QUERYING THE COMPONENT DATA OF A GRAPHICAL CADASTRAL DATABASE USING VISUAL LISP PROGRAM

Caius DIDULESCU, Associate Professor PhD eng. - Faculty of Geodesy, Technical University of Civil Engineering, Bucharest, e-mail: caiusdidulescu@yahoo.com
Adrian SAVU, Lecturer PhD eng. - Faculty of Geodesy, Technical University of Civil Engineering, Bucharest, e-mail: : adisavu2002@yahoo.com
Constantin COŞARCĂ, Associate Professor PhD eng. - Faculty of Geodesy, Technical University of Civil Engineering, Bucharest, e-mail: constantin.cosarca@geodezie.utcb.ro
Aurel SĂRĂCIN, Associate Professor PhD eng. - Faculty of Geodesy, Technical University of Civil Engineering, Bucharest, e-mail: saracinaurel@yahoo.com
Aurel Florentin NEGRILĂ, Teacher assistant, PhD eng. - Faculty of Geodesy, Technical University of Civil Engineering, Bucharest, e-mail: aurel.negrila@geodezie.utcb.ro

***Abstract**: Visual LISP is a programming environment that allows to solve specific problems, including the handling of data in graphical cadastral databases. Visual LISP allows customizing AutoCAD for specific operations, such as processing and automation of cadastral works. This article describes how the component data of a graphics database can be manipulate and an application that allows manipulation of graphical database blocks. LISP is, except FORTRAN language, the only high-level programming language that has survived from the 1960s until today, and Visual LISP as a subset of LISP language will continue to allow making routines that will improve the production in the office works in the field of surveying and cadastre.*

***Keywords:** graphical databases, Visual LISP*

1. Introduction

AutoCAD includes an integrated development environment called Visual LISP, which includes a compiler, a debugger, catalogues of functions and visual controls that simplify writing AutoLISP programs.

Visual LISP can be best described as a complete development environment, for creating and customizing AutoCAD applications using the programming language LISP. Most of the functionality and the addition of productivity that brings the Visual LISP is the fact that this is an application of Object ARX components and has a completely new LISP interpreter. The software can create Visual LISP routines that permit the manipulation of data in databases graphics.

Visual LISP permits creation and homogeneous manipulation of heterogeneous entities, a unitary treat of numbers, character strings, and geometric entities or of the multitudes formed by them. The power and the flexibility of Visual LISP routines enlarge the AutoCAD possibilities constituting an advantage for all his users [3].

Sometimes it is necessary to decide if it is more favourable to use a Visual LISP program for achieving one thing or to follow another method such as a personalised menu or a script file.

Generally Visual LISP is the best instrument for achieving the next types of activities:

1. Activities that repeat, requiring a great number of commands;

2. Operations and calculations that can lead to errors;
3. Applications of standard rules with consistency;
4. Saving working time in AutoCAD by compressing the commands;
5. Adaptation of AutoCAD for a specific operation.

Visual LISP routines were created to eliminate repetitive processes to automatically perform various operations and calculations and not least to save working time by compressing AutoCAD commands [1]. Visual LISP is noted for its strength and flexibility routines, but also it allows adapting AutoCAD to perform specific operations.

2. Accessing information from graphical databases

One of the features offered by Visual LISP is the ability to access information from the AutoCAD database. Visual LISP functions allow extraction of component data of an object, such as layer or colour. Also, data of specific items may be handled using Visual LISP functions.

2.1 Components of the entities stored in the graphical data

Each object in the database has an "entity name" assigned, unique for each object (or entity). "Entity name" is a data type, being an internal hexadecimal number uniquely assigned. This hexadecimal number is used for identification, but also the "key" to the internal list of each data object. The example below illustrates the internal list of a line.

```
((-1 . <Entity name: 7ef08290>) (0 . "LINE") (330 . <Entity name: 7ef05cf8>) (5 . "8B2")
(100 . "AcDbEntity") (67 . 0) (410 . "Model") (8 . "Releveu") (100 . "AcDbLine") (10 4.5 5.0
0.0) (11 7.5 4.0 0.0) (210 0.0 0.0 1.0))
```

