Modeling, simulation and visualization of real processes in LOGO programming language as a method of development of algorithm thinking and programming skills

S. Hubalovsky, M. Musilek

***Abstract*—** One of the most important methods in current scientific and technological research as well as in research of strategy algorithm and programming is modeling and computer simulation of real systems and real processes. System approach, modeling and simulation are discipline with its own theory and research methodology.

The paper focuses to the theory of modeling, simulation and visualization of three real processes – *Tower of Hanoi brain teaser*, decrypting of *the Daisy cipher* and *Placing of Chess Queens on the Chessboard*. Multidisciplinary approach is point out too.

The solution is demonstrated step by step – it starts with the problems definition, then the strategy solution analyses are shown and finally visualization of the processes is presented in illustrative form in LOGO programming language.

***Keywords*—** Algorithmic thinking, LOGO programming language, Simulation and modeling, Visualization, Turtle geometry.

I. Introduction

THE term algorithm can be connected with the terms *system*, *model*, *simulation*, *multidisciplinary approach*, which are important in current approach to scientific, technological and professional practice. Many universities are realizing that modeling and simulation is becoming an important tool in finding the strategy for solving and understanding numerous and diverse problems. The examples can be found e.g. in [1], [2], [3], [4] or [5].

In this paper we first briefly introduce the theory of system approach, modeling and simulation as a method of multidisciplinary and system approach to education of algorithm development.

Above mentioned functions of simulation will be represented in following part of the paper by three case studies - *Tower of Hanoi brain teaser*, *Transposition Cipher Determined by the Figure – The Daisy Cipher* and *Placing of Chess Queens on the Chessboard* so that no two queens attack each other.

The computer simulation of the case studies will be realized and visualized in LOGO programming language.

Stepan Hubalovsky is working at University of Hradec Kralove, Department of informatics, Faculty of Science, Hradec Kralove 500 38, Rokitanskeho 62, Czech republic, stepan.hubalovsky@uhk.cz.

Michal Musilek is working at University of Hradec Kralove, Department of informatics, Faculty of Science, Hradec Kralove 500 38, Rokitanskeho 62, Czech republic, michal.musilek@uhk.cz.

II. Modeling and simulation of the systems and processes

A. Modeling

Modeling is a method that is often used in professional and scientific practice in many fields of human activity.

The main goal of modeling is not only describing the content, structure and behavior of the real system representing a part of the reality but also describing the processes.

The process can be understood as series of transformations that changes the input values to output values. From the system point of view the process is dynamic system in which the values of the characteristic of the system elements are changed under the influence of the external elements.

The models are always only approaching of the reality, because the real systems are usually more complex than the models are. The system homomorphism is applied in the process of modeling, which means that each element and interaction between the elements of the model corresponds to one element and interaction of the modeled real system or real process, but the reverse is not true. The model is always to be understood as simplification of the original. If the relation of isomorphism is between the model and real system the original model we could not distinguish between the model and the original, which is discussed e.g. in [6] and [7].

The first step in the process of computer simulation is creation of conceptual model of the studied real system / real process. Conceptual model can be represented in different way. The most used representations are:

- Mathematical equitation;
- Process charts.

Mathematical equitations establishes mathematical model of the studied real system. The model can be obtained either theoretically based on basic physical properties of the system, or numerically by means of the measured values. Determination of parameters of theoretical model developed

from empirical data is called system identification.

Process charts establishes process model of the real process. The process models can be described by different way; the most common are flowcharts that described the algorithm of the modeled process.

The conceptual model must adequately describe the dependency system outputs on its inputs. Models of real process system will be shown in the following paragraphs of this paper.

B. Simulation

The process of modeling is closely related to the simulation. Simulation can be understood as process of executing the model. Simulation enables representation of the modeled real system or real process and its behavior in real time by means of computer (e.g. LOGO programming language). The simulation enables also visualization and editing of the model.

A typical simulation model can be written both through specialized programming languages that were designed specifically for the requirements of simulations, or the simulation model can be created in standard programming languages.

From the above considerations, it is clear that simulation is a process that runs on the computer. In some publications, therefore, can be found the term ”computer simulation“. It generally is valid that computer simulation is a computer-implemented method used for exploring, testing and analysis of properties of the conceptual (mathematical or process) models that describe the behavior of the real systems or real process which cannot be solved using standard analytical tools, se e.g. [8].

