Ludus Latrunculorum

Heather Anne Laudan

Follow this and additional works at: http://opensiuc.lib.siu.edu/uhp_theses

Recommended Citation
program thesis;
{&1 graph.p}
label quit;

const allstars = '************************************************************';

type
mwtype = (man, woman);

piecetype = record
  morw : mwtype;
  role : (main, chorus);
  cancel : boolean;
end;

square = record
  full : boolean;
  piece : piecetype;
end;

letterset = set of 'A'..'z';
godtype = (ares, aphrodite, notthere);
boardtype = array [1..8, 1..8] of square;
type callarray = array [1..2] of integer;

var
board : boardtype;
side, action, newgame, newboard : char;
callcount : callarray;
lastnum, oldrow, oldcol, newrow, newcol, numplayers : integer;
godcalled : godtype;
oneleft, change : boolean;
oldsquare, newsquare : square;
cutloop1, outloop2, moveset : letterset;

function keyin : char;
  var c: char;
begin
  repeat until keypressed;
  read (kbd, c);
  keyin := c;
end;
procedure play (note, time : integer);
begin
  sound (note);
  delay (time);
  nosound;
  delay (15);
end;

procedure rest;
begin
  delay (200);
end;

procedure music;

const
  n8 = 400;
  e1 = 125;
  f5 = 140;
  g = 147;
  a = 159;
  b = 170;
  cs = 180;
  d = 185;
  e2 = 193;

var n4, n3 : integer;
more : char;

begin
  textmode(c80);
  n4 := 2 * n8;
  n3 := 3 * n8;
  textbackground (black);
  textcolor (magenta);
  play (a, n8);
  play (e2, n4);
  play (e2, n3);
  play (cs, n8);
  play (d, n8);
  play (e2, n8);
  play (d, n3);
  rest;
  play (cs, n4);
  play (d, n8);
  play (e2, n8);
  play (g, n8);
  play (e2, n3);
  rest;
  play (a, n4);
  play (b, n8);
  play (g, n4);
  rest;
  play (a, n8);
  play (cs, n8);
  play (e2, n8);
  play (d, n8);
  play (cs, n8);
  play (d, n8);
  play (cs, n8);
  play (a, n4);
  play (b, n8);
  play (g, n4);
end;
procedure smallsound;

const
nC = 400;
e1 = 125;
fS = 140;
g = 147;
a = 159;
b = 170;
cS = 180;
d = 185;
e2 = 193;

var n4, n3 : integer;
more : char;

begin
n4 := 2 * n8;
n3 := 3 * n8;
play (a, n8);
play (cs, n8);
play (b, n8);
play (d, n8);
play (e2, n8);
play (cs, n8);
play (a, n8);
play (a, n4);
play (a, n8);
play (fs, n8);
play (e1, n3);
rest;
end;

procedure writestart;

var cont : char;
skipset : letterset;

begin
textmode (c80);
textbackground (black);
textcolor (magenta);
skipset := ['s', 'S'];
cirscr;
writeln;
for lastnum := 1 to 10 do
write (allstars);
writeln ('Ludus Latrunculorum');
by Heather Laudan');

writeln ('for lastnum := 1 to 10 do
write (allstars);
This is a game loosely based on an ancient game played by the Greeks and Romans. The Romans called the game Ludus Latrunculorum, or robbers. Remnants of game pieces found in tombs and ruins of ancient buildings, pictures depicting play, and references in poetry have given us the few details we have about the game. The board is a square grid of square blocks upon which two opposing teams match forces. Pieces are captured and removed from play when they are surrounded on two opposite sides by two pieces from the other side. It is debated whether there was just one type of move for all pieces or possibly two types of pieces on each side, with different allowable moves for the different pieces.

This ancient game are as yet undiscovered, but have been liberally interpreted with the help of the Greek comic playwright, Aristophanes, and his play entitled Lysistrata. The Lysistrata is the basis for this strategic battle, with the opposing sides not soldiers of opposing city-states fighting for territory or glory, but opposing sexes: men versus women in a battle for international peace as well as domestic tranquility. The women of warring city-states, tired of the toils the war is taking on their country and their private lives, unite against all their husband-soldiers and stage a sex strike until peace is declared.

