Translated by Rafał Dymek

Koszalin, 5. march, 11. may Anno Domini 2021.

Apollo – to Give Something to the World

The reasons for its birth

Someone will say: “Why write a program which does not work entirely like Delphi?”. But it's just all about its not working entirely like in Delphi. Delphi – let us pay tribute to this extraordinary application – pioneered some solutions. They were featured in it for the first time. An experienced programmer knows, however, that there are a good many burdensome aspects of writing programs on this excellent program. I understand that Delphi and its solutions have become standard. But certain models become only burdensome when one does not go beyond them for a long time. The world wouldn't develop at all if everyone kept on employing models which, as a tradition-honoured canon, strangle the will for a change. As for ourselves, let us focus on what those model programs do not offer, what has been done in a dubious manner, or even, where are their internal errors.

We do not maintain, however, that we do not make errors. But of course we do. And at present the errors presumably outnumber the right things. Yet one learns on one's mistakes, which is an invaluable asset.

Should we, then, do our utmost for the codes written in these models to operate in our program without any corrections? We have chosen not to bother how they did it in Delphi and its cumbersome copy of Lazarus, Typhon and other environments derived from FPC which have already been available. We have arranged for it to operate our way. A better way? It's not for us to judge. The program is developing just yet, although represents already a fairly compact programming environment to provide a rich array of tools. We are folowing our own way, for that's how we want it. That's what our fancy tells us.

Will those people out there run a high fever on finding with us what they haven't got themselves? We are not sisters of mercy with fever-compresses, we are writing a useful program, which is to say, one to give us satisfaction once we have perfected it. If we reach a stage where writing programs can be attained solely through our program, the basic plan will be accomplished.

And what is our program called? **Our program is Apollo program.**

Let us begin with the very structure of the program. With us **in Apollo there is**

no extra project file, as in Delphi, let alone Lazarus. No extra file-monsters wherever, let alone in XML or some other bloody thing of the kind. Our program code is self-sufficient. Apollo at program start-up – either at being opened or selected from bookmark list – automatically analyses the program code and replenishes for itself all needed information. Is the user glad about it? Why not, if he has thus everything those people out there have, and in addition something more? The disk is not stuffed with any file-monster with something scribbled into it.

The only extra file to the project is a file bearing the name of the project and extension *.apc (Apollo Project Configuration – Apollo configuration file), which contains additional options of the compiler when they are needed at the project compiling. When it is not needed, the absence of this file does not influence in any way Apollo and the work on the project. It is an optional file, only when the project requires additional compiler settings.

Next, the form is no extra file pasted to the Pascal file (*.pas). In general, there is no form file. Everything is being created in an overt method in the code itself, which is to say in Pascal file (*.pas), in the procedure `GenObjects` automatically generated by Apollo. No one, however, prohibits the programmer from fumbling in this code, provided he adheres to the basic principles of operating this weapon which is such a specific procedure. At present one had better, for example, not use comparisons to: rules (adding, concatenation and so forth), constants, alternations, and procedures. The values of the comparison must be left in their basic version. The complete record remains the complete record, the chain one remains the chain one, and so forth. And if someone is annoyed with this procedure because it's long and boring, or for some other reasons not to be mentioned here, he/she can close it with the tiny listing minus not to be disturbed any more. This method, however, gives you insight into the very creation of the objects, and without any objects inspector or supervisor, or, as a certain monkey expert on a tree will put it – a warden. One can interfere with the object generation at one right where it has originated without the need of any code correction which will effectively change its values only after their being applied in the invisible form file. The programmer is the master of the situation here, rather than any background mechanism which, true, works on its own but slows the application down at the time the objects are being created. Besides, the programmer can see how it is constructed, which allows him/her quickly to accommodate wider knowledge on the principles of building everything that the program he/she's creating fancies.

Besides, the programs themselves – in fact, their projects – are opened in optional quantities in the code editor. The icons in the bookmarks indicate at once which program, form and module is at work, or other Pascal file or even other files. **It suffices to indicate the given program bookmark for Apollo to start compiling the indicated program from its project.** This greatly facilitates work in the environment, since one can work on files shared by a lot of applications and alternately check whether our corrections and ideas fired up in the programs that are hung on to those files. We have called that something by the name of multifileness, a bit after the method of the

ignoramus Comte, the creator of the Latin-Greek vocabulary concrescence called “sociology”. A proneness to multifileness is multifileability. So Apollo is multifileable, for it makes possible the multifileness of projects, or rather, work on the projects. To be strictly correct (for the word “purist” ought to be exorcised by the lovers of the Polish language), we might call it wieloplikowość and wieloplikalność. But an Englishman or other Anglo-Saxon, or an English-speaking person, will call it multifilność (multifileness) and multifilialność (multifileability). Isn't it funny? And what fun to create applications on Apollo with that uncommon feature of it. One needn't open a second or any umpteenth copy of the programming environment, in order to open up a lot of projects. Sneeze et it. We open Apollo and ride on as many projects as we want and are in need for. Here nobody limits our needs and especially the needs of our imagination.

But, not to sound too sweet, Apollo has a modest palette of components, which in Apollo in fact are classes rather than components. Neither has it a method of pasting on new components to the palettes. Will it have one? Why not? Apollo is developing and in the folds of his bushy attire these – and not only these – colours will also eventually develop. After all, in Delphi, from which the name of Delphi is derived, there was the temple of Apollin – that is, Apollo. So Apollo rules, even though some people never realise that.

And why Apollo? I hold it against the Americans – it's true – for terminating the space Apollo programme so beautifully started out of the ideas and words of the first Catholic President of the United States of North America, John Fitzgerald Kennedy. Did America irretrievably lose the capacity for spearheading the advance of the nations into the unlimited expanse of the Universe? This I don't know. I'm writing a program. I'm writing the program Apollo. Perhaps just so as to inspire some Spirit into that faraway past, so as to prevent the dust of stupidity of politicians from covering completely a beautiful idea. Can my program prove useful in some way? I trust it can. I'm writing it to give something to the World. My supreme guideline in this is: **Apollo – to give something to the World.**

Someone will say: “But this is paganism”. Well, then the Olympic movement is also a typical form of paganism, a holiday of hecatomb – tribute to Apollo. Let us not keep thinking about it. Today Apollo is a servant of Christ, and if someone wants it otherwise, they will be hard put to prove it. But a good servant is a good servant. And such a servant we need – sowing his arrows of plague to all corners of the World, yet also dispatching his little servants Muses to look after an embattled Mankind, hopefully embattled not beyond repair. In other words, to look after each and every one of us separately, and all combined. And on this palette of colours and on those strings let us play, so that programmers can work better and their creations are better moulded... Thanks to Apollo.

God bless

Andrzej Marek Hendzel