Main Menu ☐

Main Menu ☐

PROGRAMS

- AXOLOTL
  - AXOLOTL IDE
  - BASIC CONTROLS
  - CUSTOM CONTROLS
- BUGEE#@!
  - SERVER SIDE INSTALLATION
  - HOW TO USE THE SCRIPTS
  - HOW TO USE THE CLIENT

Creating Custom Controls

In this section we will discuss about how to create a new control or edit an existing one.

Introduction

Note : In the examples below i have used the Notepad++ for editing the xml files of the controls .

All the controls are simple xml files. Every xml control file must be paired , with a png image of a size 16x16 that the Axolotl will use to create the apropriate button in its interface . For your control to be operational , you must put it inside the **controls** folder in the FrameCreator folder. If you want the control to be appeared in the Basic tab of the Axolotl , then put it inside the **Basic** folder (its png too) , if you want the control to be appeared in another tab , create a folder (e.g ExtraControls) and put your control inside it . The next time that the Axolotl opens, a new tab with the name of the folder you created (e.g ExtraControls) will appear and your control will reside there .

Warning

Do not forget that in xml the tags and attributes are case sensitive. Do not type **Basics** when do you mean **basics**

Inside the xml file there are some characters that you cannot use as a text for the attributes , like the '<' character and the '&' character. Nevertheless this characters are used inside the C/C++ code that you may write in the control xml file. This problem is solved easily if you replace this characters with the xml code equivalent. Take a look in this page : Xml Special Characters

Structure of a Control file

Basic structure

Bellow is the basic structure of the control file , with blue color are the comments that explain the functionality of the tags , with redish is the actual tags and attributes .

Powered by 000webhost

<control>The starting tag . MUST be present .

<basics type = "GroupBox" style = "CONTAINER" proc ="default" icon = "groupbox.png" tooltip = "" noparent = "true" maxchildren = "1" childtype = "" parenttype = "" >

<basics> Tag MUST be present. Represents basics of a control.

type attribute. MUST be present .

The type of the control. This attribute must be a unique (you cannot have two controls with the same type attribute.) string that is valid in C/C++ code.

style attribute. MUST be present .

The style of the control , this can have only two values , CONTAINER and NORMAL.

The CONTAINER value forces the control to behave like a parent for other controls , so in the designer if you put a control inside the client area of the control with the CONTAINER value , this control will be drawn and belongs in the container control .The NORMAL value specifies a normal control .

proc attribute. MUST be present .

This attribute is valid only with the CONTAINER style. This attribute specifies if the container will be subclassed. The Axolotl subclass the container that have set the proc attribute as "custom" with an internal callback procedure that forwards the messages to the main window. If the proc attribute is not being set or it has the "default"(in fact any value except "custom") value then the container will not be subclassed .

Note : The subclassing is crucial to make containers forward the notifications of their children to the main window .

icon attribute. MUST be present .

This atribute is the file name of the png image that the control will use as his button icon inside the designer. The string is just the image file name and not the whole path. The png must be located inside the same folder that the control xml file resides. The png must be 16x16 pixels in size .

tooltip attribute. MAY be present.

This attribute forces the Property Editor inside the designer to show a tooltip property with the text that is specified. When you declare this attribute , is for the best to leave it empty like this : tooltip="" . The most propable situation is for the user to want to set the tooltip it self .

noparent attribute. MAY be present.

This attribute is valid only with the CONTAINER style . This attribute forces the Axolotl to assign the parent of this control as the parent to his children too. This is usefull if you need a control to behave as a parent container in the designer , but in reality inside the C/C++ code it cannot be the parent of other controls. The valid values are **true** and **false**, but the absence of this attribute , or any other value except true resulting the same as **noparent = "false" .**

maxchildren attribute. MAY be present.

This attribute is valid only with the CONTAINER style . If this attribute is set , then the control cannot have more than the specified number of children. If this attribute is not set , then the number of children is infinite (or rather as much as a integer type can hold). e.g if you have set maxchildren = "10" then when you atempt to create the eleventh child, the Axolotl will pop an error message and the operation will be canceled. This attribute is usefull when you want to limit the children in a container.

childtype attribute. MAY be present.

This attribute is valid only with the CONTAINER style. If this attribute is set , the Axolotl designer forbids any other type of control to be a child of this container.

