Lucrarea Nr. 11-13
AutoLISP

Interogarea operatorului uman. Scripturi de desenare automata
Realizarea obiectelor grafice pentru un joc in AutoCAD. Memorarea obiectelor intr-o librarie.
Programarea unui joc interactiv in AutoCAD

Obiective

Aceasta lucrare este destinata aplicarii, in diferite situatii practice, a limbajului de programare AutoLISP. Situatiile practice urmaresc repetarea unei parti din desenele facute in laboratoarele precedente, dar de data aceasta studentul este invatat sa lucreze prin intermediul scripturilor de desenare automata. Studentul va invata de asemenea sa memoreze (folosind comanda BLOCK) obiectele grafice astfel realizate. Ulterior, el va putea sa le utilizeze in jocuri si aplicatii folosind comanda INSERT. Ca exemple de jocuri realizate folosind limbajul AutoLISP se dau la finalul lucrarii X&0, Spinzuratoarea si Puzzle. Studentul este incurajat sa extinda libraria de jocuri si este invatat cum sa o faca.

1. Introducere Teoretică

1.1 Aspecte generale

Dezvoltarea practic nelimitată a mediului AutoCAD, prin definirea de noi comenzi a condus la interacţiunea sa cu un alt limbaj de programare AutoLISP-ul, construit pe baza limbajului LISP (List Processing).

1.2 Utilizarea AutoLISP-ului în AutoCAD

AutoCAD-ul este un mediu deschis şi astfel utilizatorul folosind programe scrise în AutoLISP şi Visual Basic for Applications, împreună cu tehnologia ActiveX, poate defini şi redefini comenzi şi acţiuni ale sistemului.

Expresiile AutoLISP pot fi introduse direct în linia de comandă AutoCAD sau pot fi salvate în fişiere şi încărcate ulterior.

Sintaxa de bază a AutoLISP permite ca programul să evalueze liste plasate între paranteze. Pentru fiecare paranteză deschisă trebuie să existe perechea ei, paranteza închisă. Listele pot conţine la rândul lor alte liste.

Parcurgerea acestora este dinspre interior spre exterion, adică prima lista analizată este cea din interior.

Trebuie precizat faptul ca în AutoLISP, mai multe caractere în spaţiu între numele variabilelor, constantelor sau funcţiilor sunt tratate ca un singur spaţiu, aşa cum trecerea pe linia următoare este tratatată ca un caracter spaţiu.

De exemplu:

```
(setq X(*10(+ 12  14) Y (+ 21.7 74.5))
```

şi, respectiv,

```
(setq
X (*10 (+12 14)
Y (+ 21,7 74.5)
```

)

Expresiile AutoLISP în general conţin sintaxa următoare: **(nume [argumente]),** unde **numele** este numele expresiei, iar **argumentele** pot fi constante sau variabile, adică parantezele drepte care indică faptul că argumentele sunt opţionale, putând deci lipsi din expresie.

O variabilă AutoLISP este un nume simbolic atribuit obiectelor: **(set nume_variabila valoare)**.

După ce variabila a fost definită, se poate introduce, la promptul Command: **var_1**, ceea ce are ca efect returnarea valorii variabilei.

Altă modalitate de a atribui valori variabilelor AutoLISP constă în introducerea acestora de la tastatură, de către utilizator, prin functia **get.**

Codul care precedă definiţia promptului are ca efect afişarea acestuia pe o linie nouă, şi nu în continuarea promptului **Command**.

Pentru construirea listelor se pot folosi funcţiile:

(append lista_1 lista_2 ...), care formează o nouă lista, prin concatenarea listelor **lista_1**, **lista_2,** etc.

O nouă listă şi având ca prim element pe **elem** şi continuându-se cu elementele listei **lista** se obţine utilizând funcţia **lista(cons elem lista).**

<u>OBSEVAŢIE:</u>

Există şi funcţii particulare pentru a obtine elementele listei. Aceste funcţii pot fi combinate, până la nivelul patru de imbricare.

Accesarea comenzilor AutoCAD în limbajul AutoLISP se face după sintaxa: **(command argumente)**.

Introducerea argumentului constant **pause** în defiţia comenzii determină întreruperea comenzii pentru a permite introducerea unor date de către utilizator.

De exemplu, **command "circle" _1 pause "" "punct" _2 line_3** în continuare va desena un cerc cu centul în punctul 2 şi care trece prin punctul 3.

După definirea şi încărarea funcţiei prin **(load nume_fişier)**, la promptul **Command:** din AutoCAD se tastează **nume_funcţie**, având ca rezultat interpretarea funcţiei ca orice comandă AutoCAD.

Dacă din defiţia funcţiei lipsesc caracterele **C:** dinaintea numelui funcţiei, apelarea acesteia pe linia de comandă AutoCAD se face tastând numele între paranteze **(nume_functie)**.

Dacă, în definiţia funcţiei **C:nume_functie** este înlocuit prin **S::startup**, funcţia definită va fi încărcată şi executată automat, la iniţializarea oricărui desen. Pentru aceasta, defiţia funcţiei va fi salvată într-unul dintre fişierele acad.lsp sau acaddoc.lsp.

Funcţia definită astfel are numele **F1** şi nu are argumente definite. Aceasta trasează o linie, pornind de la promptul **P1** din punctul de coordonate (x,y,z) un punct **PF** indicat de utilizator, la promptul **Alege pozitia punctului final.**

Dacă defiţia funcţiei de mai sus se salvează în fişierul **Linie.lsp**, în directorul curent şi apoi se încarcă prin **(load „Linie.lsp")**, funcţia **F1** va putea fi apelată prin intermediul comenzii **F1**.

Comanda **F**1 permite utilizatorului alegerea poziţiei punctului final, urmând un grup de prompturi la care utilizarorul nu trebuie să răspundă.

OBSERVAŢIE:

Aşa cum se poate observa, chiar dacă datele de intrare se preiau din funcţia **F1**, AutoCAD-ul va afişa totuşi şi promptul specific comenzii **Line** (ultimele trei linii din dialogul de mai sus). Pentru a inhiba afişarea, pe linia a doua din definiţia funcţiei, după **C:F1** se introduce **(setvar "cmdecho" 0)**. În acest caz, ceea ce se vede pe linia de comanda va fi doar Comanda **F1**, care alege poziţia punctului final.

Pentru ca utulizatorul să evaluaeze o expresie şi, în funcţie de rezultat, să se ia o decizie, se foloseşte structura conditională **if**, în urmatoarea sintaxă:

(if conditie atunci [altfel])

Această instrucţiune funţionează astfel: dacă este îndeplinită **„conditia“**, se execută instructiunea **„atunci“**, dacă nu este indeplinită, se execută instrucţiunea **„altfel“**.

O altă funcţie condiţională AutoLISP este **cond**, care acceptă ca argumente mai multe liste:

cond (conditie_1 atunci_1) (conditie_2 atunci_2) ...)

