```
1: {*****************************************************}
2: { 4Gewinnt Game  "the fantastic four"                  }
3: { RECHNER.INC: Include-File mit der implementierten    }
4: {              Strategieroutine für 4GEWINNT.PAS       }
5: { ------------------------------------------------- }
6: { Autor       : Max Kleiner T-Ask                     }
7: { Lang        : Borland Pascal for Win
8:   loc's= 616  : 1995 - 2012  remake for maXbox        }
9: {*****************************************************}
10:
11: //Task: Set an event-handler for On_Maximize and On_Minimize to reDraw the Game!
12:
13: Const { maximale Bewertung }
14:       Unendlich = 32000;
15:       { Wert einer Reihe wo schon drei Steine einer Farbe sind}
16:       Wert2 = 8;
17:       { Wert einer Reihe wo schon zwei Steine einer Farbe sind}
18:       Wert3 = 30;
19:       N                     = 6;       //N * M   row * col
20:       M                     = 7;       //col
21:       BLAU                  = 1;
22:       ROT                   = 10;
23:       BORDER                = 20;
24:       BSUM                  = 256;
25:
26:  Type
27:       { Rechentiefe für die einzelnen Spielstärken }
28:       TRechentiefe = Array[0..3] Of Integer;
29:       TZeilenVektor = array[1..M] of Integer;    //Row inside
30:       TSpielMatrix = array[1..N] of TZeilenVektor;
31:
32:       { Wert der Stein Position - Stone Position Value SPV }
33:       {PosWert : SpielMatrix = ((3, 4, 5, 7, 5, 4, 3),
34:                                 ( 4, 6, 8,10, 8, 6, 4),
35:                                 ( 5, 8,11,13,11, 8, 5),
36:                                 ( 5, 8,11,13,11, 8, 5),
37:                                 ( 4, 6, 8,10, 8, 6, 4),
38:                                 ( 3, 4, 5, 7, 5, 4, 3));}
39:
40:  var  deepc: TRechentiefe;
41:       SM, SpM, p: TSpielMatrix;
42:       ZA, Count:  TZeilenVektor;
43:       Drei_Rot, Drei_Blau, Ende, Equal, compute,
44:                      Sieg_Rot, Sieg_Blau, ChangeColor: Boolean;
45:       RWert: Array[0..40] of Integer;
46:       CompStart, Best, Delta, StX, StY, L1, Color: Integer;
47:       Abbruch: Boolean;
48:       pForm: TForm;   // _4Gewinnt: TVierGewinnt;
49:       Grad: Byte;
50:       Score: Longint;
51:
52:
53: procedure WMRechner; forward;
54:
55: procedure initMatrix;
56: begin
57:   deepc[0]:= 4;   deepc[1]:= 4;
58:   deepc[2]:= 5;   deepc[3]:= 6;
59:
60:   //ZeilenVektor = (4,3,5,2,6,7,1);
61:   ZA[1]:= 4; ZA[2]:= 3; ZA[3]:= 5;
62:   ZA[4]:= 2; ZA[5]:= 6; ZA[6]:= 7; ZA[7]:= 1;
63:
64:   p[1][1]:=3; p[1][2]:=4; p[1][3]:=5;  p[1][4]:=7;  p[1][5]:=5;  p[1][6]:=4; p[1][7]:=3;
65:   p[2][1]:=4; p[2][2]:=6; p[2][3]:=8;  p[2][4]:=10; p[2][5]:=8;  p[2][6]:=6; p[2][7]:=4;
66:   p[3][1]:=5; p[3][2]:=8; p[3][3]:=11; p[3][4]:=13; p[3][5]:=11; p[3][6]:=8; p[3][7]:=5;
67:   p[4][1]:=5; p[4][2]:=8; p[4][3]:=11; p[4][4]:=13; p[4][5]:=11; p[4][6]:=8; p[4][7]:=5;
68:   p[5][1]:=4; p[5][2]:=6; p[5][3]:=8;  p[5][4]:=10; p[5][5]:=8;  p[5][6]:=6; p[5][7]:=4;
69:   p[6][1]:=3; p[6][2]:=4; p[6][3]:=5;  p[6][4]:=7;  p[6][5]:=5;  p[6][6]:=4; p[6][7]:=3;
70: end;
71:
72: Procedure T4GwWindow_Anfaenger;   //cm prototype 1995!