The simulation models represented by executable computer program have to be isomorphic with the conceptual model that is a representation. It means that the mathematical model and simulation model have to represent the real system, its elements, internal interactions and external interaction with the environment in the same way.

In our paper the three real processes *Tower of Hanoi brain teaser*, *Transposition Cipher Determined by the Figure – The Daisy Cipher* and *Placing of Chess Queens on the Chessboard* and computer simulation and visualization in LOGO programming language will be presented.

C. Significant function of the simulation

Simulation has from the scientific point of view several functions – see e.g. [8].

We will focus in this paper two of them and they are:

- replacing the real process;
- development of educational process.
- visualization of the process.

1) Replacement of the real process

This is an important and indispensable feature of simulations and simulation model because it allows realize a situation of the process that cannot be investigated conventionally. The main advantage of simulations is that simulations model allows providing rather big number of the process steps in relatively short time, changing of input parameters and its visualization and optimization of the process.

2) Development educational process

The simulation is very useful from educational point of view. Using the simulation model and visualization of simulation results on the screen, students can better understand the basic features of the processes and systems and develop their intuition. It is also essential that the teaching by means of simulation is much cheaper and faster than the teaching carried by real experiment. In some cases providing the real experiment cannot be feasible.

3) Visualization of real process

The term visualization is closely related to the term of simulation. Visualization is a technique that transforms, selects and shows the output data and data obtained from the simulations, allows the investigation, analysis and interpretation.

Among the commonly used tools in scientific visualization simulated data are 2D and 3D graphs, tables, flow charts, animated diagrams and more.

D. Model verification and validation

Verification and validation are important aspects of the process modeling and simulation. They are essential prerequisites to the credible and reliable use of a model and its results [9].

1) Verification

In modeling and simulation, verification is typically defined as the process of determining if executable simulation model is consistent with its specification – e.g. conceptual model. Verification is also concerned with whether the model as designed will satisfy the requirements of the intended application. Verification is concerned with transformational accuracy, i.e., it takes into account simplifying assumptions executable simulation model. Typical questions to be answered during verification are:

- Does the program code of the executable simulation model correctly implement the mathematical model?
- Does the simulation model satisfy the intended uses of the model?
- Does the executable model produce results when it is needed and in the required format?

2) Validation

In modeling and simulation, validation is the process of determining the degree to which the model is an accurate representation of the real system / real process. Validation is concerned with representational accuracy, i.e., that of representing the real system / real process in the conceptual model and the results produced by the executable simulation model. The process of validation assesses the accuracy of the models. The accuracy needed should be considered with respect to its intended uses, and differing degrees of required accuracy may be reflected in the methods used for validation.

Typical questions to be answered during validation are:

- Is the mathematical model a correct representation of the real system?
- How close are the results produced by the simulation executable model to the behavior of the real system?
- Under what range of inputs are the model's results credible and useful?

Validation and verification are both ultimately activities that compare one thing to another. Validation compares real system / real process and conceptual model. Verification compares conceptual model and executable simulation model. Sometimes validation and verification are done simultaneously in one process.

Validation of the conceptual model as well as verification of the simulation model of our real process – *passing through the labyrinth* – are to be done simultaneously by running simulation computer model.

The whole process of transformation from a real system, the simulation model and its visualization is shown in Figure 1.

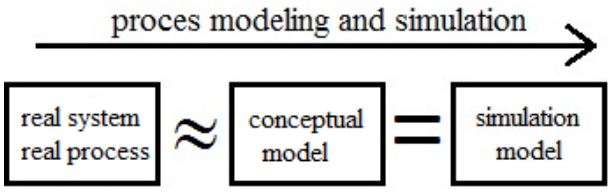


Fig. 1 Process modeling and simulation

Here again let us summarize that the mathematical model that reflects the real system / real process has some limitations and simplifying assumptions (the real system / process and conceptual model are in homomorphic relation).

In contrast, the simulation model is only the computer expression of the conceptual model (the conceptual model and simulation model are in isomorphic relationship).

