Journal of Electrical Engineering, Electronics, Control and Computer Science
JEEECCS, Volume 3, Issue 10, pages 1-4, 2017

AutoCAD Games Library for Didactic Purposes

Mădălina Măndoiu, Răzvan Alexandru Geambaşu,
Grigore-Adrian Iordăchescu, Monica-Anca Chiţă
University of Piteşti
Piteşti, Romania
e-mail: adi_iord@yahoo.com

***Abstract* – In this paper we will present a unique use of AutoCad software to realize a games library, including a series of four applications, of which the most known are „Hangman" and „Tic Tac Toe". The code behind the library is realized in AutoLISP programming language. We will present the mode of implementing these four games and their logical diagrams. This project will also contain some screenshots of the games library and our thoughts about possible updates. This library can be extended and it is suitable for motivating students to deepen their AutoCAD learning.**

Keywords- Games library; AutoCAD; AutoLISP; didactic library

I. INTRODUCTION

At the moment AutoCAD is one of the most popular and most used programs of computer assisted design. With the help of AutoCAD you can realize graphic objects in 3D or technical drawings in 2D. This program ensures also the possibility of transposition of the result on paper. Although it is most used in the constructions domain and car industry, AutoCad can also be used to realize prototypes for laptops and mobiles phones. [1]

Thanks to all of the reasons mentioned above, AutoCad has become an essential tool in the majority of technical profile universities. Among these, programming students show the biggest reticence in learning purely engineering notions, as they are having a bigger aspiration for programming. An innovative teaching idea will be to realize simple games inside of computer-aided design laboratories. In this way, teachers can stimulate attention and creativity of students and their motivation in the process of learning a purely technical program like AutoCAD. In order to achieve interactive games by using AutoCAD it is necessary to have a base knowledge about its native language, AutoLISP, which is part of LISP languages.

LISP programing language appeared around 1960. Its name comes from "List Processing" because information is structured in lists. Since 1986 AutoDesk started to develop a part of the language under the name of AutoLISP [2]. The language LISP is less known in comparison with other programming languages like BASIC, FORTRAN, PASCAL, COBOL or C. An AutoLISP program is a list of commands and functions witch used in AutoCAD can replace the human user in completing many automatic tasks. AutoLISP programs are in the form of text files with *.lsp* extension. For editing a file of this kind, any text editor that does not format the text can be used (an example is Notepad). [2]

However, sites that describe the use of AutoCAD programming and of AutoLISP applications are limited. Among these, the most important are: Afralisp [3], Wikipedia [4] and Autodesk [5].

There already are some attempts of games in AutoCAD, some of them made quite long ago, but the internet does not have many examples in this domain. Even so, the majority of them have runtime errors with newer versions of AutoCAD. For example, one can find: "Battle Ship", "GrumpyBlocks", "TROY", "MasterMind", "Slot Machine" [6], "First Pacman" [7], "Angry Birds" [8], "Asteroids" [9].

Another benefit in the programming of games in AutoCAD is the speed of passage from idea to prototype. Because of this, before implementing a game in a more developed programming language like C or JAVA, the creator can explain and test the concept by using AutoCAD's graphics interface and interact with the user by AutoLISP.

II. LIBRARY ORGANIZATION

The library that is the scope of this paper was thought to start with 4 initial games, but with the possibility of infinite expansion of the number of applications that it can support. The choice of the game is made from an intuitive menu, which presents graphically all of the library games (Fig. 1)

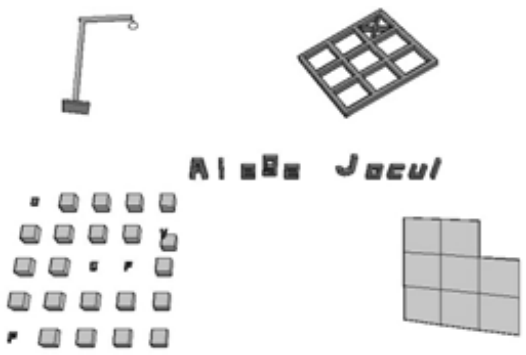

Figure 1. Main menu of the library

The code behind this menu is simple, so the students will be encouraged to extend it with games developed by them in the future.

