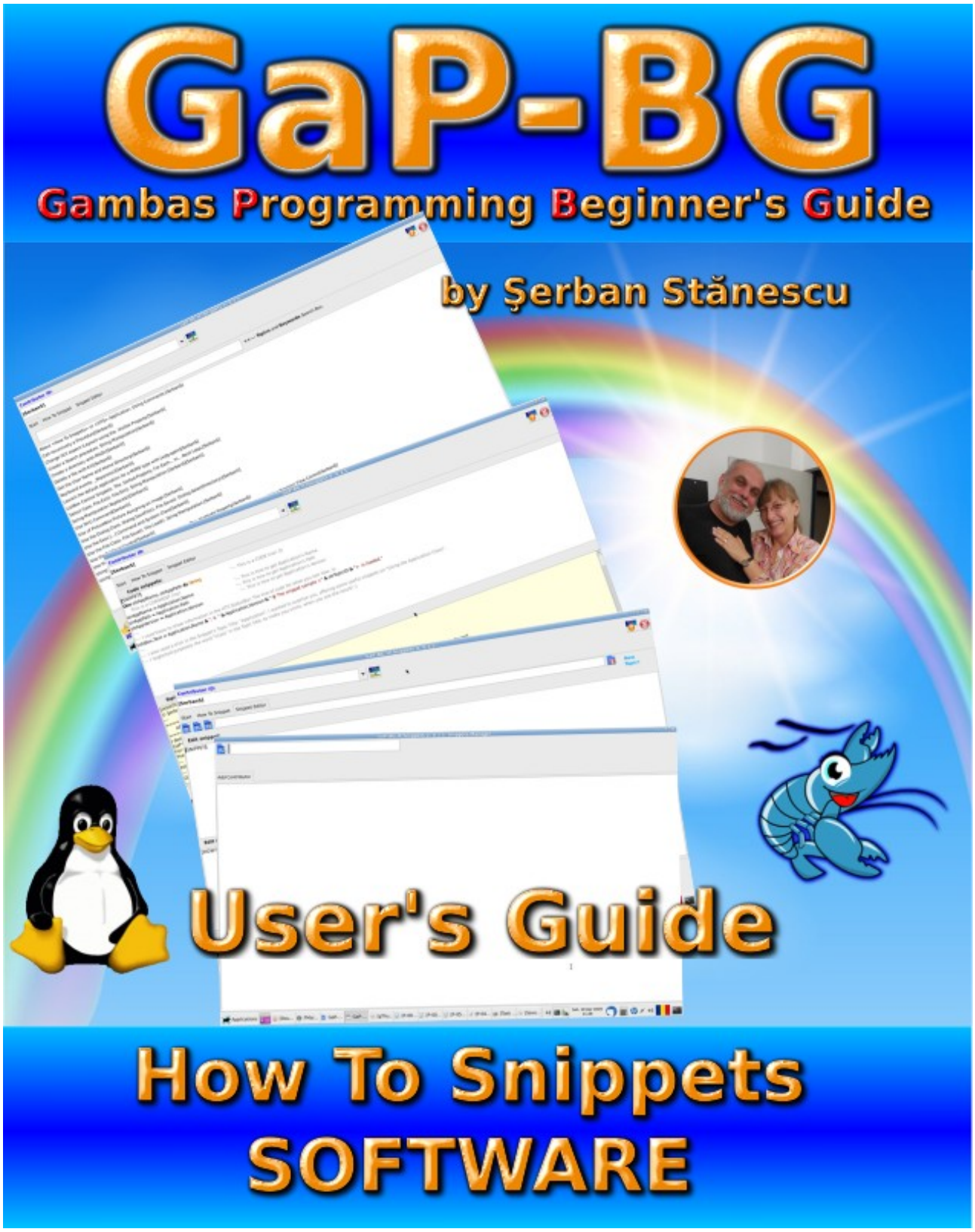

GaP-BG
Gambas Programming Beginner's Guide
by Şerban Stănescu
User's Guide
How To Snippets
SOFTWARE

GaP-BG HTSnippets User's Guide 1.0

Gambas Programming Beginner's Guide Software

Contents

[S 18-04-2020 h 01:10]

[Mo 20-04-2020 h 14:20]

Important notice:

This User's Guide, applies to HTS (How To Snippets) version 0.3.4, which is the first public release. Future versions will have different features so this Guide, will only apply partially. Look for the current version's manual. This should be available through the F1 key or the Help button on the interface. Check the document's version in the footer of each page. Should be different from "V 1.x" (V 1.0).

This version is 1.2. A minor revision of the initial release, 20-04-2020.

Snippets? Snippets!

I started writing code in Gambas, on the 1st of February 2020. That is, about two and a half month ago.

I had a previous practice with RapidQ, a very similar RAD, very much like Gambas, but *only* for Windows. I told the story in "**DirLister User's Guide**" so I'll skip the details.

So, much like everyone who wants to learn some programming language, you go for "Manuals". Documentation. Sadly, after more than 250 hours of painful research and moving back and forth through endless pages of internet, striving to find two-three lines of useful code, I realised that there's a long way ahead in the "Documentation" tale.

A question pops up in mind when you find yourself in this position:

"**Where are the snippets?**"

You know, those little, tiny lines of code that tell you WHAT to do and HOW TO do it.

I finally came to understand that the best approach, is to study the applications developped by other Gambas programmers. Sadly, there is a huge problem here, too: Way too many applications are written in Spanish, for one. Or in German.

My primary language is Romanian: **Română**.

Nothing to do with French, Spanish or German.

