Univ. u Novom Sadu
Zb. Rad. Prirod.–Mat. Fak.
Ser. Mat. 24, 1 (1994), 321-329

Review of Research
Faculty of Science
Mathematics Series

# EDUCATIONAL "TURTLE" - USING "TURTLE" GRAPHICS IN COMPUTER EDUCATION

**Zoran Putnik**
Institute of Mathematics, University of Novi Sad
Trg Dositeja Obradovića 4, 21000 Novi Sad, Yugoslavia

**Abstract**

In this paper we will show an implementation of turtle graphics features in functional programming language LISP, version LispKit Lisp, implemented at the Institute of Mathematics in Novi Sad [1], [2]. Great value of using graphics, specially turtle graphics, abilities in education - programming languages, functional programming, recursion and program modularity will be presented. Some elementary details about turtle graphics and it's implementation will be explained for better understanding of its power in process of education.

*AMS Mathematics Subject Classification (1991):* 68U05, 68N15
*Key words and phrases:* Education, Programming languages, Functional programming, Turtle graphics

## 1. Introduction

For any kind of education, it showed very suitable for better understanding of knowledge that is to be acquired, using graphical elements as much as it is possible. Starting with most elementary kind of graphics (pictures) used with given information, through more sophisticated models (simulation, animation), to individualistic creation of graphical elements by pupil, ability of visual observation of information helps in many ways in it's easier understanding. This stands as much for most elementary fields of education, as for such abstract subjects concerning computers, programming or even some more specific details of programming languages and techniques.

## 2. Preliminaries

In almost any existing programming language which has graphical abilities, there are two basic methods of using graphics: graphical statements for drawing of elementary geometric shapes and, so called, "turtle" graphics. Turtle graphics in the beginning has been introduced as a part of a programming language Logo and can be described as an "animal" (for example turtle), which crawls across a computer screen, leaving a trail, or not [3]. It's basic intention was to be used for beginning education in programming. It was considered much more suitable for children education because of (usually) more developed imagination then with elders, which enforce main advantages of turtle graphics to be used easier. Basic statements for use in turtle graphics are statements for moving turtle forward and backward, for turning turtle for a given angle and for defining whether turtle will in consecutive moves leave a trail or not. All mentioned advantages of using turtle graphics, can shortly be stated as:

- - helps in developing procedural manner of thinking
- - enables easy following of a drawing method
- - drawing results are immediate
- - it is more natural to involve recursion - one of most powerful methods in programming

Since at the Institute of Mathematics has been implemented LispKit Lisp version of a functional programming language LISP and functional programming languages are very suitable for acquiring earlier mentioned principles, turtle graphics has been implemented as a part of it.

Implementation of basic graphical functions in a LispKit Lisp programming language, for drawing a point, line, circle and ellipse have been described in [4]. All graphical functions have been implemented as external functions (in Modula-2 programming language), [5]. Besides, function SEQ, for sequential execution of given functions is available in this version of LispKit Lisp.

Only one of mentioned, function for drawing a line, is sufficient for implementation of turtle graphics. In addition, for successful implementation,

some more information is needed. Generally, it can be stated that any version of a programming language LISP, extended with:

- - graphical function for drawing a line
- - access to some global variables

can be used for implementation of turtle graphics. Details about implementation are given in [6]. As much as user is concerned, basic syntax of turtle graphics statement and also basic syntax of LispKit Lisp in general, are sufficient for successful usage.

## 3. Using turtle graphics in education

Choice of a functional programming language, gives some implicit advantages in education in a field of programming.

Definition and construction of functional programming languages, has a modular method of programming as only possible. Each function, performing certain part of a job, is defined separately (as an independent module - function) in a program. Extensions and improvements of this version of a LispKit Lisp comparing to an original version [7], enforce this element even more. In this version of a LispKit Lisp, it is possible to create libraries of useful functions, which are separately developed and tested, so that latter usage of these functions is restricted only to citation of a function name and a name of a function library in which it is stored.
Using recursion in programming is a nature of functional programming and it's most powerful part. With this method used, lots of unreadable and long programs, become much shorter and easier to understand. Sometimes, without use of recursion, some problems can't be solved. To confirm this statement, two examples will be given.

- a) Let us suppose that we want to draw a rectangle on a screen, with a length of each side equal to N pixels (points), starting from coordinate (X,Y). Using a basic set of graphical functions, program would be as follows:

```
Line X,Y      To X+N, Y;      Line X+N,Y   To X+N,Y+N;
Line X+N,Y    To X+N,Y+N; or  Line X,Y     To X+N,Y;   or...
Line X+N,Y+N  To X,Y+N;       Line X,Y+N   To X,Y;
Line X,Y+N    To X,Y          Line X+N,Y+N To X,Y+N
```

Same program, using turtle graphics functions is:

```
JumpTo X,Y;
Repeat 4    {Line N;
             Turn 90}
```

(ie. go to coordinate (X,Y), draw a line of a length N in a given direction and then turn 90 degrees. Repeat this four times.)

