USING CAD FOR GENERATING ARCHITECTURAL FORM: REVIEWING AND TRANSLATING PIONEER PROGRAMS

Gabriela Celani, Assistant professor, School of Civil Engineering, Architecture and Urban Planning, State University of Campinas, BRAZIL.

Kaya Lazarini, Undergraduate student, School of Civil Engineering, Architecture and Urban Planning, State University of Campinas, BRAZIL.

Abstract: The aim of the present research was to review and translate into a more contemporary programming language some of the pioneer shape generation programs published in the book The Art of Computer Graphics Programming, from 1987, by William Mitchell, Robin Liggett and Thomas Kvan. The implementation of ten programs from the book was developed by an architecture student under the advice of an architectural design and CAD-programming professor, as part of an initiative to encourage scientific investigation among undergraduate students at the State University of Campinas. Besides being an opportunity to apply her programming skills, the translation of the programs helped the student understanding the possible applications of programming in the process of design.

Keywords: Pascal programming; VBA programming; computational design; computational concepts in architecture.

1. INTRODUCTION

The aim of the present research was to review and translate into a more contemporary programming language some of the pioneer shape generation programs published in the book *The Art of Computer Graphics Programming*, from 1987, by William Mitchell, Robin Liggett and Thomas Kvan. The book presents a structured introduction of Pascal programming for architects and designers, emphasizing shape generation concepts such as symmetry, geometric transformations, parametric variation, recursive repetition, etc. The proposed exercises at the end of each chapter show the authors' interest in expanding the scope of the book to a deeper discussion about the nature of architectural form.

The implementation of ten programs from the book was developed by an architecture student under the advice of an architectural design and CAD-programming professor, as part of an initiative to encourage scientific investigation among undergraduate students at the State University of Campinas. To develop this study, the student received a one-year grant from the University. Before translating the programs, the student took an elective class in VBA programming for AutoCAD, popular CAD software, and reviewed theoretical references on *computational design*. Computational design is an interdisciplinary field of research that studies the use of digital means in design and aims at developing a computational theory of

design. It involves different disciplines, such as architectural and shape-generation theories, design methodology, design cognition, computer science, cybernetics and automation.

Besides being an opportunity to apply her programming skills, the translation of the programs helped the student understanding the possible applications of programming in the process of design. She started seeing computer-aided design (CAD) programs as a real means of expression and creative development and not simply as a sophisticated tool for representation, which is a common belief among architecture students.

2. COMPUTER-AIDED DESIGN (CAD)

Computer-aided design software started being developed right after the birth of the first commercial computers in the 50's. Since the early 60's it is possible to describe at least eight generations of CAD (five of which are described in Mitchel, 1999). Ivan Sutherland's SKETCHPAD, presented as a doctoral thesis at MIT in 1963, is often cited as the very first interactive CAD system, which combined data-manipulation with graphic displays (Foley 1997, Phiri 1999, Mitchell 1977). The first large-scale CAD systems were developed for the automobile and aerospace industries. With commercial mainframes, tailored CAD applications were installed at a few architectural offices by the end of the decade. At that time CAD systems were operated by technicians and had little influence in everyday life. In the 70's, with more affordable 16-bit minicomputers, a second generation of commercially available CAD systems began. The 80's saw the parallel development of three different generations of CAD, and an intention to realize its potential through new methodologies and, such as architectural expert systems. Those were "rule-based and/or frame-based systems that provided the means for codifying the problem-solving know-how of human-experts" (Schmitt 1987).

The fourth generation of CAD was developed to be used in the new 16-bit IBM and Apple Macintosh personal computers, finally making CAD affordable to small firms and independent architects. The PC's had appeared in the early 70's, but were not powerful enough for CAD applications until the 80's. The simplification of CAD systems for personal computers resulted in a radical change in the CAD culture, which Mitchell (1999) identifies as an inflection point in the history of architectural CAD, after which "the wider possibilities were largely ignored." (op.cit., p.483). The new standardized, general-use CAD applications for PC's did not aim to help architects from the early, conceptual stages of design. Such

specialized applications remained restricted to academic research, while the great majority of offices kept the use of CAD restricted to drafting and representation.