Obviously, the first step for a student will be to realize his own game. This step will be detailed in the next section. Right after completing his game, he will write his game code in a separate function with a suggestive name, like, in our case "game5".

After that, he will extend the graphical menu which contains suggestive pictures of the games by adding an icon for his new game. The last step in preparation of the menu will involve modifying geometrical coordinates which define previous pictures and add a new set of coordinates describing the new added icon.

For example, in the case of the initial library, the screen (Fig. 1) is split in 4, the division being made on the horizontal line y = - 60 and vertical line x = + 150. By adding a new object, the screen-division has to be re-written. The student will be encouraged to think this rebuild on his own.

III. BUILDING THE GAMES

The students will learn about drawing graphical objects and about AutoLISP commands by understanding the building steps of the 4 pre-existent games in the initial library.

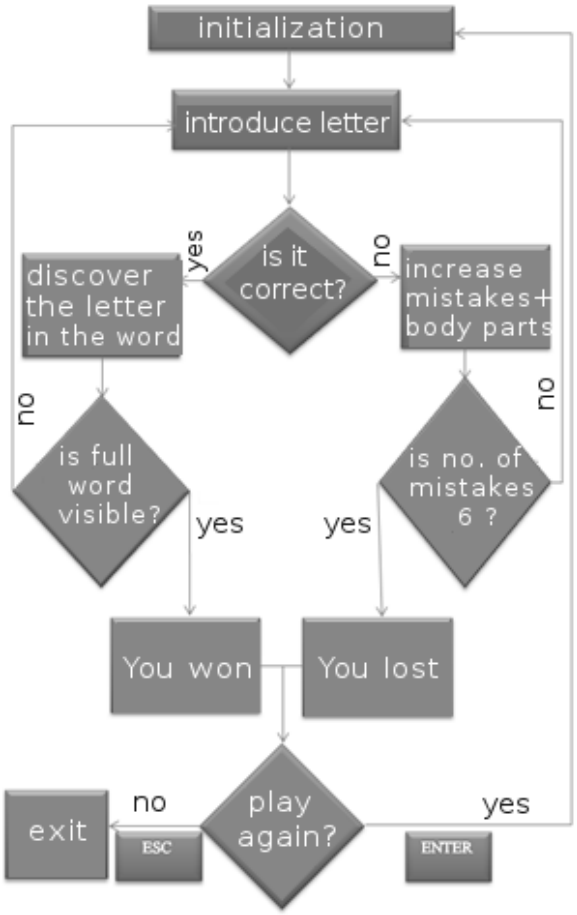

Figure 2. Logical flow diagram of „Hangman"

A. Hangman

The first step in building this game was drawing of the graphical objects, to which we allocate variables in the program. For this specific game, the graphical objects are: head, body, left arm, right arm, left leg, right leg, gallows, letter-supports and the end messages "You Win" and "You Lost". For drawing the arm for example, we used only AutoCAD commands: spline, circle, rotate3d, sweep.

The second step is drawing the general logical diagram of the program, which can be seen in Fig. 2. There can be noticed the implementation of some basic rules: the maximum number of mistakes is equal with the number of body parts that we previously drew (that means 6). There are also other rules that were not represented in the diagram, but were nevertheless implemented:

- If the player tries a letter that has been used before, then the letter will not be counted again as a mistake;
- Only letters from the English alphabet can be used;
- The graphical objects (the body parts) will be added in a predetermined order
- Computer chooses the word according to the milliseconds of the clock from a word-list entered by the programmer.
- Wrong letters are written in the bottom of the screen, as in Fig. 3

In Fig. 3 there can be seen a screenshot while playing "Hangman" in AutoCAD.