Far more than that, trying to look up some code, I found several bugs that prevented the apps from running from the start. Some of them I was able to guess — *yeah, when you're at the beginning of something, almost everything is guess-work* — so the pieces of software I was interested in, were unusable.

So, after 250+ hours of digging through the documentation, I found myself like "***Back to square one***":

"**Where are the snippets?**"

After overviewing my notes — *I work with many "logs" so to speak* — I came to the conclusion that practice is the medicine that cures the desease. Almost every desease.

After about six weeks, I was able to write DirLister and gather the first batch of practice. DirLister has over 2,500 lines of code, so yeah, some practice added up. I wrote 3 apps so far, two of them both add up to 100 lines of code or less and DirLister, which now, in my personal version, exceeds 2,500 lines.

So, again, what was the question?

"***Where are the snippets?***"

I had an idea about 20 years ago, to write an application in RapidQ, titled "***Snippets***". I had all I needed, but TIME. My life changed in a way that led me to other twisted narrow paths so, in 2006, after some five years of coding in RapidQ, I abandoned my coding plans. But as time showed, the ideas, survived.

Given the fact that in Gambas community there are some very skilled programmers, way ahead of myself, the "Snippets" idea, came back to life.

While back in february I was very confused with anything and everything, after writing DirLister, things got better.

I started to write the first book of the "**GaP-BG**" Project. Than, the second, the third, two "User's Guide" for the Apps I uploaded on Gambas Software Farm (Vendor= "***MoSeS-ZENLA***") and the "***Chapters Index***" for the Gambas Programming Beginner's Guide.

My secret dream though, was to write an application that provides samples of working code, that explain how to use one thing or another, that is easy to use and speeds up the coding.

So, here is the answer to the question:

Yeah, Snippets! Here it is! Here they are!

In Good Faith

All the work I put into the series of books "**GaP-BG**" and the apps I wrote in Gambas and made publicly available on Gambas Software Farm, were in Good Faith. Meaning that I never had the idea, or the intention of making any money from that. Nothing. Period.

I did it because I belive that if we want to live in **a better world**, is up to us to **make better things, in better ways.** It's the only way that works, despite all other ideas.

I never made a dime as a programmer or as a system admin, so this thing called "*programming*" is a **hobby**, nothing more.

To conclude, the only thing I would like you to do, is **IF** you find useful this stuff I made available, PASS IT ALONG to someone you know and is interested. I'm sure you know at least one.

Or maybe write some snippets. It's easy. You'll see in the next pages.

I made it as simple as I could and please, belive me, writing the few startup snippets, proved to be an addictive kind of "getting busy". I enjoyed writting the snippets, more than writing the programs. Difficult to know why, but I just noticed it.

I hope you'll like it, at least a quarter I liked and enjoiyed writing both the books in the series and the companion software, that the present book describes.

All the best,

Şerban Stănescu.

Overview

After this introduction, let's get to the **HTS** — ***How To Snippets*** — specifics.

The application was designed to provide an easy access to "***How to do this and that***" and some minimal help in understanding the lines of code that actually do the thing you need to do. And mostly, TESTED CODE. Meaning, ***code that was picked up from an application that actually does what was meant to do.***

The Guidelines

With those in mind, I has to folllow some basic guidelines:

- *A simpe file structure;*
- *An easy to use interface;*
- *Means to read the snippet files and show them on the screen, without tons of whistles and bells;*
- *Most of all, **A SEARCHABLE DATABASE**. Having a searchable database with a loose search engine, speeds up the process of learning and writing the code, up to 50 times. Sounds fantastic? Among others, I'm a certified Project Manager so my approach, was: IS IT EFFICIENT? Does it help me to save both TIME AND MONEY? If so, it's good. Else, it's bad;*
- *Writing capabilities. While at the beginning the experience of writing say, 150 lines apps is gathered slowly and painfully when you lack the tools to help you, the other way, using handy tools, cuts down the amount of time with at least 50%. Practice proves that **HTSnippets** does it far better.*
- *Sharing the snippets. Wow! That was the dream! When looking closely for a solution, it proved to be more difficult than it looked at first sight. I managed to put together some ideas that layed out the foundation for both WRITING AND SHARING SNIPPETS. We'll get back to that later in this User's Guide.*