In this game the game board or the screen represents the scene of most of the action of the Lysistrata, the Parthenon. The opposing sides are the women led by the militant matron Lysistrata vs. the men. There are two types of pieces for each side, 12 of each type. The shaded pieces have the most freedom and power in their movement as they are allowed to move in an L-shape in any direction. These pieces represent the main characters of the comedy, the young soldiers and their wives. The outlined pieces represent the constantly bickering old men and women of the chorus. The chorus pieces are allowed to move only one square at a time, either forward, backward, left, or right. The object of the game is to eliminate all pieces of the opposing team. Each side is helped toward its object by a patron god.
Whenever a god-call occurs, Ares and Aphrodite (who are ‘airly’)

Alternately, move one piece in each turn. 

More than one piece can be eliminated at once.

by surrounding a row (or column) of contiguous pieces at

It may be conducted between two players, or one player.

against the computer. (not yet) The players choose which sides they

want, and an initial god-call determines who goes first.

Pieces are removed when they are surrounded on two opposite

sides by 2 pieces from the opposing side. A piece in a corner is

surrounded if three opposing pieces are all around it. Teams.

alternate turns moving one piece in each turn.

by surrounding a row (or column) of contiguous pieces at.

its two ends.

Play may be conducted between two players, or one player.

against the computer. (not yet) The players choose which sides they

want, and an initial god-call determines who goes first.

Pieces are removed when they are surrounded on two opposite.

sides by 2 pieces from the opposing side. A piece in a corner is

surrounded if three opposing pieces are all around it. Teams.

alternate turns moving one piece in each turn.

by surrounding a row (or column) of contiguous pieces at.

its two ends.

Play may be conducted between two players, or one player.

against the computer. (not yet) The players choose which sides they

want, and an initial god-call determines who goes first.

 Pieces are removed when they are surrounded on two opposite.

sides by 2 pieces from the opposing side. A piece in a corner is

surrounded if three opposing pieces are all around it. Teams.

alternate turns moving one piece in each turn.

by surrounding a row (or column) of contiguous pieces at.

its two ends.

Play may be conducted between two players, or one player.

against the computer. (not yet) The players choose which sides they

want, and an initial god-call determines who goes first.

Pieces are removed when they are surrounded on two opposite.

sides by 2 pieces from the opposing side. A piece in a corner is

surrounded if three opposing pieces are all around it. Teams.

alternate turns moving one piece in each turn.

by surrounding a row (or column) of contiguous pieces at.

its two ends.

Play may be conducted between two players, or one player.

against the computer. (not yet) The players choose which sides they

want, and an initial god-call determines who goes first.

Pieces are removed when they are surrounded on two opposite.

sides by 2 pieces from the opposing side. A piece in a corner is

surrounded if three opposing pieces are all around it. Teams.

alternate turns moving one piece in each turn.

by surrounding a row (or column) of contiguous pieces at.

its two ends.

Play may be conducted between two players, or one player.

against the computer. (not yet) The players choose which sides they

want, and an initial god-call determines who goes first.

Pieces are removed when they are surrounded on two opposite.

sides by 2 pieces from the opposing side. A piece in a corner is

surrounded if three opposing pieces are all around it. Teams.

alternate turns moving one piece in each turn.

by surrounding a row (or column) of contiguous pieces at.

its two ends.

Play may be conducted between two players, or one player.

against the computer. (not yet) The players choose which sides they

want, and an initial god-call determines who goes first.

Pieces are removed when they are surrounded on two opposite.

sides by 2 pieces from the opposing side. A piece in a corner is

surrounded if three opposing pieces are all around it. Teams.

alternate turns moving one piece in each turn.

by surrounding a row (or column) of contiguous pieces at.

its two ends.

Play may be conducted between two players, or one player.

against the computer. (not yet) The players choose which sides they

want, and an initial god-call determines who goes first.