B. Puzzle

Graphical objects for this game are represented by the puzzle pieces but these are not drawn in AutoCAD. Instead they represent pieces from a .jpg image that is cut by the AutoLISP script at the beginning of the game.

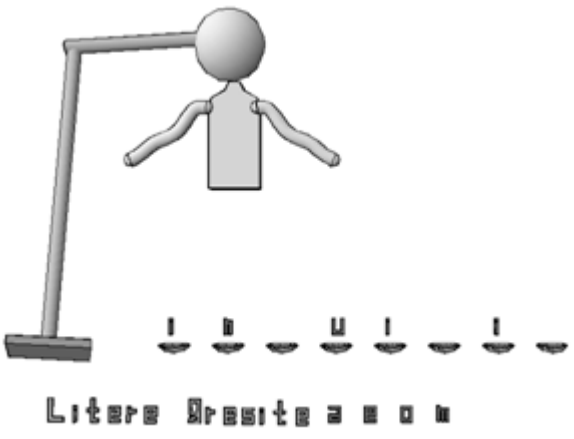

Figure 3. Screenshot from the „Hangman" game

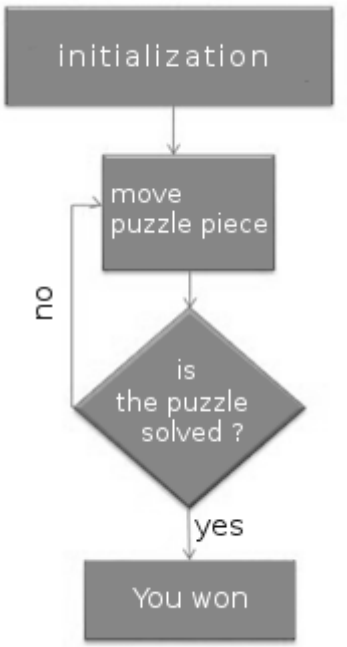

Figure 4. Logical flow diagram of the „Puzzle" game

Basic rules for this game are presented in the diagram of Fig. 4. The pieces are randomly mixed and can be moved with the mouse to solve the puzzle.

The following list presents some rules that are not in the diagram of Fig. 4:

- Because the pieces are randomly mixed, there is a chance that the puzzle cannot be solved by simple permutations;
- The final result of the puzzle should be the initial image;
- If the player selects a puzzle piece neighboring the blank piece, they will swap places (Fig. 5);
- If the player selects a puzzle piece that is not neighboring the missing piece, the program will ignore the command;

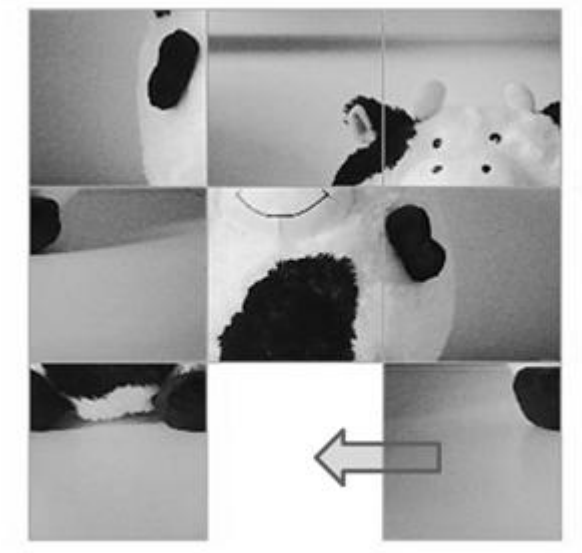

Figure 5. Screenshot of the "Puzzle" game

C. Memory game

Graphical objects needed for this game are the blocks and the objects behind them. Hidden by 24 blocks there will be 12 pairs of different graphical objects (from a larger objects library). Behind a 25th block there will be placed a forbidden object (the Joker). The diagram of this game is presented in Fig. 6 while in Fig. 7 there can be seen a screenshot of a game-play.

