LabanGraph 4P – An(other) Computer Editor for Labanotation

János Fügedi

The research was supported by the Hungarian Scientific Research Fund (OTKA NK 77922).

LabanGraph 4P (for professional presenting, printing, publishing) is a so called API, an application program interface for AutoCAD, a software for computer aided design[1]. The reason I started developing it is complex. In early 1990s I had no access to Macintosh computers, therefore I could not use LabanWriter (developed by Lucy Venable, Scott Sutherland, David Rally), but some complaines on its incabality of zoom facilites just as well as its graphic I met with low quality made me regret less the lack of possibility accessing it. I stated to work with CALABAN developed by Andy Adamson at the Birmingham University for PC, which produced an outstanding graphics, thanks to AutoCAD on which CALABAN was based on. But I had problems with the interface of CALABAN. For input it needs a digitizing tablet with a graphical menu system as it can be seen in Fig. 1. To insert a sign the pointer needs to be slided above the menu item, then the pointer has to be placed at the required point in the square at about the centre of the menu sheet. To create the notation of an average lenght of about a 10 strophe traditional dance the pointer's needed to be moved many thousand times – who tried it knows, it is a tedious work. Editing the score needs constant input from the keyboard as well, so the pointer has to be released, the keyboard handled then one could return to the pointer again – the constant change slowed down input. I also missed functionality helping notation editing, such as the interactive definition of the lenght of direction signs, the easy creation of intermediate directions, the „broken" path signs as one symbols, the input of complex, let's say „user defined" signs, and such really useful function, a solution

[1] For information on AutoCAD see the website of the producer, Autodesk (usa.autodesk.com/) or Wikipedia (en.wikipedia.org/wiki/AutoCAD).

for an „intelligent mirror". „Intelligent mirror" means, that not only the „picture" of the drawing is mirrored, but the „meaning" of notation, that is „intelligent mirror" includes the capability of mirroring properly the slanted lines of high direction signs, and it also takes into consideration the later analysis of notation, therefore a righ side direction can not be inserted on the left side as a „negative" (mirrored) image of the right, but the right side direction sign should be replaced by its identified opposite.
I decided to develop an application to speed up symbol input, and create a notation drawing which can be analyzed later. An example of analysis, a search for dance structures can be carried out by Labanatory, a software application also for AutoCAD, developed by Gábor Misi.

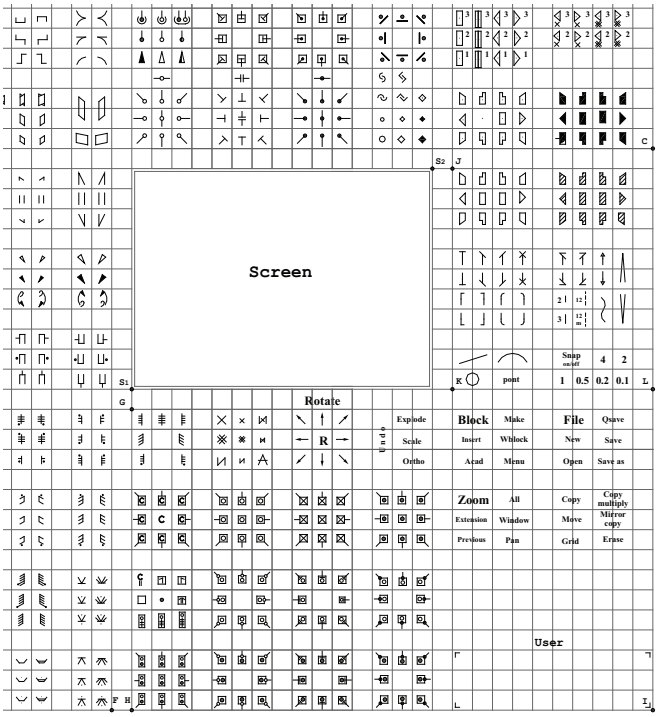

Fig. 1

LabanGraph 4P is built of a set of symbol blocks and an interpeted collection of functions written in AutoLISP language. The symbol set contains about 1400 items (some of them only help editing, but not signs for insertion) at the moment, but the set is far not full, not to mention the symbols for Motif, which are completely missing from the present set. The speciality of the blocks are their naming convention. Indentification of the blocks (except that of staves) follows a unified structure, always derived from a logically arranged matrix of symbols as shown e.g. in Fig. 2.

