Main Menu

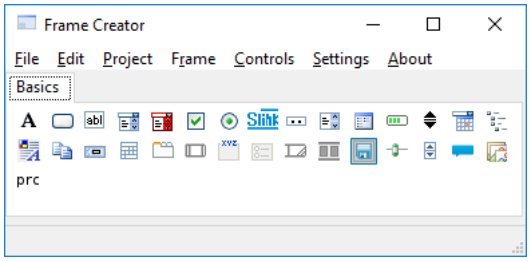
Main Menu

Operation of The Axolotl

Axolotl can operate in the follow Three modes.

The Stand Alone mode :

In Stand Alone mode the program has the following form :



The operations of this mode is as following :

Create a project.

Create frames and populates them with controls.

Save and Export the project .

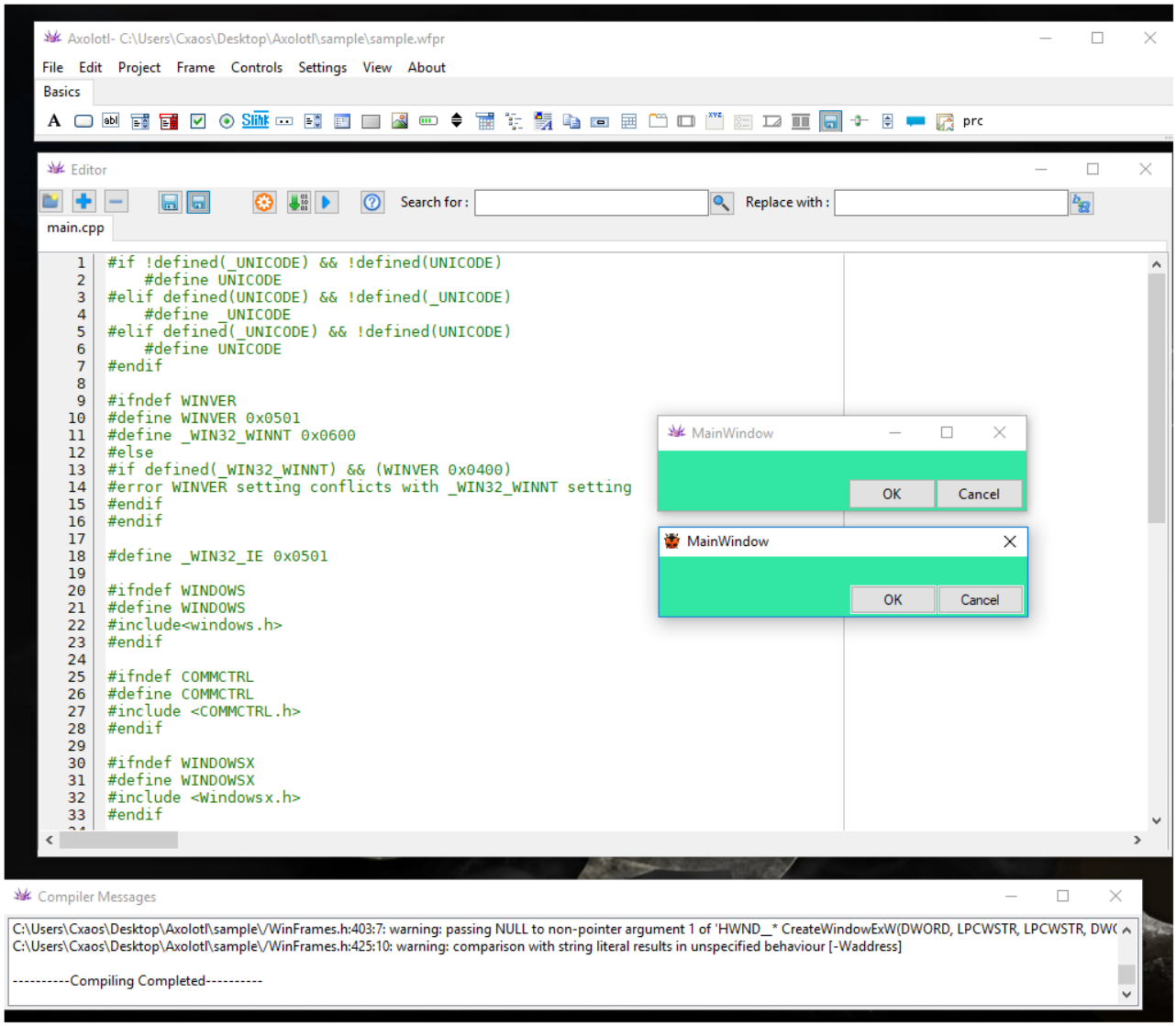
Open your favorite C/C++ IDE (i use as an example in this documents the Code::Blocks) create a new Win32 project , remove the default main , add the files that the Axolotl exported in the project. Start coding inside the wfCallbacks.h file. (see below for the exporting files and their uses) .