Instrucţiunea **while** permite repetarea unui bloc de instructiuni, cât timp o condiţie este îndeplinită. Sintaxa este:

(while conditie instrucţiuni)

Prin folosirea caracterului „**;**“, tot ceea ce urmează până la sfârşitul liniei de program este tratat ca un comentariu şi nu se execută.

Tot ceea ce se afla între caracterele „**:/**“ şi „**/:**“ este, de asemenea, interpretat ca un comentariu. Aceste caractere se folosesc pentru a plasa comentarii oriunde în cadrul unei linii sau pe mai multe linii consecutiv.

O entitate desenată este înregistrată în baza de date AutoCAD printr-un nume, pe baza căruia obiectele pot fi apelate în cadrul instrucţiunilor AutoLISP.

Definirea unor mulţimi de selecţie, care cuprind obiecte din desen, pentru a fi manipulate prin instructiunile AutoLISP se obtine prin funcţiile:

a) **(ssadd ent_nume sel_set)**, care adaugă entitatea cu numele ent_nume mulţimii de selecţie;
b) **sel_set.(ssdel ent_nume sel_set)**, care elimină entitatea cu numele ent_nume din mulţimea de selecţie sel_set;
c) **(ssget [mod_sel] [p1][p2] [lista_pcte] [lista filtre]),** care defineşte o mulţime de selecţie prin selectarea obiectelor de către utilizator, metoda de selecţie fiind indicată de argumentul mod_sel (acesta poate fi W, C, etc., similar cu selectarea obiectelor în AutoCAD); argumentele P1 şi P2 definesc colţurile ferestrei de selecţie; argumentul [lista_pcte] defineşte colţurile unei seleţii de tip poligon; argumentul [lista_filtre] defineşte filtrele de selecţie, atunci când argumentul mod_sel are valoarea X;
d) **(sslength sel_set)**, care returnează numărul de entităţi din mulţimea de selecţie sel_set;
e) **(ssmemb ent_numesel_set)**, care verifică dacă entitatea cu numele de ent_nume face parte din acea mulţime de selecţie sel_set;
f) **(ssname sel_set n)**, care returnează numele celei de-a n-a entităţi din mulţimea de selectie sel_set.

Pentru a accesa fişierele şi a prelucra datele obţinute astfel, AutoLISP foloseşte funcţii speciale.

Astfel funcţia **(open)**, are sintaxa: **(open nume-fisier arg)** deschizând fişierul **nume-fisier**. În funcţie de valorile argumentului **arg**, care pot fi **r** *(*read), **w** (write) sau **a** (append), fişierul poate fi citit, scris, respectiv completat.

Fereastra VisualLISP pentru AutoCAD din fig. 1.1 oferă o serie de facilităţi, care simplifică scrierea, editarea şi verificarea programelor AutoLISP.

Fereastra VLISP conţine propriul sistem de meniuri şi ferestre, dar nu funcţionează decât împreună cu AutoCAD-ul.

Codul AutoLISP se introduce în fereastra de editare. În cadrul acestei ferestre poate fi deschis un fişier AutoLISP pentru editare (comanda **Open** din meniul **File**) sau poate fi creat un fişier nou (comanda **New** din meniul **File**).

Pentru a accesa interfaţa **VisualLISP** (**VLISP**), se tasteaza **vlisp** în linia de comandă sau se alege, din meniul **Tools**, AutoLISP şi apoi Visual LISP Editor.

Pentru a facilita urmărirea codului din fereastra de editare **VLISP**, acesta este afişat respectând o anumită semnificaţie a culorilor. Culorile implicite pot fi modificate din meniul **Tools**, prin selectarea opţiunii **Window Attributes**.

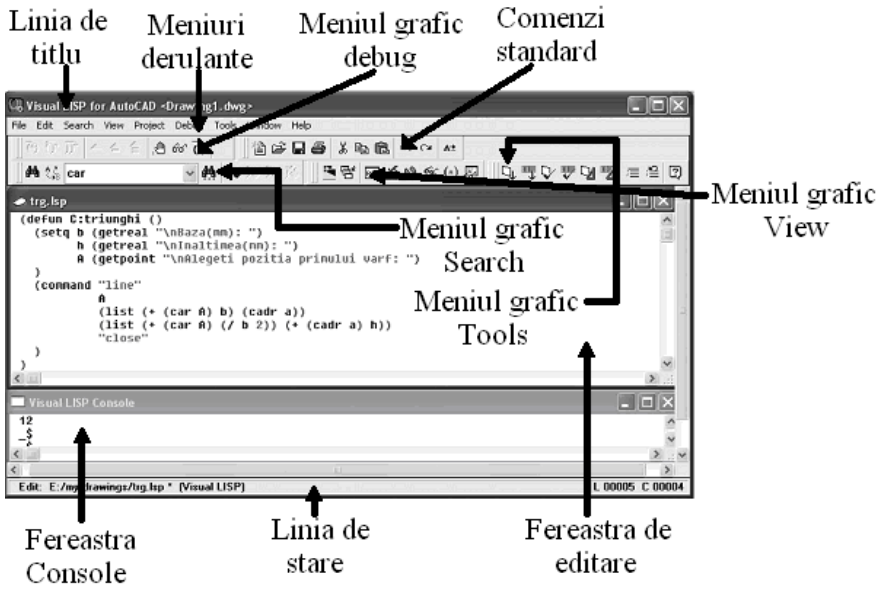


Fig. 1.1 Fereastra VisualLIPS pentru AutoCAD.

Astfel: roşu semnifică parantezele, albastru semnifică funcţii AutoLISP, violet semnifică şiruri, verde semnifică numere, mov pe fond gri semnifică comentarii, iar negru semnifică variabile introduse de utilizator şi elemente nerecunoscute de acesta.

OBSERVAŢII:

- în fereastra VisualLISP Console se pot introduce comenzi AutoLISP, similar cu fereastra AutoCAD, dar, uneori, cu o sintaxă puţin diferită, ceea ce permite evaluarea expresiiilor AutoLISP folosite la scrierea codului;

- linia de stare, situată în partea inferioară a ferestrei VisualLISP for AutoCAD, descrie ceea ce se întâmplă la momentul curent în fereastră.

Sintaxa de definire a unei liste este următoarea:

(list expresie1 expresie2 ...)

Aceasta are ca efect combinarea expresiilor **expresie1**, **expresie2,** etc. într-o singură listă.

Pentru construirea listelor se pot folosi funcţiile: **(append lista_1 lista_2 ...)** care formează o nouă listă, prin concatenarea listelor **lista_1**, **lista_2,** etc.

O altă modalitate este utilizarea **(cons elem lista)** care formează o nouă listă, având ca prim element pe **elem** şi continuând cu elementele listei **lista**.