Pieces are removed when they are surrounded on two opposite.

sides by 2 pieces from the opposing side. A piece in a corner is

surrounded if three opposing pieces are all around it. Teams.

alternate turns moving one piece in each turn.

by surrounding a row (or column) of contiguous pieces at.

its two ends.

Play may be conducted between two players, or one player.

against the computer. (not yet) The players choose which sides they

want, and an initial god-call determines who goes first.

Pieces are removed when they are surrounded on two opposite.

sides by 2 pieces from the opposing side. A piece in a corner is

surrounded if three opposing pieces are all around it. Teams.

alternate turns moving one piece in each turn.

by surrounding a row (or column) of contiguous pieces at.

its two ends.

Play may be conducted between two players, or one player.

against the computer. (not yet) The players choose which sides they

want, and an initial god-call determines who goes first.

Pieces are removed when they are surrounded on two opposite.

sides by 2 pieces from the opposing side. A piece in a corner is

surrounded if three opposing pieces are all around it. Teams.

alternate turns moving one piece in each turn.

by surrounding a row (or column) of contiguous pieces at.

its two ends.

Play may be conducted between two players, or one player.

against the computer. (not yet) The players choose which sides they

want, and an initial god-call determines who goes first.

Pieces are removed when they are surrounded on two opposite.

sides by 2 pieces from the opposing side. A piece in a corner is

surrounded if three opposing pieces are all around it. Teams.

alternate turns moving one piece in each turn.

by surrounding a row (or column) of contiguous pieces at.

its two ends.

Play may be conducted between two players, or one player.

against the computer. (not yet) The players choose which sides they

want, and an initial god-call determines who goes first.

Pieces are removed when they are surrounded on two opposite.

sides by 2 pieces from the opposing side. A piece in a corner is

surrounded if three opposing pieces are all around it. Teams.

alternate turns moving one piece in each turn.

by surrounding a row (or column) of contiguous pieces at.

its two ends.

Play may be conducted between two players, or one player.

against the computer. (not yet) The players choose which sides they

want, and an initial god-call determines who goes first.

Pieces are removed when they are surrounded on two opposite.

sides by 2 pieces from the opposing side. A piece in a corner is

surrounded if three opposing pieces are all around it. Teams.

alternate turns moving one piece in each turn.

by surrounding a row (or column) of contiguous pieces at.

its two ends.

Play may be conducted between two players, or one player.

against the computer. (not yet) The players choose which sides they

want, and an initial god-call determines who goes first.
writeln (' Welcome to the game of Ludus Latrunculorum '); writeln ('**** Battle of Lysistrata **** '); writeln ;
(* writeln (' How many players? (enter 1 or 2) '); *) (* read (numplayers); *) numplayers := 2 ;
repeat
  if numplayers = 1 then oneplayer (side1)
  else if numplayers = 2 then twoplayers (side1)
  else writeln (' wrong entry - enter 1 or 2 for number of players ');
  until (numplayers = 1) or (numplayers = 2)
end ;

procedure setboard ( var board : boardtype );
var
  row, col : integer ;
begin
  col := 1 ;
  for row := 1 to 3 do
    begin
      while col <= 8 do
        begin
          board[row,col].full := true ;
          board[row,col].piece.morw := woman ;
          board[row,col].piece.role := main ;
          board[row,col].piece.cancall := false ;
          col := col + 2 ;
          end ;
        if col = 9
          then col := 2
        else col := 1 ;
        end ;
  col := 2 ;
  for row := 1 to 3 do
    begin
      while col <= 8 do
        begin
          board[row,col].full := true ;
          board[row,col].piece.cancall := true ;
          board[row,col].piece.morw := woman ;
          board[row,col].piece.role := chorus ;
          col := col + 2 ;
          end ;
        if col = 9
          then col := 2
        else col := 1 ;
        end ;
end ;
begin
for row := 6 to 8 do
  begin
    col := 1;
    or row := 6 to 8 do
      begin
        while col <= 8 do
          begin
            board[row, col].full := false;
            col := 1;
            or row := 6 to 8 do
              begin
                while col <= 8 do
                  begin
                    board[row, col].full := true;
                    board[row, col].piece.cancel := true;
                    board[row, col].piece.morw := man;
                    board[row, col].piece.role := chorus;
                    col := col + 2;
                  end;
                if col = 9
                  then col := 2
                else col := 1;
              end;
            end;
          end;
        end;
      end;
    end;