73: Begin
74:   {MyMenu:= GetMenu(HWindow);
75:   CheckMenuItem(MyMenu,cm_Anfaenger+Grad,
76:                 mf_ByCommand+mf_Unchecked);
77:   Grad:= 0;}
78: End;
79:
80:
81: Function Auswertung(stufe: integer; rs: byte): integer;
82: var BW: integer;
83: Begin
84:   Drei_Rot:= rS=30;
85:   Drei_Blau:= rS=3;
86:   If rS>1 Then
87:     If rS=40 Then Begin
88:       result:= -30000-Stufe;
89:       If Stufe=100 Then
```

```
90:         result:= -Unendlich;
91:       Ende:= True;
92:     End Else
93:     If rS=4 Then Begin
94:       result:= 30000+Stufe;
95:       If Stufe=100 Then
96:         result:= Unendlich;
97:       Ende:= True;
98:     End Else
99:       BW:= BW + RWert[rS];
100:     //Inc(BW,RWert[S]);
101: End;
102:
103: {-----------------------------------------------------}
104: {*****************************************************}
105: { T4GwWindow.Rechner: Reaktion auf Meldung wm_rechner }
106: {                     In dieser Routine wird der Zug  }
107: {                     für den Computer mit Hilfe      }
108: {                     Minimaxstrategie und AlphaBeta- }
109: {                     Abschneidung ermittelt.         }
110: {*****************************************************}
111: {*****************************************************}
112: { Mit Hilfe dieser Funktion wird die jeweilige Spiel- }
113: { stellung bewertet.                                  }
114: {*****************************************************}
115:
116: Function Bewertung(Stufe: Integer): Integer;
117: Var BW, S, i, j, k, Help: Integer;
118:   {-------------------------------------------------}
119:   { Hilfsprozedur zur Auswertung der Spielstellung  }
120:   {-------------------------------------------------}
121: Begin
122:   BW:= 0;
123:   {-------------------------------------------------}
124:   { Bewertungskriterium 1:                          }
125:   {   Werte der einzelnen Spielsteinpositionen      }
126:   {-------------------------------------------------}
127:   For j:= 1 To M Do
128:     For i:= 1 To Count[j] Do Begin
129:       If SM[i][j]=1 Then
130:         BW:= BW+P[i][j];
131:       If SM[i][j]=10 Then
132:         BW:= BW-P[i][j];
133:     End;
134:   {-------------------------------------------------}
135:   { Bewertungskriterium 2:                          }
136:   {   Bewertung der jeweiligen Zweier-, Dreier- und }
137:   {   Viererreihen der Spielstellung                }
138:   {-------------------------------------------------}
139:   Ende:= False;
140:
141:   {-------- senkrechte Reihen --------}
142:   For j:= 1 To M Do Begin
143:     Help:= Count[j];
144:     If Help>3 Then Help:= 3;
145:     For i:= 1 To Help Do Begin
146:       S:= SM[i][j]+SM[i+1][j]+SM[i+2][j]+SM[i+3][j];
147:       result:= Auswertung(stufe,S);
148:       If Ende Then Exit;
149:       If Drei_Rot Then
150:         For k:= 0 To 3 Do
151:           If SM[i+k][j]=0 Then
152:             If i+k And 1=CompStart Then
153:               BW:= BW-RWert[3];
154:               //Dec(BW,RWert[3]);
155:       If Drei_Blau Then
156:         For k:= 0 To 3 Do
157:           If SM[i+k][j]=0 Then
158:             If i+k And 1=1-CompStart Then
159:               BW:= BW + RWert[3];
160:               //Inc(BW,RWert[3]);
161:     End; //for
162:   End; //for
163:
164:   {-------- waagrechte Reihen --------}
165:   For j:= 1 To M-3 Do
166:     For i:= 1 To N Do Begin
167:       S:= SM[i][j]+SM[i][j+1]+SM[i][j+2]+SM[i][j+3];
168:       result:= Auswertung(stufe,S);
169:       If Ende Then Exit;
170:       If Drei_Rot And (j>1) Then
171:         If j And 1=CompStart Then
172:           BW:= BW-3*RWert[3];
173:           //Dec(BW,3*RWert[3]);
174:       If Drei_Blau And (j>1) Then
175:         If j And 1=1-CompStart Then
176:           BW:= BW + 3*RWert[3];
177:           //Inc(BW,3*RWert[3]);
178:     End;
```

```
179:   {-------- diagonale Reihen --------}
180:   For i:= 1 To N-3 Do
181:     For j:= 1 To M-3 Do Begin
182:       S:= SM[i][j]+SM[i+1][j+1]+SM[i+2][j+2]+SM[i+3][j+3];
183:       result:= Auswertung(stufe,S);
184:       If Ende Then Exit;
185:       If Drei_Rot Then
