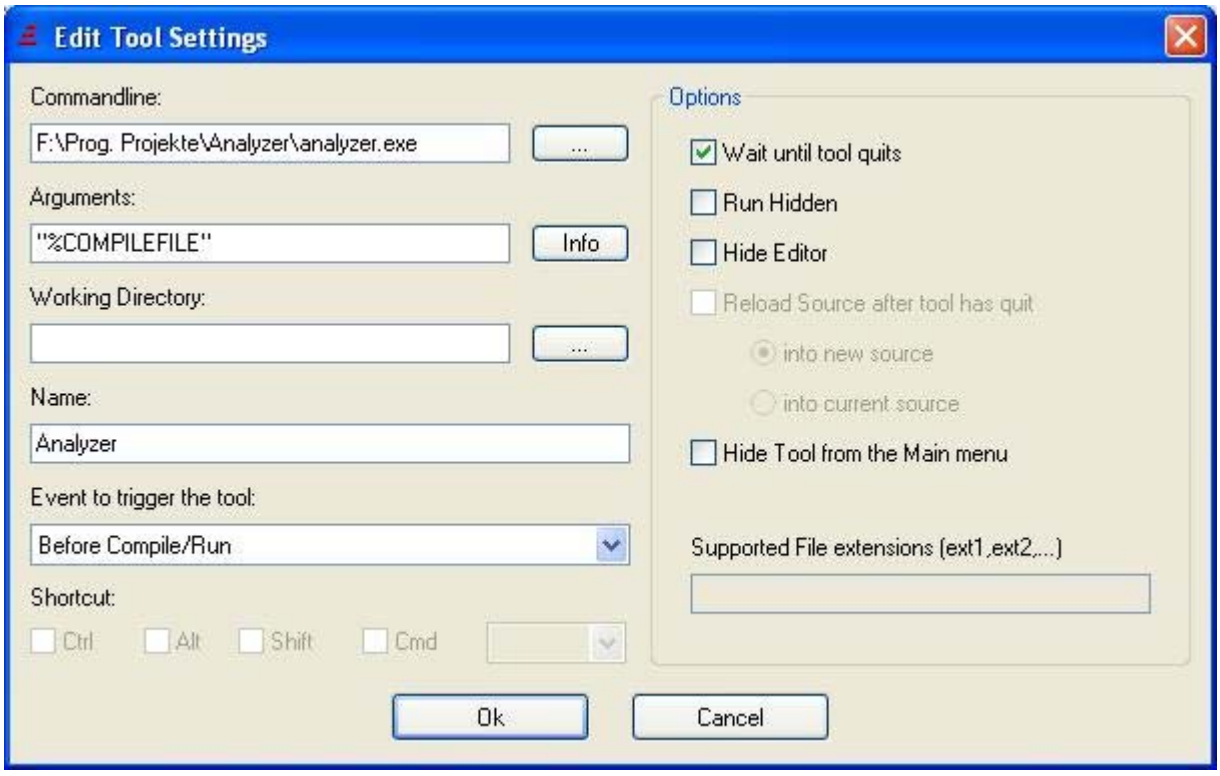
02.07.2005
E-Mail: remi_meier@gmx.ch
Website: http://www.remimeier.ch.vu
Website2: http://www.recursivemess.de.vu

Installation

Analyzer

It's an IDE plug-in for PB v3.94 but it should also work as a jaPBe plug-in and stand alone program. Just drop a temporary PB file on Analyzer.exe and it will be modified!

So, add a new tool like described in the official help file for the PB-IDE!
The **command line** points to the analyzer.exe
Arguments look like this: "%COMPILEFILE" (note that the " must be written!)
Name is what you want ;)
Event is Before Compile/Run
And don't forget **Wait until tool quits**!