The fifth generation of CAD was related to the development of a new kind of computers in the 80's: the graphic workstations. But y the end of the decade, there was no substantial difference between PC's and workstations both in price and in performance. The introduction of 3D modeling, rendering and communication capabilities to the originally simplified PC CAD systems was made possible by the increasing power of microprocessors. As opposed to the first, standardized systems, the new ones have different levels of complexity and capabilities, as well as different prices, to fit specific areas, such as drafting, graphic-layout, modeling, rendering, and surveying (not to mention applications out of the architectural scope, such as movie-making, GIS and project design). In the 2000's a last generation of parametric CAD software emerged, with new data handling capabilities.

Over the years, CAD became increasingly complex, making it impossible for architects to develop their own applications. It takes large teams of software engineers, programmers, graphic designers, and sometimes architects to developed a CAD system. Besides, there has been a drastic change in the primary objective of CAD from a problem-solver to a drafting, representation and communication tool.

The aim of the present research was precisely to rescue the original objective of CAD software, as an aid in the early creative phases of design, and not just as a representation tool, by reviewing and pioneer shape generation programs and adapting them to one of the most popular CAD programs used nowadays by architects and engineers: AutoDesk's AutoCAD.

3. PASCAL

Pascal is a computer programming language developed in the 70's by Niklaus Wirth. It is an imperative or procedural language, in which programs can be seen as a sequence of commands for the computer to perform.

The structure of a typical a Pascal program is shown in

Figure **1** where the words in bold face are key words or reserved words.

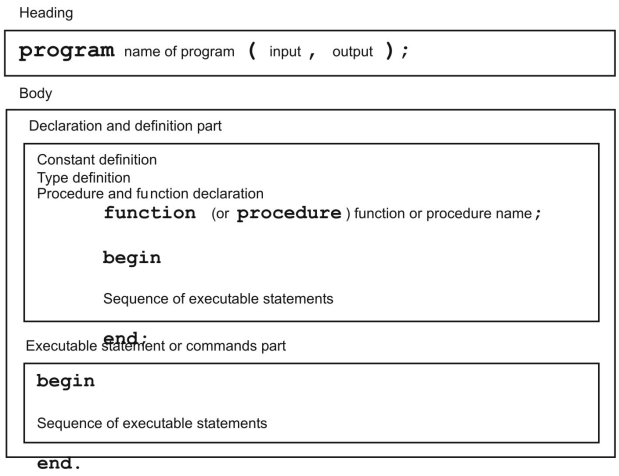


Figure 1: Typical structure of a Pascal program (after Mitchell, 1987, p.114).

Pascal programs have two main parts: the heading, where the name of the program is stated, and the body. After the name of the program, its required input and expected output (or just the expected output, in the case that there is no input needed) must be specified inside parentheses.

The body part consists of two essential parts. The first one (declarations and definitions) corresponds to the description of the data that will be used in the program. The aim of this part is to allocate the necessary memory for the data that will be used during the execution of the program (run time). Different types of memory use can be specified here. Variables, for example, can be redefined during run time. Constants, on the other hand, have a fixed value throughout execution. The type of each piece of data must also be declared. Pascal has certain predefined data types, but it is also possible to specify new ones.

In the declarations and definitions part it is possible to define constants, such as:

```
PI = 3,1417;
```

It is also possible to define new data types, and to write functions and procedures, small sequences of instructions that perform simple tasks, such as:

```
FUNCTION TAN (THETA:REAL):REAL;
    BEGIN
        TAN:=SIN(THETA)/COS(THETA)
    END;
```

Notice that the function or procedure declaration part uses the reserved words "function" (or "procedure") and "begin"/"end" to delimitate the sequence of instructions. But the most important aim of the declarations and definitions part is to declare variables, by stating their names (such as X and Y) and types (such as integers, reals, booleans and strings):

```
VAR X, Y: INTEGER;
```

In the second part of the body (executable statements or commands part) the actions that will be operated upon the data are stated within the reserved words "begin" and "end". Statements can be made of commands, conditionals, repetitions and expressions, which are made of data and operators.