It is noted that this list is an AutoLISP list, since begin and end with parentheses, being in fact a list of lists that describe all the components that AutoCAD knows about that line. In a more readable form, the list looks like this:

```
(-1 . <Entity name: 7ef08290>)
(0 . "LINE")
(330 . <Entity name: 7ef05cf8>)
(5 . "8B2")
(100 . "AcDbEntity")
(67 . 0)
(410 . "Model")
(8 . "Releveu")
(100 . "AcDbLine")
(10 4.5 5.0 0.0)
(11 7.5 4.0 0.0)
(210 0.0 0.0 1.0)
```

Each internal list was printed on a separate line. These lists are special lists, called associated lists. They are characterized by their shape:
(atom or list [.] list or atom)

If the second item in the list is associated with an atom, this list is called a dotted pair. A pair of points is a list consisting of two atoms separated by a point, with a space before and after the point.
The AutoLISP function **cons** builds dotted pairs. It's syntax is:
(cons atom atom)
In associated lists of "entity name", first item in each nested list is known as group code. Group code identifies the characteristic that is associated to the dotted pair and the second part of the dotted pair has associated the attribute value. The dotted pair (0. "LINE") indicates that the object is a line, and dotted pair (8. "Releveu") indicates that the object is on layer "Releveu".
The following table shows representative group codes and information assigned to these codes.

Table 1. Representative group codes of graphical entities.

-1	Entity name. The name changes every time when a drawing is opened. It is never fixed (saved).
0	Text string indicating the type of entity (fixed)
1	Indicates the text held by the entity if the object is text
2	Name (block name, name attribute in the block)
5	Text string up to 16 hexadecimal digits that is used for handling entity
6	Name of Line Type
7	Name of text style
8	Layer name
10	Primary point; starting point of a line or a text, the center of a circle, etc.
DXF	The primary point value X (Y and Z having followed by 20 and 30 codes)
APP	3D point (list of three real numbers)
11-18	Other points
40-48	Other values (text height, scale factor)
49	Linetype scale
50-58	Angles (values in nonius degrees in dxf files and radians through AutoLISP applications)

Each object in the drawing has a code 0 and a code 8, since these are the type of object (line, circle) and the layer on which is represented the object.

2.2. Extraction of data components of a graphical database entity

Extraction of data components of a graphical database entity is a process carried out in several stages. The first step is usually the same: getting the name of the entity. Without it, all the data components of the entity are inaccessible. AutoLISP function **entsel** (shortening for entity selection) permits to obtain the name of an entity, allows to select an entity and returns two pieces of information, respectively name of the entity and coordinates of point used to select the entity. Entity name is the first element of the list and can be isolated by using **car** function.
For extraction of data components of a circle, it is necessary to draw a circle with any centre and any radius. The Visual LISP Console are inserted the following commands:

```
_$ (setq ent (entsel))
Select object: < select the circle >
(<Entity name: 7ef0f710> (473.144 2434.44 0.0))
_1$ (setq ent (car ent))
<Entity name: 7ef0f710>
```

Ent symbol was linked to the name of the entity. Further AutoLISP provides **entget** function (derived from the get entity) [5]. This displays the data components of the entity whose name has been linked to a symbol.

```
_1$ (setq ent (entget ent))
((-1 . <Entity name: 7ef0f710>) (0 . "CIRCLE") (330 . <Entity name: 7ef05cf8>) (5 .
"D9392") (100 . "AcDbEntity") (67 . 0) (410 . "Model") (8 . "TEXT") (100 . "AcDbCircle")
(10 455.871 2453.43 0.0) (40 . 26.7985) (210 0.0 0.0 1.0))
```

At this point, the data components of circle are linked to the ent symbol.
The next step consists in extracting specific information from the data list. If it is desired to extract the layer name of the drawn entity, it is noted that the layer name is contained in the sublist having the group code 8 as a dotted pair. In the example above, this sub-list appears:
(8. "TEXT")
To remove this list, called associated list, from the entire list, it is used **assoc** AutoLISP function that requires two arguments: the group code of extracted sub-list and the symbol that is linked to the entire list of the entity.
_ $ (setq ent (assoc 8 ent))
(8. "TEXT")
This is a dotted pair containing the desired information (layer name) in the second list item. Information can be extracted using the **cdr** function.
_ $ (setq ent (cdr ent))
"TEXT"
This was the last step: it is extracted the name of the layer on which the circle was drawn and it was connected to a variable. Abridged, this can be written on a single line:
_ $ (Setq ent (cdr (assoc 8 (entget (car (entsel))))))
"TEXT"