186:         For k:=0 To 3 Do
187:           If SM[i+k][j+k]=0 Then
188:             If i+k And 1=CompStart Then
189:               BW:= BW - 2*RWert[3];
190:               //Dec(BW,2*RWert[3]);
191:       If Drei_Blau Then
192:         For k:=0 To 3 Do
193:           If SM[i+k][j+k]=0 Then
194:             If i+k And 1=1-CompStart Then
195:               BW:= BW-2*RWert[3];
196:               //Inc(BW,2*RWert[3]);
197:       S:= SM[i+3][j]+SM[i+2][j+1]+SM[i+1][j+2]+SM[i][j+3];
198:       result:= Auswertung(stufe,S);
199:       If Ende Then Exit;
200:       If Drei_Rot Then
201:         For k:=0 To 3 Do
202:           If SM[i+3-k][j+k]=0 Then
203:             If i+3-k And 1=CompStart Then
204:               BW:= BW-2*RWert[3];
205:       If Drei_Blau Then
206:         For k:= 0 To 3 Do
207:           If SM[i+3-k][j+k]=0 Then
208:             If i+3-k And 1=1-CompStart Then
209:               BW:= BW+2*RWert[3];
210:     End; //for
211:   result:= BW;
212: End;
213:
214: {*****************************************************}
215: { Ermittlung des besten Zuges für den Computer mit    }
216: { Hilfe der MiniMax-Strategie und dem AlphaBetaCut    }
217: { Diese rekursive Funktion liefert schließlich den    }
218: { Wert der Spielstellung zurück. Der beste Spielzug   }
219: { ist dann in der Variable Bester abgelegt.           }
220: {*****************************************************}
221:
222: Function MiniMax(Wert,Tiefe,Alpha: Integer): Integer;
223: Var i,j, Help, Zug, Beta: Integer;
224:               AlphaBetaCut: Boolean;
225: Begin
226:   If Not Abbruch Then Begin
227:     If (Abs(Bewertung(Tiefe+1))>=29000) OR
228:        (Count[1]+Count[2]+Count[3]+Count[4]+
229:         Count[5]+Count[6]+Count[7]>= 42) Then
230:       result:= Bewertung(Tiefe+1)
231:     Else Begin
232:       {While PeekMessage(HMsg,HWindow,0,0,pm_Remove) Do
233:         If (HMsg.Message=wm_SysCommand) And
234:            (HMsg.WParam=sc_Close) Then Abbruch:=True
235:         Else while Application.ProcessMesages do
236:          //Abbruch:= true; }
237:       If Wert=1 Then
238:         Beta:= -Unendlich
239:       Else
240:         Beta:= Unendlich;
241:       Zug:= 0;
242:       AlphaBetaCut:=False;
243:       If Tiefe>0 Then Begin
244:         For i:= 1 To M Do Begin
245:           j:= ZA[i];
246:           If (Count[j]<N) AND NOT AlphaBetaCut Then Begin
247:             Inc(Count[j]);
248:             SM[Count[j]][j]:= Wert;
249:             If Tiefe>1 Then
250:               Help:= MiniMax(Blau+Rot-Wert,Tiefe-1,Beta)
251:             Else
252:               Help:= Bewertung(Tiefe);
253:             SM[Count[j]][j]:= 0;
254:             Dec(Count[j]);
255:             If Wert=Blau Then Begin
256:               If Help>Beta Then Begin
257:                 Beta:= Help;
258:                 Zug:= j;
259:               End;
260:               If Beta>Alpha Then
261:                 AlphaBetaCut:=True;
262:             End
263:             Else Begin
264:               If Help<Beta Then Begin
265:                 Beta:= Help;
266:                 Zug:= j;
267:               End;
```

```
268:               If Beta<Alpha Then
269:                 AlphaBetaCut:= True;
270:             End;
271:           End; //If
272:         End; //For
273:         result:= Beta;
274:       End //If
275:       Else result:= Bewertung(Tiefe+1);
276:     End;
277:    Best:= Zug;
278:   End;
279: End;
280:
281: {************************************************************}
282: { Hilfsfunktion zur Bestimmung, ob das Spiel noch weiter geht  }
283: {************************************************************}
284: Function SpielEnde: Boolean;
285: Begin
286:   result:= True;
287:   If Bewertung(100)<=-Unendlich Then
288:     Sieg_Rot:= True
289:   Else
290:     If Bewertung(100)>=Unendlich Then
291:       Sieg_Blau:= True
292:   Else
293:     If Count[1]+Count[2]+Count[3]+Count[4]+
294:          Count[5]+Count[6]+Count[7]= N*M Then
295:            Equal:= True
296:     Else
297:       result:= False;
298: End;
299:
300: /////from main game form
301: Function FarbWert(W: Word): TColorRef;  //TColor?