If the program requires any input information, the READLN command can be used to read numbers typed by the user. Similarly, the WRITELN command allows to output alphanumeric data on the screen. The sequence of statements below asks the user to type in three numbers, and then reads them to use them as data to perform other tasks:

```
WRITELN ('TYPE IN 3 INTEGERS');
READLN (A,B,C);
```

Although standard Pascal does not include commands to generate graphic output, there are many implementations of Pascal that provide such functionality. Typical graphic output commands are:

```
MOVE (X,Y);
DRAW (X,Y);
```

Pascal has arithmetic (multiply *, add +, subtract -, divide /, etc.), inequality (greater than >, smaller than <) and logical operators (AND, OR and NOT). A simple statement with an arithmetic operator is:

```
AREA:=LENGTH * WIDTH;
```

A simple statement with an inequality statement is:

```
X < 0
```

In this case, the result is a Boolean value, i.e., either true or false. This type of expression can be used, for example, in a conditional statement, such as:

```
IF X < 0 THEN
     WRITELN ('X IS NEGATIVE')
ELSE
     WRITELN ('X IS POSITIVE')
```

Commands and entire statements can be repeated with structures such as:

```
FOR X:= 1 TO 10 DO
     WRITELN (X);
```

In the executable statements or commands part it is also possible to use the functions and procedures defined in the declarations and definitions part, as well as other functions provided by Pascal. Most of them perform arithmetic operations, such as in a calculator:

ABS(X) = absolute value of x

SQR(X) = square of x

SQRT(X) = square root of x

SIN(X) = sine of x

COS(X) = cosine of x

4. VISUAL BASIC FOR APPLICATIONS (VBA)

Visual Basic for Applications (VBA) is a Visual Basic implementation that has been incorporated to Microsoft Office applications and other programs, such as Word, Excel and AutoDesk's AutoCAD. VBA is not exactly a programming language, like Pascal. It is called a scripting language, because it is not compiled (i.e., converted into binary executable files) - it is interpreted by the application every time it is run. Scripting languages make applications programmable from within, allowing automating repetitive tasks. The greatest difference between Pascal and VBA is that VBA is object-oriented, which allows to create new classes of objects and to manipulate them by referring to their properties and methods. However, it is also possible to use VBA in a procedural way, much like in a Pascal program, by simply sending sequences of instructions to AutoCAD's graphic editor.

Before AutoCAD's version 2000, it was possible to develop scripts in AutoCAD with another scripting language, Auto Lisp, a version of Lisp (List Processing Language) developed especially for AutoCAD. However, Auto Lisp presented the disadvantage that it required another scripting language - *dialog control language* (DCL) - for creating graphic interfaces (also known in AutoCAD as dialog boxes), and still a third scripting language for defining menus. In AutoCAD's VBA editor (called VBAIDE) it is possible to intuitively design complex graphic user interfaces and automatically link them to a VBA code page.

A VBA program can be composed by as many code pages as necessary. There are two types of code pages: the first type, called Module, is a stand-alone code page, and the second type, called User Form's code page, is associated to a graphic interface unit. Modules can be of two types: regular Modules and Class Modules, where new classes can be defined.

The structure of a VBA program is similar, in some aspects, to Pascal. Although not mandatory, the top part of a code page can be used to declare variables and constants, much like in a Pascal program. Under the declarations, it is possible to define functions. Alternatively, functions can also be defined after the executable part, or in a separate code page. The executable part contains a type of procedure called Sub (which stands for sub-

routine), which is a sequence of executable instructions. This part can have different types of expressions, conditionals (If/Then/Else), code repetitions, commands, etc. Figure 2 shows the typical structure of a VBA Module. Just like in Pascal, VBA also uses reserved or keywords, which are displayed in bold face in the figure.