If you change something in the frame e.g the position of a control or the size e.t.c then you can Update All Frames Code (see below for this menu command) and the changes will be update the C files with out touching your files or the wfCallbacks.h file. Thats mean that you can change the interface of your program with the Axolotl and your hand made code will be safe.
NOTE : **If you delete a control , then you must delete manually all the references of the control inside your code (not the code that the Axolotl produces but your personal code) the same applies when you delete a frame. All of frame references inside your code must be deleted manually. In the case of a frame though , you must delete manually its callback routines too. Read the help files to understand further the way that Axolotl manages its code and the user code .**

The draw back of the Stand Alone mode is that you have to swap between the two programs (the editor and the Axolotl) every time you want to change something in the interface.

The Portable Mode :

In this mode the Axolotl will display a simple editor and some simple tools for writing and compiling code with the MinGW compiler.



As you can see from the above image , the Axolotl will have an editor and an output window and not only the main window. The use of this windows are very simple and it will being demostrated in the following paragraphs :

The Controls Window :

This window has all the basics win32 controls. For the use of them , read the other help files.

The Editor :

This is the simple editor that the Axolotl provides.

Features

Auto complete : The editor can show a list of all the winapi functions (and the custom functions of the Axolotl) for the user to choose with the ctrl+space combination.

Compile : The compile button can compile the current project and store the produced executable inside the "exe" folder in the project folder.

Run : The run button runs the executable of an allready compiled project and pass it as parameters the parameters that they are set in the configuration.

Configuration : This button opens the configuration form . In this window you can alter the :

"Extra Commands" field where you can add or remove switches for the compiler.

"Linking Librarys" You can add or remove librarys for the linker to link. Keep in mind that the librarys must be inside the "library path".

"Static Linking" If this is checked then the executable will be stattically linked with the libraries.

"Executable param line" The line in this field will be passed as a parameter line to the executable if the run button is used.

The Message Window :

This window display the messages from the compiler. In case of an error or warning you can double click in a message and the focus of the editor will transfer to the appropriate line. Note that

for the editor to focus to the correct line , you must double click the message in the message window when you are focusing a tab of the editor that it actually has the file that the line resides. For example , if you

double click to a message that has an error in the file win.cpp line 32 then you must have open in the editor the win.cpp file and have its tab (the editor has multi tabs) in focus.

In case you want to copy the messages then with right click you can select to copy one line or all the messages.

NOTE that the Axolotl in portable mode has not an embedded debuger.

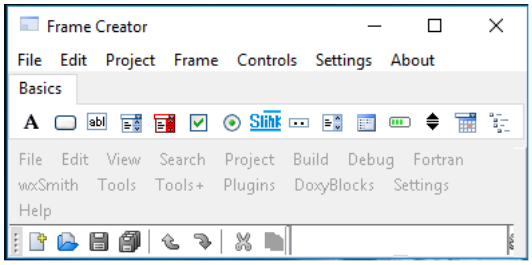
The portable form of the Axolotl it is the natural form of it. You can run it from any removable media and write any code in c++. The Axolotl use the MinGW compiler to compile and link an executable. The Axolotl is always in portable mode ,but for the editor and the message window to appear with the program start , the Set c++ Enviroment option in Settings menu, must be <NONE> . Else the Axolotl will open the external c++ enviroment and not his own. Of course you can always desplay the editor and message window from the "View" menu.

IMPORTANT For Axolotl to be able to compile your project , you MUST set the main project file (usually the main.cpp that the Axolotl generate but can be any *.cpp file that have the a Main() function) . Set the main project file in the "Setting" menu with the "Set Project MAin File" option.