O funcţie folositoare, aplicabilă unei liste, este **foreach** care, în sintaxa: **(foreach arglista expresie_1 expresie_2 ...)**, atribuie ca valoare argumentului **arg**, succesiv, fiecare element al listei **lista**; apoi execută instrucţiunile **expresie_1 expresie_2** ...

Un alt set de comenzi AutoLISP util în aplicaţiile practice AutoCAD este cel al comenzilor de citire a datelor de intrare de la tastatură:

- **(getangle *şir_caractere*):** pentru citirea unui unghi;
- **(getdist *şir_caractere)*:** pentru citirea valorii unei distanţe;
- **(getint *şir_caractere)*:** realizează citirea unei valori întregi;
- **(getpoint *şir_caractere)*:** pentru citirea coordonatelor unui punct;
- **(getreal *şir_caractere)*:** pentru citirea unei valori reale;
- **(getstring *şir_caractere)*:** realizează citirea unui şir de caractere.

Acestea sunt comenzile care fac legătura între program şi utilizator. La rularea unei astfel de comenzi, programul va afişa pe ecran textul din interiorul aştepta introducerea de către utilizator a tipului de date cerut, inainte de a continua cu rularea celorlalte instrucţiuni ale sale. Un exemplu de utilizare a unei astfel de comenzi, cu rol în citirea de la tastatură a unei valori reale, este:

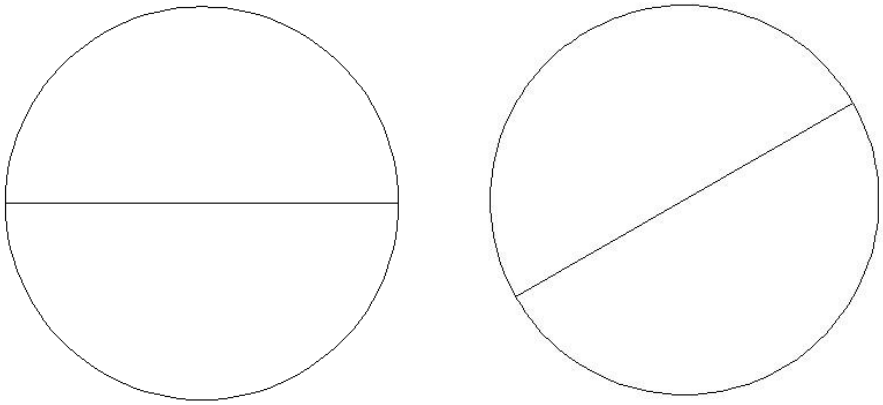
(*SETQ *unghi (getreal "\nIntroduceti valoarea unghiului dorit: "))

Utilizarea caracterelor ***"\n"***, în şirul furnizat ca argument al comenzii **getreal**, realizează trecerea la o nouă linie. Comanda **SETQ** memorează apoi valoarea introdusă de la tastatură de către utilizator în variabila denumită ***unghi***.

O altă funcție utilă în scrierea aplicațiilor este (**ENTLAST)**, ce returnează numele ultimului obiect desenat de pe planșa de lucru. Folosind această funcție putem memora obiectele de interes în variabile dedicate, pe care le putem utiliza ulterior ori de câte ori avem nevoie.

Ca un exemplu de aplicație a celor mai importante comenzi învățate în acest capitol, să desenăm de exemplu un cerc cu centrul în punctul de coordonate (0, 0, 0) și rază egală cu 5. Vom desena apoi o linie egală cu diametrul cercului, al cărei mijloc se va situa în centrul cercului. După desenarea liniei, vom folosi comanda **(ENTLAST)** pentru a memora referința într-o variabilă denumită ***lin*** (Fig. 1.2.a). Ne vom folosi apoi de referința către linie pentru a roti linia cu 30˚ în jurul mijlocului acesteia (Fig. 1.2.b).

```
(COMMAND "_.circle" "_non" '(0. 0. 0.) 5.) ; desenează cercul
(COMMAND "_.line" '(-5. 0. 0.) '(5. 0. 0.) "") ; desenează linia
(SETQ lin (entlast)) ; linia va fi referentiata de variabila „lin”
(COMMAND "_.rotate" lin "" "0,0" 30) ; rotirea liniei cu 30°
```



a) b)

Fig. 1.2. Exemplu grafic de folosire a comenzilor COMMAND, SETQ şi ENTLAST: a. desenarea unui cerc şi a diametrului său; b. rotirea diamentrului cu un unghi de 30˚.

După cum se poate concluziona din partea teoretică a acestui capitol, comenzile AutoLISP pe care le vom folosi cel mai frecvent în aplicațiile AutoCAD sunt **SETQ, COMMAND, ENTLAST,** precum și setul de comenzi de citire a datelor de la tastatură **(getangle, getdist,** etc.**).**

2. Desfasurarea lucrarii

Exercitii introductive

2.1. Desenaţi, folosind doar comenzi AutoLISP, un pătrat, un cerc şi un triunghi echilateral.

2.2. Memoraţi referinţele către obiectele grafice desenate la exerciţiul 2.1. în trei variabile denumite obiect1, obiect2 şi obiect3.

2.3. Folosiţi-vă de referinţele de la exerciţiul anterior pentru a muta obiectele grafice desenate (pătrat, cerc, triunghi) cu 10 unităţi pe axa OX.

2.4. Folosiţi-vă de referinţele de la exerciţiul 2.2 pentru a roti obiectele grafice desenate (pătrat, cerc, triunghi) cu 10 grade în jurul centrului lor de greutate (centrul cercului circumscris).

2.5. Folosiţi-vă de una din comenzile de citire a datelor de la tastatură pentru a lăsa utilizatorul să introducă singur unghiul de rotaţie al obiectelor grafice de la exerciţiul precedent.

Exercitii avansate

Exerciţiul 1

Desenaţi, folosind un script AutoLISP, figura geometrică de la exerciţiul nr. 4 din capitolul 2 (două pătrate înscrise într-un cerc de rază 10). Ca o îmbunătăţire faţă de exerciţiul iniţial, lăsaţi utilizatorul să introducă de la tastatură unghiul de rotaţie dintre cele două patrate înscrise în cerc. De exemplu, pentru desenarea Fig. 3.1, utilizatorul a

ales un unghi de 65˚ între cele două pătrate. Realizaţi şi cotarea lizibilă a razei cercului, precum în Fig. 3.1.

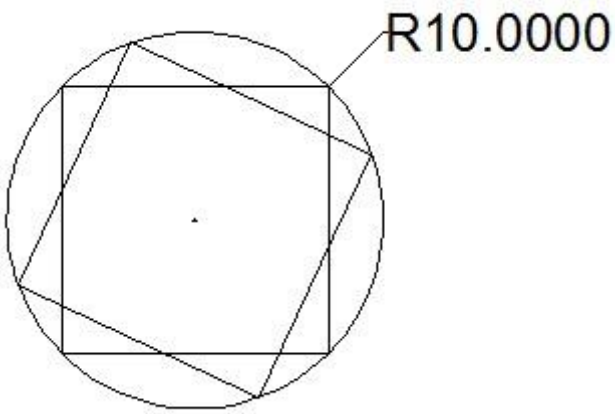


Fig. 3.1. Cerc cu raza egală cu 10 şi două pătrate înscrise în el, realizate folosind AutoLISP.