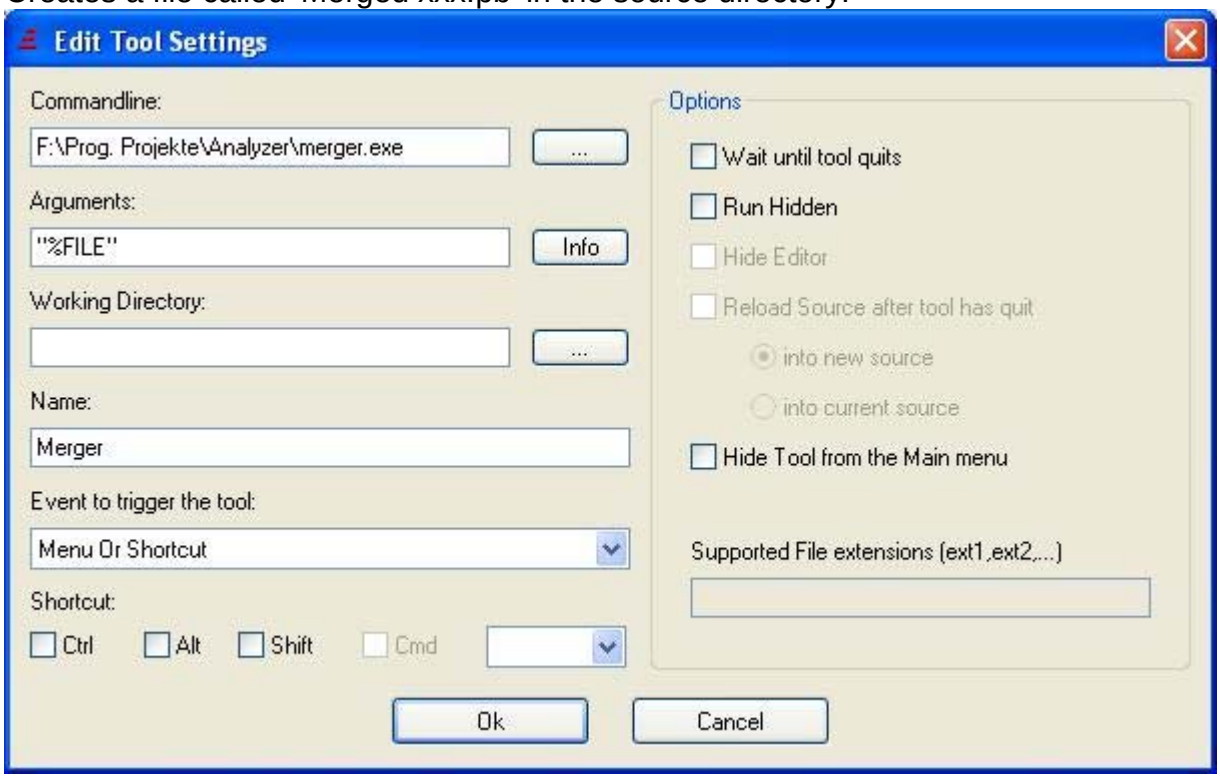


Now compile/run something ;)

Merger

It's an IDE plug-in for PB v3.94 but it should also work as jaPBe plug-in and stand alone program. Just drop a temporary PB file on Merger.exe and it will be merged!

Creates a file called 'Merged xxx.pb' in the source directory!



WHY a Merger tool??

Because the Analyzer cannot modify the include files, this tool will merge all include files together with the main file! So a whole project with many include files will be merged to ONE big file!

ATTENTION:
Lines like
Debug 1 : IncludeFile "xx.pbi"
will now be recognized but 'Debug 1' will be removed! The whole line will be removed!

In lines like
IncludeFile "xx.pbi" : IncludeFile "yy.pbi"
only the first IncludeXX will be recognized and the whole line will be deleted!