  function checkfull (newrow, newcol : integer ;
    board : boardtype ) : boolean ;
  begin
    board[newrow,newcol].full := true
    then checkfull := true
    else checkfull := false
  end ;

  function checkmove (board : boardtype ;
    oldrow, oldcol, newrow, newcol : integer ;
    side : char ) : boolean ;
  var
    wrongs, no_1, noempty, nofull, noblock, m1, m2, m3, m4, m5, m6 : string[80] ;
    rowab : integer ;
    colab : integer ;
  begin
    textmode ;
    wrongs := \You are trying to move the other side's piece - try again\;
    no_1 := \Not moving in an L shape - try again.\;
    noempty := \Can not move from an empty space - try again.\;
    nofull := \Can not move into already occupied space - try again.\;
    noblock := \Can only move this piece one square at a time. try again!\;
    end ;
"Enough of that insolent lip!"
"Gross institute. A sorry day for the force..."
"Such nanky-panky we have to thank for today's utter anarchy!"
"What-all's that bodacious ruckus?"
"Who-all's notion was this-hyer confabulation?"
\[\text{rowab} := \text{abs(oldrow - newrow);}
\text{colab} := \text{abs(oldcol - newcol);}
\text{checkmove} := \text{false;}
\text{if checkfull (oldrow, oldcol, board) = false then}
\text{begin}
\text{writeln (m1);}
\text{writeln (noempty)}
\text{end}
\text{else if checkfull (newrow, newcol, board) = true then}
\text{begin}
\text{writeln (m2);}
\text{writeln (nofull)}
\text{end}
\text{else if (board[oldrow,oldcol].piece.morw = man) and (side = 'w') then}
\text{begin}
\text{writeln (m3);}
\text{writeln (wrongs)}
\text{end}
\text{else if (board[oldrow,oldcol].piece.morw = woman) and (side = 'm') then}
\text{begin}
\text{writeln (m4);}
\text{writeln (wrongs)}
\text{end}
\text{else if board[oldrow,oldcol].piece.role = main}
\text{then if ((rowab = 2) and (colab = 1)) or ((rowab = 1) and (colab = 2))}
\text{then checkmove := true}
\text{else}
\text{begin}
\text{writeln (m5);}
\text{writeln (no_1)}
\text{end}
\text{else if ((rowab = 1) and (colab = 0)) or ((rowab = 0) and (colab = 1))}
\text{then checkmove := true}
\text{else}
\text{begin}
\text{writeln (m6);}
\text{writeln (noblock)}
\text{end}
\text{ writeln menwon (board : boardtype) : boolean ;}
\text{var}
\text{col, row : integer ;}
\text{temp : boolean ;}
\text{begin}
\text{row := 1 ;}
\text{temp := true ;}
\text{while (row <= 8) and (temp = true) do}
\text{begin}
\text{col := 1 ;}
\text{while (col <= 8) and (temp = true) do}
\text{begin}
\text{if board[row,col].full = true}
\text{then if board[row,col].piece.morw = woman}
\text{then temp := false ;
\text{else if board[row,col].piece.morw = man}
\text{then temp := true ;}
\text{else temp := false ;}
\text{end}
function womenwon (board : boardtype) : boolean;

col, row : integer;
temp : boolean;

begin
  col := 1;
temp := true;
  while (col <= 8) and (temp = true) do
    begin
      row := 1;
      while (row <= 8) and (temp = true) do
        begin
          if board[row, col].full = true
            then if board[row, col].piece.morw = man
                then temp := false;
          row := row + 1
        end;
      row := row + 1
    end;
  womenwon := temp
end;

procedure godcall (caller : char;
                  var godcalled : godtype);
var
callnum : integer;

begin
  fillscreen (-1);
  smallisound;
  randomize;
  callnum := random (5);
  case callnum of
    0, 2 : godcalled := ares;
    1, 3 : godcalled := aphrodite;
    4 : godcalled := nuthere;
  end;
  and:

procedure movepiece (var board : boardtype;
                    oldrow, oldcol, newrow, newcol : integer);

begin
  board[newrow, newcol].full := true;
  board[newrow, newcol].piece.morw := board[oldrow, oldcol].piece.morw;
  board[newrow, newcol].piece.role := board[oldrow, oldcol].piece.role;
  board[newrow, newcol].piece.cancall := board[oldrow, oldcol].piece.cancall;
  board[oldrow, oldcol].full := false
end;

procedure morecalls (newcol : integer;
                    var newsquare : square;
                    var callcount : callarray);

begin
  if newcol = 1
    then
if newsquare.piece.morw = man
    then if newsquare.piece.cancall = true
    begin
        newsquare.piece.cancall := false;
        writeln ('The men have another god-call');
        writeln ('for a total of', callcount[2]:1);
    end
    else if newcol = 8
    then if newsquare.piece.morw = woman
        then if newsquare.piece.cancall = true
        then
            begin
                newsquare.piece.cancall := false;
                writeln ('The women have another god-');
                writeln ('call for a total of', callcount[1]:1);
            end
    procedure downcount (var callcount : callarray;
        caller : char;
        var noneleft : boolean);
    const nocalls = 'you have no god-calls.';
    begin
        noneleft := false;
        case caller of
        'm' : if callcount[2] = 0
            then
                begin
                    noneleft := true;
                    writeln (nocalls)
                end
        'w' : if callcount[1] = 0
            then
                begin
                    noneleft := true;
                    writeln (nocalls)
                end
        end;
        if noneleft = false
        then
            begin
                writeln ('Remaining god-calls: ');
                writeln ('Women - ',callcount[1]:1,' Men - ',callcount[2]:1)
            end;
    end;
procedure takepieces (var board : boardtype;
    newrow, newcol : integer;
    var change : boolean);
    var 1, row : integer;
    newmorw : mwtype;
begin
    change := false;
    newmorw := board[newrow,newcol].piece.morw;

while (board[i,newcol].full = true) and (board[i,newcol].piece.morw = newmorw) and (i < newrow - 1)) do
  i := i - 1;
  if (board[i,newcol].full = true) and ((board[i,newcol].piece.morw = newmorw) and (i < newrow - 1)) then
    begin
      change := true;
      for row := (i + 1) to (newrow - 1) do
        board[row,newcol].full := false;
    end;
  i := newrow + 1;
while (board[i,newcol].full = true) and ((board[i,newcol].piece.morw <> newmorw) and (i < 8)) do
  i := i + 1;
if (board[i,newcol].full = true) and ((board[i,newcol].piece.morw = newmorw) and (i < newrow + 1)) then
  begin
    change := true;
    for row := (newrow + 1) to (i - 1) do
      board[row,newcol].full := false;
  end;
  i := newcol - 1;
while (board[newrow,i].full = true) and ((board[newrow,i].piece.morw <> newmorw) and (i > 1)) do
  i := i - 1;
if (board[newrow,i].full = true) and ((board[newrow,i].piece.morw = newmorw) and (i < newcol - 1)) then
  begin
    change := true;
    for row := (i + 1) to (newcol - 1) do
      board[newrow,row].full := false;
  end;
  i := newcol + 1;
while (board[newrow,i].full = true) and ((board[newrow,i].piece.morw <> newmorw) and (i < 8)) do
  i := i + 1;
if (board[newrow,i].full = true) and ((board[newrow,i].piece.morw = newmorw) and (i > newcol + 1)) then
  begin
    change := true;
    for row := (newcol + 1) to (i - 1) do
      board[newrow,row].full := false;
  end;
end;

procedure drawboard (board : boardtype); (* dummy *)

const alllines = ' ----------------------------------------------------------' 
                   ' -------';

var
  row, col : integer;
  bsquare : square;

begin
  cirstr ;
  writeln (' column '); 
  writeln (alllines);
  for row := 1 to 8 do
    begin
      for col := 1 to 8 do