Indicaţie: Este prezentată în continuare o variantă de script AutoLISP care, odată executat în AutoCAD, va satisface cerinţele exerciţiului:

```
(COMMAND "DIMTXT" "2") ; setează înălţimea textului folosit la dimensiuni
(COMMAND "_.circle" "_non" '(5. 5. 0.) 10.) ; desenează cercul
(COMMAND "_.dimradius" (list (entlast) '(10. 5. 5.)) "_non" '(15. 15. 0.)) ; cotarea cercului
(COMMAND "_.polygon" "4" "5,5" "I" "10") ; desenează primul pătrat
(COMMAND "_.polygon" "4" "5,5" "I" "10") ; desenează al doilea pătrat
(SETQ ent2 (entlast)) ; al doilea pătrat va fi adresat prin variabila ent2
(SETQ unghi (getreal "\nCe unghi doriti intre cele doua patrate? ")) ; aflarea unghiului dorit
(COMMAND "_.rotate" ent2 "" "5,5" unghi) ; rotirea celui de-al doilea pătrat
```

Observaţie: Pentru a putea rula programul de mai sus, acesta trebuie salvat întru-un fiţier cu extensia .lsp (pentru scrierea sa puteţi folosi orice editor simplu de text precum Notepad). Fişierul nou creat poate fi executat în AutoCAD prin simpla deplasare cu mouse-ul (drag'n drop) a acestuia în spaţiul de lucru al AutoCAD-ului.

Exerciţiul 2

Desenaţi, folosind un script AutoLISP, figura geometrică de la exerciţiul nr. 7 din capitolul 2 (un triunghi isoscel cu înălţimea de 15 şi unghiurile de la bază egale cu 45˚). Spre deosebire de acel exerciţiu însă, înlocuiţi linia mijlocie cu o linie oarecare, paralelă cu baza, situată la o distanţă faţă de bază fixată de utilizator, precum în Fig. 3.2. Distanţa faţă de bază va fi introdusă de la tastatură sub forma unui număr zecimal

subunitar *f*, reprezentând raportul dintre această distanţă şi înălţimea triunghiului (de exemplu dacă linia auxiliară se doreşte a fi linia mijlocie, utilizatorul va fixa acest raport la valoarea *f=0,5*). Se vor marca înălţimea triunghiului şi distanţa dintre bază şi linia paralelă cu aceasta.

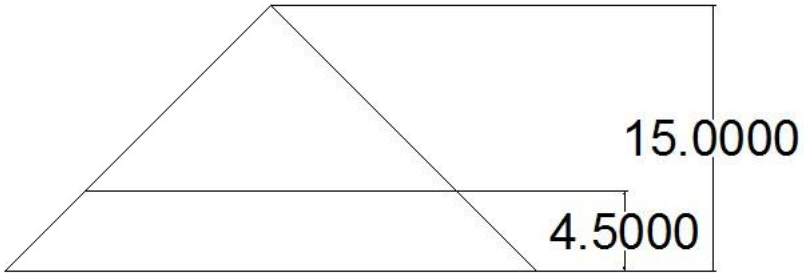


Fig. 3.2. Triunghi isoscel cu unghiurile şi înălţimea cunoscute, în care este desenată şi o linie paralelă cu baza. Distanţa dintre bază şi linia auxiliară s-a ales egală cu 0,3 din înălţime.

Indicaţie: Este prezentată în continuare o variantă de script AutoLISP care, odată executat în AutoCAD, va satisface cerinţele acestui exerciţiu, generând Fig. 3.2:

```
(COMMAND "DIMTXT" "2") ; setează înălţimea textului folosit la dimensiuni
(SETQ pt1 '(0. 0. 0.) pt2 '(0. 15. 0.) pt3 '(25. 0.)) ; marchează câteva puncte de interes
(COMMAND "line" pt1 (polar pt1 0 30) "") ; desenează linia bazei
(SETQ L1 (entlast))
(COMMAND "line" pt1 (polar pt1 PI 30) "")
(SETQ L2 (entlast))
(COMMAND "line" pt2 (polar pt2 (/ (* 7 PI) 4) 30) "") ; desenează latura din dreapta
(SETQ L3 (entlast))
(COMMAND "line" pt2 (polar pt2 (/ (* 5 PI) 4) 30) "") ; desenează latura din stânga
(SETQ L4 (entlast))

(SETQ f (getreal "\nCe distanta doriti intre baza si linia auxiliara? (0.00 - 1.00) "))
(SETQ dist (* f 15.))
(SETQ pt4 (list 0. dist 0.))

; desenează linia auxiliară:
(COMMAND "line" pt4 (polar pt4 0 30) "")
(COMMAND "line" pt4 (polar pt4 PI 30) "")
```

```
; şterge extremităţile în surplus ale liniilor
(COMMAND "trim" L1 "" "C" '(0. -1. 0.) '(25. -10. 0.) "")
(COMMAND "trim" L2 "" "C" '(0. -1. 0.) '(-25. -10. 0.) "")
(COMMAND "trim" L3 "" "C" '(29. 16. 0.) '(31. -1. 0.) "")
(COMMAND "trim" L4 "" "C" '(-29. 16. 0.) '(-31. -1. 0.) "")

(COMMAND "_.dimlinear" pt1 pt2 pt3) ; marchează înălţimea triunghiului
(COMMAND "_.dimlinear" pt1 pt4 '(20. 0.)) ; marchează distanţa dintre bază şi linia auxiliară
```

Exerciţiul 3

Modificaţi scriptul precedent pentru ca utilizatorul să fie capabil să aleagă atât valoarea unghiurilor egale ale bazei, precum şi înălţimea triunghiului isoscel.

Indicatie: Se vor introduce aşadar la începutul programului două comenzi pentru citirea de la tastatură a unghiului şi a înălţimii. Toate comenzile destinate desenării în scriptul prezentat la exerciţiul precedent vor fi influenţate de valorile introduse de utilizator. De exemplu, să considerăm unghiurile de la bază egale cu *x* şi înălţimea triunghiului egală cu *h*. Atunci punctul *pt2* va avea coordonatele *(0, h, 0)*, iar unghiul pe care latura din stânga îl va face în coordonate polare cu direcţia pozitivă a axei Ox va deveni egal cu $\pi+x$ în loc de valoarea fixă de $5\pi/4$ de la exerciţiul precedent. Analog se poate demonstra că unghiul pe care latura din dreapta îl va face cu direcţia pozitivă a axei Ox devine egal cu $\pi+x$ în locul valorii fixe de $7\pi/4$ de la exerciţiul precedent. De asemenea, lungimile laturilor triunghiului trebuiesc alese mai mari dacă unghiurile de la bază sunt mici, altfel acestea nu se vor intersecta.