E. Multidisciplinary approach

Another important benefit associated with the modeling and simulation of real processes is a multidisciplinary approach, without which the identification of the real processes using conceptual and simulation model and cannot be realized. This is also emphasized in this paper.

Multidisciplinary approach generally means that specialized disciplines are applied in a study of real process. These disciplines provide partial analysis of the process. These mono-disciplinary analyses are integrated to overall solution by integrating the solver who has basic multi-disciplines knowledge.

In our case study four disciplines are integrated, namely, algorithm development, programming and mathematics.

III. Solution of Problem of Visualization in the Programming Language LOGO

A. Programming language developing logical and creative thinking - LOGO

The first programming language designed to teaching and to developing logical and creative thinking at all *in both children and adult students* is the language LOGO. Its creation was influenced by the work of Swiss developmental psychologist and spiritual father of the constructivist school in theory of education - **Jean Piaget** (1896 - 1980). LOGO was designed by **Seymour Papert** (* 1928), Piaget's student, one of the pioneers of artificial intelligence, known American mathematician, computer scientist and educator. Papert worked with Piaget in the years 1958 to 1963 at the Geneva University. *"Nobody understands my ideas as well as Papert,"* Piaget said. Pappert created LOGO programming language in 1967 after his arrival at MIT (Massachusetts Institute of Technology). LOGO is partially based on the LISP programming language (called "artificial intelligence language"), from which it took over working with lists. A completely new idea was the so-called turtle geometry.

New implementations of LOGO programming language expanded its options on other elements. One of them can be used in the solution of problems connected with the Cartesian coordinate system in the plane based on the work with more turtles simultaneously, called *Multiturtle Mode*. Multiturtle mode is in the paper verified by the implementation of language XLOGO [9].

B. Tower of Hanoi brain teaser

The challenge is to find a suitable solution to visualize the well-known puzzle Tower of Hanoi. This puzzle was devised by French mathematician **Édouard Lucas** (1842 – 1891) in **1883**. The brain teaser consists of three pins, which can be pulled on circular disks of different radii.

All *n* disks are deployed on the left pin at the beginning, where they are sorted from the largest to the smallest disk. The task of the solver is to move all the discs on the middle pin, while the right pin is used as an auxiliary storage space. It is necessary to observe the following rules:

- You can relocate only one disc in one move.
- A move consists of taking the upper disk from one tower and placement on top of another tower.
- I t is forbidden to put a bigger disk on a smaller one.

Simple and elegant solution can be found on the Internet [10]:

```
to move :n :from :to :via
   if :n = 0 [stop]
   move :n-1 :from :via :to
   (print [Move disk from] :from [to] :to)
   move :n-1 :via :to :from
end
move 4 "left "middle "right
```

But this solution is not at all illustrative, because it provides

a text-only output. It is possible and necessary to build on this simple recursive solution of a system of interconnected components.

C. Transposition of Cipher Determined by the Figure – The Daisy Cipher

Figure 2 shows the daisy cipher with counting-out rhymes. For decryption of this cipher we start with the letter at the top (0 ° azimuth) and then expect other letters in a clockwise direction using counting-out rhymes. We scratch the used letters and they are not used repeatedly. Therefore, we must check each letter and, if used, we cannot count it further. Program to create daisy consists of function **overedge** and own procedure **daisycipher** for creating a daisy – see Figure 2.