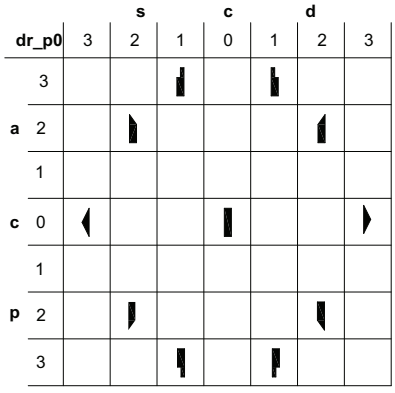

Fig. 2

The ID for e.g. forward low on the right side is **dr_p0_a3d1**. The first two characters „**dr”** are for the class (directio), the second two characters for the subclass (**p** = profundus, that is deep; **0** for the basic matrix). The „a3d1” stands for the location in the matrix, a = anterior (forward), d = dexter (right), and the digits show, in which column the symbol is placed. (As it can be seen, the classes have ancient latin names.) The number of rows and columns of the symbol matrixes can be different following the features of the signs, but the syntax of the names remain the same The naming convention helps programming symbols manipulation. From the point of the future notation analysis it is a basic requirement that all graphical elements be indentified, because line and cirlces can not be interpreted as Labanotation symbols (just like with text, the softwares recognize the letter „o” by its ASCII code and not as a cirlce).

It was a challenging task to find the tool for a fast symbol input. I feel the LabanWriter way of using the screen menus a bit overwheming. Usually many if not all the symbol class menus are needed, to seem them in an arranged and a not-covering-each-other way in a traditional single display is difficult of not impossible. A second monitor can be a solution, but not all machines are capable handling two monitors, also it is an extra cost, but mainly finding a symbol in a jungle of screen menus, or always changing the direction of attention from one monitor to the other slow down editing. I needed someting else.

For positioning signs on the screen the mouse is an excellent device. It is important not to change devices, because it makes editing difficult, therfore the right hand is for the mouse and stays there. For the left hand alone the traditional keyboard is two complex, the shifted positions of the buttons made difficult their use. I selected a special keyboard originally developed for games, now I use the type (Logitech G13) shown in Fig. 3.

Fig. 3

Fig. 4

Fig. 5

The keys can be identified to keystrokes or macros, I set the central 12 keys to correspond to letters, as it can be seen in Fig. 4. These letter issued in AutoCAD invoke basic commands such as „l” for „line”, „a” for „arc”, „c” for „circle”, and in

case of need the commands can be released easily from the special keyboard. But for the user of LabanGraph the keys are identified primarily as symbol classes and not as characters according to Fig. 5. The central 12 keys as the „zoodiac" of the symbol classes form the Main Symbol Selector as it is shown in Fig 6. First row from left to right: staves, pins and spaces, directions, foot hooks; second row: body parts, joints, space measurement, holds-carets; third row: surfaces, paths, rotations, dynamics. (I tried to arrange the classes to serve a comfortable reach for the frequently used signs.)

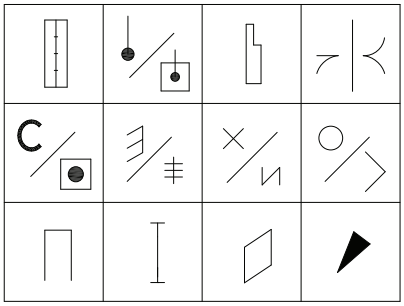

Fig. 6

The selection of a class „transforms" the keyboard to a subclass level, and the trasformation goes on until the desired sign is reached as it can be seen in Fig. 7. Selection of the „Direction" button transforms the zoodiac panel into a level selection panel. When the level is selected, the actual direction can be set. To stick to the example given above when the naming convetion was explained, for inserting a forward low direction sign in the reality the program interprets the typed „cxcp" character sequence. The user doesn't need memorizing the characters sequence, rather the change of panels.

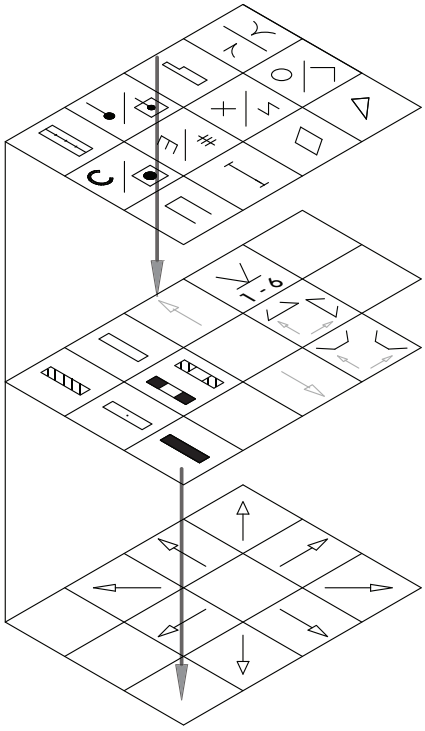

Fig. 7

At the beginning a chart is usefil which summarizes the Main Symbol Selector with all the subsequent panel. This summary is shown in Fig. 8.