302: Begin
303:   Case W Of
304:     0: result:= RGB2TColor($BF,$BF,$BF);
305:     1: result:= RGB2TColor($00,$00,$00);
306:     2: result:= RGB2TColor($FF,$FF,$FF);
307:     3: result:= RGB2TColor($FF,$00,$ff);
308:     4: result:= RGB2TColor($00,$00,$00);
309:     5: result:= RGB2TColor($00,$00,$FF);
310:     6: result:= RGB2TColor($F7,$00,$00);
311:     7: result:= RGB2TColor($7F,$7F,$7F);
312:   End;
313: End;
314:
315:
316: Procedure Reset;
317: Var i,j: Integer;
318: Begin
319:   compute:= False;
320:   Sieg_Rot:= False;
321:   Sieg_Blau:= False;
322:   Equal:= False;
323:   For i:= 1 To N Do
324:     For j:= 1 To M Do SpM[i][j]:= 0;
325:   For j:= 1 To M Do Count[j]:= 0;
326:   Delta:= 0;
327: End;
328:
329:
330: Procedure WM_SetzeStein(wparam, lparam: integer);
331: Var //DC: HDC;
332:     XPos, YPos, X, Y: Integer;
333: Begin
334:   Y:= 7-wParam Mod BSUM;
335:   X:= wParam Div BSUM;
336:   XPos:= StX+(X-1)*L1+2;
337:   YPos:= StY+(Y-1)*L1+2;
338:   //DC:=GetDC(HWindow);
339:     if changeColor then
340:       pForm.Canvas.brush.Color:= FarbWert(lparam+2)
341:     else
342:       pForm.Canvas.brush.Color:= FarbWert(lparam+color);
343:    //SelectObject(DC,Brush);
344:    pForm.Canvas.Ellipse(XPos,Ypos,Xpos+L1-3,Ypos+L1-3);
345:    //pform.Canvas.TextOut(xpos,ypos,inttostr(p[y][x])); //debug the values
346:    //ReleaseDC(HWindow,DC);
347: End;
348:
349: //********************** Set the Game Board Form ****************************
350: Procedure Spielfeld;
351: Var NRect: TRect;
352:     Breite, Hoehe, i: Integer;
353: Begin
354:   //pForm.canvas.GetClientRect(HWindow,Rect);
355:   //DC:=GetDC(HWindow);
356:   with pForm.Canvas do begin
```

```
357:     brush.color:= FarbWert(0+color);
358:     NRect:= Rect(0,0,pform.width-BORDER,pform.height-(2*BORDER));
359:     FillRect(NRect);
360:     Breite:= (NRect.Right-BORDER) Div M;
361:     Hoehe:= (NRect.Bottom-(2*BORDER)) Div N;
362:     If Breite>Hoehe Then L1:= Hoehe Else L1:= Breite;
363:     Brush.color:= FarbWert(3+color);
364:     StX:= (NRect.Right-L1*M) Div 2;
365:     StY:= (NRect.Bottom-L1*N) Div 2;
366:     Rectangle(StX,StY,L1*M+StX+1,L1*N+StY+1);
367:     For i:= 1 To M-1 Do Begin
368:       MoveTo(L1*i+StX,StY);
369:       LineTo(L1*i+StX,StY+L1*N);
370:     End;
371:     For i:= 1 To N-1 Do Begin
372:       MoveTo(StX,L1*i+StY);
373:       LineTo(L1*M+StX,L1*i+StY);
374:     End;
375:   End; //with
376:   //Sbutton.top:= pForm.height-4*BORDER; debug
377:   //ReleaseDC(HWindow,DC);
378: End;
379:
380: Procedure Gewonnen;
381: Var mRect: TRect; GMsg: PChar;
382: Begin
383:   GMsg:='';
384:   If Sieg_Rot Then GMsg:='  Wow Gratulation !!!  ';
385:   If Sieg_Blau Then GMsg:='  Sorry, you lost !!!  ';
386:   If Equal Then GMsg:='  Same for two !!!  ';
387:   If Sieg_Rot Or Sieg_Blau Or Equal Then Begin
388:     //GetClientRect(HWindow,Rect);
389:     mRect.Top:= 2;
390:     mRect.Bottom:= BORDER;
391:     //Showmessage(GMsg);  //debug
392:     pform.Canvas.TextOut(mRect.top, mrect.bottom-BORDER+5, GMsg);
393:   End;
394: End;
395:
396: //*************************** Event Handler ******************************
397: Procedure WM_Paint(Sender: TObject);