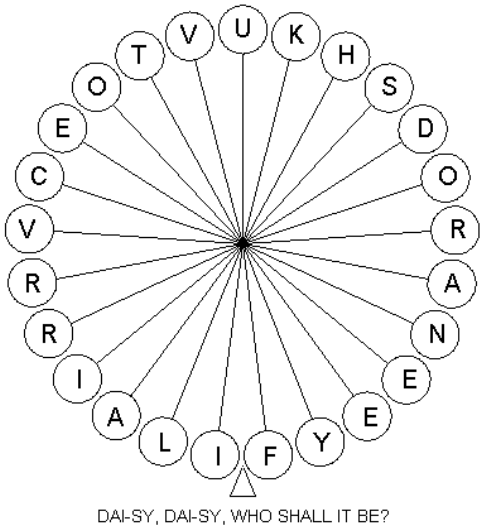


Fig. 2 The daisy cipher with counting-out game

```
to overedge :b :a
 if (:b <= :a) [output :b]
 if (:b > :a) [output sum mod :b :a 1]
end

to daisycipher :n :k :text :countingout
 make "outward list 1 2
 make "i 2
 repeat :n-2 [make "i :i+1
  make "outward lput :i :outward]
 make "i 0 make "position 1
 make "counter 0
 repeat :n
  [make "i :i+1
   make "outward replace :outward :position
item :i :text
   make "j :position
   repeatuntil
    [make "j overedge :j+1 :n
     if number? item :j :outward
      [make "counter :counter+1]]
   [or :counter = :k :i =:n]
   make "position :j
   make "counter 0]
 print :outward
 cs setfontsize 20 setfontjustify [1 1]
 make "i 0
 repeat :n
  [rt 360/:n*:i make "i :i+1
   fd 150 rt 90 repeat 120 [fd 1 lt 3]
   pu lt 90 fd 20 setheading 0 label item :i
:outward home pd]
 setfontsize 14 pu bk 215
 label :countingout fd 15
end
```

D. Placing of Chess Queens on the Checkboard

The challenge is to find a visualized solution of puzzle known also as Eight Queens Puzzle. This puzzle was originally proposed in 1848 by the chess player and composer Max Friedrich William Bezel (1824 – 1871).

The task is to place on the board 8 x 8 squares eight queens so that they do not endanger each other.

Generalization of this problem by means of mathematics model is as follows: The task is placed on an $n \times n$ chessboard fields n queens so that they do not endanger each other. It can be easily found that the job is not interesting for $n = 1, 2, 3$. For $n = 1$ it is the trivial solution, because one queen does not threaten the queen herself. For $n = 2$ or $n = 3$ problem has no solution. The situation is shown on Figures 3 and 4.

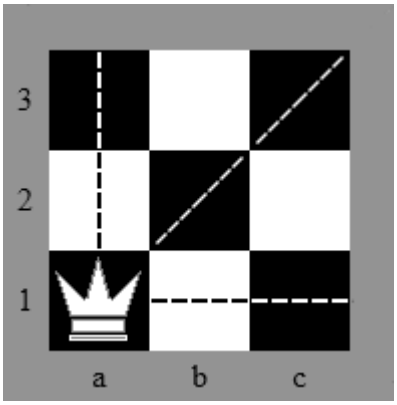


Fig. 3 Three Queens Puzzle – step 1

On board 3×3 arrays there are basically only three ways to place the first queen.

The first option is to place it on the central field b2, but then it will endanger all squares.

The second option is to use the field at the middle of the square, then remain free two corner boxes.

The third option is shown in Figure 4. Queen is located in the corner boards and two fields at the middle side of the square remain free.

By placing the other queen either in the second or in the third option, all field will be endangered - see Figure 4. The third queen, therefore, is not possible to place. The task has no solution.

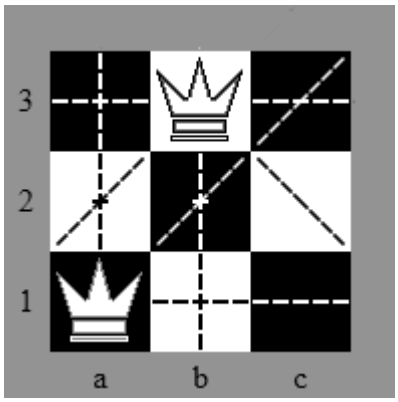


Fig. 4 Three Queens Puzzle – step 2

It can be shown that for all other n = 4, 5, 6, ... the puzzle has always solution, and even the number of different solutions with increasing n is increasing.

