Re BASIC! Tab generation and management

Disclaimer

This SOFTWARE PRODUCT is provided by El Condor "as is" and "with all faults." El Condor makes no representations or warranties of any kind concerning the safety, suitability, lack of viruses, inaccuracies, typographical errors, or other harmful components of this SOFTWARE PRODUCT. There are inherent dangers in the use of any software, and you are solely responsible for determining whether this SOFTWARE PRODUCT is compatible with your equipment and other software installed on your equipment. You are also solely responsible for the protection of your equipment and backup of your data, and El Condor will not be liable for any damages you may suffer in connection with using, modifying, or distributing this SOFTWARE PRODUCT.

You can use this **SOFTWARE PRODUCT freely, if you would you can credit me in program comment:**

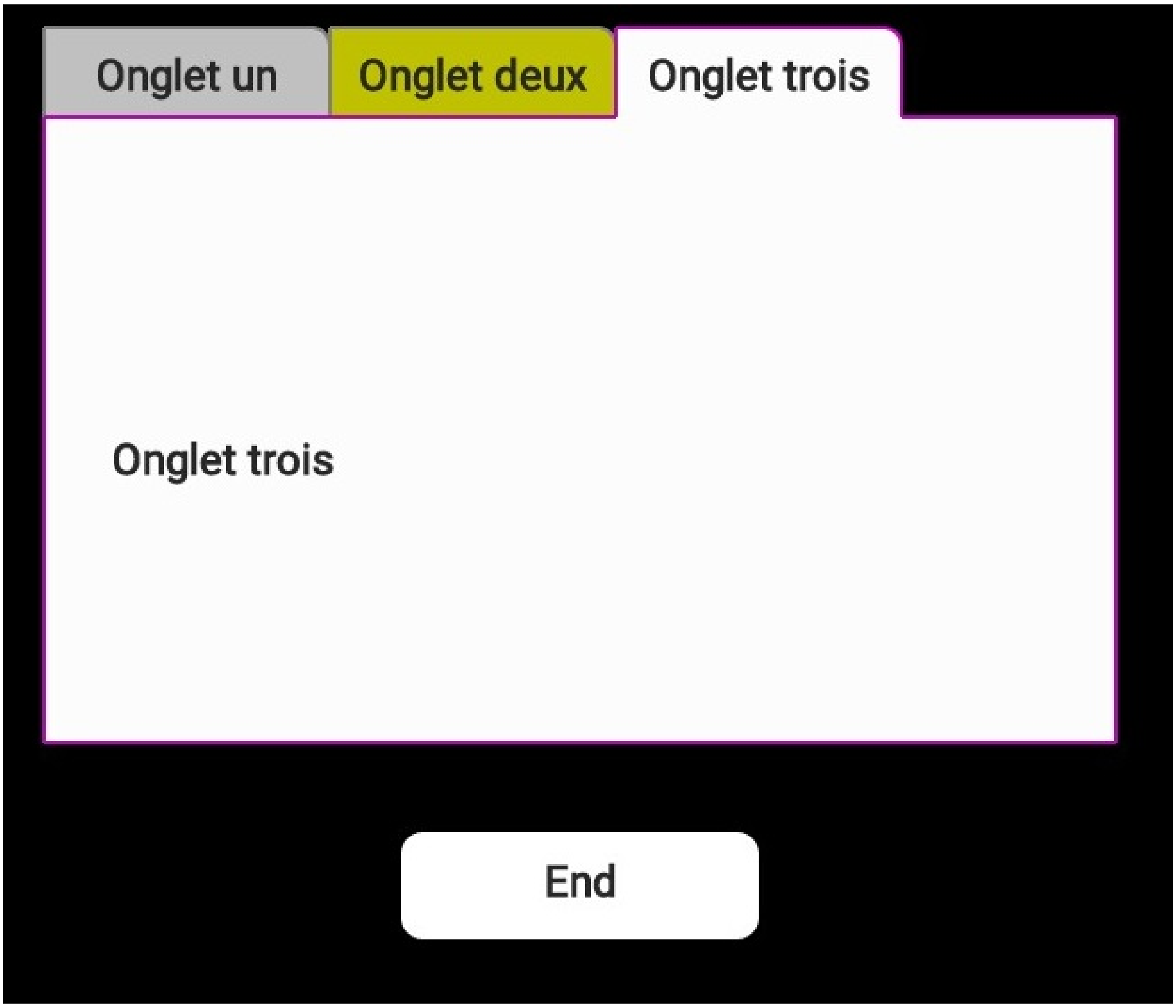
El Condor – CONDOR INFORMATIQUE – Turin

Comments, suggestions and criticisms are welcomed: mail to rossati@libero.it

Conventions

Commands syntax, instructions in programming language and examples are with font `COURIER NEW`**. The optional parties of syntactic explanation are contained between** `[square parentheses]`**, alternatives are separated by** `|` **and the variable parties are in** *`italics`*.