The childtype MUST be a valid control type of the Axolotl , this means that the xml control file must being inside the **controls** folder of the Axolotl.

If this attribute is not set , then the container will have not any restriction.

e.g childtype = "Edit"

parenttype attribute. MAY be present.

This attribute forces the Axolotl designer to check wherever the parent of the control meets the parenttype attribute and if not, the child will not be created. This means that if you have set the parrenttype = "GroupBox" , then if you try to put this control in a panel or a frame the designer will pop an error message and the operation will be aborted .

A typical caption that shows Text .

Free text. The above text between <basics> and </basics> is the text that the property editor will show when the user clicks the **<SPECS>** button. You can write as much text as you want , but keep the context relevant to the uses and the usability of the control that you created.

</basics>The closing tag .

<codevars>This Tag CAN be present.

Inside this tag you can put what ever code you want , and it will be inserted in the code at WinFrames.h exported file. You can define values , create variables or class headers. Note that this code will be inserted only once even if you have several of this type of control inside you project.

Powered by 000webhost

```
#ifndef ILC_ORIGINALSIZE
#define ILC_ORIGINALSIZE       0x00010000
#define ILC_HIGHQUALITYSCALE    0x00020000
#endif
```

</codevars>The closing Tag

<uservars>This Tag MUST be present even if its empty

Inside this tag the user can define variables that will be visible inside the Property Editor of the designer .

This variables can be used inside the C/C++ code as we will see later .

A variable can be declared as this : <**NAME type**="[one of the valid types]" **value**="the value of the variable" **cvalues**="a,list,of,values"/>

The valid types of the variables are :

string

The variable can have any combination of leters , symbols and numbers as its value. This is the true type of all variables.

filelist

This type forces the property editor to show a dialog that the user can use to load files as strings.

boollist

This type forces the property editor to show a list with the values **true** and **false** for the user to select .

valuelist

This type forces the property editor to show a list with the values that you have specified in the cvalues attribute , for the user to select. The values must be comma seperated.

checklist

This type forces the property editor to show a dialog with the values that you had set , as cheklist items. This type is optimized for showing a checklist of flags like **|ILC_PALETTE,|ILC_MIRROR,|ILC_PERITEMMIRROR,** but it can work whith any type of comma seperated values .

cntrlist

This type forces the property editor to show a list with the controls of the project.The cvalues attribute can be the type of any control of the Axolotl , or simply the * character to force the property editor to show a list of all the controls in a project .

For clarification see the below examples .

```
<Text type = "string" value = "Text" />
<Files type = "filelist" value = "" />
<CreateMask type = "boollist" value = "false" />
<ImageType type = "valuelist" value = "IMAGE_BITMAP" cvalues = "IMAGE_BITMAP,IMAGE_CURSOR,IMAGE_ICON"/>
<Flags type = "checklist" value = "ILC_COLOR"
cvalues="|ILC_COLOR16,|ILC_COLOR32,|ILC_PALETTE,|ILC_MIRROR,|ILC_PERITEMMIRROR,|ILC_ORIGINALSIZE"/>
<ImageList type = "cntrlist" value ="NULL" cvalues = "ImageList" />
```

</uservars>The closing Tag

<drawing>This Tag MUST be present.

```
<controlframe width = "80" height = "15" const = "none" bgcolor = "f0f0f0" linecolor = "adadad" border = "0"/>
```

The **<controlframe>** Tag is responsible for the initialization of the control.

width attribute MUST be present.

height attribute MUST be present.

const attribute MUST be present.

This attribute declares that some of the dimensions of the control will be constant , and the user will be unabled to change them. The valid values that this attribute can have are :

none

This value has priority over every other value , except **absolute** and it means that the control will be freely manipulated

Powered by 000webhost

top

The control will being autosized in width and it will position itself at the top of the frame. This value can be combined with the height value.

bottom

The control will being autosized in width and it will position itself at the bottom of the frame. This value can be combined with the height value.

left

The control will being autosized in height and it will position itself at the left of the frame. This value can be combined with the width value.

right

The control will being autosized in height and it will position itself at the right of the frame. This value can be combined with the width value.

dimensions

The control has all of its dimensions locked and can be only moved . This value has priority over the other values except **absolute** and **none**.

absolute

The control is an immovable and not resizable control. This value has absolute priority .

width

The control will have his width locked.

height

The control will have his height locked.

