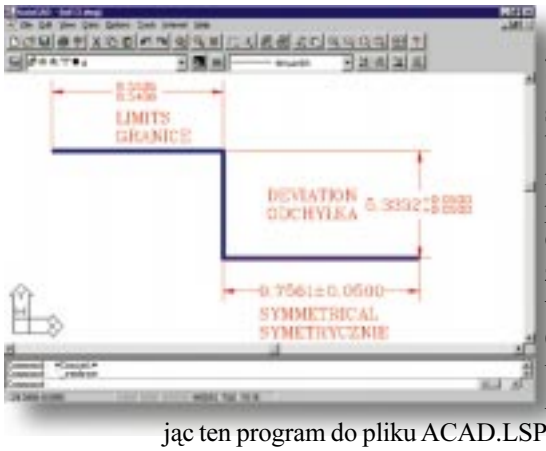
KONTROLOWANIE TEKSTU WYMIARUJĄCEGO

W artykule tym poznamy sposób działania polecenia DIM, sterującego wyświetlaniem tekstów wymiarujących. Peter Chay z Singapuru chciałby zmienić wartość tekstu wymiarującego, jednakże zmiana ta powoduje zniknięcie tolerancji oraz granic wymiarowania. Rozwiązaniem problemu naszego czytelnika jest program `DIMUTILS.LSP`, za pomocą którego możesz zmienić nie tylko sam tekst wymiarujący, lecz także określić styl tolerancji oraz granic wymiarowania. Program ten działa w programie AutoCAD, w wersjach 12 i 13 [angielskich – dop. red.].

Program AutoCAD r12 udostępnia trzy sposoby określania tolerancji, jak to zostało przedstawione na rysunku 1. Styl *Limits* powoduje wyświetlenie dwóch liczb jedna nad drugą. W stylu *Variance*, z prawej strony głównego tekstu wymiarującego, wyświetlane są dwie liczby – dodatnia i ujemna wartość niezgodności. Jeśli obie te wartości są identyczne, to wyświetlana jest tylko jedna z nich; w takim przypadku, pomiędzy tekstem wymiarującym i wartością niezgodności umieszczony zostaje znak ±.

W programie AutoCAD r13 dostępne są także trzy sposoby prezentacji tekstu wymiarującego oraz tolerancji: *Limits*, *Symmetrical* oraz *Deviation*; zostały one zaprezentowane na rysunku 2.

Jak działa program DIMUTILS



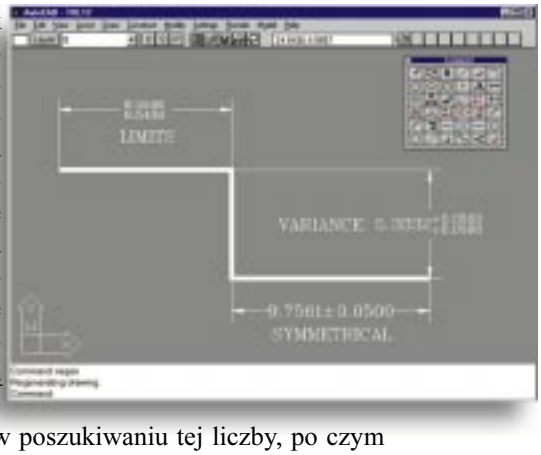
Skopiuj program `DIMUTILS.LSP` do swojej kartoteki roboczej lub do pomocniczej kartoteki programu AutoCAD. Następnie załaduj ten program wpisując polecenie **(Load "dimutils")**, używając polecenia Appload lub dodając ten program do pliku ACAD.LSP (w tym przypadku program zostanie załadowany automatycznie).

Po załadowaniu programu wywołaj polecenie **DT**, aby uruchomić aplikację. W pierwszej kolejności program *Dimutils* prosi o podanie wymiaru, a następnie, w zależności od wersji programu AutoCAD-a, zażąda podania odpowiednich wartości tolerancji lub wartości granicznych. Za każdym razem dostępne są wartości domyślne, będące oryginalnymi wartościami wymiarów, dzięki czemu jeśli niczego nie będziesz chciał zmieniać, wystarczy nacisnąć klawisz <Enter>.

Zmieniać możesz tylko wymiary liniowe, tolerancje i wartości graniczne. Jeśli wybierzesz wymiar, który nie posiada tolerancji, to nie stanie się nic, a program zakończy działanie. Polecenie to działa także w AutoCAD-zie r13 w przypadku podawania wymiarów promienia posiadającego tolerancję; wówczas nową wartość promienia powinieneś poprzedzić literą **R**, na przykład R3500.

