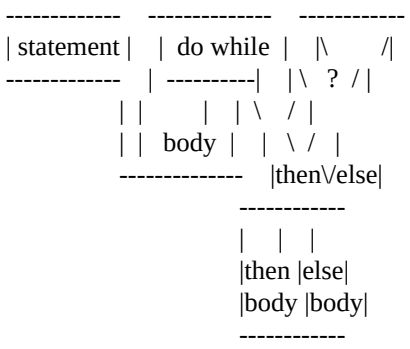
ALBIZURIROMERO84
Albizuri-Romero, Miren B.; A Graphical Abstract Programming Language; ACM SIGPLAN Notices 19, 1 (January 1984) pp. 14-23.

This article presents a graphical language for describing program structures. GAL (for Graphical Abstract Language) is based on the Nassi-Schneiderman structured flow charts which provide the usual looping and conditional constructs and do not allow GOTOs. These flow charts do seem much better than conventional ones and tend to closely mirror the control structures in the actual programs. For programming in the small, flow charts can more directly represent the programmer's thinking than a linear stream of text. Abstraction and modularization are easier at the statement level, and it is clearer where more detail is needed.

Some examples of GAL's structures are:




There are about 10 different structures which cover all the basic control mechanisms of Pascal. To sequence statements, the boxes are simply put one on top of another; no arrows are used.

GAL programs have the interesting characteristic that most of the syntax of conventional languages is eliminated. GAL itself is only an "abstract" programming language, which means that it is an informal notation that "embodies in a practical and tested form the theory of structured programs". In order to be used, of course, GAL needs a graphics program to help with input. This is described in the paper "GRASE--A Graphical Syntax-Directed Editor for Structured Programming," by Miren Bogona Albizuri-Romero, SIGPLAN Notices. 19, 2 (Feburary 1984) pp. 28-37. GRASE supports Pascal programming, but the output is shown in the GAL format. Although the user interface to this editor is extremely poor, it demonstrates that GAL structures can be generated automatically.

The actual GAL language is not especially exciting because it maps directly to conventional languages like Pascal. The interesting part of this article, however, is the idea of using graphics to enhance the programming task. Programs may be easier to construct, understand, and maintain if presented in a graphical manner, and future research might investigate other graphical languages and whether or not they make programmers more productive.