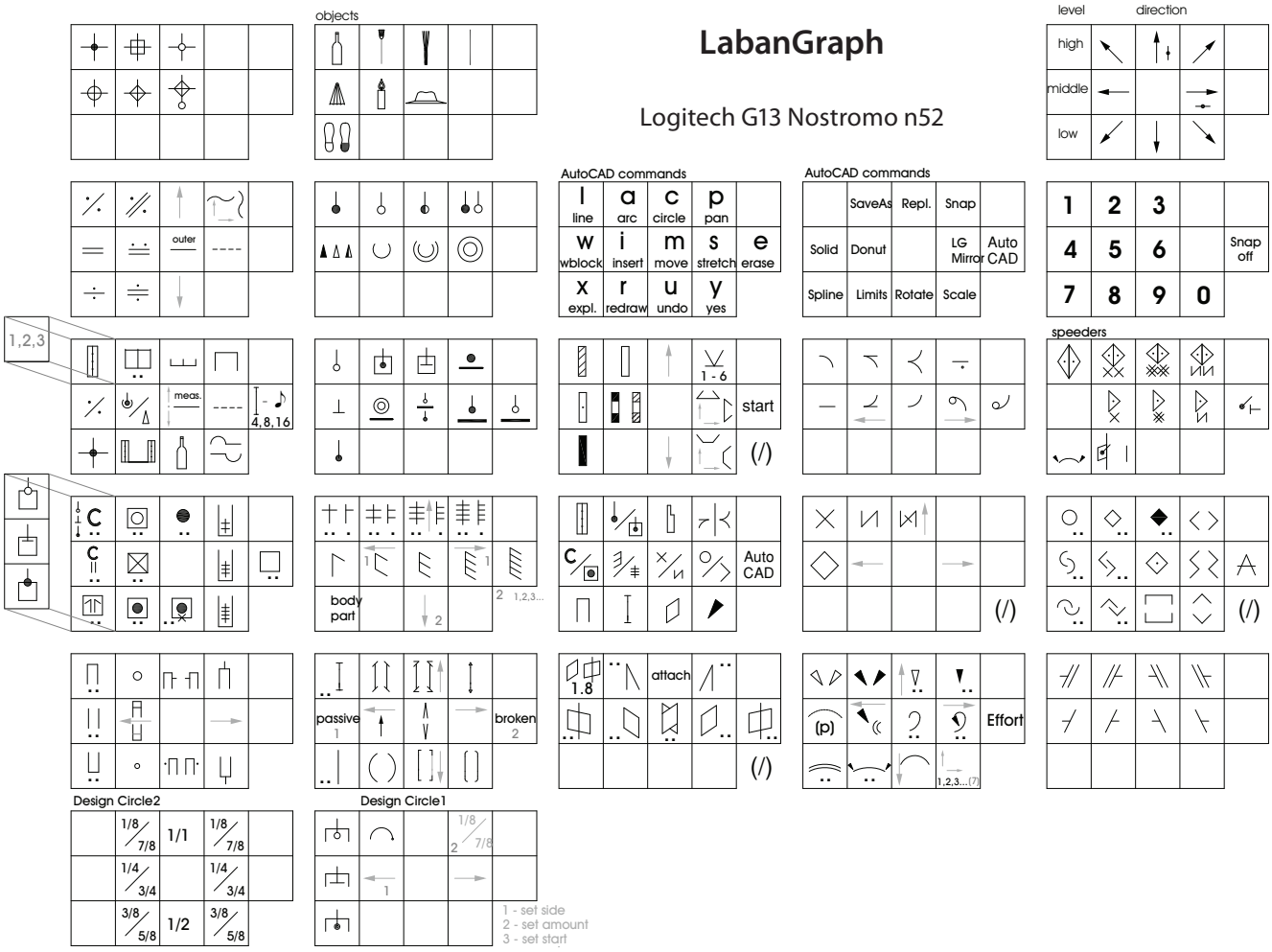

Fig. 8

Though LabanGraph works well, the separate panel overview needs certain memorizing and a comparatively advanced knowledge of Labanotation to be able to use it. I had a course on it with my students, the basics could be learnt during two sessions, each lasting two hours. Of course, for using it needs some elementary knowledge of AutoCAD itself.

Functionality

It can be debated (it was debated on the Labantalk mailing list), whether for editing Labanotation an original, independent and self-developed software is needed (possibly with open-code), which is not depending on a commercial one, or we can rely on the „mercy" of the software giants. I try to compare the advantages and disadvantages of a computer application for a commercial software (e.g. AutoCAD):

Advantages:
1. The developer follows the constant and rapid changes of IT world. An earlier developped application must runs with the new versions.
2. Commercial CAD softwares run on both platforms (PC and Mac).
3. Printing quality meets any professional standards.
4. Applications can be written in many compiler languages.
5. Application directions can be shared (e.g. Labanatory for search and analysis, LabanGraph for editing).

Disadvantages:
1. Difficulties if the developer breaks. (In case of AutoCAD, there are such a huge amount of dwg file formats all over the world, that solutions definitely willbe found.)
2. A commercial program costs money, AutoCAD is especially expensive. (Though educational institutions can purchase it on a discout price, about 10% of the original price.)
3. Commercial softwares are usually highly complex, learning it needs considerable time.
4. For years I had trouble getting a solution to import AutoCAD graphics into the generally used publishing softwares (e.g. Quark, nowadays InDesign). (It takes several steps to achieve it, but its quality worth the struggle.)

The Hungarian team interested in developing Labanotation applications decided to follow the „application for a commercial program" direction.