Zwróć uwagę, że jeśli użyjesz polecenia *Update* lub jeśli spróbujesz podać nowy tekst wymiarujący, to podany wymiar będzie przeliczony, a wyniki działania programu *Dimutils* zostaną utracone. Programu *Dimutils* powinieneś używać jako ostatecznej metody modyfikowania tekstu wymiarującego.

Uwagi dotyczące programu



Podobnie jak czyni to moja procedura obsługi błędów oraz procedura zarządzająca zmiennymi systemowymi, także i główna procedura programu (*chdimtxt*) prosi o podanie liczby reprezentującej wybieraną opcję, następnie przegląda tablicę w poszukiwaniu tej liczby, po czym przekazuje ją do funkcji (`do-r12`) lub (`do-r13`), w zależności od używanej wersji AutoCAD-a.

Funkcja (`do-r13`) przechowuje informacje dotyczące wymiaru jako pojedynczy łańcuch znaków, w którym zapisane są wszystkie dane dotyczące zarówno tekstu wymiarującego, jak i tolerancji. Funkcja analizuje łańcuch znaków próbując ustalić użyty sposób prezentacji tekstu wymiarującego – *limits*, *symmetrical* lub *deviation*. W zależności od sposobu prezentacji wykonywana jest następnie jedna z trzech funkcji – (`do13-l`), (`do13-s`) lub też (`do13-d`). Każda z tych funkcji zwraca nowy łańcuch znaków, zmodyfikowany na podstawie informacji podanych przez użytkownika, po czym za pomocą funkcji (*entmod*) i (*entupd*) modyfikowany jest rysunek.

Funkcja (`do-r12`) pod wieloma względami przypomina swoją odpowiedniczkę – funkcję (`do-r13`). Główna różnica polega na tym, iż tekst wymiarujący przechowywany jest w trzech oddzielnych zmiennych wewnątrz bloku. Odpowiednik symetrycznej niezgodności – tekst poprzedzony znakiem ± – przechowywany jest w zmiennej , zawierającej znak ± reprezentowany za pomocą wyrażenia %%p. Jeśli został użyty sposób prezentacji *Limits*, to dolna i górna wartość graniczna przechowywana jest w dwóch zmiennych; natomiast w przypadku użycia sposobu *Variance* – informacje zapisywane są w trzech zmiennych.

Tony Hotchkiss

Przedruk i tłumaczenie za zgodą –
Reprinted with permission from –
CADALYST magazine, 1997, 01, Tony Hotchkiss