3. Handling of data components of an entity in Visual LISP programs

Extraction of data component of entities allows programs to save time and to increase productivity in office work.
It is desirable to use an existing block in the drawing to be inserted in other positions. The first stage consists in running the steps above, namely finding the block name.

```
_$ (setq ent (entsel))
(<Entity name: 7ef03860> (588064.0 361078.0 0.0))
_$ (setq ent (car ent))
<Entity name: 7ef03860>
_$ (setq ent (entget ent))
((-1 . <Entity name: 7ef03860>) (0 . "INSERT") (330 . <Entity name: 7ef13d00>) (5 .
"3ACC") (100 . "AcDbEntity") (67 . 0) (410 . "Model") (8 . "DN") (100 .
"AcDbBlockReference") (2 . "borna-hecto-m") (10 588065.0 361076.0 0.0) (41 . 1.0) (42 .
1.0) (43 . 1.0) (50 . 0.0) (70 . 0) (71 . 0) (44 . 0.0) (45 . 0.0) (210 0.0 0.0 1.0))
```

It is noted that the block name is found in the associated list with the group code 2. Extracting its name is the same as above:

```
_ $ (setq ent (assoc 2 ent))
(2. "borna-hecto-m")
_ $ (setq ent (cdr ent))
"borna-hecto-m"
```

Further it can be inserted the desired block, using **command** AutoLISP function.

```
$ (command "insert" ent)
nil
```

The result in Visual LISP Console is nil because the **command** function returns nil, but the effect is seen in AutoCAD, where the block is waiting to be inserted.
The result of the command line is as follows:

```
Command: 'VLIDE insert Enter block name or [?] <borna-hecto-m>: borna-hecto-m
Units: Millimeters Conversion: 1000
Specify insertion point or [Basepoint / Scale / X / Y / Z / Rotate]: Specify insertion point or
[Basepoint / Scale / X / Y / Z / Rotate]: <select insertion point>
Enter X scale factor, specify opposite corner, or [Corner / XYZ] <1>: <Enter>
Enter Y scale factor X <Use scale factor>: <Enter>
Specify rotation angle <0.0000g>: <Enter>
```

The result above can be written in a program:

```
(defun C: IB (/ ent)
  (setq ent (entsel))
  (setq ent (car ent))
  (setq ent (entget ent))
  (setq ent (assoc 2 ent))
  (setq ent (cdr ent))
  (command "insert" ent)
)
```

It is selected the code and it is loaded with Load Selection command from the Tools menu. It can be observed in the Visual LISP Console window that new command **C: IB** is loaded. After calling the new AutoCAD **IB** command, it is selected the desired block and inserted in another location specified by the user.
It is possible for the user to select an object that is not a block. In this case, the routine fails. It extracts the code group 2, which only block items have it. In this case **ent** variable will be bound to nil. It is necessary to introduce a conditional ramification to allow the routine to run when one block was selected and otherwise to print an error message.

```
 ( defun C: IB ( / ent )
 ( setq ent ( entsel ) )
 ( setq ent (car ent ) )
 ( setq ent ( entget ent ) )
 ( setq ent ( assoc 2 ent ) )
 ( setq ent ( cdr ent ) )
 (if ent
   (command "insert" ent )
   ( prompt " You have not selected a block , try again . " )
 )
 ( princ )
)
```