- b) Let us suppose that we want to draw a polyhedra of M sides, with a length of each side equal to N pixels, starting from coordinate (X,Y). Using a basic set of graphical functions, this problem can not be solved. There is not enough information for calculation of beginning and ending coordinates for each line, it is even not known in advance how many lines should be drawn, so with a procedural programming, this picture can not be achieved.

  Using turtle graphics, problem is solved almost identically as the first one:

```
JumpTo X,Y;
Repeat M    {Line N;
             Turn 360/M}
```

While first example could be a matter of discussion and programmers preference, second example proves a need for existence and usage of turtle graphics in creation of even slightly more complicated pictures. Also, it should be noted, that first example, with usage of basic set of graphical functions, can be executed in any order of a given four statements. Excluding the first one, none of the others have any value in developing a procedural thinking method, on the contrary, it can only be confusing to a beginner. Using turtle graphics and recursion, all problems from this class, are solved in a very similar and easy to remember, method.

As much as education is concerned, several things can be noted. Besides natural involvement of recursion in a drawing process it is easy to see

how obvious results are. In mentioned examples, in execution of drawing a rectangle (or polyhedra) on a computer screen, pupil can easily follow turtle movement and compare it with a given program. Obvious correspondence of these two things, clearly connects recursion (used in a program) and turtle behavior (movement on a screen), so that recursion is in a best manner explained to a pupil and assembled in his imagination.

After accepting this program (via usage of a REPEAT loop) in which nature is involved recursion, it is relatively easy to make a step forward to a method of recursion most difficult to understand - selfrecursion. Given program, with a relatively small level of changes can be turned into the following:

```
Procedure Rectangle (N, I)
  If I = 4
      Then
        Stop
      Else
        {Line N
         Turn 90
         Rectangle N I+1}
  EndIf
```

In this phase, pupil is ready to, comparing a program and turtle movement on a screen, accept this, most classic, but still most difficult to understand, recursion method.

As much as modularity is concerned, on a simple example will be shown how turtle graphics, implemented in a functional programming language, helps in comprehending this field.

Sketch of a house can be represented by already given function for drawing a rectangle, and a picture of a triangle (house roof), on the top of a rectangle. Program for drawing a triangle with turtle graphics functions in a LispKit Lisp, can be achieved via given "technique" for drawing a polyhedra:

```
Procedure Triangle
Repeat 3    {Line N;
             Turn 120}
```

Appending these two functions, as separate modules in a main program, can be in a pseudo-code represented as:

```
Procedure House
  {Rectangle,
   Triangle}
```

Execution of this functions (modules), can be easily followed on a screen. Picture is achieved, part by part, giving a complete sketch. On the other side, comparison of a picture on a screen and written program is simple and analogy is obvious. Since each of the modules can work as a separate program (function) and can be used in any other program, separation of a program into functional parts and separate programming ( = modular programming) becomes a natural way of thinking.

Method for creation of separate modules in (most of) the other programming languages, specially ones of procedural type, is usually complicated and asks for separate connecting (linking) of modules (ie. library functions and procedures) with a main program. As a consequence, it often happens that for a pupil it is easier to program again even those functions and procedures which already exist in a program libraries, to achieve complete programs. Similarly, using a function/procedure which pupil created in one program in some other program, usually is just a "cut and paste" method. Since this method can easily have a consequence of bad notion and fancy about modular programming, it is not specially needed to emphasize a affirmative influence which in this field functional programming and usage of turtle graphics can have.

## 4. Conclusion

Power of visual methods in showing and explaining certain subjects, has a greater value as a field which is to be learned, more complicated. At the beginning of a course in programming, subjects as recursion, modular programming (and many others), are fields too complicated to be explained efficiently without a usage of pictures, animation, simulation or some other visual method.

On the other hand, method of individual, independent work, specially with creative results, showed it's value many times. In a field of computer

education, when it is relatively easy and simple to enable individual work on a computers and programming of a small functions which show most important attributes of a given lessons, this method is specially useful.

Suitable choice of a programming, language which implicitelly contain most of up-to-date trends in development of programming, as much as all the classic methods of programming - which is a great deal truth for a functional programming languages, can show as a very important thing in creating future programmers "habits" and "programming styles".