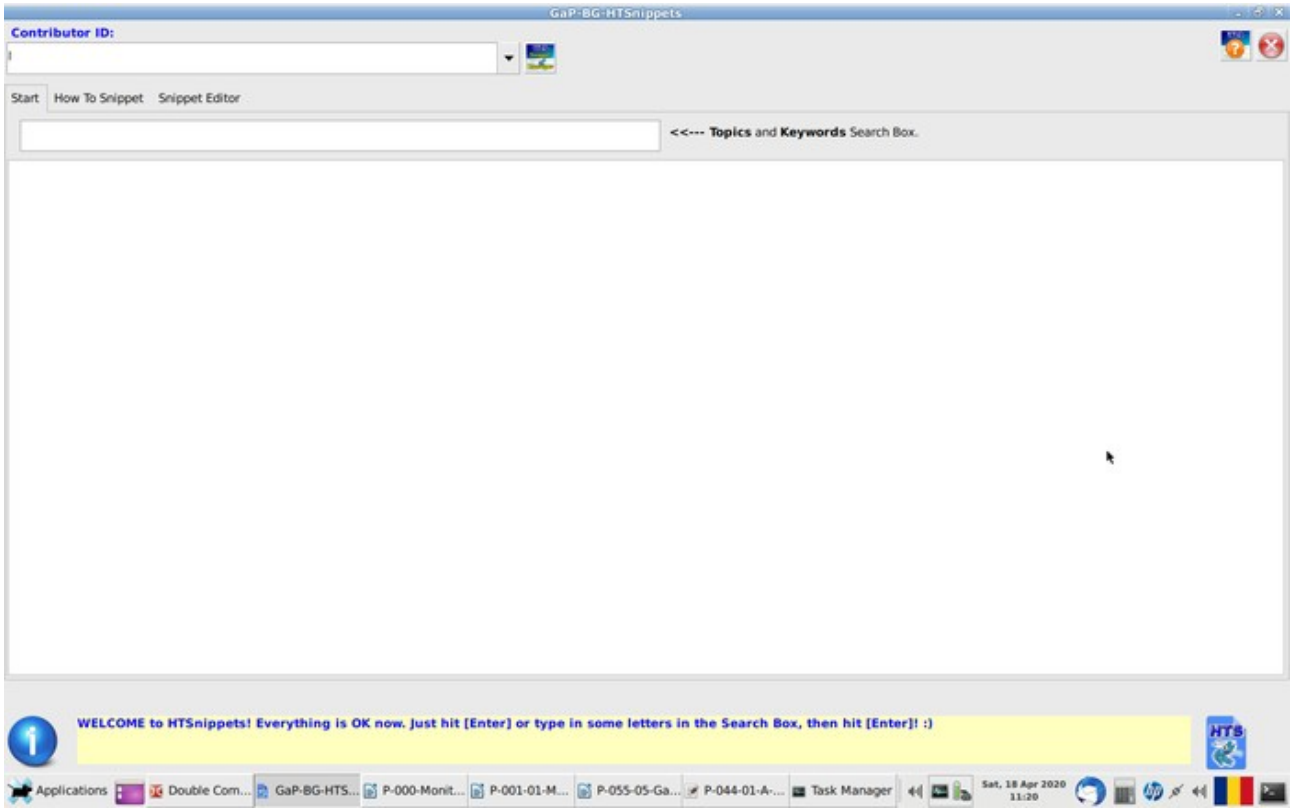
The basic features

- **Contributor Management.** When there's only one developer, it's easy! Still, this application was designed from the beginning FOR THE COMMUNITY! So as long as you write snippets for yourself and keep it this way, ***it's easy.*** What happens when a second programmer has tons of ideas and writes snippet after snippet? How does he distinguish the snippets that were initially in the database and the ones he added? Taugh situation! So, I introduced a very simple Contributor tracking mechanism, based on a token, [**Contributor ID**]. A very short or at least short string, that distinguishes one contributor from the others. You'll see it on the first screen: The ComboBox. It holds all the Contributor's aliases ("Tokens"). Based on that, the files are individualised based on this criteria: [**Contributor ID**]. There is an internal basic management engine, a very simple one, but it does the job, if used properly;
- **Searching the DataBase.** Without search capabilities, I spent frustrating hours moving back and forth through HTML pages, clicking countlessly on tons of links that got me far away from what I was looking for. So, that was among the main goals: SEARCH CAPABILITIES. There are two approaches in **HTS**: Search with an EMPTY string (Search Box is empty), meaning you just press [**Enter**], which leads to a full listing of all available snippets. Second, is a loose search engine, based on a keyword. The search engine, will return all topic matches found, that contain the keyword. While the database is small, is easy to list all available Snippets. When the number of Snippets exceeds what the ListBox can hold and show in a screen, you start scrolling. That's a waste of time, again. So, it's better to use a keyword, get the job done, then you can return to the [**Start**] screen and search for the next keyword

and so on, until you get all you can.

- **Viewing Snippets.** This had to be something very significant: my idea was to see the snippets with SYNTAX HIGHLIGHT. And it so happens that Gambas provides such means: the [**TextEditor**] control. So, for the Snippet part of the snippet file, I used the SYNTAX COLORING feature. It's easier to follow the code, and it's the closest match to the IDE itself.
- **Creating and Editing Snippets.** It's easy! You'll see! It's a four steps path: **(1)** Copy some working and tested **code** from an App, then Paste it into the code area (the upper edit field); **(2)** Fill the **Topic Title**, with a few meaningful keywords; **(3)** Fill the lower editing field with some help information; **(4)** Hit the "**Save Snippet**" button. That's it! Deleting a snippet is as simple as pressing the [**Delete Snippet**] button.
- **Snippets Manager Window.** This is a work in progress. For now (***Version 0.3.4***), it only allows to add a **Contributor** to the Contributor's list. Future versions will have other features, but for now, the basic skeleton is layed out. All I need is to add step by step, the minimum functionality for Snippet management. Might take some time but in the end, will be there. Nothing fancy, though. Just basics to make the import of new Snippets easier to handle. Feedback from Contributors, will shade the light on what might be useful. Being a community driven project, I assume that feedback will bring new ideas and solutions, as the Project grows and, obviously, glitches show up in the process.

First Screen: Start



The first screen, [**Start**] is divided into 3 sections:

1. **The Toolbar;**
2. **The TabPanel control**, which is now the white space. In fact it's a ListBox control there.
3. **The StatusBar**.

Choosing the Contributor

On the ToolBar, there is a ComboBox. This is the "**Contributors**" list. Click on this if you are a contributor and you will be presented the list of contributors. Click on your Contributor's ID.

If you are using HTS for the first time, and just want to browse through the available Snippets, that's enough so far.