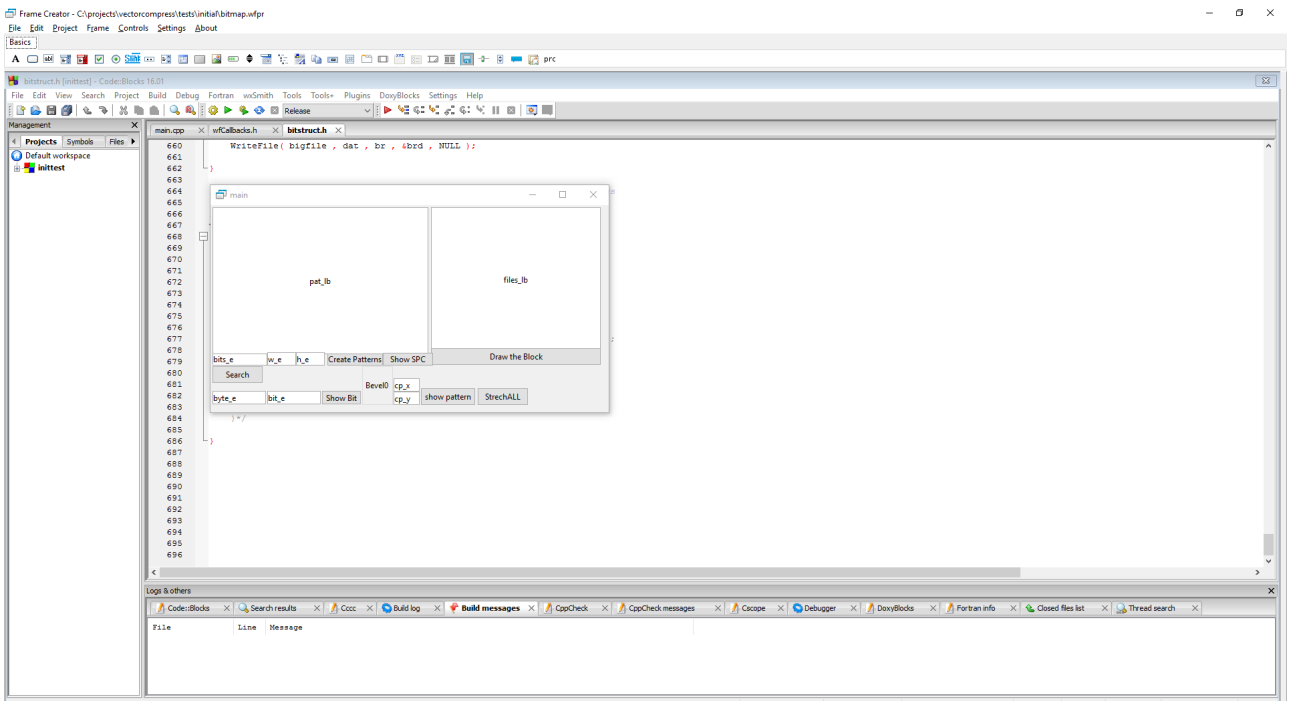
All third party programs are property of their correspondent owners.

The Embeded mode :

In embeded mode the program has the following form (if Code::Blocks is used) :



And the following form after a resize :



Now the Axolotl have in its client area the editor , and that makes the use of the program more flexible.

The draw backs of this mode is that (as with the tests i have do with Code::Blocks) , the user can be confused with the two sets of menus , and while saving the Axolotl project it will forget to save the change in code in the Code::Blocks.

Note that the window of the Code::Blocks have a system button (the x button in the right top corner) so you can close it if you want. If you close the Axolotl the Code::Blocks will

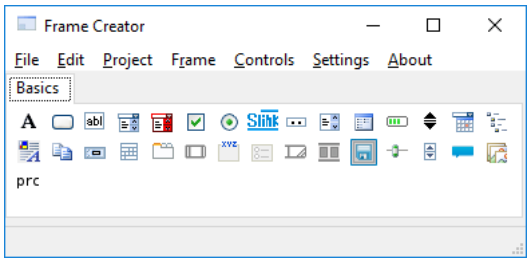
stay open as stand alone program , and you must to close it manually.

Now that you know the threedifferent modes of the Axolotl , lets see how we can use it and create an interface .

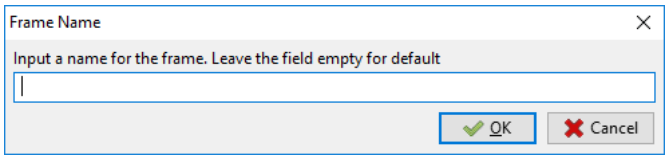
Starting the Program

When the program is started for the first time , it opens in the default state that is the portable mode.