O altă variantă (mai simplă) de rezolvare a exerciţiului constă în desenarea directă a laturilor triunghiului fără a se mai face apel la linii ajutătoare, linii ale căror extremităţi în surplus le-am şters la finalul exerciţiului precedent. Pentru aceasta ne putem folosi de coordonatele geometrice ale celor trei vârfuri ale triunghiului:

- A(0, h): vârful de sus;
- B(−h·ctg(x), 0): vârful din stânga;
- C(+h·ctg(x), 0): vârful din dreapta.

Punctele de intersecţie ale dreptei auxiliare (paralelă cu baza şi aflată la o distanţă faţă de aceasta de $f \cdot h$) cu laturile triunghiului vor fi:

- D(−h·ctg(x)·(1-f), f·h): vârful din stânga;
- E(+h·ctg(x)·(1-f), f·h): vârful din dreapta.

Exerciţiul 4

Desenaţi, folosind un script AutoLISP, dispozitivul de afişaj de la exerciţiul nr. 9 al capitolului 3. Spre deosebire de exerciţiul iniţial, de data aceasta însă lăsa şi utilizatorul să introducă de la tastatură numărul de pini de pe fiecare parte a

displayului. Pentru realizarea Fig. 3.3, utilizatorul a ales un număr de 6 pini pe fiecare parte a displayului.

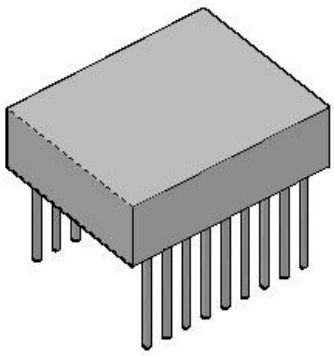


Fig. 3.3. Celulă de afişaj cu 9 pini de control pe fiecare parte a sa.

Indicaţie: Este prezentată în continuare o variantă de script AutoLISP care, odată executat în AutoCAD, va satisface cerinţele acestui exerciţiu şi va putea genera componenta din Fig. 3.3:

```
; memoreaza in variabila p numarul de pini de pe fiecare parte a afisajului:
(SETQ p (getint "\nCati pini de control doriti pe fiecare parte a afisajului? (1 - 10) "))

; curăță ecranul
(COMMAND "ERASE" "ALL" "") ;
(COMMAND "GRID" "OFF")

; desenează corpul celulei de afișaj:
(COMMAND "rectang" '(0. 0. 0.) '(10. 8. 0.))
(SETQ E1 (entlast)) ;
(COMMAND "extrude" E1 "" 3)

; desenează un model de pin de control și memorează rezultatul în blocul PIN:
(COMMAND "circle" '(20. 20. 0) 0.15) ;
(SETQ E2 (entlast))
(COMMAND "line" '(0. -1. 0) '(5. -1. 0) "") ;
(SETQ E3 (entlast))
(COMMAND "sweep" E2 "" E3) ;
(SETQ E4 (entlast))
```

```
(COMMAND "rotate3D" E4 "" "Y" '(0. -1. 0) 90)
(SETQ E4 (entlast))
(COMMAND "Block" "PIN" '(0. -1. 0) E4 "")

(COMMAND "line" '(0. 0.5 0.) '(10. 0.5 0.) "");
(SETQ E5 (entlast))
(COMMAND "line" '(0. 7.5 0.) '(10. 7.5 0.) "");
(SETQ E6 (entlast))
(COMMAND "divide" E5 "B" "PIN" "N" (+ 1 p))
(COMMAND "divide" E6 "B" "PIN" "N" (+ 1 p))

; șterge acum liniile ajutătoare
(COMMAND "ERASE" E5 E6 "") ;

; fixează dimensiunile ecranului pentru a putea vedea întreaga figură rezultată:
(COMMAND "zoom" "a") ;
```

Observaţie: După executarea scriptului de mai sus, obiectul 3D creat va fi iniţial afişat de deasupra. Pentru a putea observa obiectul dintr-un unghi mai favorabil, folosiţi comanda *3dorbit*.

Exerciţiul 5

Realizati o librarie de jocuri in AutoCAD, care sa includa cel putin doua dintre urmatoarele jocuri: X si 0, Spinzuratoarea, Puzzle.

Indicatie: Librăria care face obiectul acestui exercitiu este gândită să pornească inițial cu 4 jocuri, dar cu posibilitatea de extindere facilă a numărului de aplicații pe care le poate suporta. Alegerea jocului ar trebui facuta dintr-un meniu intuitiv în care sunt prezentate grafic toate jocurile din librărie.

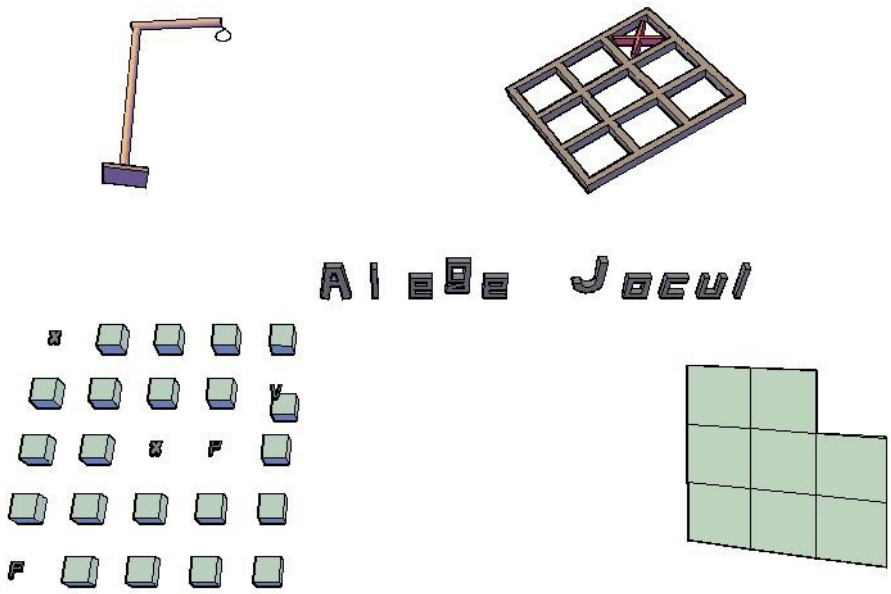


Figura 3.4. Meniul principal al librăriei

Codul din spatele acestui meniu este unul simplu, iar studenții sunt încurajați să-l extindă ținând cont de jocurile create de ei ulterior.

Primul pas pe care un student ar trebui să-l facă este bineînțeles realizarea jocului său propriu. Acest pas va fi detaliat în secțiunea următoare. Imediat după realizarea jocului, el va trece codul acestuia într-o funcție separată AutoLISP.

