

**SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN**

Ivan Vlašić

IMPLEMENTACIJA CONNECT FOUR IGRE U PYTHONU S MINIMAX ALGORITMOM

PROJEKT

UVOD U UMJETNU INTELIGENCIJU

Varaždin, 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Ivan Vlašić

Matični broj: 0016152618

Studij: Informacijski i poslovni sustavi

**IMPLEMENTACIJA CONNECT FOUR IGRE U PYTHONU S MINIMAX
ALGORITMOM**

PROJEKT

Mentor:

Prof. dr. sc. Markus Schatten

Varaždin, Siječanj 2024.

Ivan Vlašić

Izjava o izvornosti

Izjavljujem da je ovaj projekt izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvatanjem odredbi u sustavu FOI Radovi

Sažetak

Ovaj rad istražuje implementaciju Connect Four igre koristeći programski jezik Python, s posebnim naglaskom na upotrebu umjetne inteligencije. Teorijsko-metodološko polazište temelji se na analizi algoritma minimax. Kroz projekt, razvijena je simulacija igre s grafičkim sučeljem koristeći pygame biblioteku. Glavne teze rada obuhvaćaju jasnu strukturu koda, praktičnu primjenu minimax algoritma te implementaciju grafičkog sučelja koje poboljšava korisničko iskustvo. Kroz analizu funkcionalnosti poteza igrača i umjetne inteligencije, rad se bavi praktičnom izvedbom igre te naglašava raznolikost poteza umjetne inteligencije. Zaključci ukazuju na uspješnost implementacije, pružajući uvid u primjenu umjetne inteligencije u kontekstu jednostavne igre. Ovaj rad služi kao koristan resurs za razumijevanje umjetne inteligencije u igrama te može poslužiti kao temelj za daljnje istraživanje i edukaciju.

Ključne riječi: Četiri u nizu, Connect four, game, AI

Sadržaj

1. Uvod	1
2. Četiri u nizu ili Connect Four	2
3. Pravila i Logika Igre "Connect Four"	3
3.1. Cilj igre	3
3.2. Struktura Ploče	3
3.3. Sudionici	3
3.4. Potezi	3
3.5. Pobjednički Uvjeti	3
3.6. Neriješeno Stanje	3
3.7. Strategija	4
4. Kritički Osvrt na Implementaciju Connect Four Igre u Pythonu	5
4.1. Korištene Komponente	5
4.2. Algoritam	5
4.3. Praktična Izvedivost	5
4.4. Primjena	6
5. Programski kod	7
6. Objašnjenje koda	16
6.1. Bibiloteke i moduli	16
6.2. Funkcije	16
6.2.1. Organizacija funkcija	16
6.2.1.1. Funkcije Grafičkog Sučelja	16
6.2.1.2. Logika igre	17
6.2.1.3. Parametri	17
6.2.1.4. Minimax algoritam	17
6.2.2. Glavna Petlja Igre	18
7. Prikaz rada aplikacije	19
8. Zaključak	24
Popis literature	25
Popis slika	26

Popis isječaka koda	27
----------------------------	-----------

1. Uvod

U današnjem digitalnom dobu, umjetna inteligencija (AI) igra ključnu ulogu u razvoju sofisticiranih sustava s mogućnošću donošenja odluka. Tema ovog seminarskog rada usmjerena je na implementaciju igre s prediktivnim AI protivnikom, posebice korištenjem algoritma minimax. Odabir ove teme proizlazi iz želje za istraživanjem dubljih aspekata umjetne inteligencije kroz prizmu igara, pri čemu će se fokusirati na Connect Four ili na hrvatskom četiri u nizu, popularnu igru koja kombinira strategiju i taktiku.

Motivacija za odabir ove teme leži u želji za razumijevanjem kako algoritmi umjetne inteligencije mogu unaprijediti korisničko iskustvo u igrama te kako se takvi algoritmi mogu prilagoditi različitim scenarijima. Kroz implementaciju Connect Four igre u programskom jeziku Python, očekuje se razvoj AI protivnika koji će koristiti minimax algoritam kako bi pružio izazov igraču.

U glavnom dijelu rada pružit će se definicije ključnih pojmova poput minimax algoritma te teoretski okvir na kojem se temelji opisani formalizam. Osim toga, razmatrat će se praktična izvedivost i primjena predloženog pristupa, istražujući kako se algoritam može prilagoditi dinamici Connect Four igre.

2. Četiri u nizu ili Connect Four

Connect Four, poznata i kao "Četiri u nizu" ili "Četiri u liniji", je popularna strategijska igra koja se igra na ploči s 7 stupaca i 6 redaka. Cilj igre je postaviti četiri svoja znaka u nizu, bilo vodoravno, okomito ili dijagonalno, prije suparničkog igrača. Connect Four, ili "Četiri u nizu", je igra koja ima zanimljivu povijest i postala je jedna od najprepoznatljivijih i najigranijih igara na svijetu. Igra je izvorno patentirana 1974. godine od strane američkog dizajnera Howarda Wexlera i njegovog sina Nila Wexlera.

Inspiraciju su crpili iz tradicionalnih igara poput "Tic Tac Toa" i "Gomoku". Američka tvrtka za igračke i igre, stekla je prava na igru i proizvodnju, te je od tada distribuirala igru pod svojom popularnom markom "Connect Four". Igra je doživjela mnoge varijacije i digitalne implementacije. Pojavile su se razne verzije s dodatnim značajkama, ali osnovni koncept ostaje nepromijenjen. Različite varijacije igre su PopOut, Pop10, Five in a row, Power up. . .

