

# 3D AutoLISP Mesh Generation

Mohammed Amine Dirhar<sup>1</sup>, Essaid El Kennassi<sup>2</sup>, Khalid Idrissi Janati<sup>3</sup>,  
Lahbib Bousshine<sup>4</sup>

## Abstract

This work is the result of 3D mesh generation AutoLISP's code. We create the code for parallelepiped shape and we generate the x, y, z coordinates for each vertex. We obtain for the mesh generation: coordinates, nodes, and elements. This data is affected to a data structure file. The results obtained can be directly used for engineering need.

**Keywords:** AutoLISP, Mesh generation, Data structure

## Introduction

One of the most popular graphical software in CAD systems is AutoCAD. It is used by a very large modeling design community because of its numerous advantages. AutoCAD incorporate the AutoLISP standard for the customizing requirements [1]. AutoLISP is a programming language designed for extending and customizing the functionality of AutoCAD [1], [2]. Numerical computing is a set of calculus which is executed in an information system. To simulate natural phenomena like for example we use numerical computing to determine materials deformations caused by constraints. To do this computing, we need a simple personal computer which actually can elaborate complex calculus. Mechanical engineers use Finite Element Method FEM to solve problems related to mechanical structures, vibration, heat transfer, limit analysis and much more. The Finite element method uses a set of nodes which describe the topology of any geometrical shape [3]-[5]. Then, we can do the simulations based on the x, y, and z of each node. It is in this context, that we create AutoLISP's code for 3D mesh generation. This work begins with the methodology used to obtain a parallelepiped mesh. In second position we give results of our meshing code. Finally we give a conclusion.

## Method

AutoLISP programs elaboration for AutoCAD is based on writing code. This later used a text editor, and then executing them. The program debugging is done by the use of instructions adding. We used a set of instructions in the following form,

## Function Arguments

Each instruction is between parentheses. Each instruction give a value, this later can be used by another expression. These instructions are called lists. The geometry used is relatively simple because it shows how basic shapes are created and meshed. In this way, we can create more complex shapes. We used a rectangle defined with four points (vertex or nodes). Each two points form a line (edge). This rectangle is positioned in the XY plane. Then, we reproduce this rectangle along Z axis with the use of repeat command.

---

<sup>1,2,3,4</sup>Ensem Hassan II University, Casablanca, Morocco.

**Correspondence:** Mr. Essaid El Kennassi, Ensem Hassan II University, Casablanca, Morocco.

**E-mail Id:** [essaid.elkennassi@orange.fr](mailto:essaid.elkennassi@orange.fr)

**How to cite this article:** Dirhar MA, Kennassi EE, Janati KI et al. 3D AutoLISP Mesh Generation. *J Adv Res Mech Engi Tech* 2017; 4(1&2): 1-6.

ISSN: 2454-8650

After the nodes creation and mesh generation, we generate a set of space coordinates  $x$ ,  $y$ , and  $z$ . These coordinates correspond to the nodes. We affect results to a .dat file with the use of setq combined with the rtos commands.

The second step is the elements determination. For this, we used the same procedure to the coordinate determination. We affect elements results to a .dat file in the data structures need.

### Results

The results obtained are shown in Figure 1 and Figure 2, in which we used the code specified in appendices from A to F. In Figure 1, we have a parallelepiped meshed into four elements.