I would be very thankful if somebody would write a better tool, but for me it stops here! It should be a help tool for the Analyzer and that's what it is! I won't write a whole PB parser

Information

The Analyzer only modifies the currently compiled file WITHOUT its include files! If you want them to be included in the analysis, you have to merge them together to **ONE PB file!**

When the Analyzer could correctly modify the compile file, at the start of the program a message box should appear ("Analyzer", "Started")

If you want to know how it works, look into the "PB_EditorOutput.pb" of your compilers directory!

Outputfile

The Analyzer creates an "analysis.txt" in the directory of your source file.
The **line number** ('line') indicates the line in your code!
'**total time**' is the accumulated time this code line used during the run of your program (only times greater than 0 are shown).
'**percentage**' is the time divided by the total runtime of your program.
'**time per call**' is an average value that indicates the time this line/block took in one call.
'**called**' shows you how many times this line/block was called!
The last column is the **code line** itself!

ATTENTION:
The compile time can extremely increase and also the start up time!

Special Comments

Pause - Continue

You can stop analyzing for blocks i.e. ASM with 2 comments:
; pause
... ASM stuff
; continue
Take a look at “Example pause-continue.pb”

ATTENTION:
If you use ASM-Code outside of “; pause” and “; continue”, your code probably won't work anymore!

Blocks

If you want to test how long a whole block of code takes (like loop or procedure), you can use
; block [name]

... loops, code, ...
; endblock
to start a time measurement on your own (put in your own name for [name])!
You can also nest these comments like in “Example blocks.pb”.

Turn it on/off

To turn the Analyzer on or off, just select the Analyzer over the menu *"tools"* and it will change its state.