You can combine the values like this : **const = "bottom,height"**

Note that you must be carefull when combine values for having the desirable outcome.

bgcolor attribute MUST be present.

This atribude is the background color of the control. The value is a hexadecimal value.

linecolor attribute MUST be present.

This attribute is the color of the border line of the control. The value is a hexadecimal value.

border attribute MUST be present.

This atribute determines when the control will have a border or not. The correct values are 0 for no border and 1 for border. Other values grater than one will make the border to being drawn weird and values smaller that 0 are undefined.

<custom>This tag MUST be present.

Inside this tag you can declare drawnable elements. This elements will be drawn inside the client area of the control.

<element x = "1" y ="0" width = "0" height = "0" bgcolor = "e1e1e1" transparent = "true" linecolor = "adadad" border = "0" valign= "center" halign = "none" style = "rect" pic = "" autosize = "false" >$Text$</element>

<element> This Tag MAY be present.

This Tag can have the following attributes

x,y,width,height

This attributes MUST be present and must be set. They can have an negative value , useful for when the **autosize** attribute is set to true.

bgcolor

This attribute MUST be present and must be set . This attribute is the background color of the element, its value is a hexadecimal value.

linecolor

This atribute MUST be present and must be set. This attribute is the color of the border , if exists.

border

This attribute MUST be present and must be set. This attribute can have values from 0 to any valid number for an integer type.

The border line will be thick as this value. Negative values are undifined.

transparent

This attribute MUST be present. This attribute can have two values , true or false. If the value is true , then the element will not being drawn with a background and it will be transparent. If the attribute value is false or the attribute is absent , the element will being draw with his background color .

valign

This attribute MUST be present. This attribute can have the following values : **center** , **top** , **bottom** , **none**. The values are self explanatory and if one of them is set the element will align itself acordinaly .

halign

This attribute MUST be present. This attribute can have the following values : **center** , **left** , **right** , **none**. The values are self explanatory and if one of them is set the element will align itself acordinaly .

style

Powered by 000webhost

This attribute MUST be present. This attribute can have the following values : **rect** , **line** , **image** .

The rect value tells the control to draw the element as a rectangle with the top,left,width,height attributes as dimensions and position values.

The line value tells the control to draw the element as a line with the top,left to have the values for the start point , and the width,height to have the values for the end point.

The image value tells the control to draw an image in the specific top,left coordinations. The width and height must match the image dimensions , and the image it self is the value of the **pic** attribute .

pic attribute

This attribute MAY be present.This attribute is valid only if the **style** attribute is **image.** Its value must be the name of a png that it resides inside the **control folder of the Axolotl.** The png must not be inside another folder.

autosize

This attribute MUST be present. This attribute can have one of this two values : true or false. If the value is false the element is drawn normally. If the value is true,the element have the same size as the control. Note that if the autosize attribute is true , if you have set values to x,y,with,height this values will be calculate and the actual width and height of the element may vary . e.g.

```
<element x = "12" y ="5" width = "12" height = "-5" bgcolor = "e1e1e1" transparent = "true" linecolor = "adadad" border = "1"  valign = "none" halign = "none" style = "line" pic = "" autosize = "true" Visibleif =" $Orientation$=Vertical" ></element>
```

In the above example , the autosize is true but the x attribute says that the line will start from the 12pixel of the control on x axis, from the 5th pixel in y axis , the x of the second point it will be in the 12th pixel (the type is line) and the height will be as the control height -5.

Try to expirement to understand the versatility of this attributes.

Visibleif

This attribute expects a condition as a value and if this condition is true , then the element will being drawn. In the above example the **Orientation** is a uservar that is declared in the **<uservar>** section. So if the Orientation equals the string Vertical in some point inside the Axolotl designer , this element will be made visible , else this element will not being drawn.

Note that in the condition you must use only one expresion , only with the equal sign e.g the following is forbidden : $X$>0 ,
and you can use only **uservars** , **systemvars** (we will talk about them later) and literals. To differentiate between the literals and a variable ,
the name of the var(iable) must be enclosed by the $ sign e.g the top is literal the $top$ is a var .

</custom>The closing Tag

</drawing>The closing Tag

<code>This Tag must be present.

<create>This Tag must be present.

<proc header = "" call = ""> This Tag must be present even if it's empty. This is the controls creation function.