398: Var i, j: Word;
399: Begin
400:   Color:= 4;
401:   Spielfeld;
402:   For i:= 1 To M Do
403:     For j:= 1 To Count[i] Do Begin
404:       If SpM[j][i]=Rot Then
405:          WM_setzestein(i*BSUM+j,2);
406:       If SpM[j][i]=Blau Then
407:          WM_setzestein(i*BSUM+j,1);
408:     End;
409:   Gewonnen;
410: End;
411:
412: procedure FormCloseClick(Sender: TObject; var Action: TCloseAction);
413: begin
414:   //myImage.Free;
415:   Writeln('4Gewinnt Form Closed at: '+ TimeToStr(Time));
416:   //pFrm.Free;
417:   Abbruch:= True;
418:   Action:= caFree;
419: end;
420:
421:
422: //Procedure WMMouseMove;
423: procedure GewinntMouseMove(Sender: TObject; Shift: TShiftState; X,Y: Integer);
424: Var XPos, X1: Integer;
425:     Help1, Help2: Integer;
426: begin
427:   If Not compute Then Help1:= crArrow
428:     Else Help1:= crHourglass;
429:   If Not compute Then Help2:= crCross  //idc_cross
430:     Else Help2:= crHourglass;
431:   XPos:= X;
432:   If (XPos>StX) AND (XPos<StX+M*L1) AND NOT
433:      (Sieg_Rot OR Sieg_Blau Or Equal) Then
434:      Begin
435:        X1:=(XPos-StX) Div L1+1;     //shows possible move
436:        If X1>7 Then X1:= 7;
437:        If X1<1 Then X1:= 1;
438:        If Count[X1]<N Then Screen.Cursor:= Help2
439:          Else Screen.Cursor:= help1   //SetCursor(LoadCursor(0,Help1));
440:      End
441:   Else Screen.Cursor:= help1;
442: End;
443:
444: //Procedure T4GwWindow.WMLButtonDown;
445: procedure MouseDownLeft(sender: TObject; Button: TMouseButton;
```

```
446:                          Shift: TShiftState; X, Y: Integer);
447: Var XPos, X1, cntint: Word;
448: Begin
449:   XPos:= X;
450:   If (XPos>StX) AND (XPos<StX+M*L1) AND NOT
451:     (Sieg_Rot OR Sieg_Blau OR Equal) Then
452:   Begin
453:     X1:= (XPos-StX) Div L1+1;
454:     If X1> M Then X1:= M;
455:     If X1< 1 Then X1:= 1;
456:     If Count[X1] < N Then Begin
457:       Inc(Count[X1]);
458:       If Count[X1]= N Then Inc(Delta);
459:       cntint:= Count[X1]
460:       SpM[cntint][X1]:= Rot;
461:       WM_Setzestein(X1*BSUM+Count[X1],2);
462:       WMRechner; //Bewertung, Auswertung(1);
463:     End;
464:   End;
465: End;
466:
467:
468: Procedure InitGame;
469: Begin
470:   //TWindow.Init(NIL,AName);
471:   //Attr.Menu:=LoadMenu(HInstance,'MENU');
472:   Grad:= 1;         //levels 0 - 3;  3 as Expert
473:   CompStart:= 1;
474:   changeColor:= false;
475:   Reset;
476: End;
477:
478: procedure ButtonReset(sender: TObject);
479: begin
480:   InitGame;
481:   Spielfeld;
482: end;
483:
484: procedure EChangeColor(sender: TObject);
485: begin
486:   changeColor:= NOT changeColor;
487: end;
488:
489: procedure EChangeLevel(sender: TObject);
490: begin
491:   Grad:= 3;  //highest level
492: end;
493:
494:
495: procedure FormTCreate(Sender: TObject);
496: //var label1: TLabel; bevel1,bevel2: TBevel;  for future expansion
497: var mi, mi1, mi2: TMenuItem;
498:     mt: TMainMenu;
499:     sbutton: TButton;
500: begin
501:   //SetFigures;
502:   //RedrawSheet:= True;
503:   {bevel1:= TBevel.create(pform)
504:   bevel1.parent:= pForm;
505:   bevel2:= TBevel.create(pform)
506:   bevel2.parent:= pForm;
507:   label1:= TLabel.create(pform)