As you can see the program display all of its controls in one horizontal line, for rearreanging the controls when the Axo;ot; window change size , go and check the Settings -> Resize Tabs Control Height. The result is the following in case that you have resize the Axolotl window as this :



Creating a new Frame is easy as clicking the menu Frame -> New Frame
The follow dialog will appear :



You can input a valid frame name (valid for C/C++ standard) or leave this field empty ,the program will create a default name such as "Frame0" or "Frame125" The new frame will have the default dimensions, it will have as title the frame name and it will being devoid of any control or menu .To export this project as C/C++ code go to the file menu and select **Save And Export**.
If you already have saved the project then the exported files will be reside inside the folder that the project is saved , if you do not have saved the project before , a save dialog will appear for the project to be saved .The C/C++ files that will being exported are the following :

main.cpp
WinFrames.h
wfCallbacks.h
resources.rc

Of this files the user can edit ONLY the main.cpp and the wfCallbacks.h files. This two files are not being accessed from the program after their initial creation so its safe to put code in there with the certainity that the Axolotl will not change their contents when the interface changes .

Analysis of the files :

main.cpp :

The main.cpp is the main C/C++ file that has the main() function for the program to start. In this file all the initialization routines of the interface is being called .

WinFrames.h :

This file is updated with every **Update Code** command from the Axolotl (see bellow for the menu commands) DO NOT put any code inside this file EXCEPT that are you sure that you will not use the Axolotl to change the interface anymore. Any code that the user wrote inside this file will be replaced with the new code that the Axolotl will generate.

wfCallbacks.h :

In this file are the callback functions that the frames (windows) will use . You can edit this file as you can see fit , moreover a function with the name InitCode() {} exists in this file, that is for the user to put any additional code that it wants to execute after the initialization of the interface , like to change dynamically some positions or to add some items in a listbox e.t.c. Of course nothing stops you to declare other functions or create other .h and cpp files to use .

WARNING : This file will be automatic edited to the follow situations :
If the File->Save And Export menu command being used or the Project->Export Code menu command,if thats the case the file will be reset. If a new Frame will be created, a new callback function for this frame will be added. In that case the file will not being reset, it will being merely edited , so the user code will not being lost.

Attention : If a frame has its name being changed while the C/C++ files have already being exported , then the user must manually change the name of the callback function inside the wfCallbacks.h file. For example if a frame with a name "Frame0" is being changed to "MainWindow" then the user must find the Frame0wclbproc and rename it to MainWindowclbproc. This is aplying generally , e.g. if inside your code (not inside WinFrames.h this file is updated by the Axolotl) there is instances of the previous name , they must be changed to the new one.

resources.rc :

This file is a resource file , inside it are the bitmap , icon or string resources , this file is edited inside the Axolotl , DO NOT edit this file outside of the Axolotl , EXCEPT if you are positive that you will not use the **Frame->Update Code** any more .

Run the exported code (Code::Blocks example)

Open Code::Blocks , go File->New->Project...Choose Win32GUI project , In the next screen choose Frame Based (if the screen is empty press next and the following one will have the option) In the next screen type the project title and choose the folder to create the project in (its advisable that the Code:Blocks project are being saved in the same location with the Axolotl project)Press next and finish the new project wizard . In the project tree to the left you can see the new project that you created. Open the tree and in the "source" folder you will see a "main.cpp" file remove it from the project .Now do a right click to the project and select the option "Add Files..." An open dialog window will open. Select the main.cpp , WinFrames.h , wfCallbacks.h and resources.rc that you have export and click open. The Code::Blocks will add the files in your project. Save the project , compile and run it. That's All!

Attention : To avoid the annoying messages of the Code::Blocks every time that the Axolotl change a file , (File (...) is modified outside the ide ...) do not have open in the Code::Blocks IDE the WinFrames.h and the resources.rc.

Menu Commands

File

New Project	Creates a new empty project .
Open Project	Opens a project .
Recent Projects	A list with the three latest projects that were opened .
Save Project	Saves the current project .
Save Project As	Saves the current project with a new name .
Save and Export	Saves the current project AND exports the C/C++ files. **The previews exported files will be overwriten and all the changes and the user w command its designed to used only once in a project , or when the user has not write any code.**
Close Program	Closes the program .

