Programmeren in AutoLISP

Les twee

AutoLISP in een notendop

Het begin

door

Joop F. Moelee

een gelovig volger van
“de Sacrale Kunst van Luiheid”
zijn Hoge Priester LISP en Acoliet Script

1) Inhoudsopgave

2) Verantwoording

Dit document is gebaseerd op en/of bevat delen van de artikelen geschreven door Kenny Ramage eigenaar/ beheerder van de site http://www.afraLISP.com/
Kenny heeft de schrijver dezes toestemming gegeven tot gebruik van zijn teksten.
Hiervoor heel hartelijk bedankt, Kenny.

Voor alle duidelijkheid eerst even de verklaringen van de diverse tekstsoorten.
Italic--Engelse woorden en termen die geen AutoLISP en/of AutoCAD termen zijn.
Italic en vet – AutoCAD termen, commando's en menu opdrachten.
Kleiner en vet – regels programmacode en AutoLISP termen.

3) Wat is AutoLISP

LISP staat voor: *List Processor*. Oftewel het Manipuleren van Lijsten.
AutoLISP is ontwikkeld uit LISP om AutoCAD aan te passen aan jouw specifieke eisen en wensen.
AutoLisp is een geïnterpreteerde taal en niet een gecompileerde zoals veel andere talen (b.v. VBA). Dit betekent dat als het programma in het geheugen wordt ingelezen elk programma instructie wordt geëvalueerd en vertaald.
Dit in tegenstelling van een gecompileerd programma dat van tevoren door een externe compiler is vertaald naar machine niveau.
Elk programma instructie in AutoLISP bestaat een set functies en andere door de functies gebruikte data ingesloten door haakjes, de zogenaamde *parentheses*.
Een voorbeeld hiervan is:

(command "line" pt1 pt2 pt3 pt4 "close")

De functie is hier **command** dat vooraf gaat aan de bekende AutoCAD opdracht ***line*** en de subopdracht ***close***. De inhoud van de haakjes is een volledige opdracht om een lijn te tekenen van punt1 naar punt2, verder naar punt3 en naar punt4 en om tot slot de figuur te sluiten door een lijn naar het beginpunt (punt1) te tekenen. Van tevoren moet je wel de vier punten definiëren zodat AutoCAD ze kan gebruiken om de vierhoek te tekenen.

4) Invoer via Console window

Laten we nu eens beginnen met enkele dingen uit te proberen.
Start AutoCAD op en open de Visual LISP Editor. Type in de **Console window**:

_$ (alert "Aanvalluh!!!")

Voor de eerste en laatste keer: uiteraard zonder de **_$** want die staat er al.
Druk nu op ***enter*** en voilá:

Goed zo! Je hebt net AutoLISP gebruikt om AutoCAD iets te laten doen.
Ja ja, ik weet het. Bijna alle andere cursussen voor het programmeren gebruiken "Hello World". Maar dit is een Nederlandse les en dus gebruik ik een welbekende Nederlandse term.

Ik weet niet of het je opgevallen is, maar door het gebruik van de **alert** functie werd een ***dialogue box*** op het scherm afgebeeld. Verder werd in de **Console window** de waarde **nil** geretourneerd. Dit betekent in dit geval: alles is goed verlopen en opdracht afgesloten.

Nu iets anders. Type het volgende op de **Console prompt** en druk op ***enter***:

_$ (setq a (getpoint))

Kies ergens op het scherm een punt en een lijst met nummers verschijnt in de **Console window**. De lijst lijkt op dit:

(496.0 236.5 0.0)

Geloof het of niet: dit zijn de x, y en z coördinaten van het gekozen punt.

x = 496.0
y = 236.5
z = 0.0

De AutoLISP code **(setq a (getpoint))** betekent in normaal Nederlands:

> Neem van de gebruiker een punt en sla de x, y, en z waarden op in de variabele a in de vorm van een lijst .

5) Haakje openen en sluiten

Is het je opgevallen dat alles is ingesloten met haakjes?

