

```

/* Zufallszahlen wirklich zufällig machen, da sonst bei jedem Programmstart die selben Zahlen kommen.*/
set_random_state (make_random_state(true))$

/* Die Spielfunktion, die eine neue leere Spielmatrix erzeugt und dann abwechselnd Spieler und KI zum Zug
kommen lässt.*/
nofix("tictactoe")$
"tictactoe"():= block([M, s, tmp],
  M: zeromatrix(3,3),
  s: read("wer soll beginnen? (Computer=1, Sie=2, dann mit Semikolon abschließen)"),
  if s=1 then (M: randomZug(M), ausgabe(M))
  else ausgabe(M),
  do(
    M: spielerZug(M),
    ausgabe(M),
    if feldvoll(M) or (tmp: gewinnAbfrage(M))#0 then return(gewonnen(tmp)),
    M: ki(M),
    ausgabe(M),
    if feldvoll(M) or (tmp: gewinnAbfrage(M))#0 then return(gewonnen(tmp)) ))$

/* Eine Funktion zur Gewinnabfrage, sie gibt entweder 0 für unentschieden, 1 für den Sieg des Spielers oder
-1 für den
Sieg der KI zurück.*/
gewinnAbfrage(M):=block([t, ergebnis:0],
  for i: 1 thru 3 do(
    t: M[i,1],
    if t#0 and t=M[i,2] and t=M[i,3] then return(ergebnis:t),
    t: M[1,i],
    if t#0 and t=M[2,i] and t=M[3,i] then return(ergebnis:t)),
  if ergebnis=0 and (t: M[2,2])#0 and ( t=M[1,1] and t=M[3,3] or t=M[1,3] and t=M[3,1]) then ergebnis:t,
  ergebnis)$

/* feldvoll prüft ob noch Züge möglich sind.*/
feldvoll(M):= is(length(moeglicheZuege(M))=0)$

/*Eine Funktion zur ermittlung aller möglichen Züge, die als Liste zurückgegeben werden.*/
moeglicheZuege(M):= block([L:[],e],
  for z:1 thru 3 do(
    for s:1 thru 3 do(
      if M[z,s]=0 then (
        L:cons([z,s],L) ))),
  L)$

```

```

/* Abfrage des gewünschten Zuges des Spielers, dessen Prüfung auf Gültigkeit und Ausführung.*/
spielerZug(M):=block([input, zeile, spalte, text],
do(
    text: "Spieler bitte zug eingeben
        (mit Semikolon[;] abschließen und Enter drücken z.B. \"b2;\"):",
    input: charlist(string(read(text))),
    [zeile, spalte]: map('cint, input) -[96,48],
    if member(zeile, [1,2,3]) and member(spalte,[1,2,3]) and M[zeile, spalte]=0 then return()),
M[zeile, spalte]: 1,
M)$

/*Funktion der KI, die den MiniMax-Algorithmus aufruft und dann den zug ausführt.*/
ki(M):=block([zug],
    zug: kiZugMin(M)[2],
    M[zug[1],zug[2]]: -1,
M)$

/* Ein Zufallszug, der nur ausgeführt wird, wenn die Ki als erste am Zug ist, da der erste Zug bei einem
leeren Feld
für die Gewinnchance unerheblich ist und so Rechenzeit gespart wird.*/
randomZug(M) := block([z,s],
    [z,s]: map('random, [3,3]) + 1,
    M[z,s]: -1,
M)$

/* erster Teil MiniMax-Algorithmus, weitere Erklärungen in der Dokumentation.*/
kiZugMax(M) := block([besterZ, besterw: 'minf, tmpM, bewertung],
    for zug in moeglicheZuege(M) do(
        tmpM: copymatrix(M),
        tmpM[zug[1], zug[2]]: 1,
        sieger: gewinnAbfrage(tmpM),
        if sieger=1 or feldVoll(tmpM) then return([besterw, besterZ]: [sieger, zug]),
        bewertung: kiZugMin(tmpM)[1],
        if bewertung>besterw then (besterZ: zug, besterw: bewertung)),
    [besterw, besterZ])$

/* zweiter Teil MiniMax-Algorithmus, weitere Erklärungen in der Dokumentation.*/
kiZugMin(M) := block([besterZ, besterw: 'inf, tmpM, bewertung],
    for zug in moeglicheZuege(M) do(
        tmpM: copymatrix(M),

```

```

    tmpM[zug[1], zug[2]]: -1,
    sieger: gewinnAbfrage(tmpM),
    if sieger=-1 or feldvoll(tmpM) then return([besterw, besterZ]: [sieger, zug]),
    bewertung: kiZugMax(tmpM)[1],
    if bewertung<besterw then (besterZ: zug, besterw: bewertung)),
[besterw, besterZ])$

/* Eine Ausgabefunktion, die das Spielfeld mit Spalten- und Zeilennummerierung ausgibt.*/
ausgabe(M) := block([aListe],
  aListe: [[1,88],[-1,165],[0,48]],
  print("  1  2  3"),
  for z:1 thru 3 do(
    sprint(ascii(z+96)),
    for s:1 thru 3 do(
      sprint("", ascii( assoc(M[z,s], aListe) )),
      newline()),
    newline())$

/*Diese Funktion ist für die Ausgabe nach Beendigung des Spiels zuständig.*/
gewonnen(sieger) :=
  if sieger=0 then "Unentschieden!"
  else if sieger=-1 then "Der Computer hat gewonnen!"
  else "Sie haben gewonnen!"$

/* Compiler-Ausgaben in Datei umlenken: */
define_variable(output, ?\*standard\-output\*, any)$
?\*standard\-output\* :
  openw(sconcat(maxima_tempdir,"/maxima_compile.temp"))$

/*Alle Funktionen Compilieren, damit später REchenzeit gespart wird.*/
compile("tictactoe",gewinnAbfrage,feldvoll,moeglicheZuege,spielerZug,
  ki,randomZug,kiZugMax,kiZugMin,ausgabe,gewonnen)$

/* Zurücklenkung der Ausgabe, damit das Spiel beginnen kann.*/
?\*standard\-output\* : output$
print("Start mit der Eingabe von \"tictactoe;\".")$

/* Tic Tac Toe mit Gegenspieler für Maxima von Jonathan Wendt*/

```