Because of simple usage and small hardware requirements, and also already developed additional modules for usage of classic computer graphics and turtle graphics, LispKit Lisp version of a programming language LISP, implemented at the Institute of Mathematics, is a very suitable choice. Needed programming support, LispKit Lisp simulator of a SECD machine and LispKit Lisp executor, are used at the Institute for a long time for education in functional programming and programming in general. As a literature, textbook "LISP by examples", in which all basic programming problems are solved, and all basic programming techniques are shown, can be very helpful for pupils, and also for teachers on computer subjects.

In addition, real-life experience in work with students, show correctness of this concept. Education in a field of computer science, specially learning functional programming languages, with all their main characteristics improved since LispKit Lisp version of a programming language LISP, expanded with computer graphics abilities and turtle graphics, have been introduced. Simple usage of existing functions, initiate selfinitiative adding of new functions, specially graphical ones by students, existing programs for demonstration of graphical abilities of a LispKit Lisp and functions for geometric manipulation of picture representations are extended with new functions. Functional programming, earlier "difficult", "unlogical" and "unattractive", become excellent background for development of creative work for interested students.

## 5. Appendix

List of all basic statements for turtle graphics:

- Forward N - moves turtle N pixels in a given direction and (depending on a pen position) draws a line or not
- Backward N - same as for function Forward, except that turtle is moving in an opposite direction
- Turn N- changes turtle direction, relative to a previous direction, for an angle of N degrees
- Up - Lifts a pen. From now on, turtle will not leave a trail while moving.
- Down - Puts a pen down. From now on, turtle will leave a trail while moving.

List of all additional statements for turtle graphics:

- DrawTo X Y - Draws a line from a current position of a turtle to (absolute) position given by coordinates X and Y. Direction of a turtle is not changed. Line is drawn, even if pen is up.
- JumpTo X Y - Similar as a function DrawTo, moves a turtle to a given position, but without drawing a line (even with pen down).
- Angle N - Turns a turtle to a given (absolute) direction.
- Home - Puts a turtle to a starting position (coordinates (0,0), turns it in a starting direction (angle of 0 degrees), and puts a pen in a starting state (down). This statement can be treated as a turtle initialization.
- Repeat N F A-Consecutive, for N times, execution of a given function F, with a given parameter list A.

## References

[1] Budimac, Z., Ivanović, M., A Useful Pure Functional Language Interpreter. In Proceedings of XII International Symposium "Computer at the University" (Cavtat, Yugoslavia) 1990, 3.17.1-3.17.6

[2] Budimac, Z., Ivanović, M., An Improved Implementation of the LispKit Lisp Language. In Proceedings of 16. Summer school "Programming '91" (Sofia, Bulgaria), 1991, 101-104.

[3] Ross, P., Logo Programming for the IBM PC, Addison Wesley & IBM UK International 1985.

[4] Budimac, Z., Ivanović, M., Putnik, Z., A Representation Scheme for Graphical Objects Based on Function Definitions in Functional Language, In Proceedings of XIII International Conference "Information Technology Interface", Cavtat, 1991, 255-260

[5] Ivanović, M., Budimac, Z., Involving Coroutines in Interaction between Functional and Conventional Language, ACM SIGPLAN Notices, 25,11 (1990), 65-74.

[6] Putnik, Z., Budimac, Z., Ivanović, M., Turtle Walk Through Functional Language, ACM SIGPLAN Notices 26,2 (1991), 75-82.

[7] Henderson, P., Functional programming - Application and Implementation, Prentice-Hall International, 1980.

[8] Budimac, Z., Ivanović, M., Putnik, Z., Tošić, D., LISP by examples, Institute of Mathematics, University of Novi Sad, 1991 (in Serbian).

**REZIME**

## OBRAZOVNA "KORNJAČA" - KORIŠĆENJE "TURTLE" GRAFIKE U RAČUNARSKOM OBRAZOVANJU

U radu je prikazana implementacija "turtle" grafike ("kornjačine" grafike) u programskom jeziku LISP, verzija LispKit LISP, koja je kreirana na Institutu za Matematiku, Prirodno-matematičkog fakulteta u Novom Sadu, [2]. Prikazan je i objašnjen značaj korišćenja mogućnosti grafike, a posebno "turtle" grafike u obrazovanju iz oblasti informatike - programskih jezika, funkcionalnog programiranja, rekurzije i modularnog programiranja. U cilju boljeg razumevanja mogućnosti korišćenja, prikazani su i neki od osnovnih detalja u vezi sa implementacijom "turtle" grafike u funkcionalnom programskom jeziku.

*Received by the editors June 4, 1991*