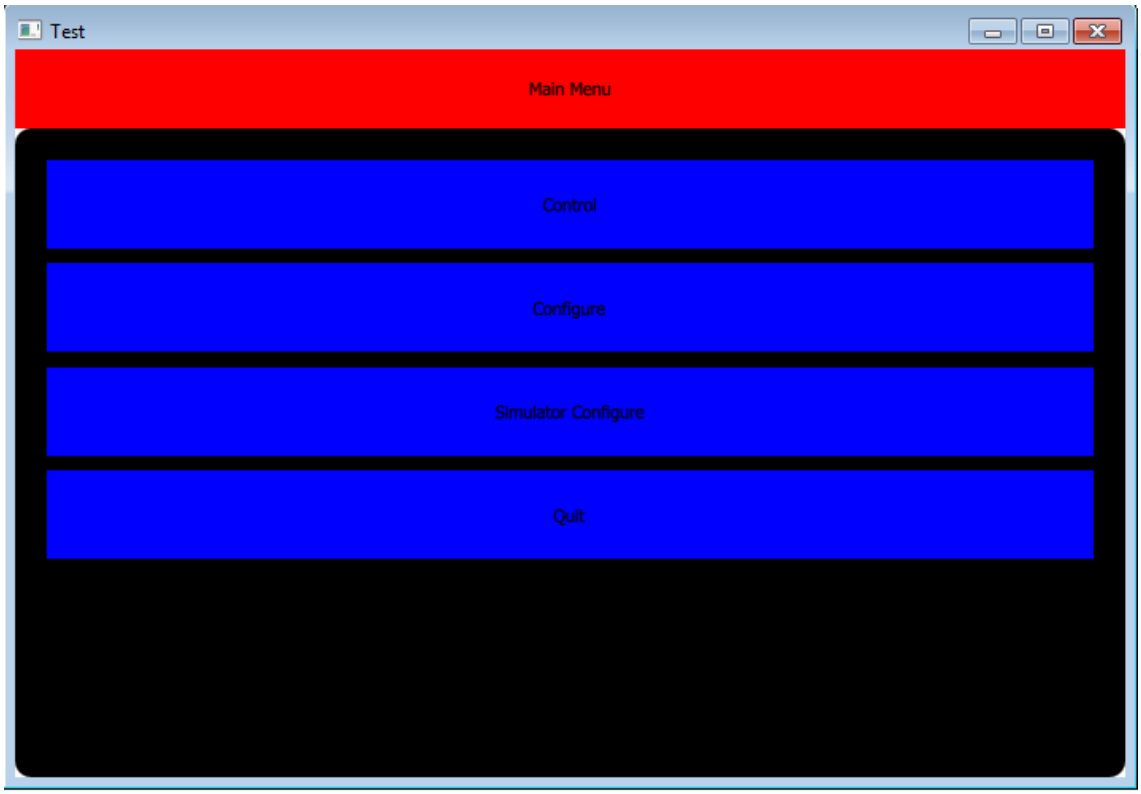
```
            }
        }//fin Menu
        Menu {
            title: qsTr("Edit")
            MenuItem {
                text: qsTr("Copy")
            }
            MenuItem {
                text: qsTr("Cut")
            }
            MenuItem {
                text: qsTr("Paste")
            }
        }// Fin menu
        Menu {
            title: qsTr("Help")
            MenuItem {
                text: qsTr("About")
            }
        }// End Menu
    } //fin MenuBar

    Rectangle{ // add a Rectangle on top
        id: rect1
        width: parent.width
        anchors.top: parent.top // Locate on the top of the window
        height: 50
        color: "red"
        Text{
            id: textMenu
            anchors.horizontalCenter: parent.horizontalCenter
            anchors.verticalCenter: parent.verticalCenter
            text: "Header App"
```

```
        }//Fin Text
    }//Fin rectangle

    Rectangle{ // add a Rectangle on the bottom
        id: rect2
        width: 640
        anchors.bottom: parent.bottom //Locate the bottom of the window
        height: 50
        color: "blue"
        Text{
            id: textMenu2
            anchors.horizontalCenter: parent.horizontalCenter
            anchors.verticalCenter: parent.verticalCenter
            text: "Botton App"
        }//Fin Text
    }//Fin rectangle

    Button {
            text: "Click Me to exit"
            anchors.horizontalCenter: parent.horizontalCenter // Locating
element within the window

            anchors.verticalCenter: parent.verticalCenter
            onClicked: Qt.quit()  // action to exit the application

        }
}
```

Archivo: main.qml //versión 2

IMPORTANT NOTE: EVERY TIME EDIT FILE QML MUST GIVE THE OPTION "CLEAN PROJECT" WHICH IS IN THE MENU "BUILD" AND THEN GIVE "REBUILD PROJECT" AND TO EXECUTE A "RUN" IF YOU DO THIS MAY HAVE PROBLEMS THAT CHANGES ARE WE DO NOT APPLY.

Activity: The objective is to do an interface with one button made from a rectangle. To do this we will have to define a .qml file within the rectangle have the description and the main.qml file a ListView element where it was invoked to rectangle.qml and will give you attributes and values there will be created.



5 Seminar

5.1 Introduction

This seminar will be to work about the GUIs development and several alternatives in its implementation. On the second part the seminar focuses on the importance of a good GUI.