Program `dimutils.lsp` dostępny jest na stronie www 3D: `http://www.helion.com.pl/3d/index/3dindex.htm`

```
;; DIMUTILS.LSP - a lisp program to change dimension
;;  text with limits and tolerances.
;;  Copyright 1996 by Tony Hotchkiss.
;************************************
(defun err (s)
    (if (= s "Function cancelled")
        (princ "\nDIMUTILS - cancelled: ")
       (progn (princ "\nDIMUTILS - Error: ")(princ
    s)
       (terpri))
   ); if
    (resetting)
    (princ "SYSTEM VARIABLES have been reset\n")
    (princ)
); err
(defun setv (systvar newval)
    (setq x (read (strcat systvar "1")))
    (set x (getvar systvar))
    (setvar systvar newval)
); setv
(defun setting ()
    (setq oerr *error*)
    (setq *error* err)
    (setv "CMDECHO" 0)
    (setv "BLIPMODE" 0)
); end of setting
(defun rsetv (systvar)
    (setq x (read (strcat systvar "1")))
    (setvar systvar (eval x))
); restv
(defun resetting ()
    (rsetv "CMDECHO")
    (rsetv "BLIPMODE")
    (setq *error* oerr)
); end of resetting
(defun dxf (code ename)
    (cdr (assoc code (entget ename)))
); dxf
(defun ch-dimtxt (/ ename bname tblist dname)
    (setq ename (car (entsel
        "\nSelect dimension: ")))
    (while (or (null ename) (/= (dxf 0 ename)
       "DIMENSION"))
       (setq ename (car (entsel
           "\nNot a dimension; select again: "))
      ); setq
   ); while
    (setq bname (dxf 2 ename)
          tblist (tblsearch "BLOCK" bname)
          dname (cdr (assoc -2 tblist))
   ); setq
    (if (wcmatch (ver) "*13*")
        (do-r13 ename dname)
        (do-r12 ename dname)
   ); if
    (princ)
); ch-dimtxt
(defun do-r13 (ename dname / oldlist oldtxt bkslsh
               plusminus str str2 len j dimtype cr
               oldstr nwstr newtxt newlist)
    (while (/= (dxf 0 dname) "MTEXT")
        (setq dname (entnext dname))
   ); while
    (setq oldlist (entget dname))
    (setq oldtxt (assoc 1 oldlist))
    (setq bkslsh "\\"
          plusminus (chr 177)
          str (cdr oldtxt)
          str2 (substr str 5)
          len (strlen str2)
         j 1
          dimtype nil
   ); setq
    (if (and (= (substr str2 1 1) bkslsh)
             (= (substr str2 2 1) "S")
      ); and
       (setq dimtype "Lim")
       (progn
         (repeat (- len 2)
            (setq cr (substr str2
    (setq j (1+ j)) 1))
           (if (= cr plusminus)
    (setq dimtype "Sym"))
           (if (and (= (substr str2 j 1) bkslsh)
                    (= (substr str2 (1+ j) 1) "S")
              ); and
              (setq dimtype "Dev")
          ); if
        ); repeat
       ); progn
   ); if
    (setq oldstr (dxf 1 dname))
    (setq nwstr
      (cond
          ((= dimtype "Lim") (do13-l oldstr))
          ((= dimtype "Sym") (do13-s oldstr))
          ((= dimtype "Dev") (do13-d oldstr))
        (T nil)
      ); cond
    ); setq
    (if nwstr
      (progn
          (setq newtxt (cons 1 nwstr)
                newlist (subst newtxt oldtxt oldlist)
         ); setq
          (entmod newlist)
          (entupd ename)
      ); progn
   ); if
); do-r13
(defun do13-l (str / prefx str2 len pos ul ll caret
               msg1 msg2 newul newll nwstr)
    (setq prefx (substr str 1 6)
          str2 (substr str 7)
          len (strlen str2)
          pos (/ len 2)
          ul (substr str2 1 (1- pos))
          ll (substr str2 (+ pos 1) (1- pos))
          caret (chr 94)
          msg1 (strcat "\nNew upper limit <"
                ul ">: ")
          msg2 (strcat "\nNew lower limit <"
                ll ">: ")
          newul (getstring msg1)
          newll (getstring msg2)
   ); setq
    (if (= newul "") (setq newul ul))
    (if (= newll "") (setq newll ll))
    (setq nwstr (strcat prefx newul caret newll ";"))
); do13-l
(defun do13-s (str / prefx str2 plusminus len i cr
               dmval pmtol msg1 msg2 newv1 newv2 nwstr)
    (setq prefx (substr str 1 4)
          str2 (substr str 5)
          plusminus (chr 177)
          len (strlen str2)
         i 0
   ); setq
    (while (/= cr plusminus)
       (setq cr (substr str2 (setq i (1+ i)) 1))
   ); while
    (setq dmval (substr str2 1 (1- i))
          pmtol (substr str2 (1+ i))
          msg1 (strcat "\nNew dimension text <"
                dmval ">: ")
          msg2 (strcat "\nNew plus-minus tolerance <"
                pmtol ">: ")
          newv1 (getstring msg1)
          newv2 (getstring msg2)
   ); setq
    (if (= newv1 "") (setq newv1 dmval))
    (if (= newv2 "") (setq newv2 pmtol))
    (setq nwstr (strcat prefx newv1 plusminus newv2))
); do13-s
```

```
(defun do13-d (str / prefx1 str2 caret i cr dmval