Edit

Undo	Undo the previous action if its one of the following : Moving a control , resize a control , delete a control . This is the only actions that the undo c
Copy	Copy the selected controls .
Paste	Paste the controls .
Delete	Deletes the selected controls .

Project

Set Project Name	The project name , this is the name that your executable will have if you are running the Axolotl in portable mode.
Set Project Icon	Sets the project icon that will being the icon of the executable .
PopUp Menus	Shows the dialog that the user can use to create popup menus for the application .
Edit Resource File	Shows the editor for the resource file .
Export All Code	Same as the Save and Export command menu except that it will not save the project .

Frame

New Frame	Creates a new empty frame
Clone Current Frame	Clones the current frame (creates a new frame with the same controls as the current one)
Grid	If this option is checked, then the frame has a grid .
Frame List	Shows the dialog that enumerates the frames and the controls in the project.
Update Code	Click this command for updating the changes of the frames in to C/C++ files .

Controls

Show Properties	Shows the control properties window .

Settings

Resize Tabs Control Height	If this option is checked , the controls will wrap and will be displayed in multilines.
Set c++ Enviroment	Shows the dialog that the user can type , the path of the enviroment that the Axolotl will embed in his own client area .

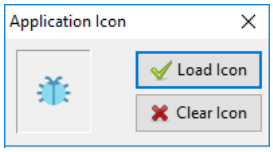
About

Info about the program !

Dialogs Specifications

In this section all the important dialogs of the Axolotl will be explained .

Project -> Set Project Icon



This dialog loads an icon for the application.

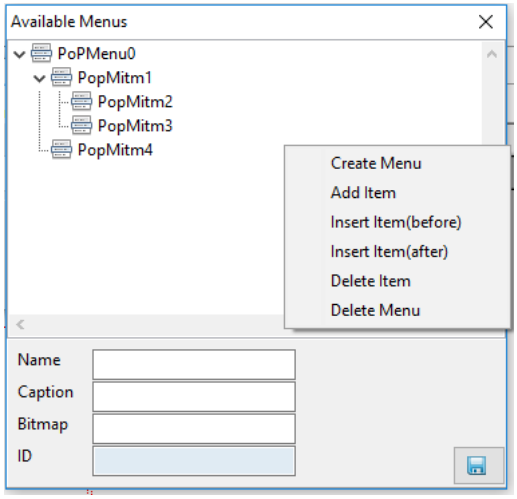
If an icon is set (as in the example) then the Axolotl will insert an icon inside the resources.rc file with the **ID 201**.

Actual lines :

#define ID_MAIN_ICON 201

ID_MAIN_ICON ICON "c:\..\myicon.ico"

Project -> PopUp Menus



This dialog is for creating PopUp Menus .

With right click the menu for the creation is being visible , as the image shows. After the successfull creation of the items you can change the following properties :**Name**This is the name of the item , the name is used for naming the HWND variable inside the code (if its necessary) and for creating the unique control ID .e.g if you named an item "CloseFile" then inside the C/C++ code a variable will be created as : HWND CloseFile ;

(Warning! Changing the name often may result to inconsistencies inside the user code , resulting to a need for constant corrections in the user code . If it is mandatory to change the name , do it after the creation of the item and before use it inside the user code !)

Caption : The text of the menu item **.**

Bitmap : The Bitmap that the menu may have . This value is an ID that is already set inside the resources.rc. For example , if in the resources.rc you have set the following value :

#define ID_BIT1 202

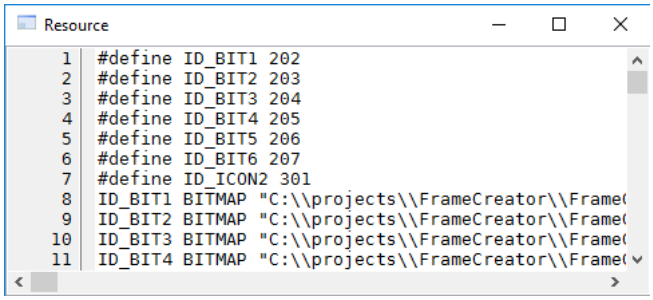
ID_BIT1 BITMAP "C:\\progressbar.bmp"

And you have set the Bitmap property of the menu item to 202 then the menu item will have the progressbar.bmp bitmap as image .

ID : This is a read only property , this property is the menu id that the menu sends to the windows callback when it is clicked .

Do not forget to press the save button in every change that you do!