One of the basic rules presented in the diagram is that the player has only some seconds to memorize the place of the graphical objects. After this time lapse the objects will be covered by cube-blocks. Then he has to choose a pair of blocks. If the graphical objects behind two consecutively chosen blocks are the same, they will stay revealed for the rest of the game. If they are not the same, the number of mistakes increases. This repeats until the number of mistakes is 12 or the player uncovers the joker by mistake. In this last scenario, the player loses the game. If the player finds all pairs correctly he wins.

The initial graphical objects of this game are the alphabet letters from A to Z. The game can be further developed by the students by replacing these letters in the library with more interesting objects, and, the student will be encouraged to draw them by themselves.

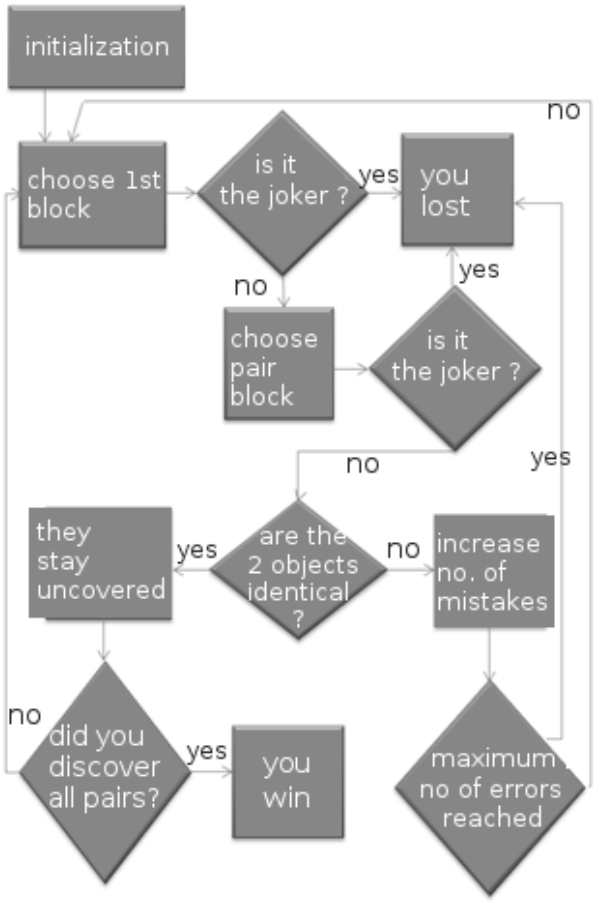

Figure 6. Logical flow diagram of the Memory game

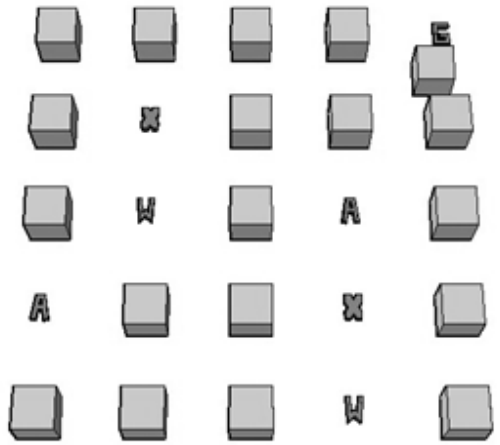

Figure 7. Screenshot from the Memory game

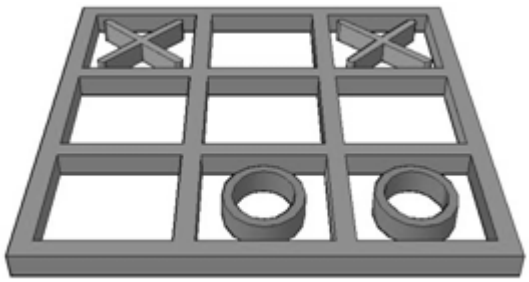

Figura 8. Screenshot from Tic Tac Toe

D. Tic Tac Toe

As graphical objects for this game there can be seen the characters of X and 0, the game table (Fig. 8) and the 3D win-lose messages at the end of the game.