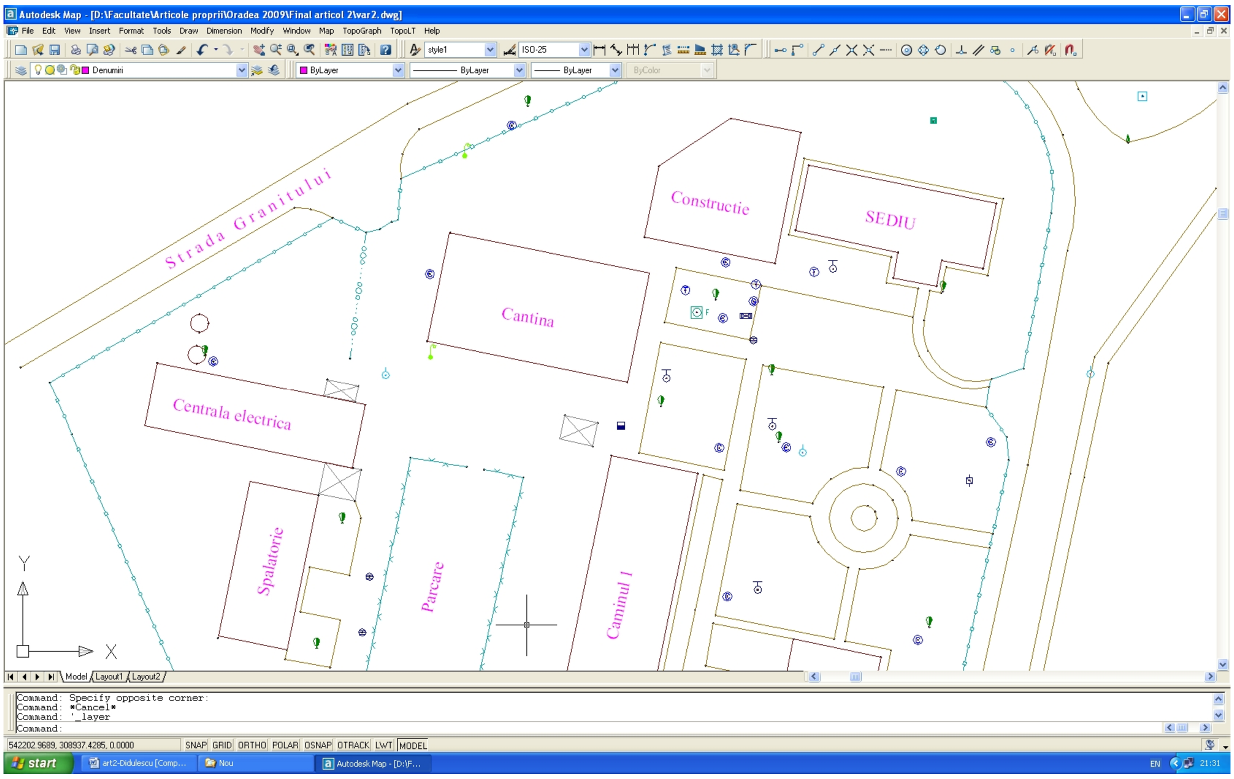
3.1 Retrieving coordinates of blocks inserted in a graphical database.

If the coordinates of the blocks in the database cannot be found in the files of coordinates, they can be recovered. The steps are: it queries the entire database, extract only the blocks from the selected area and the results will be written in an external file.

```
(defun C:ptbl (/ ent)
 (setvar "OSMODE" 0)
 (print "Selectati  blocurile:")
 (setq fis (open "fiscoo.txt" "a"))
 (setq sp (ssget))                    ; create a selection set
 (setq sa (sslength sp)) ; returns the number of entities in the selection set
 (setq index 0.)
 (while (< index sa)
   (setq ent (ssname sp index)); returns the name of the entities in the selection set
   (setq ent (entget ent))
   (setq ent1 (assoc 2 ent)); check whether the selected entities are blocks
   (if    (/= ent1 nil); condition the extraction of coordinates only for block entities
     (progn
           (setq XYZ (cdr (assoc 10 ent)))
           (command "point" XYZ)
           (setq x (car XYZ))
           (setq y (cadr XYZ))
           (Setq z (car XYZ))
           (write-line
             (strcat "    " (rtos x) "    " (rtos y) "     " (rtos z))
             fis
           )
     )
   )
   (prompt " You have not selected a block , try again.")
 (princ)
 (setq index (+ index 1.))
   )
 (close fis)
(princ)
)
```

The program opens a text file for writing, creates a selection set, returns the number of entities found in the selection set and check whether selected entities are blocks. “

Further, is placed the condition of extraction coordinates only for block entities and the obtained values are written in the open file. If there are not selected blocks, the program displays the message "You have not selected a block, try again”.