Alle AutoLISP functies worden omsloten door haakjes. Je kunt je functies ook nesten. Denk er wel aan dat je het nest moet verlaten met een gelijk aantal haakje sluiten als dat je voor openen gebruikt hebt. Bijvoorbeeld:

(DoeIets (DoeIetsAnders (EnDoeNogWatAnders)))

Je hebt met drie haakjes geopend dus moet je met drie haakjes sluiten.
Om het beter leesbaar te maken kun je bovenstaande ook schrijven als:

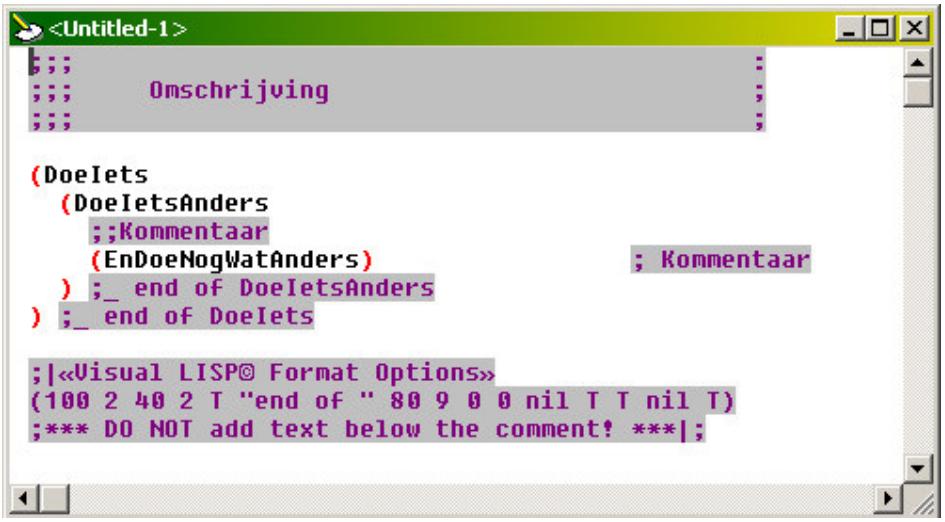
```
(DoeIets
        (DoeIetsAnders
                (EnDoeNogWatAnders)
        )
)
```

Nu zie je meteen waarom LISP ook vertaald wordt als *Lost In Stupid Paratheses.* In het Nederlands: Gek Worden Van De Haakjes.

6) Commentaar toevoegen

Je kunt ook commentaar toevoegen aan je code.
Alles wat na een puntkomma komt wordt door de interpreteerder genegeerd.

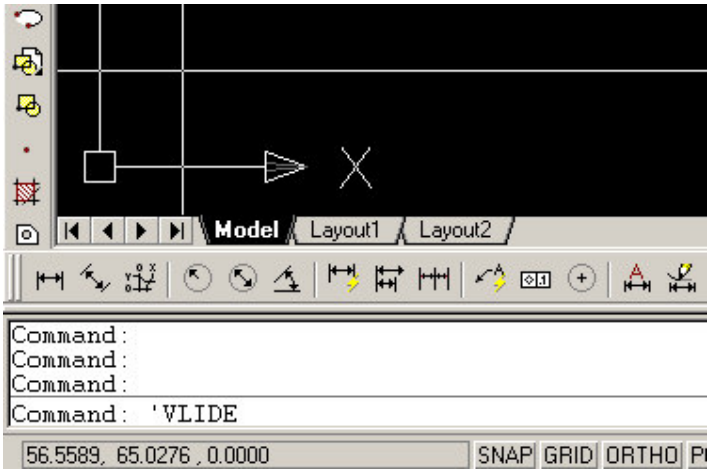
;;; →blijft aan het begin van een regel staan.
;; →verplaatst het commentaar naar de volgende regel.
;_ →blijft achter het laatste haakje op dezelfde regel staan.
; →verplaats de tekst naar het einde van de regel.
;| en |; →tekst tussen deze karakters kan over meerdere regels verspreidt zijn.



Je weet het: commentaar maakt het programma begrijpelijk, ook na 5 jaar nog.