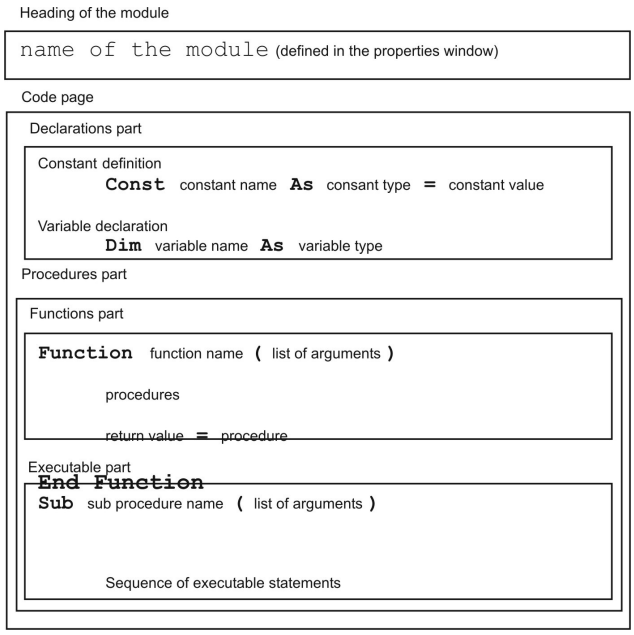


Figure 2: Example of a typical VBA code structure in a Module.

The basic VBA data types are similar to Pascal's. They can be Integers, Booleans, Strings, etc. However, VBA also features a large number of interface object types (such as User Forms, Command Buttons, etc.) and AutoCAD specific object types (such as ACADLines, ACADCircles, ACADLayers, ACADLineStyles, etc.). Each of these data types has their own properties and methods (functions), which are already defined in the system.

In regards to syntax, VBA uses the "dot syntax", which is a typical way of writing object-oriented programs:

```
objectname.property(value)
objectname.method(parameters)
```

However, it is also possible to simply write series of AutoCAD commands as strings (much like when using AutoCAD's command prompt area) by using the SendCommand command, as shown below:

```
ThisDrawing.SendCommand ("line 0,0 5,10  ")
```

Like in Pascal, VBA language also includes similar arithmetic, inequality and logical operators and pre-defined functions, such as the Date function, which returns the current system date. VBA offers more options than Pascal in terms of repetition structures (Do

While... Loop, Do... Loop While, Do Until... Loop, For... Next, For Each... Next), but the basic concept is similar.

AutoCAD's VBA development environment (VBAIDE) is interactive and automatically provides a list of available properties and methods after the name of an object is typed. The environment also provides a useful help with examples of codes.

5. CHOOSING THE PROGRAMS

Ten programs from the book were selected for being translated. The programs were chosen according to the following criteria: (1) they had to have different levels of difficulty, (2) they had to clearly exemplify the main characteristics of a computer program, such as the use of variables and constants, code repetition, and the use of conditionals, and (3) they had to exemplify the use of programs in the generation of architectural forms or in the solving of architectural problems.

In "The theoretical foundation of computer-aided architectural design" Mitchell (1975) proposed 4 levels of "ambition" in the use of computers in design. It is possible to say that the programs from the book The Art of Computer Graphics Programming that were translated in this research fall into category 3.a. in Table 1.

1. Representation	"The least ambitious level of use of the machine is to allocate to it only tasks of representation (...), and to leave all solution-generation and solution-evaluation responsibilities to the human designer" (Mitchell, 1975, p.354).
2. Evaluation	"Rather more ambitiously the machine might be allocated the task of evaluating solutions produced for consideration by the human designer" (ibid.).
3. Automated generation of solutions	
a) In well-defined problems	"Where both the set U of potential solutions and the solution criteria can be clearly specified, that is where the problem is completely well-defined, then the design process can potentially be completely automated" (ibid.).
b) In ill-defined problems	"The most ambitious potential level of use of the machine is to attempt to develop systems capable of dealing intelligently and flexibly with ill-defined problems, that is of displaying the capabilities characteristic of a good human designer" (ibid.).