After choosing your ***ContributorID***, the screen looks like this:

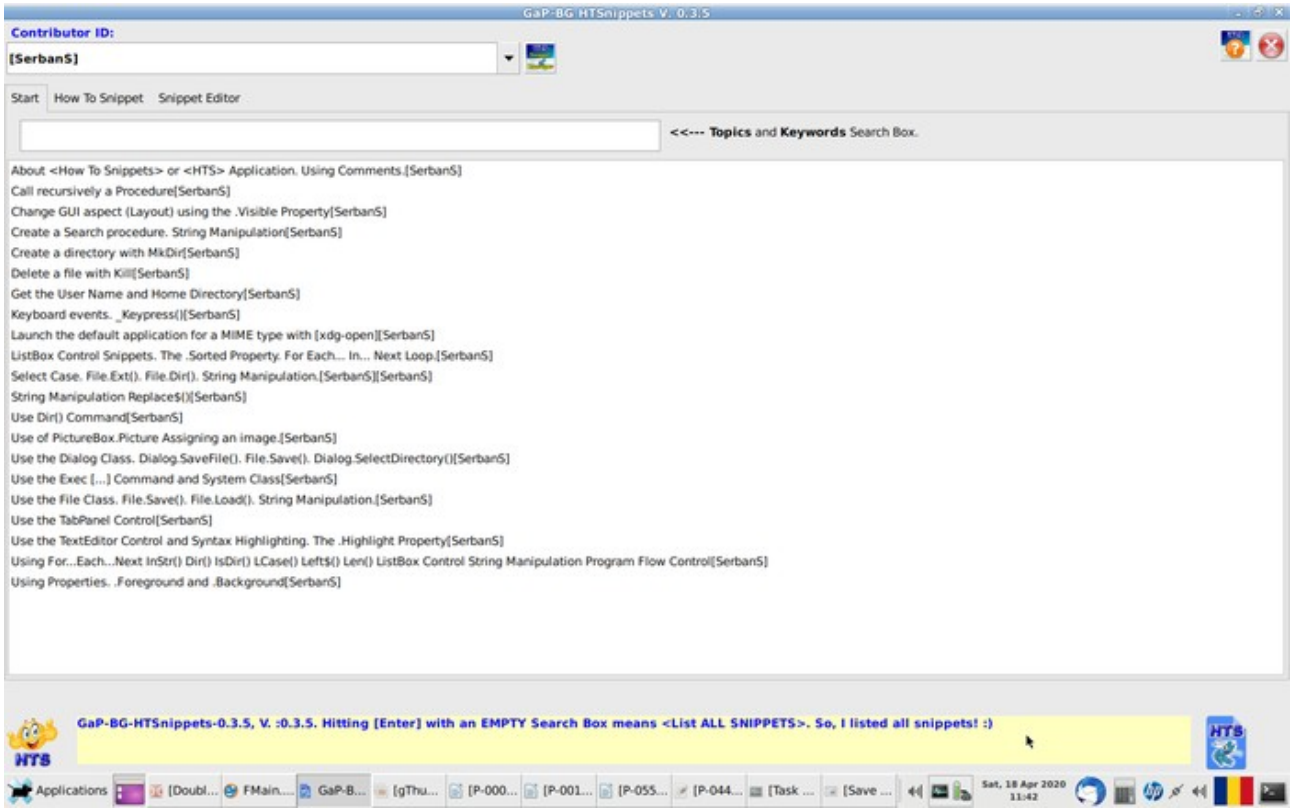
Listing the Snippets

It's very easy!

Two approaches are available:

1. **List All Snippets.** Just select the Edit field and hit [**Enter**]. All available snippets will be listed in the ListBox below the Search Box. While now it's easy to follow the list, in time it will become longer and longer and will be very difficult to spot a certain topic so you will choose the "search" feature.
2. Search for a topic. When the list is long, it's very difficult to see which topic coveres exactly what you are interested in. So you type in a keyword and all matches are listed, instead of all Snippets. It's a very simple, very basic search engine so only one keyword is accepted or else, it's highly unlikely that any result will be rendered. Maybe in a future release I'll change it, or maybe a Contributor — **Are you The One?** — will come with a better idea. Until then, that's it...

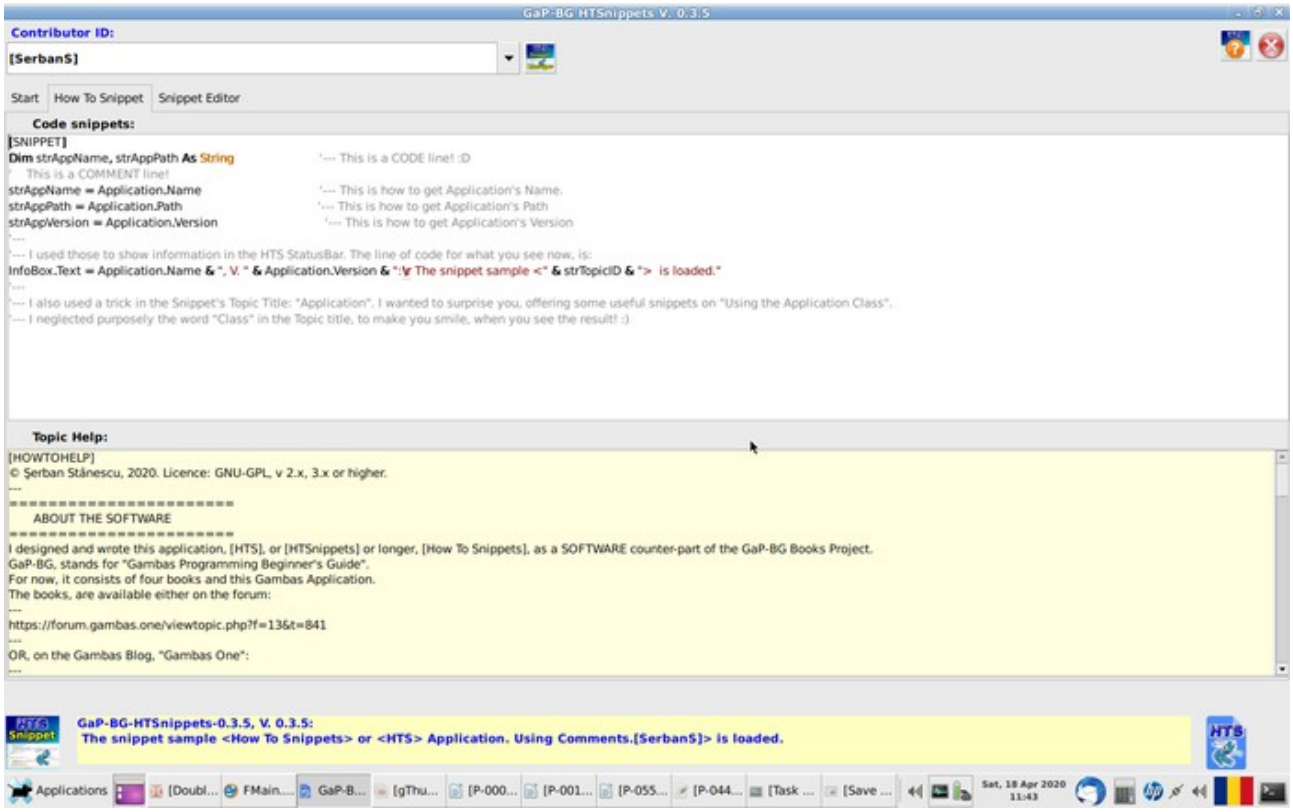
You will be presented with this screen:



Now, if you find what you were looking for, click on the topic, the selected topic will be loaded into the Snippet Viewer and you'll be directed to the next screen: [**How To Snippet**].

Second Screen: How To Snippet

This is the second screen:



As you can see, we have four regions (sections) on this screen. The **ToolBar**, the **Snippet Field**, the **Help Field** and the **StatusBar**.

The ToolBar

I spoke already about the **ContributorID** ComboBox.

There are now three buttons more on the **ToolBar**:

1. **Launch Snippets Manager.** A work still in progress.
2. **Launch Help File.** Launches this file.
3. **Close Application.** Closes **HTSnippets**.

The Snippet Field

You can easily observe that the **Snippet Field** offers a **Syntax highlited** version of the code, to make things easier to read and follow. They look very much like in the Gambas IDE. You can study the code here and if it fits your needs, you can even Copy/Paste it into your program, then adjust the code for the specific needs of the context.

Although the main purpose was to offer you this exact feature, the opposite is also available, meaning that in the [**Snippets Editor**], you can do the reversed operation: Copy from the IDE then Paste into HTSnippets code field. Actually, that is how I wrote all snippets, except the first, that was written with Pluma (Text Editor). Far more than that, actually I wrote only a few words and I updated the rest in **HTS** itself, when this was possible.

The Help Field

This is the next editing field. You can see the token at the beginning of the Help text: [**HOWTOHELP**]. You can see a similar token at the beginning of the Snippet code: [**SNIPPET**].

This section is meant to offer a minimum guidance on how the code works and, eventually, ***a reference*** to the Application where it is used.

Any kind of useful information regarding the Snippet or where it can be found and seen working, goes here.