5.2 Objectives

The main objectives are:

- To relate the GUI concepts with other technical concepts that usually are studied in different subjects, in a contextualized way.
- Acquire team-working skills, discussion the GUI development topic.
- Acquire documentation and presentation skills.
- Acquire critical searching of information skills, necessary to find new resources to develop modern, intuitive and impressive GUIs.

5.3 Advanced GUI design: external libraries and other possibilities

This seminar will be to work on the advanced aspects of GUIs.

With internet we will collect the following information:

- Some companies that develop GUIs.
- Comparative review of advanced components for professional GUIs.
- Library would be chosen as the most suitable for the realization of an advanced GUI.
- Possibilities of integration of the chosen development library in an industrial application.

After that students should work in groups of 4 people, in order to think about the resources and possibilities to develop professional and advanced GUIs for industrial applications.

At the end of the seminar they must expose and discuss their experiences to the whole group.

6 Mini-project

6.1 Implementation of the GUI: Advanced controls

In this second mini-project activity related with GUI topic, the student should improve the basic GUI implemented in the mini-project with advanced controls and features that provide a professional aspect to the under development mini-project.

The student should apply all the resources learned during the lecture and laboratory sessions.

In this activity students should use for the graphical interface design at least the following elements:

- QML code
- customized drawings
- customized components

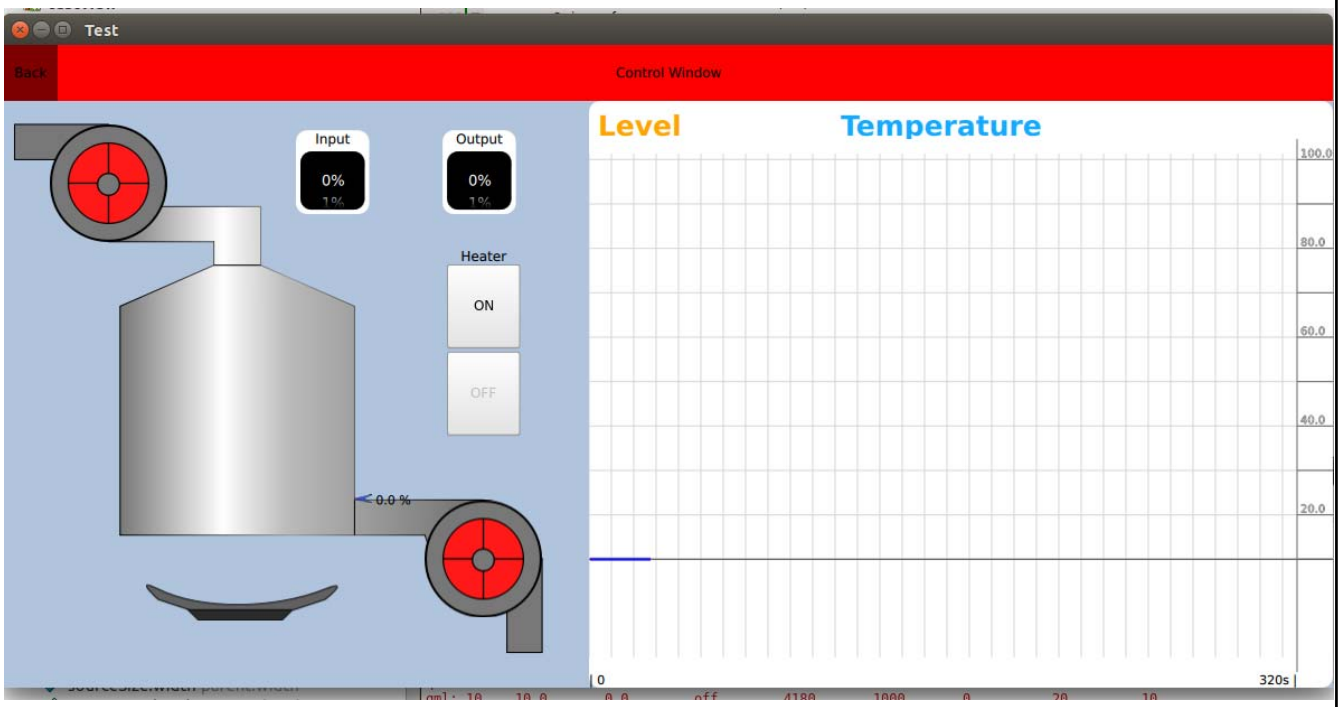


Figure 20. Example of GUI for the liquids tank project

Finally the student should validate the developed GUI in the mini-project.

7 Bibliography

Jeff Johnson , *Designing with the mind in mind : simple guide to understanding user interface design rules*. Ed. Morgan Kaufmann, 2010, ISBN 9780123750303.

Foley, James. D., Andries van Dam, S.K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice*, second edition, Addison-Wesley, Reading, MA, 1990.

Soren Lauesen, *User interface design : a software engineering perspective*. Pearson Addison Wesley, 2007. ISBN 9780321181435.

Ballard, Barbara. *Designing the mobile user experience*. Ed. Wiley, 2007. ISBN 9780470033616.

Rogers, David F. and J. Alan Adams, *Mathematical Elements for Computer Graphics*, second edition, McGraw Hill, New York, NY, 1990.

Wilbert O. Galitz, *The essential guide to user interface design : an introduction to GUI design principles and techniques*, 2nd ed. John Wiley & Sons, 2002. ISBN 0471084646.

http://www.labview.com

Digia, Libreria Qtqml.Models, 2013, http://qt-project.org/doc/qt-5/qtqml-models-qmlmodule.html