Table 1: Levels of "ambition" in the use of computers in design.

6. TRANSLATION METHOD

After the study of the main characteristics of Pascal and VBA, a comparison table was produced to help in the translation from one language to the other. Some of the programs chosen were actually exercises proposed at the end of each of the book's chapters. In this case the work started with the development of an abstract program structure to perform the required tasks.

The translation process started with an analysis of the Pascal program and the identification and color-coding of each of its main sections, such as constant, function and procedure definitions, variable declarations, executable statements, and the presence of code loop and conditional structures, as shown in Figure 3.

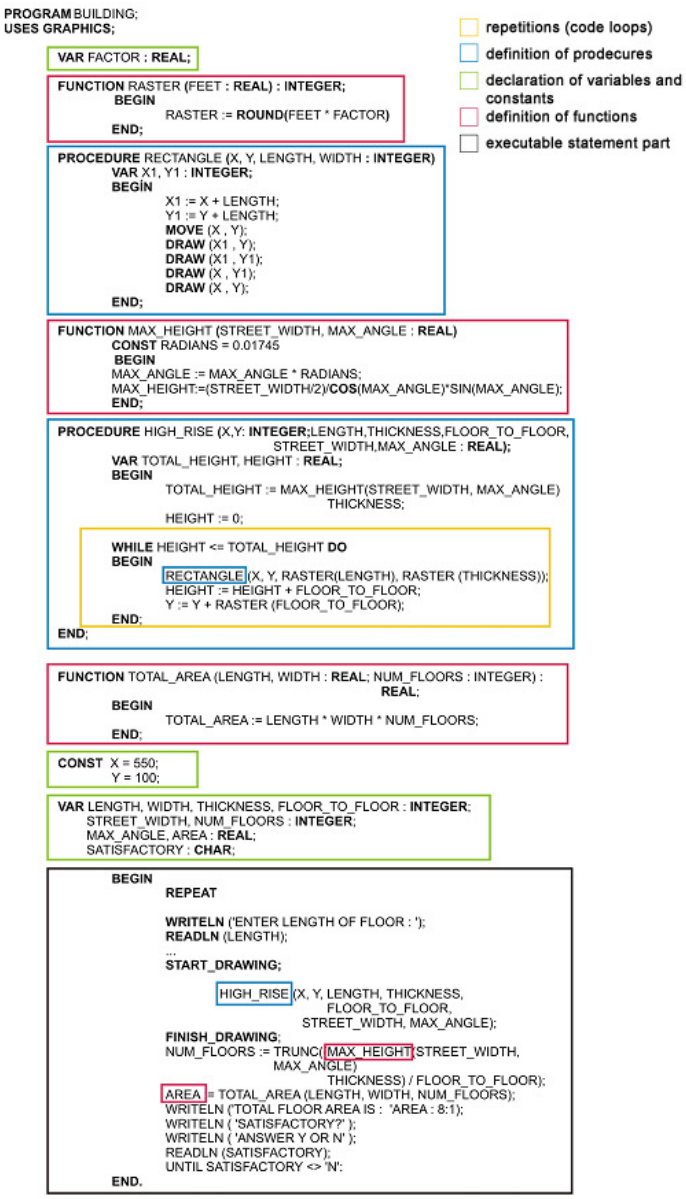


Figure 3: Color-coding of the main sections of one of the Pascal programs that were translated.

After the analysis, the same variables, functions and procedures were created in a VBA Module, and the same executable statements were re-written as sub-routines. Two examples of the translated programs, with different levels of complexity, are described below.

A program called "Mies Van Der Rohe" (which was actually an exercise at p.198) draws parametric versions of the Brick Country House plan, based on the cross shape. The program uses random values to place the walls and define their lengths.