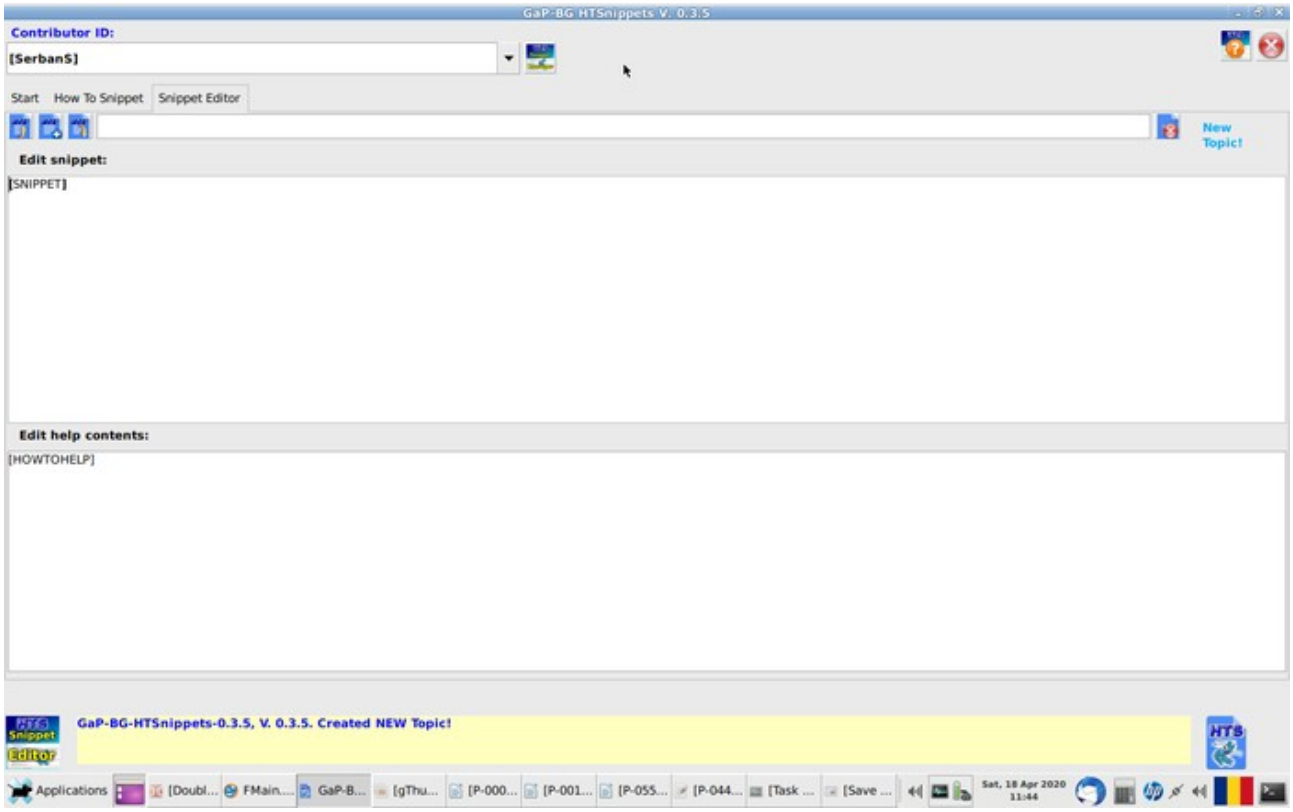
The StatusBar

Is a very simple one, that shows some different icons that are relevant for the current context and messages regarding what is happening "behind the scenes". They are mostly informative. I used Message Boxes only when ***a decision is required*** to be able to go further with the program.

Blocking the program, has to have a good reason! :)

Third Screen: Snippet Editor

Let's start with the screen itself:



As seen in the above capture, the [**Snippet Editor**] screen, is slightly different from the [**How To Snippet**].

While the main ToolBar is still available, there is a **second ToolBar**, specific to the tasks of the Editor:

The Editor ToolBar

- **Leftmost button: Open Existing Topic.** The button with the "***Up Arrow***" icon. Does the same thing as searching with the "" string: Lists all available topics. While in time this might be more appropriate to be on the second place, during development — *which, by the way, it's still what's happening* — this was the most used button so I placed it on the first position. The usual layout though, is "***Create New Topic***". Since

this application is meant for developpers rather than for other audience, I belive it's on the right place. Ideas come and go, better ones always pop up and we all end up improving one algorithm or another so, updating a previously created Snippet, it's likely to happen often. It happened to me already.

- **Second button: Create New Topic.** The button with the "***Plus***" sign icon. Self explanatory: Creates an empty snippet, ready to be filled up. Some refinements are still to come I guess, such as predefined signatures for each contributor. But these are secondary aspects for now.
- **Third button: Save Snippet.** The button with the "***Down Arrow***" icon. Saves the current snippet. For now, if you want to keep different versions of one snippet for whatever reason, *"**Save As...**", while less obvious, it works very simple:* Change at least one character in the Topic Title — say, put a "1", or better (1) in the title — and will be saved as a different file. It's faster than clicking a different button, than doing the exact same thing in the Dialog window. You do it in place.
- **Topic Title field.** Just add some relevant words for your topic. Say... "**Open a .csv file**". You might add even related keywords such as "***Using Databases***". If the Topic Title field is empty, you'll get a Message Box telling you to fill the field in.
- **Delete Snippet.** The button with the large "***X***" on red background icon. **It deletes the Snippet for good.** Use it with caution! I put it beside the Topic Title edit field purposely. Initially it was beside the "**Save Topic**" with some space between the two, but when working, I was tempted too many times to click on it and I moved it, to avoid the accidental deletion of a snippet. Might rethink this "Delete" strategy in time, though. Maybe a cleanup procedure on exiting the App? Just an idea...
- **Snippet (Topic) status label.** The rightmost control. Shows the type of task ongoing on the snippets: "***New***", "***Editing...***". Just a handy reminder.

The Snippet Code Editor

Right below the ToolBar. Allows you to enter or rather Copy/Paste some code, showing it highlighted, like in the IDE code editor window.

The best approach is obviously to Copy/Paste code from an application that works and has been tested.

You can also write the code directly into the editor, working very much like in the IDE, except the fact that you'll have only the text editing capabilities available, without the ability to run the code. That is impossible now, except the situation when **HTS** would be capable of running it, which is a distant dream for now. I'll get back to that in the chapter "**Dreaming of...**".

The Help Editor

Has basic text editing capabilities like the other one, although during development I encountered some trouble pasting text with [**Ctrl+V**] from another program, such as Pluma. The "***Paste***" though, works using "***Right click***", then "***Paste***". I assume that my knowledge on using those controls, need more study and practice. A snippet on that, would be useful! I guess that has something to do with the [**Clipboard**] class in conjunction with keyboard events, which, by the way, already create problems (GTK warning). I never used the [**Clipboard**] class and is almost undocumented so, there is ground enough for tons of Snippets!

Might be something like:

```
TextArea.Clipboard Paste("text/plain")
```

put on [**TextArea_KeyPress**] or something, for the [**Ctrl+V**] event type. While this approach might work for "*text/plain*" MIME type, *which is all needed here afterall*, I need to test that some time to make sure it works.

What about pasting a formatted text? Would it be treated as "text/plain"?

Maybe in the next version?

The size of a Snippet

Although the control allows to write somewhere above 2,600,000 lines, depending on the amount of RAM available on your system, I discourage this kind of verbosity, as well as I discourage anyone leaving the section [**HOWTOHELP**] empty and passing it for public use this way.

How do I know the amount of lines that might be written in a [TextArea] or [**TextEditor**]?