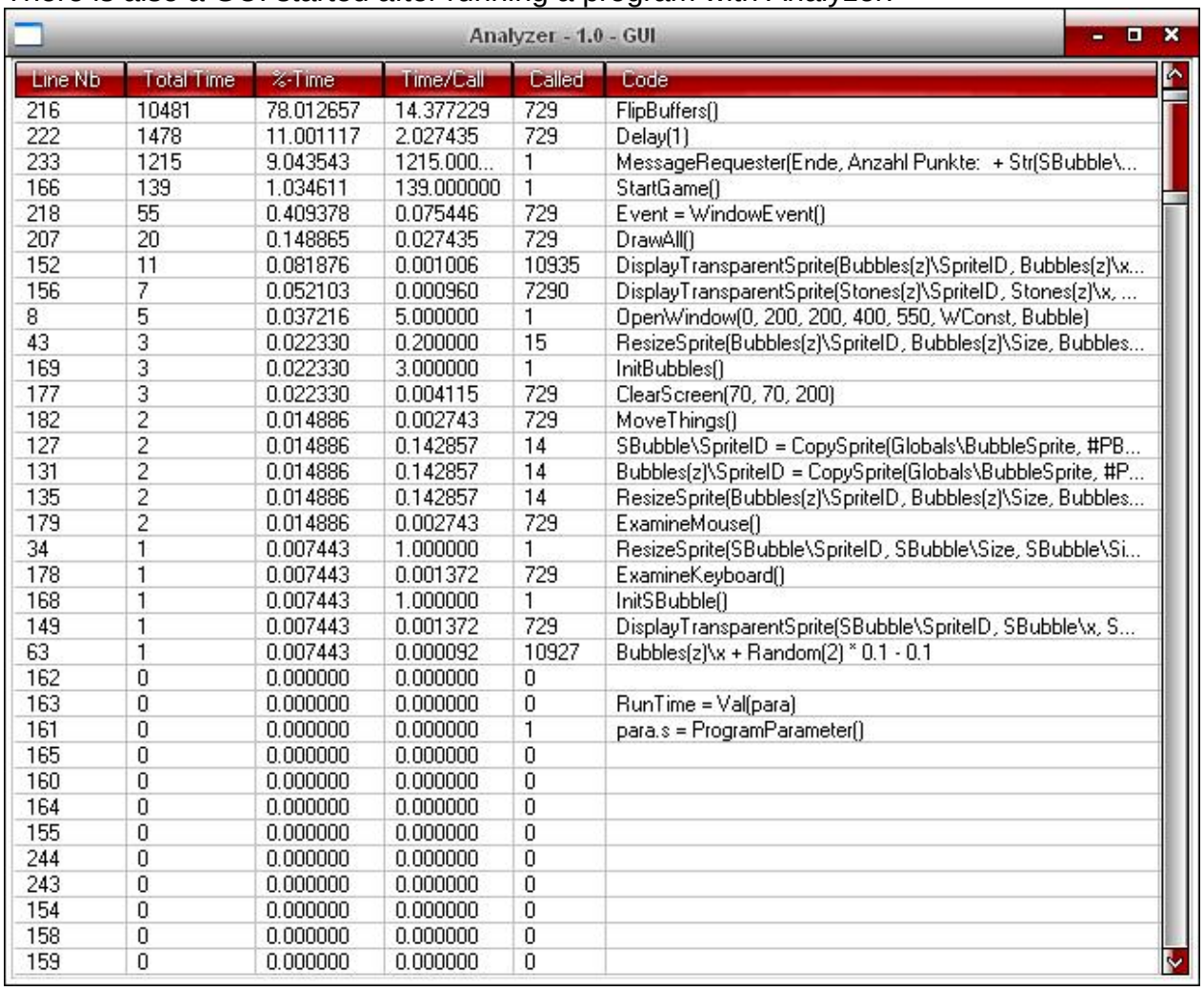
Output tips / Analysis.txt / GUI

Analysis.txt

After a run with the Analyzer, a file called “analysis.txt” will be created in the current folder of your application. It is a plain text file you can read with every editor.

GUI

There is also a GUI started after running a program with Analyzer.



Line Nb	Total Time	%-Time	Time/Call	Called	Code
216	10481	78.012657	14.377229	729	FlipBuffers()
222	1478	11.001117	2.027435	729	Delay(1)
233	1215	9.043543	1215.000...	1	MessageRequester(Ende, Anzahl Punkte: + Str(SBubble\...
166	139	1.034611	139.000000	1	StartGame()
218	55	0.409378	0.075446	729	Event = WindowEvent()
207	20	0.148865	0.027435	729	DrawAll()
152	11	0.081876	0.001006	10935	DisplayTransparentSprite(Bubbles(z)\SpriteID, Bubbles(z)\x...
156	7	0.052103	0.000960	7290	DisplayTransparentSprite(Stones(z)\SpriteID, Stones(z)\x, ...
8	5	0.037216	5.000000	1	OpenWindow(0, 200, 200, 400, 550, WConst, Bubble)
43	3	0.022330	0.200000	15	ResizeSprite(Bubbles(z)\SpriteID, Bubbles(z)\Size, Bubbles...
169	3	0.022330	3.000000	1	InitBubbles()
177	3	0.022330	0.004115	729	ClearScreen(70, 70, 200)
182	2	0.014886	0.002743	729	MoveThings()
127	2	0.014886	0.142857	14	SBubble\SpriteID = CopySprite(Globals\BubbleSprite, #PB...
131	2	0.014886	0.142857	14	Bubbles(z)\SpriteID = CopySprite(Globals\BubbleSprite, #P...
135	2	0.014886	0.142857	14	ResizeSprite(Bubbles(z)\SpriteID, Bubbles(z)\Size, Bubbles...
179	2	0.014886	0.002743	729	ExamineMouse()
34	1	0.007443	1.000000	1	ResizeSprite(SBubble\SpriteID, SBubble\Size, SBubble\Si...
178	1	0.007443	0.001372	729	ExamineKeyboard()
168	1	0.007443	1.000000	1	InitSBubble()
149	1	0.007443	0.001372	729	DisplayTransparentSprite(SBubble\SpriteID, SBubble\x, S...
63	1	0.007443	0.000092	10927	Bubbles(z)\x + Random(2) * 0.1 - 0.1
162	0	0.000000	0.000000	0	
163	0	0.000000	0.000000	0	RunTime = Val(para)
161	0	0.000000	0.000000	1	para.s = ProgramParameter()
165	0	0.000000	0.000000	0	
160	0	0.000000	0.000000	0	
164	0	0.000000	0.000000	0	
155	0	0.000000	0.000000	0	
244	0	0.000000	0.000000	0	
243	0	0.000000	0.000000	0	
154	0	0.000000	0.000000	0	
158	0	0.000000	0.000000	0	
159	0	0.000000	0.000000	0	