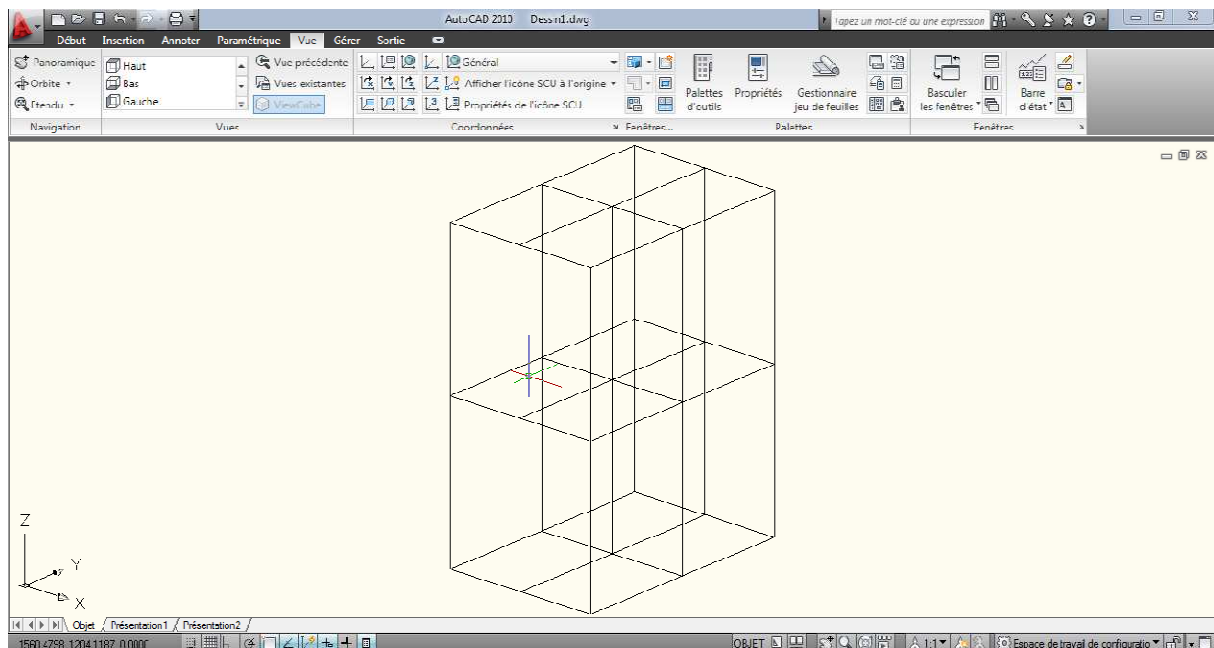


Figure 1.Parallelepiped meshed into four elements.

In Figure 2, we numbered the 8 elements from 1 to 8, and the 27 nodes from 1 to 27. Figure 3 shows the details about data structures and the mesh data. We

have the nodes numbers and the  $x$ ,  $y$  and  $z$  coordinates.

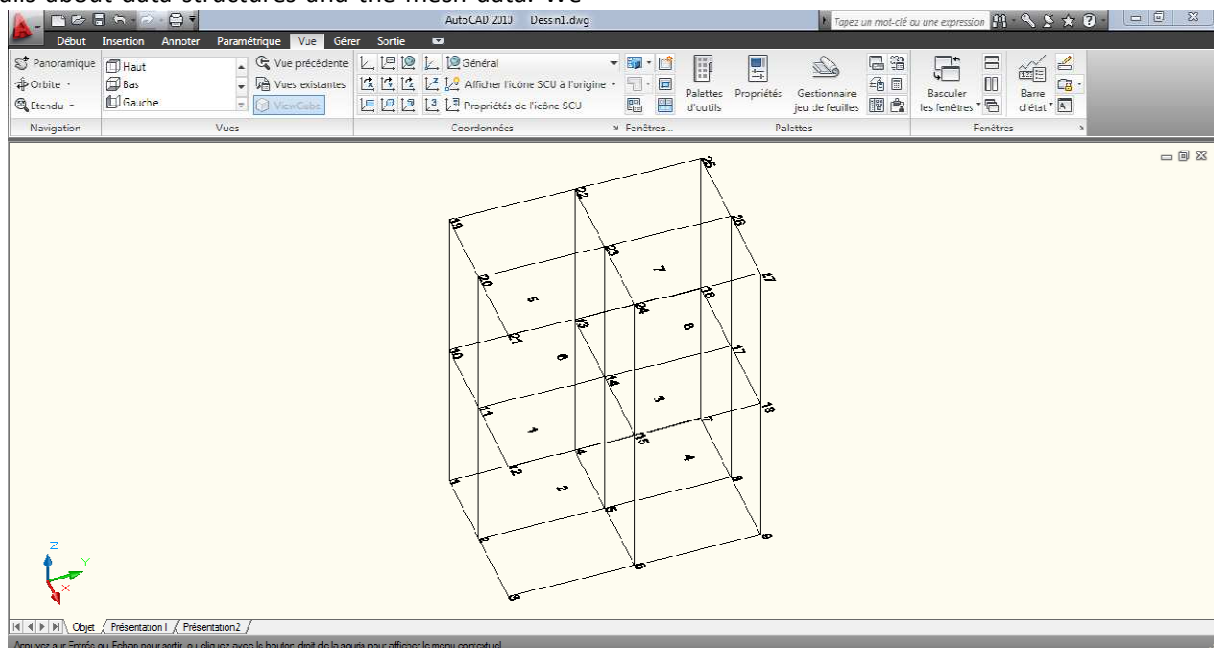


Figure 2.The elements and nodes are numbered for the affectation needs.