I made a ***mistake*** when working on DirLister. A line of code copied from a different procedure, was supposed to be say, at line 831 and accidentally, I pasted it at line 833. The line above, line 832 was doing something related to adding lines to the destination file. I ended up in an infinite loop that generated a 2,3 GB file that froze my computer, filled up to the brink the SWAP and ate up all my RAM. **:o**

I had to cold restart the computer and try to find out what happened. That's a funny story now, but was frustrating at the time it happened... I was very tired, after a some 17 hours work marathon...

Now, hoping that I made you smile, the idea is that as far as I can understand, the amount of data you can write into a text control, is limited only by the amount of RAM. I doubt that you really want to write a kernel code, or a DE one into the Snippet editor!

The general idea is to write some lines of code that are relevant for a very specific task, such as "Resizing a window runtime" or the like. A few lines of ***code***, a few lines of ***help*** and that's it! Hit the "**Save Snippet**" button and move to the next topic. Even if you generate a large number of snippets, in time **HTS** will have the capabilities to manage the snippets collection in a more friendly way. We'll get to that too, soon!

The tokens

It's about the [**SNIPPET**] and [**HOWTOHELP**] tokens.

Each record in the corresponding editors, begins with one of this **TOKENS**.

These are in fact **Section Delimiters**, that are used to Read/Write the two sections from and into the corresponding files. Both contents of the two editing fields, are in the same file and the amount of data corresponding to each editor, is delimited by those tokens. Very much like in a [**something.conf**] file.

Deleting or altering those tokens, lead to reading errors so **those are mandatory and have to be left as they are.**

You just hit [**Enter**] ***after the token***, and write whatever you need, below the tokens.

This approach is necessary for the Snippet sharing capabilities that will follow in the next versions. While an "all-in-a-file" approach sounds interesting, my practice says that this leads to countless trouble in time, although updating the database looks easier this way: just copy/import the new file and that's it.

Problem is that the Read/Write procedures get very complex and very difficult to maintain. I like to keep it simple!

The number of Snippet files is irrelevant. We can deal with them. Further more, the update procedure can be very fine tuned this way and the database structure, can also be improved in time. As I said before, there are lots of things to improve... But **now, there is SOMETHING to improve!**

Second Window: Snippets Manager

While this might look like almost useless now, in time many features related to Snippet Management will be mandatory.

The most obvious situation:

- ***I upload HTS to Gambas Software Farm.*** How many snippets were in the first public release? That is mandatory information since adding snippets AFTER the release, requires a method to address that. How does one update the database? Overwriting it? What if the user is a CONTRIBUTOR? How do we update without altering HIS database? How does MY **HTS** copy know what's on the other machine?

As you see, too many questions, one on top of the other and, it seems to me an endless story.

"Step by step", this is how anyone gets to a goal. Any goal, as a matter of fact.

This is a brief overview of "***Why another window***" topic.

There are many things to cover and only a community driven approach can get out the minimum further requirements, new features and the like.

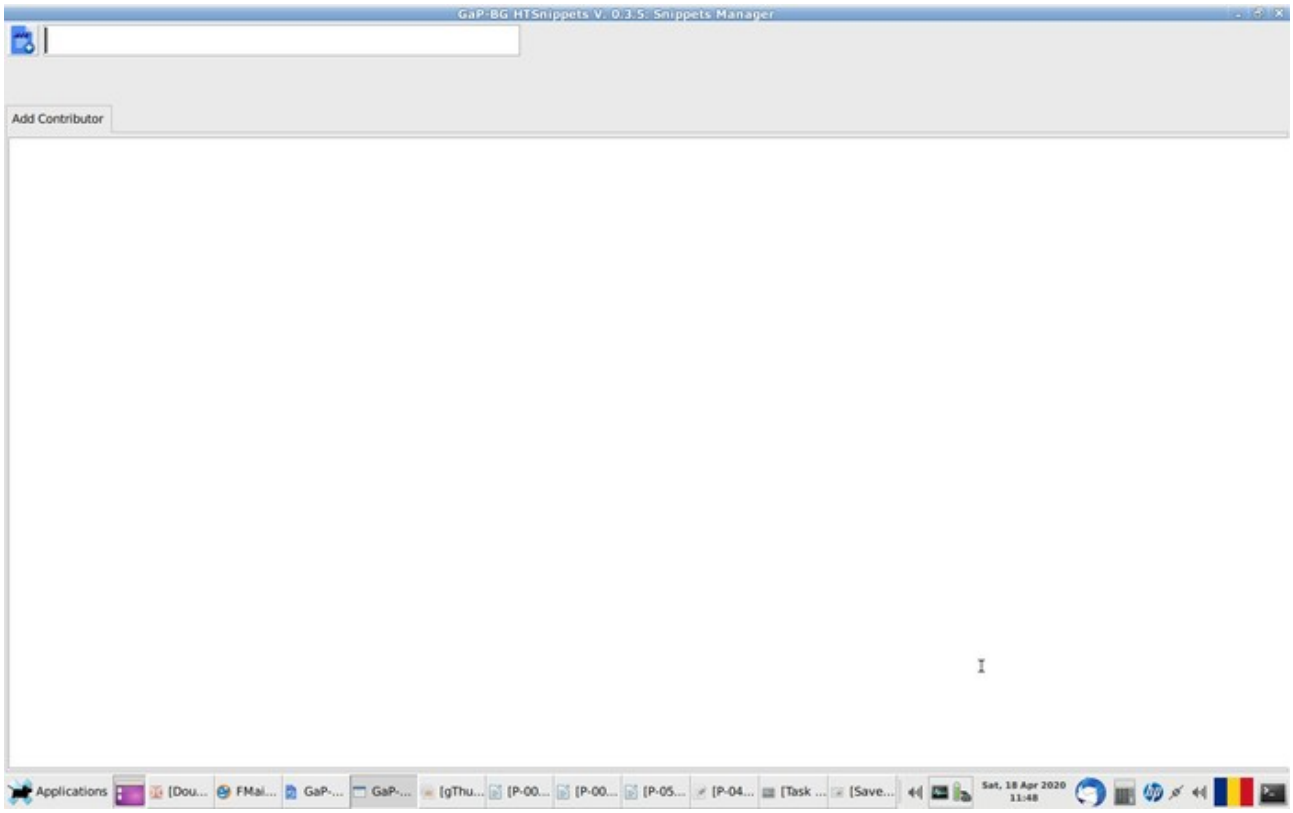
That means releasing a working version, even if unfinished.