Inside this tag you can write code for the initialization of the control. The Axolotl will take this code and it will encapsulate it inside a function that the program will call inside the InitControls function .Inside the code you can use the systemvars and the uservars as you can see fit.

The uservars and the systemvars can be used if you enclose their name inside $ e.g **$realparent$** .

SystemVars

left , top , width , height

This vars have the values for the position and dimensions of the control .

parent

The parent of the control (the parent inside the Axolotl Designer)

realparent

The realparent of the control , the parent that the control will have inside the C/C++ code .

CID

The unique identifier of the control .

name

the name of the control. This is also the name of the control handle that the Axolotl will create inside the Code.

e.g **HWND edit1 ;**

frame

The name of the frame that the control reside , e.g mymainwindow . This is also the name of the handle of the window.

Powered by 000webhost

```
 $name$ = CreateWindowEx(WS_EX_CONTROLPARENT, // with out this the IsDialogMessage cause the program to freeze .
                L"STATIC",  // Predefined class; Unicode assumed
                L"$Text$",      // Text
                WS_CHILD | WS_VISIBLE | SS_SIMPLE ,  // Styles
                $left$,         // x position
                $top$,          // y position
                $width$,        // width
                $height$,        // height
                $realparent$,      // window
                (HMENU) $CID$, //ID
                (HINSTANCE)GetModuleHandle(NULL),
                NULL    // Pointer not needed.
               );

if ( $name$==NULL )
{
  MessageBox($frame$,L"Handle Error !" ,L"Error",MB_ICONERROR );
  return ;
}

HFONT hFont = (HFONT)GetStockObject( DEFAULT_GUI_FONT );
SendMessage( $name$ , WM_SETFONT, (WPARAM)hFont, 0);
```

</proc>The closing Tab

</create>The closing Tab

<extracode>This Tag MUST be present even if its empty.

<proc header = "int TBStrLen(const WCHAR * str)" >This Tag CAN be present as many times it is neccesary.

This Tag is a function that the C/C++ code will have. This function can be called inside the control creation function and inside the C/C++ code.

header attribute

The header of the C/C++ function .In the body of the tag you can write any code you want , note that this functions are not per instance , this means that you cannot use the uservars or the systemvars. Your code inside this tag has nothing to do with the control. It's just pure code. When the Axolotl exports the C/C++ files , this functions will be present only once in the files even if multiple control instances are present in the project.

```
int cnt = 0;
if (str[0] == 0) return 0 ;
do
{
  cnt ++ ;

}while( str[cnt] != 0 );

return cnt ;
```

</proc>The closing Tag

</extracode>The closing Tag

<initcode>This Tag MAY be present. This Tag gives the capability to create functions per control instane.

<procs>This Tag MUST be present. Inside this tag you can create as many functions you need.

<proc header = "void BDCreateBand$name$()" > This Tag Can be present. You can use any uservar or system var in the header, the correct way is to use the name of the component for the header. In this tag you can write any code and use any uservar or systemvar .

Powered by 000webhost

```
ULONG flags = RBBIM_STYLE | RBBIM_ID ;

REBARBANDINFO rbBand = { sizeof(REBARBANDINFO) };

rbBand.fStyle = $Style$;
```

</proc>

</procs>The closing Tag.

<plaincode>This Tag MAY be present .Inside this tag you can write code that can use the uservars and the system vars. As a result you can call the functions from the **<procs>** tag .This code is executed after the control creation code.

```
BDCreateBand$name$();
```

</plaincode>The closing Tag.

<children> This Tag MAY be present. In this Tag you can write code that can use the systemvars , the uservars and a special variable , the **$child$**. The **$child$** variable has in its value the name of the current child. The code that you will write in here , will be repeated for every child that this parent have. (The control must be a CONTAINER ofcourse).

```
PGSetChild($name$ , $child$ );
```

</children>The closing Tag.

</initcode>The closing Tag.

</code>The closing Tag.

</control>The closing Tag.

This summarize the most crucial elements of the control xml file. If you really want to create a control file , please study the control files that already the Axolotl have. If you create a control that you think is really usefull please share it with us!

CONTACT US

- **@ELSE-RETURN0**
- **@NIKOS MOURGIS**

Donate

A small donation could help me invest more time sleepin... errr in my free projects!

Donate

Powered by 000webhost