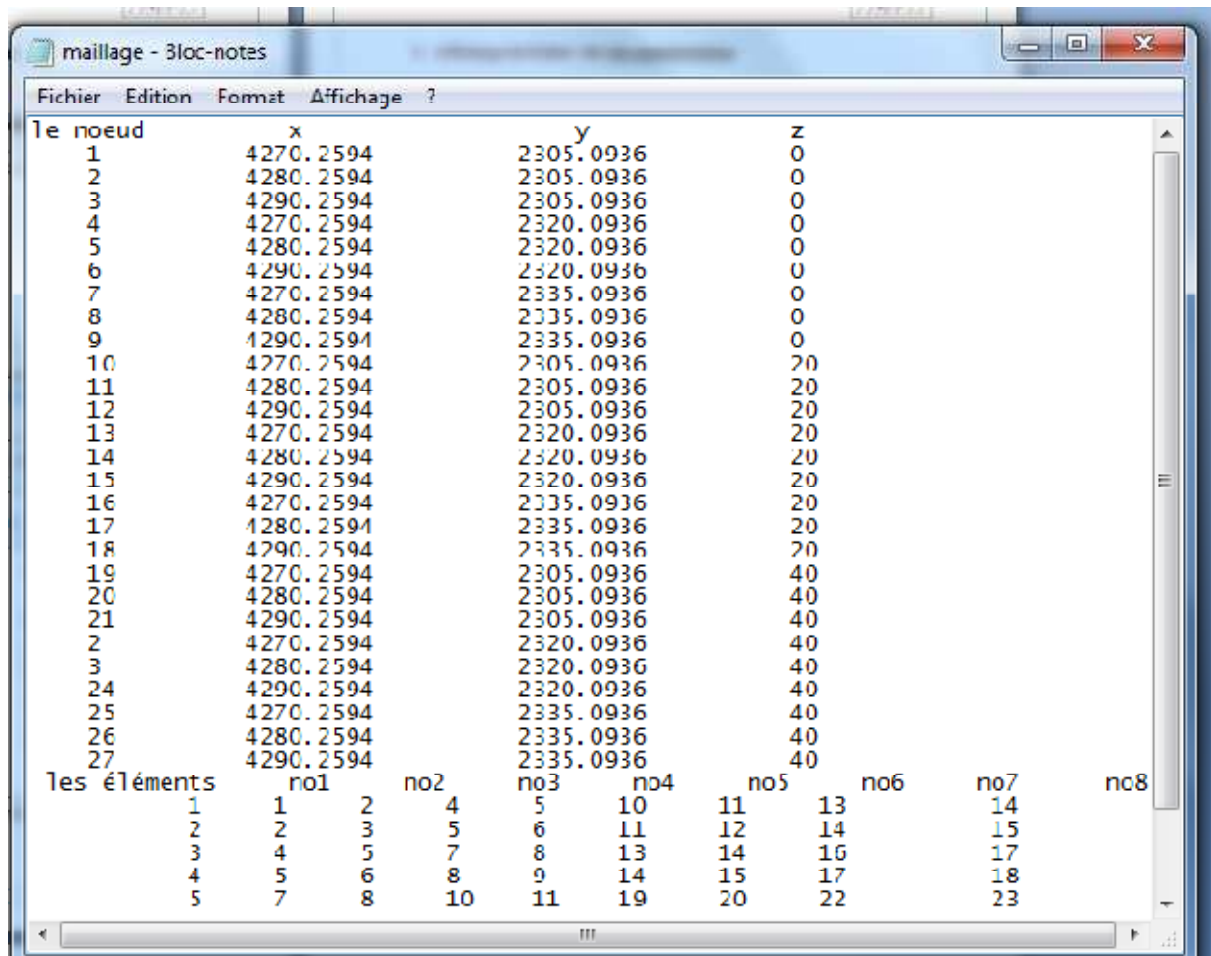


Figure 3. Data structures of the parallelepiped mesh

Then the data obtained from AutoLISP mesh generation can be used by the engineers.