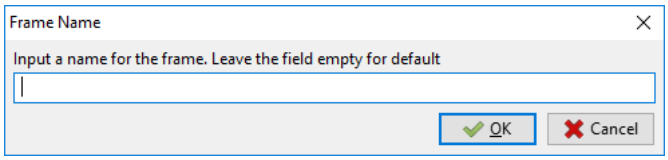
Project -> Edit Resource Files



A simple editor for editing the resources.rc

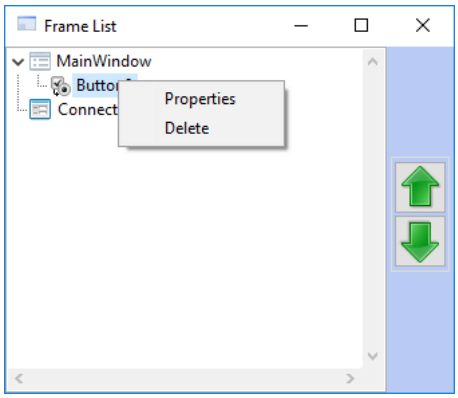
Note : The Axolotl is starting the count for the ID' s of its controls from the 300 and counts.

Frame -> New Frame



If you type a frame name the new frame will have this name , if you leave the field blank then the new frame will being assigned a default name.

Frame -> Frame List



This dialog shows all the frames in the projects and their controls.

With right click on a item the menu with the options Properties and Delete will appear . (**notice that the only way to access the properties of a frame is by this menu**) With doubleClick the Frame will be appear if its not visible.

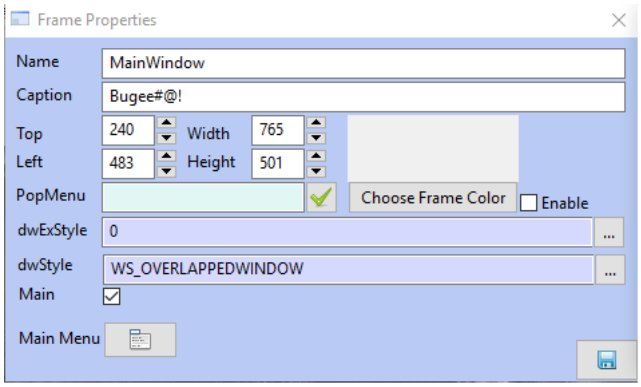
ATTENTION ! The up and down arrows at the right size of the dialog , is being used to change the order of the controls inside a frame.

THIS IS VERY IMPORTANT!

The order of the creation of the controls inside the C/C++ file is relevant to this order in the Frame List. If a control is dependant to another control then the control that must be created first , MUST be ABOVE of it in the Frame List. In the most of the cases the natural order of creation is enough to assures that every control will be created in the right order . But in rare cases , like when you add an Image List to the Frame, you must manually set the control at the top of the list. For example, If you have a ListView and suddenly you decide that this ListView will be cooler with images in it , you can add an ImageList and then set this ImageList to belong to the ListView. All will seem to be okay. But in reallity in the actual code, the image list will be created after the creation of the ListView , so the ListView will silently fail to add the ImageList. The solution is to simple move the ImageList on top of the ListView and preferably to the top of the list.

Moreover , if you add the WS_TABSTOP flag to your control , then the control will have a tab order. This order is relevant to the creation order. First come , first serve!

Frame List->Properties



This dialog is beign show when the user selects the menu command **Properties** when it has select a frame (in the Frame List Dialog).

Properties :

Name The name of the Frame .

Caption The text in the title bar of the Frame .

Top , Left , Width , Height Position and dimension of the Frame. This properties can be changed directly from draging the Frame and its border .

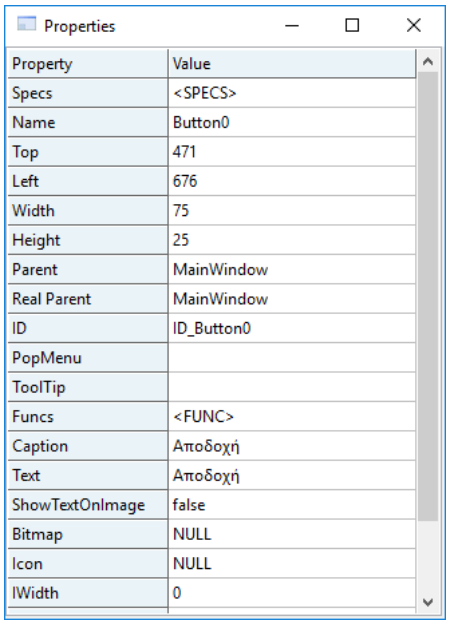
PopMenu With the press of the button right of the property , a dialog with the available PopUp menus will be show. This selection will be the PopUp menu of the Frame.

dwExStyle , dwStyle With the press of the button right of each of this properties , a dialog with checkable flags will appear for selection. The dwExStyle is the ExStyle for the window creation in C/C++ code , and the dwStyle is the Style for the window creation in C/C++ code.