Va trebui apoi să extindă meniul grafic care conține câte un desen semnificativ pentru fiecare joc prin adăugarea unei iconițe semnificative pentru noul său joc. În ultima etapă a pregătirii meniului, va trebui să modifice coordonatele geometrice care definesc desenele anterioare și să adauge coordonatele noii iconițe într-un bloc condițional de decizie.

De exemplu, pentru cazul librăriei inițiale, ecranul este împarțit în 4 regiuni, diviziunea făcându-se la linia orizontală $y = -60$ și la linia verticală $x = +150$. La adăugarea unui nou obiect această împărțire trebuie refăcută. Studentul va fi încurajat să gândească singur această reîmpărțire.

Studenții vor învăța atât despre desenarea obiectelor grafice precum și despre comenzile AutoLISP-ului prin înțelegerea modului de realizare a celor 4 jocuri existente în libraria inițială.

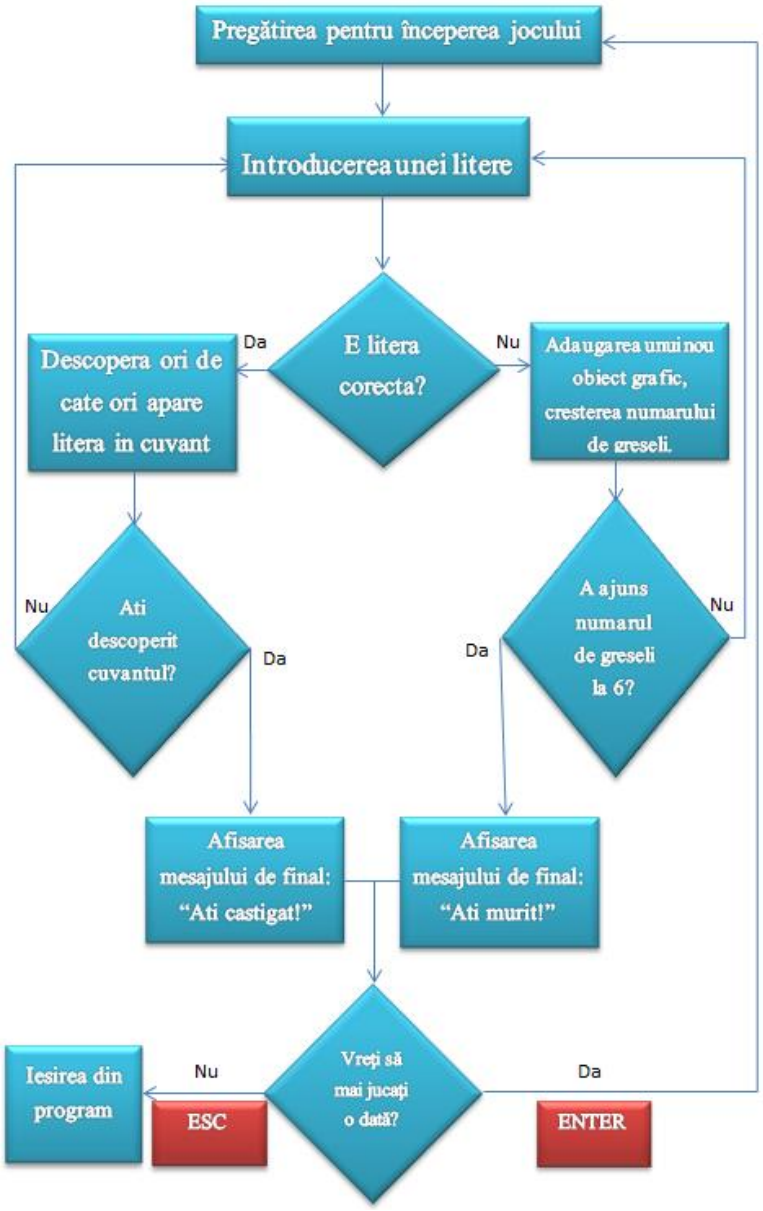


Figura 3.5. Organigrama jocului „Spânzurătoarea”

Jocul „Spânzurătoarea”

Primul pas în realizarea acestui joc este desenarea obiectelor grafice, cărora trebuie sa le alocăm și variabile în program. Pentru cazul de față obiectele grafice sunt următoarele: capul, trunchiul, mâna stângă, mâna dreaptă, piciorul stâng, piciorul drept, stâlpul spânzurătorii, suporţii pe care stau literele, mesajele de sfarșit “Ai murit” și ”Ai câştigat”. Pentru desenarea de exemplu a mâinii am folosit doar comenzi AutoCAD: SPLINE, CIRCLE, ROTATE3D, SWEEP.

Al doilea pas este desenarea organigramei generale a programului, pe care o aveți în Fig. 3.5. Se pot observa pe organigramă implementarea câtorva reguli de bază ale programului: de

exemplu numărul maxim de greșeli este egal cu numărul părților corpului pe care le-am desenat anterior, adică 6. Ceea ce nu se poate observa însă, este prezența și altor reguli:

- ❖ Dacă utilizatorul reintroduce o literă pe care a mai utilizat-o, atunci litera nu va mai fi luată în considerare;
- ❖ Se pot folosi doar litere din alfabetul englezesc;
- ❖ Nu exista remiză;
- ❖ Obiectele grafice (părțile corpului) sunt introduse într-o ordine prestabilită, în funcție de numărul greșelilor de la momentul respectiv;
- ❖ Calculatorul alege cuvântul în funcție de sutimile de secundă de la ceasul calculatorului, dintr-o listă prestabilită de programator.
- ❖ Literele greșite sunt trecute în partea de jos a ecranului, precum în Fig. 3.6

Mai jos, în Fig. 3.6, se poate observa o captură de ecran din timpul unei partide de „Spânzurătoarea” în AutoCAD.

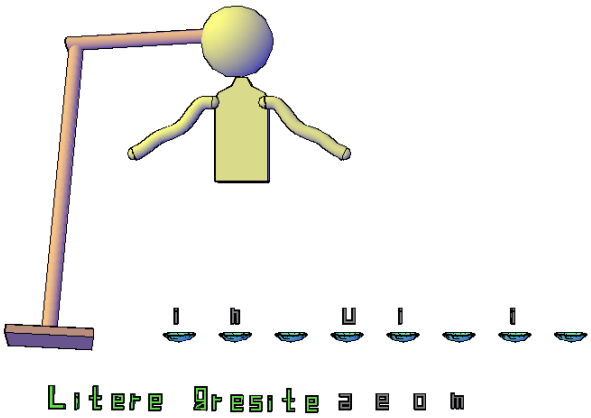


Figura 3.6. Captura de ecran din timpul jocului „Spânzurătoarea”

Jocul „Puzzle”

Obiectele grafice în realizarea acestui joc sunt reprezentate de piesele de puzzle, dar acestea nu sunt desenate in AutoCAD, ci decupate prin intermediul AutoLISP dintr-o imagine .jpg.