**Conclusion**

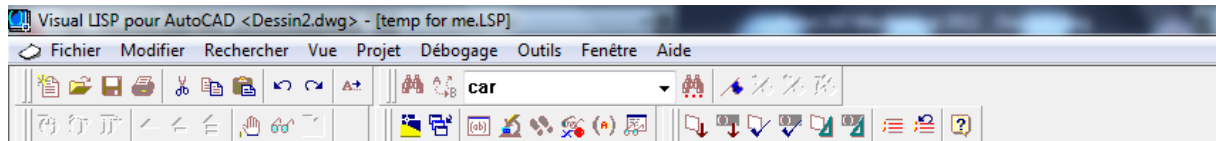
The used CAD computer aided design AutoCAD, is the most popular program of graphics and design aided by computer. It is used in several domains such as: architecture, mechanical design, aeronautics, vehicles design and much more. The methods presented in this paper refer to the volume mesh generation. We used for this several commands, such as setq, getpoint, getreal, getint, caddr, cadr, car, repeat, list, write-line, strcat, rtos, initget, getkword, while, if, prong. We show the mesh of a parallelepiped, with the space coordinate of each node. Thus, the results obtained in this paper allow us to suggest that our technical method for meshing could be applied to any form.

**References**

1. AutoCAD, "AutoLISP developer's guide." USA, Autodesk Inc. 2012.
2. R. Green, "AutoLISP level two: beyond the crash course." Autodesk University, 2008.
3. P. L. George, "Automatic mesh generation and finite element computation." P. G. Ciarlet and J. L. Lions (Eds.), Handbook of Numerical Analysis, Vol. IV, Elsevier Science B.V., 1996.
4. T. J. Baker, "Automatic Mesh Generation for Complex Three-Dimensional Regions Using a Constrained Delaunay Triangulation." Engineering with Computers, vol. 5, pp. 161-175, 1989.
5. H. Edelsbrunner, "Geometry and Topology for Mesh Generation." Cambridge, United Kingdom, Cambridge University Press, 2001.

## Appendices

### Appendix A. Parallelepiped conception



```

Visual LISP pour AutoCAD <Dessin2.dwg> - [temp for me.LSP]
Fichier Modifier Rechercher Vue Projet Débogage Outils Fenêtre Aide
car
(defun c:maillagetemp ()
;dessin du rectangle
  (setq pt (getpoint "cliquer:"))
  (setq l (getreal "\ndonner la longueur:"))
  (setq e (getreal "donner la largeur:"))
  (setq h (getreal "donner la hauteur:"))

  (setq n1 (getint "entrer le nombre d'éléments suivant x:"))
  (setq n2 (getint "entrer le nombre d'éléments suivant y:"))
  (setq n3 (getint "entrer le nombre d'éléments suivant z:"))

  (setq d1 (/ l n1))
  (setq d2 (/ e n2))
  (setq d3 (/ h n3))

  (setq z1 (caddr pt))
  (setq x1 (car pt))
  (setq y1 (cadr pt))
  (setq c z1)
  (repeat (+ n3 1)

  (setq pt1 (list x1 y1 c))
  (setq pt2 (list (+ x1 l) y1 c))
  (setq pt3 (list (+ x1 l) (+ y1 e) c))
  (setq pt4 (list x1 (+ y1 e) c))

  (command "ligne" pt1 pt2 pt3 pt4 pt1 ""))

```

### Appendix B. Parallelepiped meshing

```

; maillage
  (setq i (+ x1 d1))
  (repeat (- n1 1)
    (setq pt5 (list i y1 c))
    (setq pt6 (list i (+ y1 e) c))
    (command "ligne" pt5 pt6 "")
    (setq i (+ i d1))
  )
  (setq c (+ c d3))
  (setq v x1)
  (repeat (+ n1 1)
    (setq pt7 (list v y1 z1))
    (setq pt8 (list v (+ y1 e) z1))
    (setq pt9 (list v (+ y1 e) (+ z1 h)))
    (setq pt10 (list v y1 (+ z1 h)))

    (command "ligne" pt7 pt8 pt9 pt10 pt7 ""))

    (setq j (+ y1 d2))
    (repeat (- n2 1)
      (setq pt11 (list v j z1))
      (setq pt12 (list v j (+ z1 h)))
      (command "ligne" pt11 pt12 "")
      (setq j (+ j d2))
    )
  )
  (setq u (+ v d1))
  (setq w y1)
  (repeat (+ n2 1)

  (setq pt13 (list x1 w z1))
  (setq pt14 (list (+ x1 l) w z1))
  (setq pt15 (list (+ x1 l) w (+ z1 h)))
  (setq pt16 (list x1 w (+ z1 h)))

  (command "ligne" pt13 pt14 pt15 pt16 pt13 ""))

  (setq k (+ z1 d3))
  (repeat (- n3 1)
    (setq pt17 (list x1 w k))
    (setq pt18 (list (+ x1 l) w k))
    (command "ligne" pt17 pt18 "")
    (setq k (+ k d3))
  )
  )
  (setq w (+ w d2))