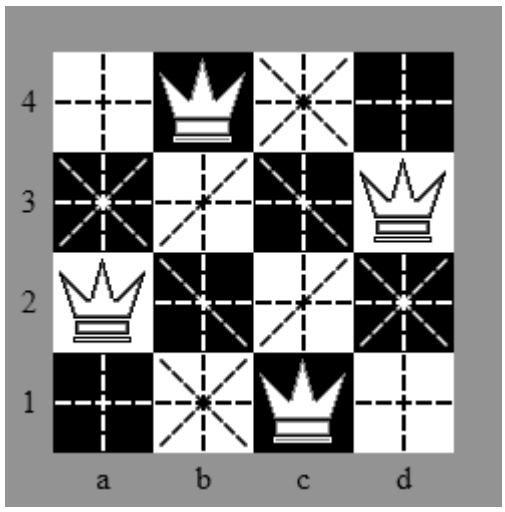


Fig. 5 Four Queens Puzzle – solution

In Figure 5 we can see a solution for n = 4, which is known as the Four Queens puzzle. In addition to solution shown on the Figure 5 there exist also a solution which is axially symmetrical along the vertical axis. We therefore distinguish the number of all solutions and the number of unique solutions that cannot be converted to another with axial symmetry, center of symmetry or rotation. Numbers of solutions for n = 1 to 10 we can see in the Table 1:

Table 1 – number of solutions of Queens Puzzle

n	1	2	3	4	5	6	7	8	9	10
u_s	1	0	0	1	2	1	6	12	46	92
a_s	1	0	0	2	10	4	40	92	352	724

Algorithm of solution the puzzle of n queens on the chessboard is described in standard programming literature – see e.g. [11]. It is a classic example of backtracking.

The algorithm implemented to the LOGO programming language, however, uses a different data structure that is used in standard programming language – e.g. in Pascal.

On the other hand program in Pascal cannot visualize how to find the solution, which is priority for the simulation of the process in LOGO language.

IV. Solution of the Problem

In this section the case study of *Tower of Hanoi brain teaser* will be extended by visualization in LOGO and the case study *Transposition Cipher Determined by the Figure – The Daisy Cipher* will be reached on new procedure for decrypting and its visualization. In the third case study *Placing of Chess Queens on the Chessboard* senses of computer turtle are used. The turtle is able to determine its actual position in Cartesian coordinates and also determine azimuth of direction of the orientation.

A. Visualization of solution Tower of Hanoi brain teaser

Which are the interconnected components in our system solution Tower of Hanoi brain teaser? The first is the aforementioned recursive core. The second component of system governs the current storage of the composition of the three towers into three lists. The third component, based on LOGO language and turtle graphics, is used for rendering current assembly of each tower.

The program is implemented the following four procedures. The procedure rectangle draws a rectangle with unusual way from the center of bottom side. The procedure drawtower draws tower based on the current list of discs. The individual discs are represented by rectangles, which are also filled with colors for clarity – see Figures 6, 7 and 8.

```
to rectangle :a :b
    right 90 forward :a/2 left 90
    forward :b left 90 forward :a left 90
    forward :b left 90 forward :a/2 left 90
end

to drawtower :a
    if not (:a = [])
       [make "base first :a
        make "residue butfirst :a
        make "width :base*20
        rectangle :width 20
        pu fd 10 setpc :base fill fd 10 pd
        setpc black drawtower :residue]
end
```