7) Stel een vraag

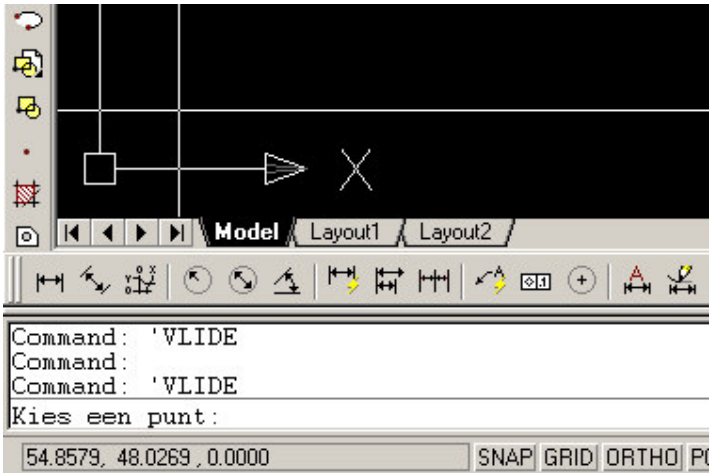
Met de opdracht **(setq a (getpoint))** weet de gebruiker niet echt wat er van hem/haar verwacht wordt. Het scherm schakelt naar AutoCAD en op de ***command line*** verschijnt ***_VLIDE*** en de cursor verandert in kruisharen zonder ***pickbox***.



Als je de opdracht verandert in:

_$ (setq a (getpoint "\nKies een punt: "))

vraagt AutoCAD in het ***command window*** aan de gebruiker wat er van hem/haar verwacht wordt.



Een beetje communiceren kan nooit kwaad en maakt het werken met programma's een stuk leuker.

8) Het eerste programma

8.1) Rechtstreekse invoer

Type op de **console prompt**:

_$ (setq a (getpoint "\nKies beginpunt: "))

Druk op ***ENTER*** en kies een punt als startpunt. Type nu:

_$ (setq b (getpoint "\nKies eindpunt: "))

Druk weer op ***ENTER*** en kies een ander punt als eindpunt. Type ten slotte:

_$ (command "Line" a b "")

en druk weer op ***ENTER***. Ga naar het AutoCAD venster en daar staat de lijn getekend tussen de punten die je eerst gekozen hebt.
De **command** functie wordt gebruikt om AutoCAD te vertellen wat het moet doen:

"Line" → Teken een lijn
a → Vanaf het punt opgeslagen in variabele "a"
b → Tot het punt opgeslagen in variabele "b"
"" → ***ENTER*** om het ***line*** commando af te sluiten

8.2) Het programmabestand

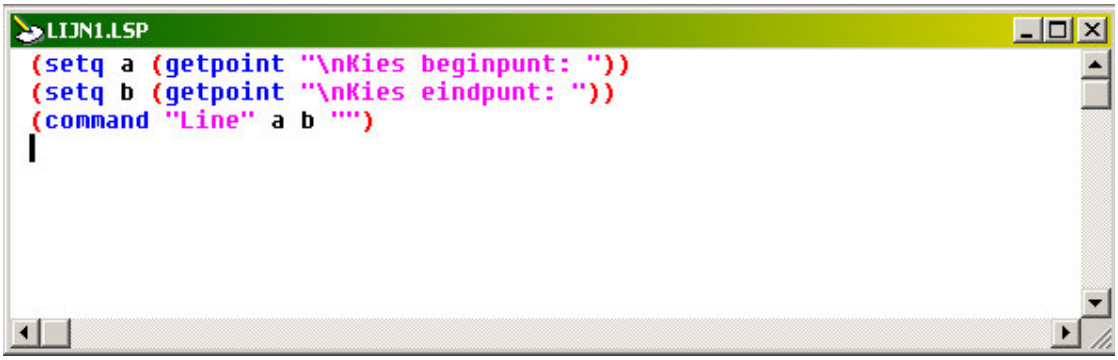
Maar wat als ik deze routine weer wil uitvoeren? Moet ik dan alles opnieuw intypen?
Gelukkig niet. AutoCAD kan deze routine ook zelf vanuit een bestand inlezen, mits er aan een paar voorwaarden voldaan wordt:

- De opdrachten moeten in een ongeformateerd tekstbestand opgeslagen zijn. Dit wil o.a. zeggen dat er geen vette of schuine karakters in mogen staan.
- Het bestand moet de extensie LSP hebben. Er bestaat ook nog de mogelijk het programma te comprimeren naar een *.FAS of *.VLX bestand, maar dat wordt later behandeld.
- Het bestand moet opgeslagen zijn in een directory die in het zoekpad van AutoCAD is opgenomen.