508:   label1.parent:= pForm;}
509:
510:   pform:= TForm.Create(self);   //constructors
511:   sButton:= TButton.Create(pform)
512:   with pform do begin
513:     caption:= '4Gewinnt GameBox 2012';
514:     //BorderStyle:= bsDialog;
515:     Position:= poScreenCenter;
516:     onMouseDown:= @MouseDownLeft;
517:     onMouseMove:= @GewinntMouseMove;
518:     onPaint:= @WM_Paint;
519:     onClose:= @FormCloseClick;
520:     //KeyPreview:= true;
521:     ClientWidth:= pForm.Width+150;
522:     ClientHeight:= pForm.height+150;
523:     Show;
524:   end;
525:   with SButton do begin
526:     parent:= pForm;
527:     caption:= 'Reset'
528:     top:= pForm.height-4*BORDER-5;
529:     width:= 4*BORDER;
530:     onclick:= @ButtonReset;
531:   end;
532:   mt:= TMainMenu.Create(pForm)
533:   with mt do begin
534:    //parent:= frmMon;
```

```
535:   end;
536:   mi:= TMenuItem.Create(mt)
537:   mi1:= TMenuItem.Create(mt)
538:   mi2:= TMenuItem.Create(mt)
539:   with mi do begin
540:     //parent:= frmMon;
541:     Caption:='New Game';
542:     Name:='ITEM';
543:     mt.Items.Add(mi);
544:     OnClick:= @ButtonReset;
545:   end;
546:   with mi1 do begin
547:     //parent:= frmMon;
548:     Caption:='Change Color';
549:     mt.Items.Add(mi1) ;
550:     OnClick:= @EChangeColor
551:   end;
552:   with mi2 do begin
553:     //parent:= frmMon;
554:     Caption:='High Level';
555:     mt.Items.Add(mi2);
556:     OnClick:= @EChangeLevel;
557:   end;
558:   Spielfeld;
559:   //Grad:= 1;
560:   Score:= 0;
561: end;
562:
563: {*************************************************}
564: {      Hauptteil der Methode T4GwWindow.Rechner       }
565: {*************************************************}
566: //Procedure T4GwWindow_Rechner;
567:
568: procedure WMRechner;
569: var i,j: Integer;   // from Rechner
570: begin
571:   For i:=0 To 40 Do RWert[i]:= 0;
572:   RWert[3]:= Wert3;
573:   RWert[30]:= -Wert3;
574:   RWert[2]:= Wert2;
575:   RWert[20]:= -Wert2;
576:   SM:= SpM;
577:     {for I:= 1 to N do
578:       for j:= 1 to M do SM[i][j]:= SpM[i][j];}
579:   If Not SpielEnde Then Begin
580:     Screen.Cursor:= crHourglass;//SetCursor(LoadCursor(0,idc_wait));
581:     compute:= True;
582:     Abbruch:= False;
583:     MiniMax(Blau, deepc[Grad]+Delta,Unendlich);
584:     If Abbruch Then Showmessage('PostQuitMessage(0) or Game Closed')
585:     Else
586:     If (Count[Best]<N) AND (Best>0) Then Begin
587:       Inc(Count[Best]);
588:       If (Count[Best]=N) AND (Grad>0) Then
589:         Inc(Delta);
590:       SpM[Count[Best]][Best]:= Blau;
591:       WM_Setzestein(Best*BSUM+Count[Best],Blau);
592:       SM:= SpM;  //!
593:     End; //If
594:   End;
595:   SpielEnde;
596:   Gewonnen;  //SendMessage(HWindow,wm_gewonnen,0,0);
597:   compute:= False;
598:   //SetCursor(LoadCursor(0,idc_arrow));
599:   Screen.Cursor:= crArrow;
600: end;
601:
602: Procedure InitComputerStart;
603: Begin
604:   Reset;
605:   CompStart:= 1;
606:   WMRechner;
607: End;
608:
609:
610: Begin     //Main Control
611:   initMatrix;
612:   initGame;
613:   FormTCreate(self);
614:   WMRechner;
615: End.
616: {*****************************T-Ask maXPlay Series**********************}
```