That is the idea of this extra-window: it lays out the foundation for what's to come.

I might give up developping HTS for whatever reason, bad things can happen to me, so making tedious statements is pointless. I can even die today, as anyone else, as a matter of fact. What will happen then, depends on **what I have accomplished NOW.**

Now, let's take a look at this screen, as it is now.

The Snippets Manager Window



The ToolBar

For now, it's more like a sketch than a real toolbar, but there are things to come, as discussed in the previous subtitle.

The **button**, [**Add New Contributor**] allows you to add a Contributor and now, it only clears the contents of the two editing fields. That's it.

At the right side of the button, is the editing field, where you need to enter the desired contributor name. If you are on the Gambas Software Farm and have a "**Vendor ID**", this would be very convenient to use. Otherwise, think of something short and meaningful.

After entering the desired **ContributorID**, hit [**Enter**] or [**Return**] and the constributor is added to the database.

From now on, when you start **HTS**, click on the Combo Box and select

your **ContributorID**, and you will be able to add snippets that are associated with your **ContributorID**. That is the approach.

You can list all your snippets, by entering the token into the Search Box in the [**Start**] screen. For now, it's the only way to do that.

In time I hope that will change, but his takes time and for now, it's a "***One Man Show***"...

The Task Monitor Output

The TextArea below the toolbar, is very much like a console window. It ouputs various messages according to what was done at a certain step of the current activity.

As the application evolves, it will be very important to know what is happening so this will be more and more useful.

Work in progress: The ToDo List

Here is a very short list of what I am thinking now:

- **Make a DB Snapshot.** Necessary. I already discussed that.
- **Filter by Contributor.** Mentioned that in the previous chapter.
- **Backup DB.** Important. Whatever method will be used, it's important to be able to backup the snippets on a regular basis.
- **Restore DB.** Needless to say anything.
- **Build Update List.** ..
- **Import Snippets.** ..
- **Export Snippets.** ..
- **Create compressed package.** ..
- **Create Category.** ..
- **Sort by Category.** ..
- . ..
- . ..

That is what I can see so far. After that, who knows what might pop up in my mind or anyone else's mind.

It depends on too many factors. Let's see what happens!

HTSnippets Lines of Code?

The third book of the series of the Project "**GaP-BG**", ***Gambas Programming Beginner's Guide***, is in fact a different approach to "**How To Snippets**": I picked up the **DirLister** code, I followed the first set of chains of events and described it and what does every interface element of DirLister, and what happens behind the GUI; **what does what, when it does and how it does.**

The people who got to read the book, are probably going to ask themselves if I will go on the same path with **HTS**.

At this point, I am wondering myself if this is a good idea.

I feel rather attracted by the idea of developing a similar story, but **INSIDE HTS**.

That means that I like better the idea of writing some snippets and adding the help contents making showcases from the parts of **HTS** itself.

It's a different approach and I guess will be more attractive than writing a book on the same topic.

On the other hand, I am planning to write a new book "Gambas Lines of Code", to describe the next features of **DirLister**. Overlapping this project with writing a similar book on HTS, seems a bit less original idea.

I'd rather go on the idea of using **HTS** to build new snippets, than writing another book on **HTS** snippets. Seems to me more appropriate than writing a book.

There is only one argument, but it's powerful enough: Copy/Pasting from **HTS** is easier than from a book. I know that a decent PDF reader has search capabilities, I use them often, still... I feel more comfortable with the idea of using **HTS** to find a snippet, than searching within a PDF for the same result.

Just a different perspective I guess.

Dreaming of...

We got to *the final chapter...*

Do I have dreams?

Yes. At least one. It is called **PLUGINS.**

As I said in the first two chapters, this project has some 20 years of age.

I'll skip the details. I'll only mention that in RapidQ, one of the RapidQ community fellows, maybe Jordi Ramos, I forgot who exactly, wrote an application that was able to list all the "Include" files.

Those were the equivalent of the [**.class**] but more like [**gb.component**] files in Gambas, so you get the idea: could have been used as parts of the program, you could browse parts of the code and get ideas.

The only downside of the application was that the "**Include**" files were mostly large files, some of them containing thousands of lines of code, most of them, unappropriate for study, being either uncommented or poorly commented. Some of the includes, were written in a "*set and forget*" style and even the authors had trouble reading and understanding their own code.

The ***naming variables*** problem, was I guess the most frequent of them.

Variables like "cstmk", or "cbstk", "ic", "hdc1" or many others like this, have no meaning even after 4 or 6 weeks after writing the code, if you break development for a few weeks for some reason.

Now I belive you see the idea behind **HTS**.

But the idea of **using PLUGINS to demonstrate the validity of the code**, well, goes beyond a simple paragraph of help. Let alone the aspects regarding of "How To Write a plugin".

The possibilities, are virtually endless.

Still, as I said in the chapter's title, this is **A DREAM** so far.

It's good to know, anyway! **All the best!**

Contents	EOF		End Of File