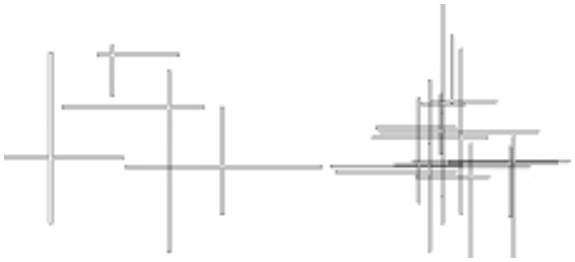
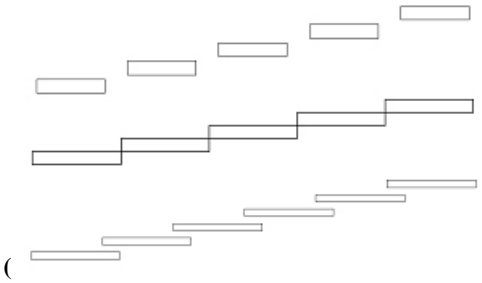


Figure 4: Exercise proposed at the end of chapter 8, and two different wall arrangements produced with the "Mies Van Der Rohe" program.

The program called "Stairs" (p.221) draws stair profiles in a parameterized way



(

Figure 5). It implements a repetition structure (For/Next). The program takes as arguments (input values) numbers typed by the user for the X and Y values of the initial step, the length, width and height of the threads, and the total number of steps.

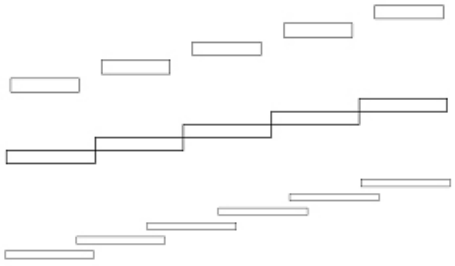


Figure 5: Three different stair profiles produced by program "Stairs".

The first program is an example of an implementation for an ill-defined architectural problem, in which many different plans in the Miesian vocabulary can be randomly generated by the computer, and then evaluated by a human designer. The second program is an example of a

typical well-defined architectural problem, in which space and ergonomic constraints lead to a single solution.

The other programs translated during this research were Square (p. 140), Snap (p.155), Draw square (p.173), Building (p.217), Roof trusses (p.249), Temple (p.289), Façade (p.295) and Column (p.331).

7. CONCLUSION

By learning a new programming language (Pascal), and translating programs from Pascal to VBA, the student responsible for the translation of the programs (Kaya) was able to improve her knowledge of VBA, acquire more confidence about her programming skills, and understand the advantages of using an object-oriented programming language in relation to a procedural language. More than that, the student was able to fully understand the objectives of the book and the range of possibilities in the use of computer programming for generating architectural form.

All the programs translated in this research are available for download at http://www.fec.unicamp.br/~celani/pascal.htm

8. ACKNOWLEDGEMENTS

The authors would like to thank computer scientist Fábio Azevedo, for his help with the programming, and UNICAMP's *Serviço de Apoio ao Estudante* -SAE (Students' Support Service) for providing the one-year scholarship to Kaya.

9. REFERENCES

Mitchell, W. J. (1975) The theoretical foundations of computer aided architectural design. Environment and Planning B, 2. 127-150.

Mitchell, W. J., Liggett, R. S., Kvan, T. (1987) The Art of Computer Graphics Programming, a Structured Introduction for Architects and Designer. New York: Van Nostrand Reinhold Company.

Foley, J. D., Van Dam, A., Feiner, S. K., Hughes & J. F., Phillips, R. L. (1997) Introduction to computer graphics. Reading, MA: Addison-Wesley.

Phiri, M. (1999) Information technology in construction design. Bath: Thomas Telford.

Schmitt, G. (1987) Expert systems in design abstraction and evaluation. In Y. Kalay (Ed.) Computability of design. (p.213-244) New York: John Wiley and Sons.

Mitchell, W. J. (1990) “A New Agenda for Computer Aided-Design. In McCullough, M. The Electronic Design Studio Cambridge, MA: The MIT Press, p.483.

Mitchell, W. J. (1977) Computer-aided architectural design. New York: Petrocelli/Charter.