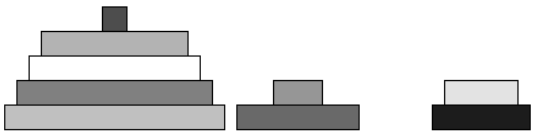


Fig. 6 First steps of solving the puzzle

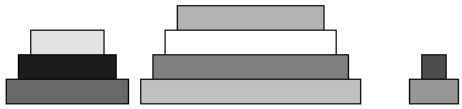


Fig. 7 An advance in solving the puzzle

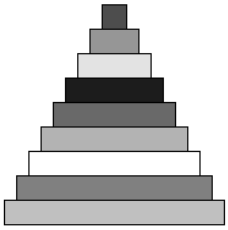


Fig. 8 Target position of solving the puzzle

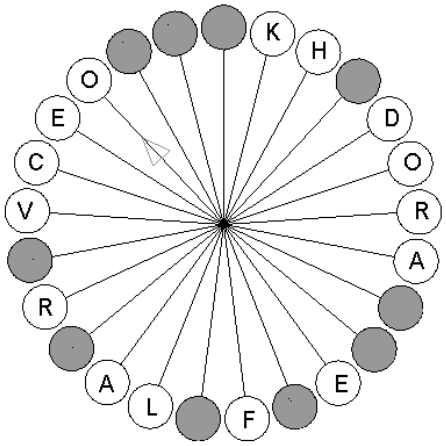
Next procedure **move** is the key, used to perform one move and includes two recursive calls to itself. The last procedure **hanoi** then just set the default lists and graphics, and starts the process of solution.

```
Procedure move:
to move :n :from :on :via
 if (:n = 0) [stop]
 move :n-1 :from :via :on
  if (:from = 1)
   [make "top last :lefttower
    make "lefttower butlast :lefttower]
  if (:from = 2)
   [make "top last :middletower
    make "middletower butlast :middletower]
  if (:from = 3)
   [make "top last :righttower
    make "righttower butlast :righttower]
  if (:on = 1) [make "lefttower lput :top :lefttower]
  if (:on = 2) [make "middletower lput :top
:middletower]
  if (:on = 3) [make "righttower lput :top
:righttower]
  cs
  pu setXY -150 0 pd drawtower :lefttower
  pu setXY 0 0 pd drawtower :middletower
  pu setXY 150 0 pd drawtower :righttower
  ht wait 30 st
  move :n-1 :via :on :from
end

Procedure hanoi:
to hanoi :n
 make "lefttower []
 make "i 0
 Repeat :n
  [make "i :i+1
   make "lefttower fput :i :lefttower]
 make "middletower []
 make "righttower []
 cs pu setXY -150 0 pd drawtower :lefttower
 ht wait 60 st
 move :n 1 2 3
end
```

B. Visualization of decrypting the Daisy cipher

Much more interesting then creation of daisy cipher figure is however the system of procedures for visualizing solution of daisy cipher. The procedure **fillcircle** is used to visualize the scratching a specific letter – see Figure 9. Competent circle is filled with orange color and has also not counted for next rounds. The procedure **writeletter** gradually writes individual letters of the plaintext obtained by applying counting-out rhymes. The main procedure, managing the decryption system is called **countyngoutgame**.



U N I V E R S I T Y

Fig. 9 Partially deciphered daisy cipher figure

```
to fillcircle :n :f
   pu home rt 360/:n*:f setpc orange
   repeat 12 [fd 13 wait 4] pd
   repeat 15 [fill fd 1] make "i 0
   repeat 15 [make "i :i+1 circle :i]
   fd 1 make "i 0
   repeat 15 [make "i :i+1 circle :i]
end

to writeletter :x :n :letter
   pu
   setxy difference :x*20 (:n+1)*10 minus 210
   setheading 0 pd setpc 0
   label :letter wait 40
end

to countingoutgame :n :k :text
   cs setfontsize 20 setfontjustify [1 1]
   make "i 0
   repeat :n
```

```
      [rt 360/:n*:i make "i :i+1
       fd 150 rt 90 repeat 120 [fd 1 lt 3]
       pu lt 90 fd 20 setheading 0
       label item :i :text home pd]
    make "outward list item 1 :text
    item :k+1 :text
    make "used list 1 :k+1
    make "thelast :k+1
    fillcircle :n 0
    writeletter 1 :n item 1 :outward
    fillcircle :n :k
    writeletter 2 :n item 2 :outward
    make "x 2
    repeat :n-2
      [make "counter 0
       make "j 0
       make "x :x+1
       repeatuntil
          [make "j :j+1
           if not member?
           overedge :thelast+:j :n :used
          [make "counter :counter +1]]
      [:counter = :k]
      make "outward
      lput item overedge :thelast+:j :n :text
         :outward
      fillcircle :n overedge :thelast+:j-1 :n
      make "used lput overedge :thelast+:j :n
         :used
      make "thelast overedge :thelast+:j :n
      writeletter :x :n item :x :outward]
    print :outward ht
end
```