Table of Contents



1 Tab generator

The script is a demo that contains functions to generate tabs in RFO Basic (version 1.91).

1.1 Calling the tab generator

The function `createTab` has one string parameter that contains informations for generate one tab; it returns a bundle of data.

```
answer = createTab("Title=Onglet un,color=FFC0C0C0,position=1,width=600,height=400)
```

1.2 Parameter description

The parameters have the form `parameter=value[,parameter=value [,...]]`, where *parameter* is indifferent to the case.

All parameters are optional and have a default value.

Parameter	Default value	Note
`Border`	`FF7F7F7F`	Border color
`Color`	`FFFCFCFC`	Background color (silver)
`Height`	`90% of screen height`	Tab height
`Left`	`5% of screen width`	Left position
`Position`	`1`	Tab position
`Title`	`Tab` *position*	Tab title
`Top`	`5% of screen height`	Top position
`Width`	`90% of screen width`	Tab width

1.3 Information returned

The function returns some information contained on a bundle useful for manage the form:

Key	Type	Note
`glist`	numeric list	list of object number used to create the tab
`touch`	string	containing the coordinate of rectangle: *Left*,*Top*,*Right*,*Bottom*
`label`	numeric	object number of the tab title

For act on tabs `glist` and `touch` must be stored: in this script `glist` is stored in an array and `touch` in a bundle with name `Tab`*Position* (see below).

```
bundle.create touchs       % key = fieldname, dtaa = left, top, right, bottom
bundle.create tabs         % key = TabPosition, data = gList
array.load arrayGList[],0,0,0,0,0      % grafic list array of tabs
...
answer = createTab("Title=Onglet un,position=1,width=600,height=400,LEFT=50,Top=50")
arrayGList[1] = getBundle(answer,"glist")
bundle.put touchs,"Tab1",getBundle$(answer,"touch")
```

1.4 Guideline

1.4.1 Adding graphic object to tab

A graphic object can be added to a tab by adding its object number to the list of tab objects:

```
gr.text.draw txt,150,300,"Onglet un"
list.add arrayGList[1],txt  % this text is added to tab
```

1.4.2 Changing a Tab presentation

All drawing object are contained in a list and are drawn from the first one: we must rearrange the list for send at bottom the graphic object list of the selected tab (see the script above fragment):

```
FN.DEF changeTab(tabNumber,aGList[],otherGraphics)
  ARRAY.LENGTH numberOfTabs, aGList[]
  list.create N,newLst
  For i=1 to numberOfTabs
    if aGList[i] > 0 Then if i <> tabNumber Then list.add.list newLst,aGList[i]
  Next
  list.add.list newLst,aGList[tabNumber]  % this is the tab to show
  if otherGraphics > 0 then list.add.list newLst,otherGraphics
  list.toArray newLst,newArr[]
  GR.newDL newArr[]  % replace the display list
  GR.render
FN.END
```

2 Functions

2.1 Create button

The function create a button with rounded corner; it returns the Object number list list.
Syntax:

`buttonGraphicList = button(text$,left,top,width,height)`

2.2 Test string by Regular Expression

The function `matchRE` return true if a string match a regular expression:

Syntax:

`boolAnswer = matchRE(stringToTest,regularExpression)`

```
FN.DEF matchRE(field$,re$)    % retun true if match
      split a$[],"*"+field$+"*",re$
      array.length ll,a$[]
      FN.RTN ll-1
FN.END
...
re$ = "[aAbBcCdDeEfF\\d]{8}"
if matchRE("f7c0c0c0",re$) Then ...  % this matches
```

2.3 Show tab selected

The function force the redraw of tabs.
Syntax:

`changeTab(tabNumber,arrayOfGraphicsList,otherGraphicsList)`

`otherGraphicsList` is a graphic list of objects outside of Tab or `0` if there are none.

2.4 Test choice

The function returns a name of the object touched or an empty string.
Syntax:

`nameOfObject = button(touchsBundle)`

3 Open problems

When Tabs contain touchable objects the script is unable, currently, may be unable to locate the correct object.

4 Technical notes

4.1 Arrays

Name	**Note**
`arrayGList`	Graphic list of Tabs

4.2 Bundles

Name	**Key**	**Value**	**Note**

`answer`	*`parameterName`*	*`parameterValue`*	Returned parameters
`touches`	*`Name`*	*`touchRectangle`*	