Goed, open een nieuw bestand in de Visual LISP Editor (zie hiervoor Les een hoofdstuk 5.1).

8.3) Directe programma executie

Type of kopieer de vorige drie opdrachten in dezelfde volgorde naar het **Editor window** en sla het programma op als LIJN1.LSP.
De **Editor window** ziet er dan zo uit:



Als we dit programma laden zoals het is, worden de opdrachtregels ogenblikkelijk uitgevoerd. Probeer maar uit. Type op de Command prompt van AutoCAD ***(load "lijn1.lsp")*** en druk op ***ENTER***.

Onthoud dit goed, want dit is een zeer bruikbare eigenschap die we nog vaak zullen gebruiken. Als je bij het laden van een programma wil dat er meteen iets gebeurt dan plaats je dat aan het begin van het programma buiten de feitelijke programmacode. Zoiets als:

```
(alert "Copyright : \n\n\tIkke")
```

maar je kunt ook controleren of de gebruiker wel rechten heeft om jouw programma te gebruiken. Dit is echter voor gevorderden en niet voor nu. Normaal gesproken willen we het programma echter starten door het intypen van een commando. Bijvoorbeeld ***lijn1***. We gaan de code zodanig aanpassen dat dit mogelijk is.

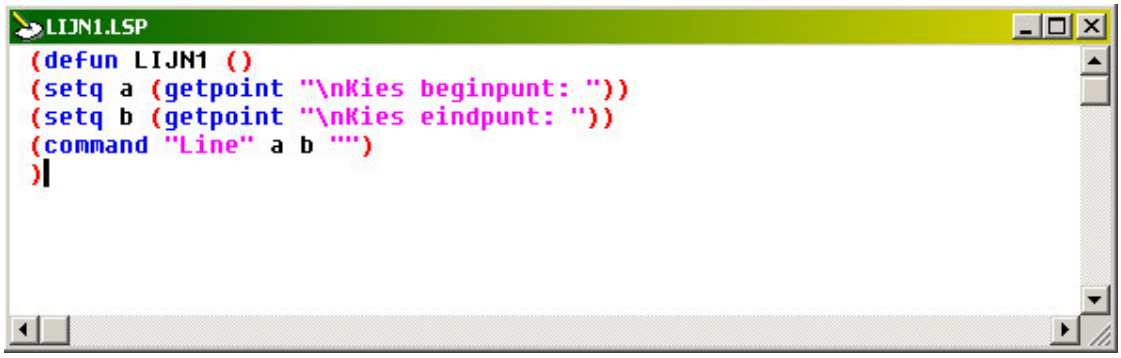
8.4) De code formatteren

Voor nu en voor de toekomst:

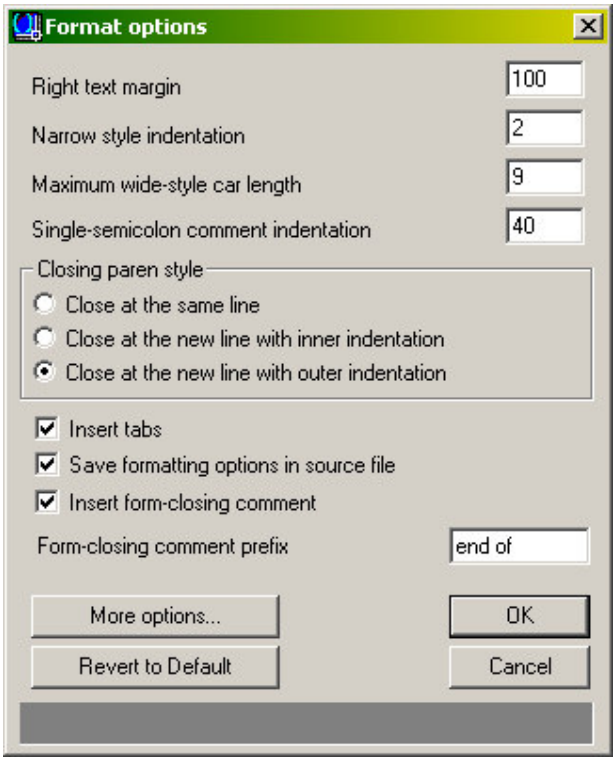
Bij de code die je vanaf nu in deze lessen tegenkomt geldt dat **nieuwe** code in het rood is getypt, **oude** code in het zwart en **vervallen** code in het groen.
Bij wijzigingen in een code geldt dan ook: voeg de rode code toe en haal de groene code weg.

```
(defun LIJN1 ()
   (setq a (getpoint "\nKies beginpunt: "))
   (setq b (getpoint "\nKies eindpunt: "))
   (command "Line" a b " ")
)
```

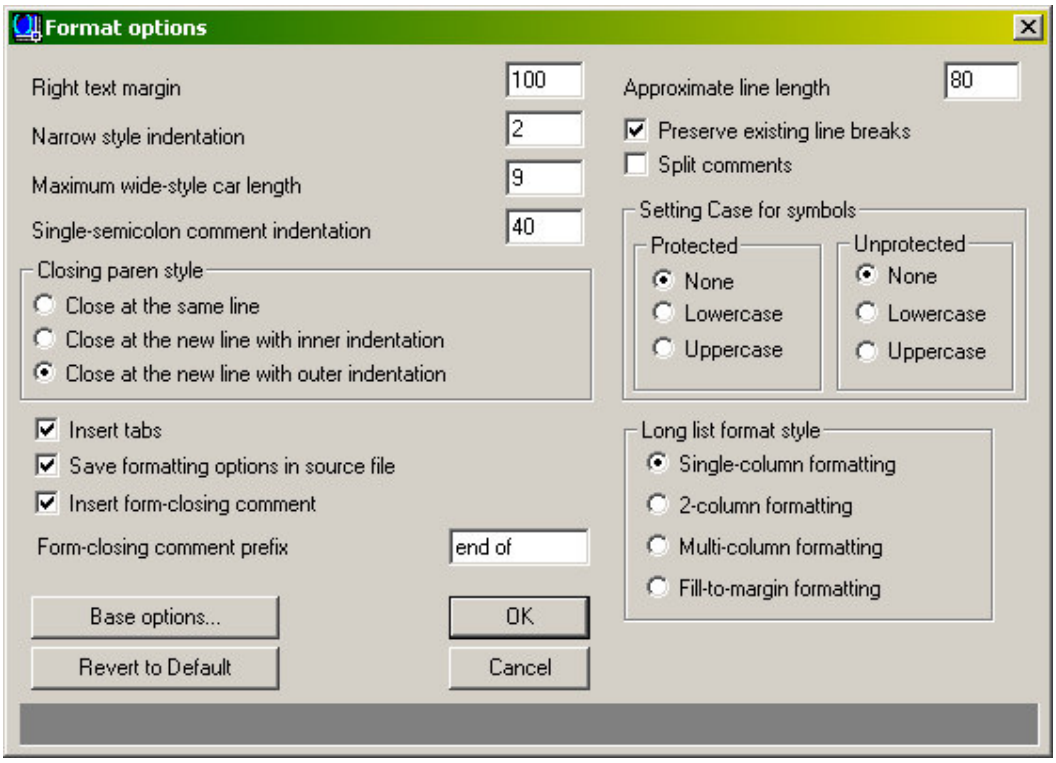
De **Edit window** ziet er dan zo uit:



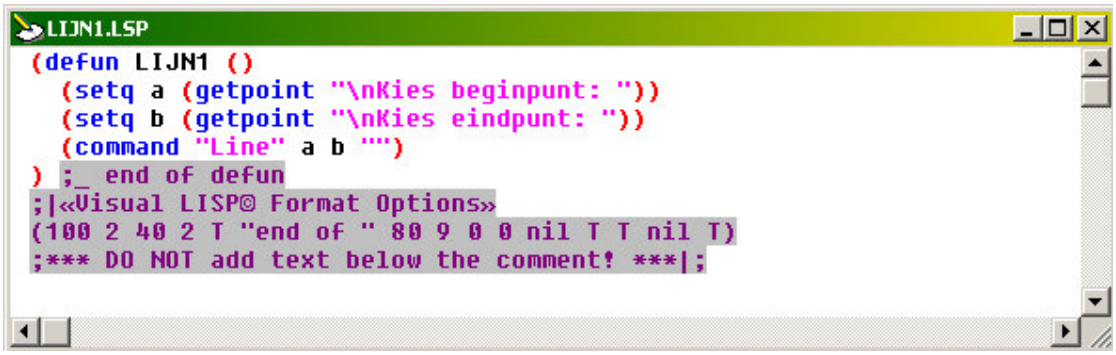
Om dit beter te kunnen lezen zou een beetje formattering, zoals inspringingen welkom zijn. Wil je meer weten over de formattering van programmacode lees dan **Programmeren in AutoCAD** van ondergetekende en te downloaden van http://www.cadsite.be/lisp/Programmeren_in_AutoCAD.pdf .
Ga naar ***Tools → Environment Options → Visual LISP Format Options*** .



Klik op ***More options…***



Speel met deze instellingen tot je een *format* hebt die je wel bevalt. Niet goed: klik op ***Revert to Default*** en de fabrieks instellingen worden hersteld.
Om de code te formatteren: ***Tools → Format code in Editor*** of druk op ***CRTL+ALT+F***. Als je de hier getoonde instellingen gebruikt ziet het venster er daarna zo uit:



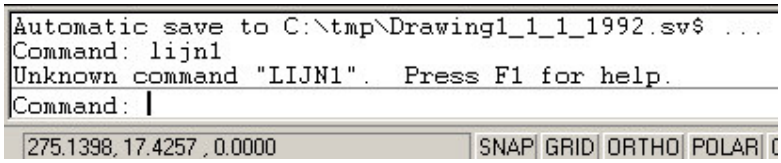
Als je naar het begin van regel 1 en regel 5 kijkt zie je de verplichte en bij elkaar behorende haakje openen en haakje sluiten.

8.5) Een nieuw AutoCAD commando

De AutoLISP functie **defun** geeft aan dat er een nieuwe functie is gedefinieerd met in dit geval de naam LIJN1.
Laadt dit programma in de **Console window** en type op de prompt:

_$ (lijn1)

Het programma wordt uitgevoerd.
Nu ga naar de AutoCAD ***Command window*** en typ daar ***lijn1***.



Foutmelding! Het programma wordt niet uitgevoerd.
Type nu eens ***(LIJN1)***. Het programma wordt nu wel uitgevoerd, maar wel als AutoLISP functie.
HELA!! We willen geen nieuwe AutoLISP functie maar een nieuw AutoCAD commando!
Terug naar de code en maak er een AutoCAD **commando** van:

```
(defun C:LIJN1 ()
   (setq a (getpoint "\nKies beginpunt: "))
   (setq b (getpoint "\nKies eindpunt: "))
   (command "Line" a b " ")
)
```

(defun C:LIJN1 vertelt AutoCAD dat het programma een nieuw commando is.
Sla de code op en laad het opnieuw in de **Console window**.
Start het programma nu vanaf de ***command prompt*** en het werkt als een bijtje in een honingpot.

9) Tot slot

We hebben gezien hoe we een AutoLISP functie maken en hoe een AutoCAD commando. En dat alles in een notedop.
In Les drie gaan we hier dieper op in en zullen we het o.a. hebben over de verschillende soorten data die AutoLISP kent en hoe we moeten omgaan met variabelen.

Tot dan,

Joop