C. Visualization of solution

Placing of Chess Queens on the Chessboard Puzzle

The first two procedures only draw a square box of the board. The procedure **square** draws white box. The procedure **blsquare** draws black box. The third auxiliary procedure forms of these white and black boxes complete chessboard. Chessboard pattern is used for man's orientation man who looks at the visualization solutions. For solution of algorithm it has no meaning. It can be said that turtles would be oriented even without this pattern.

```
to square
  pu fd 30 pd lt 90 fd 30 lt 90
  repeat 3 [fd 60 lt 90]
  fd 30 pu lt 90 fd 30 lt 180
end

to blsquare
  square pu fd 10 pd fill pu back 10 pd
end

to chessboard :n
  cs
  make "z mod :n 2
  make "k div :n-:z 2
  make "r :k
  repeat :n
   [
     make "s minus :k
     repeat :n
       [
          pu SetXY :s*60 :r*60
          if (mod :r+:s 2) = 0 [square]
          if (mod :r+:s 2) = 1 [blsquare]
          make "s :s+1
     make "r :r-1
   ]
end
```

The core of the solution is procedure, respectively function **place :i :n**. It is a function because its recursive call returns a boolean value of **true** or **false** depending on the turtle is placed (or not) to the unthreatened position in the *i*-th row (counting down) of the $n \times n$ chessboard arrays.

The program of algorithm solution written in LOGO language uses different data structure than program written e.g. in Pascal language. Queens are represented by *n* turtles. The program works in the Multiturtle Mode, in which is shown more turtles at one time. The program chooses among the turtles only one active turtle by using the command **setturtle**. All subsequent commands are applied to this selected turtle.

Each turtle moves in a predetermined rows, i.e., only horizontally. For each specific location the coordinates of all turtles in the rows above her home range (i.e. all turtles with a lower number) is detected. Then it is checked if any of the turtles do not lie above it in the same column (angle 0°), or in the same diagonal (angles 45° or 315°). If the position of the turtle is safe, the function **place** is recursively called for the location of the turtle below i.e. the turtle with the serial number of the one above. Position the turtle is evaluated retroactively as unsatisfactory in that case, although it is safe, but the turtle with a number below cannot be safely placed in any of the positions in the entire row. The solution is achieved and the program is terminated when all turtles are safely placed.

```
to place :i :n
   setturtle 0
   setshape 0
   SetXY 17 37 ht
   make "w quotient 125 :n*:n
   if (:i > :n)
     [
      output false
     ]
   if (:i <= :n)
     [
      localmake "j 0
      localmake "errors 0
      repeat :n
        [
         setturtle :i
         setshape 3
```

```
        localmake "j :j+1
        make "s :j-:k-1
        make "r :k+1-:i
        make "q true
        pu st SetXY :s*60 :r*60 pd
        wait :w
        localmake "z 0
        localmake "test true
        repeat :i
          [
           setturtle :z
           make "p pos
           setturtle :i
           make "azimut towards :p
           if (:azimut = 0) [make "test false]
           if (:azimut = 45) [make "test false]
           if (:azimut = 315) [make "test false]
           make "z :z+1
          ]
         if (and :test :i=:n) [stopAll]
         if (:test) [make "q place :i+1 :n]
         if (not :q) [make "errors :errors+1]
        ]
       if (:errors>=:n) [output false]
       if (:errors<:n) [output true]
      ]
end
```

Last short procedure **solve: n** only sets the number of turtles in Multiturtle Mode to *n*, clears the workspace, draws $n \times n$ chessboard array and starts depth searching by calling the function **place 1: n**, thus execution the location of the first turtle in the first row from the top. The result of the program for $n = 8$ is shown in Figure 10.

It is very significant that this solution enables real-time visualization of solution of the problem.

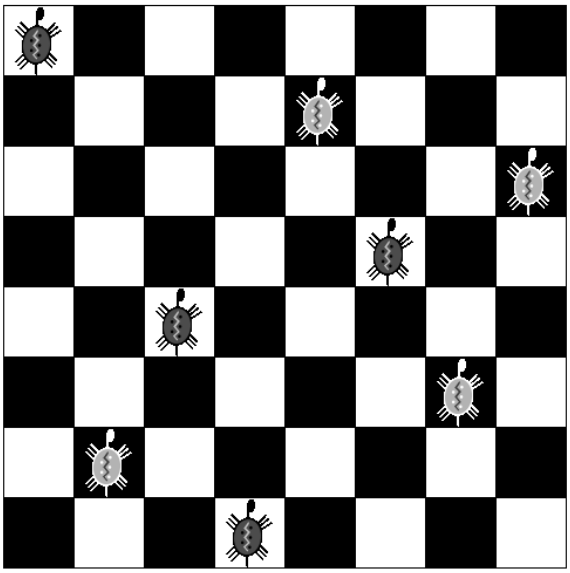


Fig. 10 Eight Queens Puzzle – visualization of problem solving in XLOGO environment

```
to solve :n
   setturtlesmax :n+1
   cs
   chessboard :n
   make "aaa place 1 :n
end
```

V. Conclusion

There are various approaches how to provide training in algorithms, how to introduce and develop basic algorithmic thinking of students.