First column shows you the line number, second the total time, third the percentage of time this line/block took to run, fourth the time per call, fifth the number of calls and sixth the code line itself.
You can sort them when clicking on a column’s header and you can resize the window as you want. It provides the same information as the “analysis.txt”.

What does the output tell me?

Introduction

Let's presume that you get something like that:

```
line 216:  total time 9231   percentage: 79.550156  time per call: 14   called: 640   FlipBuffers()
line 222:  total time 1278   percentage: 11.013444  time per call: 1    called: 640   Delay(1)
line 233:  total time 857    percentage: 7.385385   time per call: 857  called: 1     MessageRequester("", "")
line 166:  total time 159    percentage: 1.370217   time per call: 159  called: 1     StartGame()
... (Time < 10)
```

So now you have to get out some info about your code!

First rule:
Everything with a total time below 10 shouldn't affect your program that much!

What does the total time tell me?

It's an accumulation of all the milliseconds a line/block needed in the whole runtime of your program!

Second rule:
Higher the time, higher the importance of optimization!

Percentage!

The percentage is sometimes the most expressive value because it's relative to the number of times this line/block was called! We can't use this value for big programs, because it will be very small (if not -> O_O).

Time per call!

This is also relative and because of that very expressive! It tells you how much time this line/block takes to be executed once! If this time is really high, you should take a look at it!

Called.

Called is mostly uninteresting but it's also nice to have this information. If you enable “showallanalyzedlines” in the preferences, you can also see which lines weren’t called at all!

How can I find the code where the problem occurs?

First information: Line number (it's the number in your source code)!
Second information: The last column shows the code line itself, search them in your source code!

Examples

Third rule:
You have to know how the command you want to replace works and how it could be replaced!

FlipBuffers() without the flag = 0 wastes a lot of time in waiting, but this waiting isn't really CPU economical. So perhaps it's better to increase the Delay()!

Delay() This command is like a buffer of time! If you can increase the value of Delay() the CPU speed isn't wasted and other programs get some speed, too! If you are short of time, you can go down with the value till 0!

MessageRequester() This one isn't critical at all! It's just waiting for user input.

WaitWindowEvent() This one is very important for GUI applications! If you don't develop a really special application, it's not good if WaitWindowEvent() DOESN'T stay on top of the list! This command is very friendly to the CPU! Let it stay on top!

StartDrawing() This is a really slow command and often stays on top of a game application. Try to avoid using it like replacing with sprites or even better 3D sprites!

Own procedure There you know what's inside and you can optimize it! Sometimes it's just a procedure where you collect some other procedure calls, there you can't really optimize much.

Fourth rule:
Try to let the CPU economical procedures stay on top!

But I don't know how I could optimize this procedure!

Ask on the forums like http://www.pure-board.de (DE) or http://forums.purebasic.com (GB)

Preferences

There are some preferences you can change☺. The first one is when you click on the menu item of the Analyzer in the tools menu (see Turn it on/off).
But there are some more you can't adjust there! All preferences are stored in the Analyzer.ini of your Analyzer-directory.

```
analyze = 1
inlineproc = 1
showallanalyzedlines = 1
```

analyze

This is the option to turn the Analyzer on/off (on = 1, off = 0).

inlineproc

This one is to enable the feature to inline the calls of "__an_timer(LineNb) " which is called twice every analyzed line/block! With this feature there will be more code but less overhead and therefore more accurate results. For big projects it's better to disable (0) it.

showallanalyzedlines

If this is turned off (0), only the lines with more than 0 ms (milliseconds) of total time will be shown in analysis.txt!