begin
  if col = 1
    then write (row:1, ' | ! | '); 
  bsquare := board[row,col]; 
  write (' '); 
  if bsquare.full = true 
    then begin
      if bsquare.piece.morw = man 
        then write ('m-')
        else write ('w-')
      if bsquare.piece.role = main 
        then write ('m :')
        else write ('c :')
    end
    else write (' '); 
  writeln (' : ');
  writeln (alllines)
end;

procedure drawwc ;
var x1,x2,y1,y2,angle,color,radius : integer ;
begin
  x1 := 3 ;
  y1 := 6 ;
  radius := 3 ;
  color := 3 ;
  angle := 180 ;
  arc (x1,y1,angle, radius, color);
  x1 := 9 ;
  arc (x1, y1, angle, radius, color);
  x1 := 2 ;
  x2 := 8 ;
  y2 := 14 ;
  draw (x1, y1, x2, y2, color);
  x1 := 15 ;
  draw (x1, y1, x2, y2, color);
end;

procedure drawmm ;
begin
  drawwc ;
  fillshape (9, 9, 3, 2);
end;

procedure drawmc ;
begin
  draw (7, 2, 11, 2, 3); 
  draw (7, 2, 7, 4, 3); 
  draw (11, 2, 11, 4, 3); 
  draw (3, 4, 7, 4, 3); 
  draw (11, 4, 15, 4, 3); 
  draw (3, 4, 3, 6, 3); 
  draw (15, 4, 15, 6, 3); 
  draw (3, 6, 7, 6, 3); 
  draw (11, 6, 15, 6, 3); 
  draw (7, 6, 9, 14, 3); 
  draw (11, 6, 9, 14, 3);
end;

procedure drawmm ;
begin
  drawwnc;
  fillshape (8, 3, 3, 2)
end;

procedure drawboard2 (board : boardtype);

var wt, wb, wl, wr, row, col, color : integer;
  apiece : piecetype;

begin
  palette (2);
  colortable (3, 2, 1, 0);
  graphcolormode;
  graphbackground (0);
  color := 3;
  writeln ('1 2 3 4 5 6 7 8');
  writeln;
  for row := 1 to 8 do
    begin
      wt := 16 + (16 * (row - 1));
      wb := wt + 16;
      writeln;
      writeln (row);
      for col := 1 to 8 do
        begin
          wl := 12 + (18 * (col - 1));
          wr := wl + 18;
          graphwindow (wl, wt, wr, wb);
          draw (1, 1, 18, 16, color);
          draw (1, 16, 18, 16, color);
          draw (18, 1, 18, 16, color);
          if board [row, col].full = true
            then
              begin
                bpiece := board [row, col].piece;
                if bpiece.morw = woman
                  then if bpiece.role = chorus
                        then drawwc
                        else drawmm
                    else if bpiece.role = chorus
                        then drawmc
                        else drawmm
                end;
                graphwindow (0, 0, 319, 199);
              end;
        end;
    end;
end;

procedure display;

var cont : char;

begin
  graphcolormode;
  graphbackground (0);
  palette (1);
  colortable (3, 2, 1, 0);
  writeln ('Heart shaped pieces are women.');
  writeln ('Swords are the men.');
  writeln ('Solid pieces are main actors -- they move in an L - shape.');
  writeln ('Outlined pieces are the chorus --');
end;
writeln ("they move one square at a time.");
graphwindow (20,120,40,160);
crawnc;
graphwindow (60,120,80,160);
drawwm;
graphwindow (100,120,120,160);
drawmc;
graphwindow (140,120,160,160);
drawmm;
writeln ("(Press any key to continue.)");
cont := keyin;
textmode;
end;

procedure makechoice (   side : char;
    var action : char;
    var oldrow, oldcol, newrow, newcol : integer);
var oldsq, newsq : integer;
begin
writeln;
graphwindow (170,1,320,200);
if side = "m"
    then write ("Men - make your move.");
    else write ("Women - make your move.");
writeln ("= move, q = quit, c = god-call");
action := keyin;
write (" = move, q = quit, c = god-call");
if action = "m"
    then
begin
    writeln ("Enter old square then new");
    writeln ("square number.");
    readin (oldsq, newsq);
    oldrow := ((oldsq - 1) div 8) + 1;
    oldcol := ((oldsq - 1) mod 8) + 1;
    newrow := ((newsq - 1) div 8) + 1;
    newcol := ((newsq - 1) mod 8) + 1;
end;
end;