The well-known basic rules are presented in the diagram of Fig. 9. Computer has the first move. The user can move only in the empty spaces. The board format has 9 empty spaces, aligned in a 3x3 matrix.

The goal of the game is to complete 3 identical symbols on a line, column or diagonal line. If the board is full but none of the players has managed to win, the game ends in a draw. A rule that is not highlighted in the logical diagram is that if the user clicks off the board or a busy space, the move is repeated.

IV. CONCLUSIONS

Making applications in AutoLISP is not a well-covered area. Sites describing the interaction of AutoCAD and AutoLISP are scarce, so students are not very interested in this area. However, an original way to attract their attention is developing a game library.

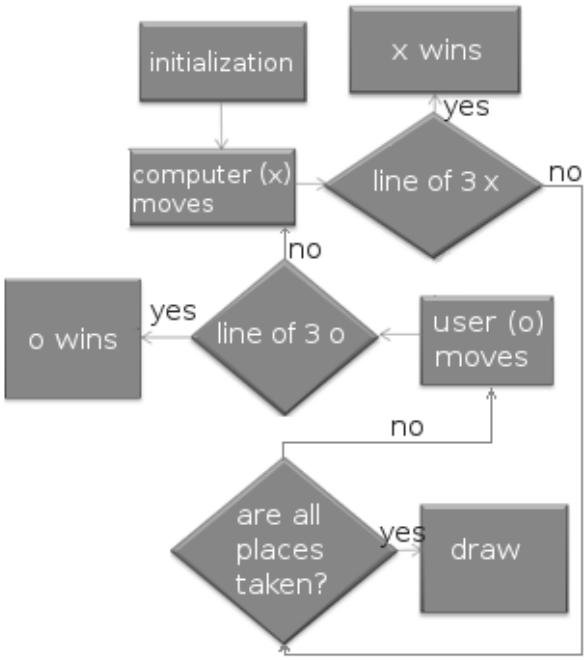

Figure 9. Logical flow diagram of the Tic Tac Toe game

The original library presented in this paper has four games and can be easily understood and expanded by the students. Expansion can be done by adding new games or by improving the four existing ones in the library. The building and operation of these games were briefly explained in this paper.

The library described in this paper was uploaded by the authors and can be found online [10] as an archive containing the drawing file and the AutoLISP script. The library was tested by the authors in AutoCAD 2012 and AutoCAD 2017. In order to test it, one should open the drawing file in AutoCAD, drag and drop the script file over the drawing and enter the GAME command.

REFERENCES

[1] What is and what are the uses of AutoCAD, https://teach-ing.ro/ce-este-si-la-ce-foloseste-autocad-ul/
[2] AgoraCAD: introduction to AutoLISP, http://agoracad.blogspot.ro/2009/04/introducere-in-autolisp.html
[3] Learn AutoLISP for AutoCAD productivity, http://www.afralisp.net/index.php
[4] AutoCAD, wikipedia article https://en.wikipedia.org/wiki/AutoCAD
[5] AutoCAD For Mac & Windows https://www.autodesk.com/products/autocad/overview
[6] CAD Studio - files and utilities - download Games http://www.cadstudio.cz/en/download.asp?ccat=32
[7] First pacman game for AutoCAD (using Lisp programming), https://www.youtube.com/watch?v=7A3jeDVcrIs
[8] AngryBird in AutoCAD, https://www.youtube.com/watch?v=tt_XP5aRsYU
[9] Game in AutoCAD, https://www.youtube.com/watch?v=HnjMr1-vT8o
[10] AutoCAD Games Library: http://upit.eu5.org/Laboratoare/gamesLibrary.rar