The paper offered one of the kinds of the possible teaching / learning strategies using the system approach including problem analysis. The system approach can be set as the default paradigm for a wide integration of the principles of the algorithm development into education. The authors highlight importance of utilizing systematic problem-solving approaches, like problem analysis, for tackling problems in algorithmic tasks in teaching algorithm development and programming. The paper emphasizes the fact that the algorithm development of the real processes should be supported by computer simulation.

Solutions of the processes of **The Tower of Hanoi**, **The Daisy Cipher** and **Placing of Chess Queens** have been chosen as three possible case studies of how to present the application of the system approach.

Acknowledgment

This research has been partially supported by the Specific research project of the Faculty of Education of University of Hradec Kralove No. 2102.

References

[1] S. Hubalovsky, J. Jelinek, J. Sedivy, „Mathematical modeling and computer simulation of optimal reaction time of the Lupine protein hydrolysis using fermented whey", *International Journal of Mathematical Models and Methods in Applied Sciences*. vol. 6, No. 2., 2012.

[2] J. Sedivy, S. Hubalovsky, "Mathematical foundations and principles in practice of computer aided design simulation", *International Journal of Mathematics and Computers in Simulation.* vol. 6, No. 1. 2012.

[3] O. Horák, L. Mitrovič, "Description of the Basic Algorithm Blocks and Structures Representation in Courses of Algorithm Development", *WSEAS Trans. on Information Science & Applications on Advances in Engineering education.* vol. 9, No. 2. 2012.

[4] J. Šedivý, **"**Multimedia support of parametric modeling", in *Proc. 9th WSEAS International Conference on Engineering Education (EDUCATION '12).* WSEAS Press, 2012.

[5] K. Dvořák, J. Šedivý, "CAx application in the teaching of engineering subjects", in *Proc. 9th WSEAS International Conference on Engineering Education (EDUCATION '12).* WSEAS Press, 2012.

[6] S. Hubalovsky," Modeling and computer simulation of real process – solution of Mastermind board game", *International Journal of Mathematics and Computers in Simulation.* vol. 6, No. 1. 2012.

[7] J. Bailer, M. Daniela, "Tracing the Development of Models in the Philosophy of Science", *Magnani, Nersessian and Thagard*, 1999.

[8] S. Hartmann, "The World as a Process: Simulations in the Natural and Social Sciences", In R. Hegselmann, et al., *Modelling and Simulation in the Social Sciences from the Philosophy of Science Point of View*, Theory and Decision Library. Dordrecht: Kluwer, 1996.

[9] LE COQ, L. *XLOGO: Reference Manual*, 1st ed. XLOGO Comunity: 2009. Translation: G. Walker.

[10] http://rosettacode.org/wiki/Towers_of_Hanoi#Logo..
[11] Wirth Niklaus. *Algorithms + Data Structures = Programs.* 1st ed. Prentice-Hall, New Jersey,:1975.

Stepan Hubalovsky was born in 1970 in Czech Republic. He obtained master degree in education of mathematics, physics and computer science in 1995 and doctor degree in theory of education in physics in 1998 both in Faculty of Mathematics and Physics, Charles University in Prague, Czech Republic. He works as associated professor on University of Hradec Kralove His scientific activities are system approach, modeling and simulation.

Michal Musilek was born in 1963 in Czech Republic. He obtained master degree in education of mathematics and physics in 1988, in education of computer science in 1993 and doctor degree in theory of education in physics in 2009 all in Faculty of Education, University of Hradec Kralove, Czech Republic. He works as assistant professor on University of Hradec Kralove. His scientific activities are theory of education in informatics includes children's programming languages, using ICT in education of mathematics and physics include computer modeling and simulation.