Figura 3.7. Organigrama jocului „Puzzle”

Regulile de bază descrise de organigramă (Fig. 3.7) sunt simple: piesele sunt amestecate aleatoriu și ele pot fi mutate cu ajutorul mouse-ului astfel încât să se obțină imaginea inițială.

Ceea ce nu se poate observa însă în organigramă este prezența și altor reguli:

- Piesele fiind amestecate aleatoriu, există șansa să cădem peste o combinație fără rezolvare;
- Rezultatul final al puzzle-ului este mereu același: reconstrucția imaginii inițiale;
- Dacă utilizatorul selectează o piesă de puzzle vecină cu piesa lipsă, atunci va avea loc o permutare între cele două locații (Fig. 3.8).
- Dacă utilizatorul selectează o piesă de puzzle care este înconjurată în toate cele 4 direcții de alte piese, atunci calculatorul va ignora alegerea;

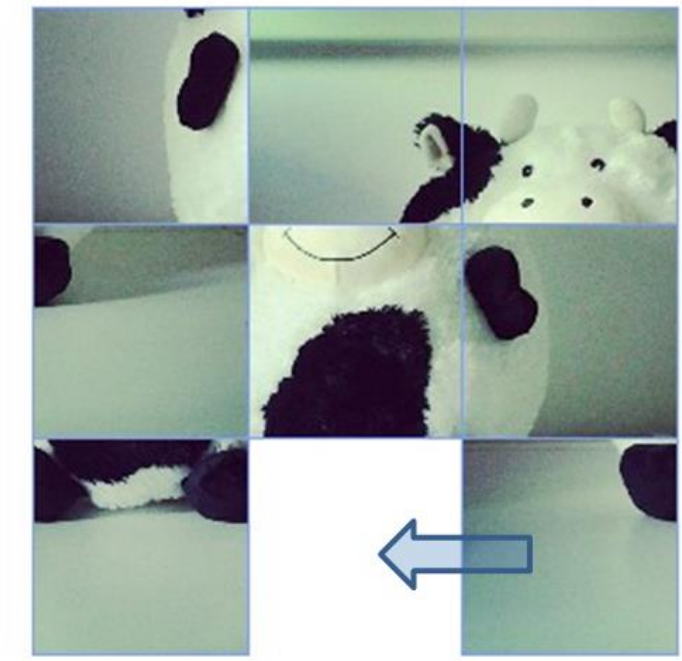


Figura 3.8. Captură din timpul jocului „Puzzle”

Jocul de memorie

Obiectele grafice necesare la realizarea acestui joc sunt cuburile și obiectele din spatele lor. In spatele celor 25 de cuburi vor fi asezate 12 perechi de obiecte grafice diferite (dintr-o librărie mai mare de obiecte). In spatele celui de-al 25-lea cub se va afla un obiect interzis (joker). Organigrama acestui joc este afișată în Fig. 3.9 iar în Fig. 3.10 apare o captură de ecran din timpul unei partide.

Regulile de bază implementate prin intermediul organigramei sunt: utilizatorul este lăsat să memoreze locul obiectelor preț de câteva secunde, după care are loc alegerea unei perechi de blocuri. Dacă acestea coincid, atunci vor rămâne descoperite. In caz contrar, numărul de greșeli este incrementat. Procedura se continuă până când numărul de greșeli ajunge la 12 . Dacă din greșeală utilizatorul decoperă jokerul, partida se încheie cu înfrângerea utilizatorului.

Obiectele din librăria inițială constau în litere ale alfabetului de la A la Z. Jocul poate fi îmbunătățit prin înlocuirea acestor litere cu obiecte mai interesante, iar studentul va fi încurajat să deseneze singur aceste obiecte.