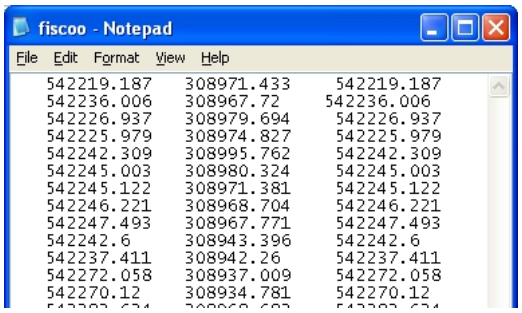

Autodesk Map - [D:\Facultate\Articole proprii\Oradea 2009\Final articol 2\war2.dwg]
File Edit View Insert Format Tools Draw Dimension Modify Window Map TopoGraph TopoLT Help
Strada Granitului
Constructie
SEDIU
Cantina
Centrala electrica
Spalatorie
Parcare
Caminul 1
Model Layout1 Layout2

```
Command: Specify opposite corner:
Command: *Cancel*
Command: '_layer
Command:
```

542202.9689, 308937.4285, 0.0000 SNAP GRID ORTHO POLAR OSNAP OTRACK LWT MODEL

Fig. 1 Running subroutine ptbl

The result of applying subroutine lead to an automatic file that containing the coordinates of the selected blocks.



```
    542219.187    308971.433     542219.187
    542236.006    308967.72     542236.006
    542226.937    308979.694     542226.937
    542225.979    308974.827     542225.979
    542242.309    308995.762     542242.309
    542245.003    308980.324     542245.003
    542245.122    308971.381     542245.122
    542246.221    308968.704     542246.221
    542247.493    308967.771     542247.493
    542242.6      308943.396     542242.6
    542237.411    308942.26      542237.411
    542272.058    308937.009     542272.058
    542270.12     308934.781     542270.12
```

Fig. 2 Getting the text file with coordinate values after applying the routine

4. Conclusion

Visual Lisp is an integrated programming environment where AutoLISP programs are developed and debugged. It should not be regarded as a substitute for AutoLISP language, but rather as an extension of AutoLISP language [3]. Adding a fully integrated development

environment (IDE) simplifies and speeds up the creation, debugging, and even packaging applications made by users in AutoLISP.
The advantages of using Visual LISP [4]:

- quality and work speed with a high level of automation;
- low cost of deployment by making an easy to install product, used and adapted to user needs;
- Increase the capacity of communication between design teams, a better integration of projects through the development of new facilities for connecting the design phases.

Visual LISP helps AutoCAD to become such a CAD system that combines "computer assisted drawing" with "computer aided design" [2]. With Visual LISP routines, measurement specialists can tailor AutoCAD to facilitate the way of obtaining the graphical objects, important part of the content of cadastral documentation.

5. References

1. *Didulescu Caius:"Automation of cadastral plans for the location and delimitation of properties in Romania using AutoLISP routines", International Symposium 'GeoPreVi' of Faculty of Geodesy, Bucharest, 2011*
2. *Didulescu Caius: „Automation of Cadastral Works Using Visual LISP Routines", Annals of University of Oradea", „Constructions and Hydro-Utility Installations Fascicle", Volume XIII-2, ISSN: 1454-4067, University of Oradea Publishing House, pag.93-98, 2010.*
3. *Didulescu Caius: Infografică pentru măsurători terestre şi cadastru – note de curs şi lucrări practice", Editura Conspress, Bucureşti, ISBN 978-973-100-091-6, 2009*
4. *Didulescu Caius: Some contributions in achievement of numerical cadastral plans., PhD Thesis, T.U.C.E.B., Bucharest, 2003.*
5. *Stăncescu C. (1996), AutoLISP – Manual de programare, Editura FAST 2000, Bucureşti.*