procedure godtake (   godcalled : godtype;
            var board : boardtype);
var
    pieceout : boolean;
    outrow, outcol, outsq : integer;
    osquare : square;
begin
    textmode;
    if godcalled = notthere
    then
begin
    writeln ("The gods are not at your beck and call - this is trivia.");
    writeln ("They have more fundamental business to engage in.");
    writeln ("Presently they are on a picnic.");
    delay (6000);
end
else if godcalled = ares
then
begin
    writeln ("Ares has won the battle - the leader of the men should give th
e");
repeat
  drawboard2 (board);
  writeln;
  writeln ("Ares take your captive.");
  readln (outsq);
  outrow := ((outsq - 1) div 8) + 1;
  outcol := ((outsq - 1) mod 8) + 1;
  osquare := board[outrow, outcol];
  if (osquare.full = true) and (osquare.piece.morw = woman) then
    begin
      pieceout := true;
      board[outrow, outcol].full := false
    end
  else
    begin
      pieceout := false;
      writeln ("There is no women's piece");
      writeln ("in that square - try again.");
    end;
  until pieceout = true
end;

procedure switchsides (var side : char);
begin
  if side = 'm'
    then side := 'w'
  else side := 'm'
end;

procedure compmove (var board : boardtype ; (* dummy *)
  side : char);
begin
end
begin
	textmode(30);
	noveset := ['m', 'M'];

toutloop1 := ['Y', 'y', 'M', 'm'];

toutloop2 := ['Y', 'y'];

repeat
  writeln.start;
  display;
  callcount[1] := 3;
  callcount[2] := 3;
  setboard (board);
  getnum (side, numplayers);
  repeat
    godcall (side, godcalled);
  until godcalled <> notther;
  if godcalled = ares then
    begin
      writeln ('Ares won the first battle, the men will start the game.');
side := 'm';
    end
  else
    begin
      writeln ('Aphrodite won the first battle, the women will start the game.');
side := 'w';
    end;

delay (6000);

newboard := 'n';

repeat
  repeat
    if lnewboard = 'n') or (action = 'n')
  then
    begin
      newboard := 'a';
      drawboard2 (board);
      end;
      makechoice (side, action, oldrow, oldcol, newrow, newcol);
      if action = 'c'
      then
        begin
          downcount (callcount, side, noneleft);
          if noneleft = false
            then
              begin
                godcall (side, godcalled);
                godtake (godcalled, board);
                newboard := 'n';
              end;
            end
      else if action = 'q'
      then
        begin
          textmode;
          writeln ('Are you sure that you want to quit - type y or n');
          action := keyin;
          if action in outloop2
            then goto quit;
        end
  else if action = 'm'
  then if checkmove (board, oldrow, oldcol, newrow, newcol, side) = true
  then
  else

movepiece (board, oldrow, oldcol, newrow, newcol);
end
else action := 'n';
delay (2500)
until action in outloop1;
drawboard2 (board);
takepieces (board, newrow, newcol, change);
if change = true
then
    begin
        writeln;
        writeln ('press any key to next board.');
        newboard := keyin;
        newboard := 'n';
    end;
if numplayers = 2
then switchsides (side)
else compmove (board, side);
until menwon (board) or womenwon (board);
if not (action in outloop2)
then drawboard2 (board);
if menwon (board)
then writeln ('Ares and the men of Greece win the game.\n')
else if womenwon (board)
then
    begin
        writeln ('Aphrodite and the women led\n');
        writeln ('by Lysistrata have won the game.\n');
    end;
quit: textmode;
writeln ('Type the letter n if you want to start a new game.\n');
writeln ('Any other key will exit the program.\n');
newgame := keyin;
until newgame <> 'n'
end.