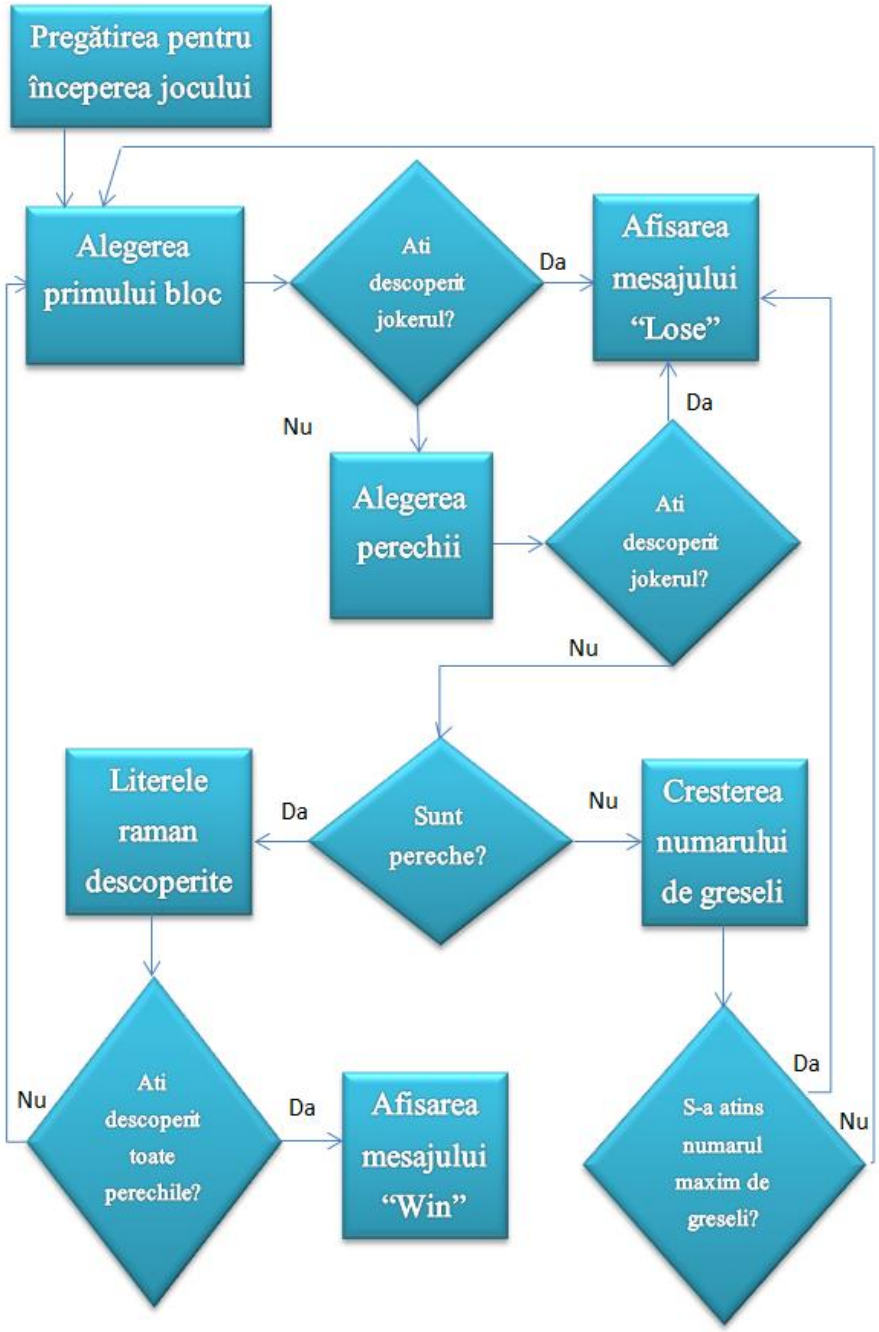


Figura 3.9. Organigrama jocului de memorie

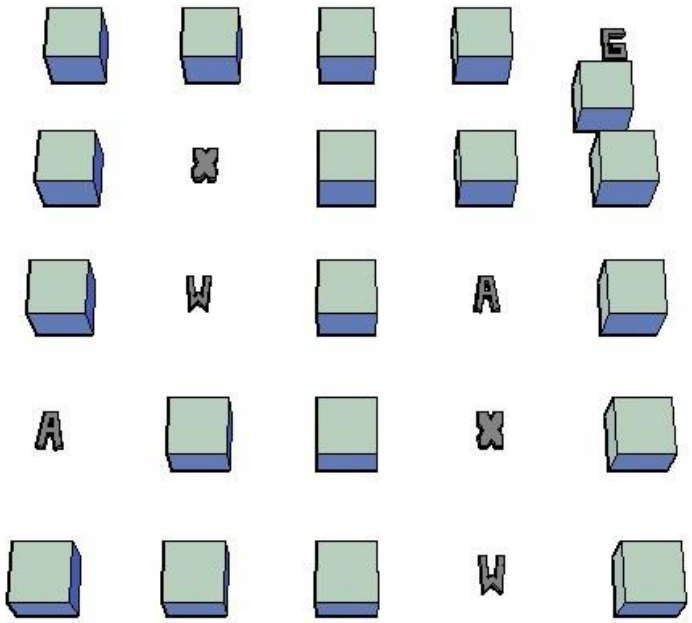


Figura 3.10. Captură din timpul jocului de memorie

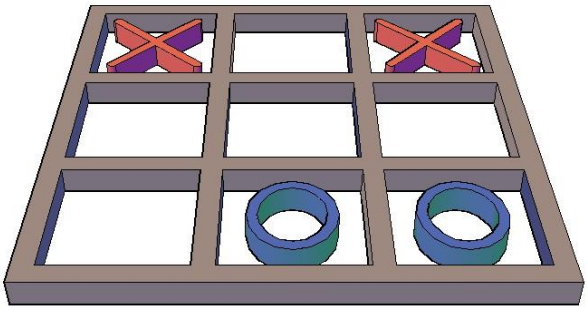


Figura 3.11. Captură din timpul jocului „X și O”

Jocul „X și 0”

Obiectele grafice în realizarea acestui joc sunt: litera „X” si cifra „0”, tabla de joc (Fig. 3.11). Regulile de bază descrise de organigramă (Fig. 3.12) sunt cunoscute. Prima mutare este cea a calculatorului. Se poate muta doar în pătrăţelele libere. Tabla este alcătuită inițial din 9 pătrăţele goale, aliniate 3x3. Scopul jocului este acela de a completa 3 simboluri identice pe o linie, coloană sau diagonală. În cazul în care tabla este plină însa niciunul dintre jucători nu a reuşit să castige, atunci jocul se termină cu remiză. Regulile care nu sunt evidențiate în organigramă: dacă utilizatorul dă click în afara tablei sau pe o pătrățică ocupată, mutarea se repetă.

Realizarea de aplicații în AutoLISP nu este un domeniu mediatizat. Site-urile care descriu interacțiunea AutoCAD-ului cu AutoLISP-ul sunt în număr redus, drept pentru care studenții nu sunt foarte interesați de aceast domeniu. Cu toate acestea, un mod inedit de a le atrage atenția este realizarea la laborator a unei librării de jocuri interactive.

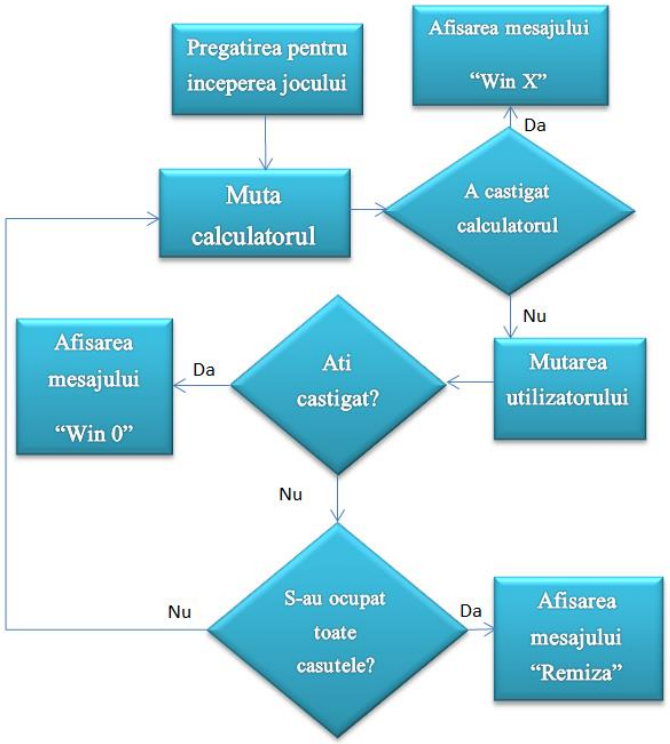


Figura 3.12. Organigrama jocului „X și O"

Librăria inițială de patru jocuri poate fi ușor înțeleasă și extinsă de către studenți. Extinderea poate fi făcută prin adăugarea de noi aplicații, sau prin îmbunătățirea celor patru jocuri deja existente în librărie. Realizarea și funcționarea acestor jocuri au fost succint explicate în lucrarea de față.

Codul integral al librariei de jocuri (fisierul game.lsp) precum si obiectele grafice folosite pentru rularea sa se gaseste la adresa:

http://upit.eu5.org/Laboratoare/gamesLibrary.rar

Pentru rularea librariei, este nevoie de:

- dezarhivarea fisierelor din arhiva de mai sus (.dwg si .lsp)

- incarcarea prealabila a fisierului cu extensia .dwg in AutoCAD (prin drag-n-drop)

- incarcarea fisierului cu extensia .lsp in AutoCAD (prin dran-n-drop)

- rularea librariei folosind in AutoCAD comanda "game" care activeaza meniul