                    str3 prefx2 len halflen numlen ul ll
                    msg1 msg2 msg3 newv1 newv2 newv3 nwstr)
    (setq prefx1 (substr str 1 4)
          str2 (substr str 5)
          caret (chr 94)
          i 0
    ); setq
    (while (/= cr "\\")
        (setq cr (substr str2 (setq i (1+ i)) 1))
    ); while
    (setq dmval (substr str2 1 (1- i))
          str3 (substr str2 (+ i 2))
          prefx2 "\\S+"
          len (strlen str3)
          halflen (/ len 2)
          numlen (- halflen 2)
          ul (substr str3 2 numlen)
          ll (substr str3 (+ halflen 2) numlen)
          msg1 (strcat "\nNew dimension text <"
                dmval ">: ")
          msg2 (strcat "\nNew upper limit <"
                ul ">: ")
          msg3 (strcat "\nNew lower limit <"
                ll ">: ")
          newv1 (getstring msg1)
          newv2 (getstring msg2)
          newv3 (getstring msg3)
    ); setq
    (if (= newv1 "") (setq newv1 dmval))
    (if (= newv2 "") (setq newv2 ul))
    (if (= newv3 "") (setq newv3 ll))
    (setq nwstr (strcat prefx1 newv1
                  prefx2 newv2 caret "-" newv3 ";")
    ); setq
); do13-d
(defun do-r12 (ename dname / dimtype dn1 dn2 dn3
          dname)
    (setq dimtype nil dn1 nil dn2 nil dn3 nil)
    (while (/= (dxf 0 dname) "TEXT")
        (setq dname (entnext dname))
    ); while
    (setq dn1 dname)
    (setq dn2 (entnext dn1))
    (if (= (dxf 0 dn2) "TEXT")
        (progn
            (setq dn3 (entnext dn2))
            (setq dimtype "Limits")
        ); progn
        (setq dimtype "Single")
    ); if
    (if (and dn3 (= (dxf 0 dn3) "TEXT"))
        (setq dimtype "Variance")
    ); if
    (cond
        ((= dimtype "Single") (do12-s dn1))
        ((= dimtype "Limits") (do12-l dn1 dn2))
        ((= dimtype "Variance") (do12-v dn1 dn2 dn3))
    ); cond
    (entupd ename)
); do-r12
(defun do12-s (dn1 / j str len cr dmval pmtol
               msg1 msg2 newv1 newv2 newstr)
    (setq j 0)
    (setq str (dxf 1 dn1))
    (setq len (strlen str))
    (repeat len
        (setq cr (substr str (setq j (1+ j)) 1))
        (if (= cr "p")
            (progn
                (setq dimtype "Sym" i j)
            ); progn
        ); if
    ); repeat
    (if (= dimtype "Sym")
        (progn
            (setq dmval (substr str 1 (- i 3))
                  pmtol (substr str (1+ i))
                  msg1 (strcat
                        "\nNew dimension text <"
                         dmval ">: ")
                  msg2 (strcat
                        "\nNew plus-minus tolerance <"
                         pmtol ">: ")
                  newv1 (getstring msg1)
                  newv2 (getstring msg2)
            ); setq
            (if (= newv1 "") (setq newv1 dmval))
            (if (= newv2 "") (setq newv2 pmtol))
            (setq nwstr (strcat newv1 "%%p" newv2))
            (switch dn1 nwstr)
        ); progn
        (entmod (entget dn1))
    ); if
); do12-s
(defun do12-l (dn1 dn2 / ll ul msg1 msg2
               newv1 newv2)
    (setq ll (dxf 1 dn1)
          ul (dxf 1 dn2)
          msg1 (strcat
                "\nNew lower limit <" ll ">: ")
          msg2 (strcat
                "\nNew upper limit <" ul ">: ")
          newv1 (getstring msg1)
          newv2 (getstring msg2)
    ); setq
    (if (= newv1 "") (setq newv1 ll))
    (if (= newv2 "") (setq newv2 ul))
    (switch dn1 newv1)
    (switch dn2 newv2)
); do12-l
(defun do12-v (dn1 dn2 dn3 / dimval ll ul msg1
               msg2 msg3 newv1 newv2 newv3)
    (setq dimval (dxf 1 dn1)
          ll (substr (dxf 1 dn2) 2)
          ul (substr (dxf 1 dn3) 2)
          msg1 (strcat
                "\nNew dimension text <" dimval ">: ")
          msg2 (strcat
                "\nNew lower limit <" ll ">: ")
          msg3 (strcat
                "\nNew upper limit <" ul ">: ")
          newv1 (getstring msg1)
          newv2 (getstring msg2)
          newv3 (getstring msg3)
    ); setq
    (if (= newv1 "") (setq newv1 dimval))
    (if (= newv2 "") (setq newv2 ll))
    (if (= newv3 "") (setq newv3 ul))
    (switch dn1 newv1)
    (switch dn2 (strcat "-" newv2))
    (switch dn3 (strcat "+" newv3))
); do12-v
(defun switch (dname newval)
    (setq olst (entget dname)
          otxt (assoc 1 olst)
          ntxt (cons 1 newval)
          nlst (subst ntxt otxt olst)
    ); setq
    (entmod nlst)
); switch
(defun DIMUTILS ()
    (setting)
    (ch-dimtxt)
    (resetting)
    (princ)
); end of defun DIMUTILS
(defun c:dt () (dimutils))
(prompt "\nCopyright (c) 1996 CAD/CAM Technologies")
(prompt "Enter the command \"DT\" to start.")
(princ)
```