Problems

Results cannot be true!

This can happen because of several reasons!

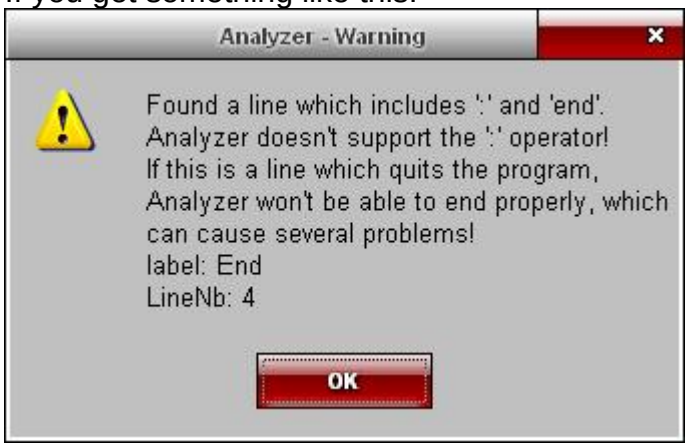
1. If a block or a procedure doesn't have the expected time, it can be because the program ends within that procedure/block. If there is an 'End' inside it, the block won't be finished and the time cannot be measured because it's not updated ("; endblock").
2. IncludeFiles won't be modified, therefore not analyzed!
3. To every analyzed line/block there will be two procedure calls added to __an_timer(LineNb), so that may cause some overhead and can change the expected result!

Slowdown!

If your program doesn't run as smooth as before, this can be because of the two procedure calls added by the Analyzer to every analyzed block/line! It's like debugging. If there are really some time critical blocks of code you can put them inside "; pause" and "; continue"!

Warning message!

If you get something like this:



Then you may have a problem. The Analyzer has a limitation in interpreting lines which are merged with a ":". It will simply leave out lines with ":" in it (strings are of course left out). But if this special line contains an "End" (the PureBasic keyword to quit a program), Analyzer will tell you that there may be a problem! Every time the Analyzer finds this keyword, a call to the end-procedure of the analysis will be inserted. But merged lines are not 100% sure to analyze at the moment, so they will be left out and the end-procedure won't be called at the end of the program. This can cause several problems and you won't get any result of the analysis. The last two lines show you the content of the line and the line number.
So please eliminate such lines for now.

Bugs / Contact

After you read this whole document (for finding your own errors), you can check these things, which would really help me for finding the bug:

- If the debugger shows an error line, this line does not belong to your code! It belongs to “PB_EditorOutput.pb” in the PB\Compilers directory! It would really help me if you could go to this line in the “PB_EditorOutput.pb” and send me this block of lines.
- If you even recognize the part of the error in “PB_EditorOutput.pb”, it would help even more if you could send me this part in the original file, too!
- If possible, send the whole project or a code snippet which reproduces the bug.

If you think you have found a bug, want to help or have some feature requests, you can easily contact me by my e-mail address remi_meier@gmx.ch or through a PM (private message) in one of these forums:
http://www.pure-board.de
http://forums.purebasic.com
http://www.purebasic-lounge.de

Don’t hesitate; I’m a nice person ;)

Credits / Contact

You can easily contact me by my e-mail address remi_meier@gmx.ch or through a PM (private message) in one of these forums:
http://www.pure-board.de
http://forums.purebasic.com
http://www.purebasic-lounge.de

Don’t hesitate; I’m a nice person ;)

Thanks to va!n!!
And Kiffi