Main If this is cheched the Frame is the main window of the Application. This is the only Window that the Axolotl will set automatic to open at the startup of the application (except if you have select the WS_VISIBLE flag in the dwStyle), and all the other Frames will have this window as their parent. This window will also call the PostQuitMessage(0) when it will receive the WM_DESTROY message .

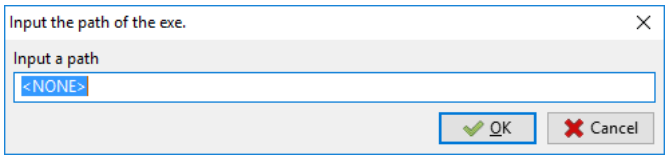
Main Menu Opens a dialog that you can create and configure the main menu of the Frame. (the dialog is similar with the one for the PopUp menus)

Controls -> Show Properties



The properties of the selected control. The properties sheet can show only the properties of one conrol at the time. In a multiselection the properties sheet wil be blank. **<SPECS>**Pressing the button in this option, the specification of the control will be shown .**<FUNC>**Pressing the button in this option , the unique functions of the control will be shown . With double click or right click you can copy the function that you want and you can paste it in the code of the your editor .

Settings->Set c++ Enviroment



If you set this to <NONE> or to blank the Axolotl will open his own editor window and compiler message window. If however you set a path to a program , then the Axolotl will open this program inside its client area after restarts. For example you can set the following string to open Code:Blocks: C:\Program Files (x86)\CodeBlocks\codeblocks.exe

How To Use Axolotl

Start the program .

Now select Frame->New Frame and press OK. Click on the second control in the Basic Tab (its tooltip says Button) and then click anywere in the frame or drag the mouse to create a rectangle. Release the mouse button and a simple button will appear in the frame.

Like this you can create any of the controls.

Note : The right click of the mouse is the absolute cancel , it cancels the current action.The keyboard arrows , can move a control very precisely .The delete button deletes the current selection . ctrl + V , ctrl + C and ctrl + Z are for paste , copy and undo respectively .

Attention!

When you are trying to put controls inside a container like the GroupBox or the Panel , pay attention that the creation rect that you are drawing with the mouse, is inside the bounds of the container. If you fail to do so , or if when you release the mouse , the cursor is not inside the client area of the container (even if its upon a child of the container) then the new control will be beloning to the Frame and not to the container. That may lead to strange behavior after compiling your application. If you experience strange position behavior of some control , pay attention to its parent property in the control properties sheet to be

manifestate itself if you are going to paste some controls and you have not select the right container.
In this situation the controls will belong to the frame and not to the container .

If you want to have a nice interface do not forget to enable visual styles . This can be achieved easly; if you insert the following line in the resource file : 1 24 "Visual.Manifest" and copy the file visual.manifest that you will find inside the folder of the Axolotl inside the folder of the exported files , or somewere that your compiler can find it!

WARNING!

When you create the frames , choose a name wisely; if you decide later on the development of your program that you want to change the name of a frame (a window) then be sure to :

1. Rename first the callback function of the window and

2. Rename the controls callback function of the window.

Example : If your frame is named **mainw** then inside the wfCallbacks.h you must have a

LRESULT CALLBACK mainwclbproc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)

and a

void mainwMessageProc

function. If you change the name of the frame to **secondw** then the functions must be renamed as :

LRESULT CALLBACK secondwclbproc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)

and a

void secondwMessageProc

If you do not change this two functions , then when you do the **Update All All Frames Code** operation , the FC will think that there is not the correct functions for this frame and it will recreate it with the proper naming. In this situation you must delete the two new created functions, and procceed to rename the old ones as depicted above.

CONTACT US

@ELSE-RETURN0

@NIKOS MOURGIS

Donate

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