```

## Appendix C. Data structure .dat file creation

```
; creation dans le fichier
```

```
(setq f (open "A:\\maillage.dat" "w"))
(setq no "le noeud")
(setq cordx "x")
(setq cordy "y")
(setq cordz "z")
(write-line (strcat no " " " cordx " " " cordy " " "cordz) f)
```

## Appendix D. File's nodes affecting

```
;affectation des noeuds au fichier
```

```
(setq d 1)
(setq ss 1)
(setq o z1)
(repeat (+ n3 1)
  (setq q y1)
  (repeat (+ n2 1)
    (setq s x1)
    (repeat (+ n1 1)
      (setq pt19 (list s q o))
      (setq kk (rtos d))
      (setq kk1 (rtos s))
      (setq kk2 (rtos q))
      (setq kk3 (rtos o))
      (setq line1 (strcat kk " " " kk1 " " " kk2 " " " kk3))
      (write-line (strcat " " " line1) f)
      (setq d (+ d 1))
      (setq s (+ s d1))
      (command "texte" pt19 "1.1" (/ pi 2) ss)
      (setq ss (+ ss 1))
    )
    (setq q (+ q d2))
  )
  (setq o (+ o d3))
)
```

## Appendix E. Elements numbering

```
;numerotation elements
```

```
(setq sss 1)
(setq oo (+ z1 (/ d3 2)))
(repeat n3
  (setq qq (+ y1 (/ d2 2)))
  (repeat n2
    (setq u (+ x1 (/ d1 2)))
    (repeat n1
      (setq pt30 (list u qq oo))
      (command "texte" pt30 "1.1" (/ pi 2) sss)
      (setq sss (+ sss 1))
      (setq u (+ u d1))
    )
    (setq qq (+ qq d2))
  )
  (setq oo (+ oo d3))
)
```

## Appendix F. File's elements affecting

```
;affectation des éléments au fichier
```

```
(setq n4 1)
(setq n5 1)
(setq hh 0)

|
  (setq ele "Les éléments")
  (setq nno1 "no1")
  (setq nuu2 "uu2")
  (setq nuu3 "uu3")
  (setq nno4 "no4")
  (setq nno5 "no5")
  (setq nno6 "no6")
  (setq nno7 "no7")
  (setq nno8 "no8")
  (setq line3 (strcat ele " " nno1 " " nno2 " " nno3 " " nno4 " " nno5 " " nno6 " " nno7 " " nno8))
  (write-line (strcat " " line3) f)

(repeat n3
  (repeat n2
    (repeat n1
      (setq n6 (+ n2 1))
      (setq n7 (+ n1 1))
      (setq n8 (* n6 n7))
      (setq n9 (+ n8 1))
      (setq no1 n4)
      (setq no2 (+ no1 1))
      (setq nu3 (+ nu1 (+ n1 1)))
      (setq nu4 (+ nu1 (+ n1 2)))
      (setq no5 (+ n9 hh))
      (setq no6 (+ no5 1))
      (setq no7 (+ no6 n1))
      (setq no8 (+ no7 1))

      (setq n41 (rtos n5))
      (setq no11 (rtos no1))
      (setq no21 (rtos no2))
      (setq no31 (rtos no3))
      (setq no41 (rtos no4))
      (setq no51 (rtos no5))
      (setq no61 (rtos no6))
      (setq no71 (rtos no7))
      (setq no81 (rtos no8))

      (setq line4 (strcat n41 " " no11 " " no21 " " no31 " " no41 " " no51 " " no61 " " no71 " " no81))
      (write-line (strcat " " line4) f)
      (setq n4 (+ n4 1))
      (setq n5 (+ n5 1))
      (setq hh (+ hh 1))
    )
    (setq n4 (+ n4 1))
    (setq hh (+ hh 1))
  )
  (setq hh (+ hh (+ n1 1)))
)
)
```