Četiri u nizu se često koristi u školama i obrazovnim ustanovama kao sredstvo za razvoj logičkog razmišljanja i strategijskih vještina kod djece.

Connect Four, poznata i kao 'Četiri u nizu' ili 'Četiri u liniji', nije samo popularna strategijska igra, već je i interesantna u kontekstu umjetne inteligencije. Razmatrano je kako bi se računalima moglo omogućiti izvođenje zadataka za koje ljudi trebaju inteligenciju. Ovo istraživanje reflektira razmišljanje o potencijalima računala i inteligencije, što se može povezati s evolucijom igre 'Connect Four' i njenom adaptacijom kroz godine.

„One of the first domains in Artificial Intelligence (AI) research, has been computer chess. This is not surprising, if we consider the possibilities people thought computers would have in the near future. Using these possibilities, people thought it would be possible to let the computer perform tasks for which humans need intelligence, whatever that may be.” [1, str. 5]

3. Pravila i Logika Igre "Connect Four"

Connect Four je igra koja spaja jednostavnost s dubokom strategijom, pružajući igračima uzbudljivo iskustvo taktičkog razmišljanja. Evo detaljnijeg pregleda pravila i logike igre:

3.1. Cilj igre

Cilj igre je postaviti četiri svoja žetona u nizu, bilo vodoravno, okomito ili dijagonalno, na ploči veličine 7x6. Ova jednostavna pravila čine osnovnu strukturu igre, ali taktički aspekt dodaje dubinu i kompleksnost.

3.2. Struktura Ploče

Igra se odvija na vertikalnoj ploči s 7 stupaca i 6 redaka, ukupno 42 polja. Svaki igrač ima zadatak pametno postavljati svoje žetone na ploču kako bi stvorio pobjednički niz.

3.3. Sudionici

Igra se između dva igrača, pri čemu svaki igrač kontrolira svoje žetone koji su obično različito obojeni. Ova jednostavna podjela stvara osnovu za dinamičan dvoboj između dva igrača.

3.4. Potezi

Igrači naizmjenice stavljaju svoje žetone u jedan od sedam stupaca. Žeton zauzima prvo slobodno mjesto na dnu odabranog stupca. Ova slobodna forma postavljanja čini svaki potez važnim, jer igrači moraju odabrati mudro kako bi postigli svoje ciljeve.

3.5. Pobjednički Uvjeti

Pobjednik je igrač koji prvi postavi četiri svoja žetona u nizu, bez obzira na smjer: vodoravno, okomito ili dijagonalno. Ovaj dinamički aspekt povećava potrebu za pažljivim planiranjem svakog poteza.

3.6. Neriješeno Stanje

Ako su svi stupci ispunjeni, a nijedan igrač nije postigao pobjedu, igra završava neriješeno. Ovo dodaje dodatnu tenziju jer igrači moraju balansirati između napada i obrane kako bi izbjegli pat poziciju.

3.7. Strategija

Connect Four zahtijeva duboko taktičko razmišljanje. Igrači moraju simultano blokirati protivničke poteze i graditi vlastitu liniju prema pobjedi. Razvijanje dugoročne strategije i brza prilagodba tijekom igre ključni su za postizanje uspjeha u ovoj dinamičnoj igri. Sposobnost predviđanja poteza protivnika i postavljanje zamki često čine razliku između pobjede i poraza.

Connect Four je dvoslojna igra s potpunim informacijama, suparnička s nul-sumom. Postoje 4,531,985,219,092 mogućih položaja na ploči 7x6. Ukratko, Connect Four je teorijski riješena igra, gdje prvi igrač može pobijediti uz savršeno igranje.

4. Kritički Osvrt na Implementaciju Connect Four Igre u Pythonu

U okviru ovog rada, izrađena je simulacija igre Connect Four koristeći programski jezik Python. Ovaj projekt koristi se za istraživanje umjetne inteligencije u kontekstu jednostavne igre.

4.1. Korištene Komponente

- Numpy: Korišten za manipulaciju višedimenzionalnim poljima, posebno za stvaranje i upravljanje igraćom pločom.
- Pygame: Grafička biblioteka koja omogućuje vizualno sučelje igre, čineći je pristupačnom i zabavnom.
- Math: Korišten za osnovne matematičke operacije potrebne tijekom izvođenja igre.

4.2. Algoritam

- Minimax: Ključni dio implementacije za odlučivanje poteza umjetne inteligencije. Minimax pristup omogućuje sustavno ocjenjivanje svih mogućih poteza. ovaj algoritam se primjenjuje u igrama gdje postoji dvoje igrača koji se izmjenjuju u odabiru poteza, a broj mogućih poteza za određenu poziciju u igri je unaprijed poznat.

„Minimax algorithm can be applied providing there are two players in the game who take turns at playing with a given number of possible moves for a given position in the game. The game is determined, i.e. the game does not employ dice rules of moves. The game is characterized by information transparency, i.e. each player knows the whole state of the game at each position. The leaves of the game tree present the final game positions where the outcome of the game is obvious. The aim of the minimax search in the game tree is to find the optimal strategy as a sequence of best possible moves of a given player taking into account possible moves of the other player up to a given depth.” [2, str. 2]

4.3. Praktična Izvedivost

- Jednostavnost Implementacije: Kod je jasno strukturiran i pristupačan, što olakšava razumijevanje i eventualne modifikacije.
- Grafičko Sučelje: Korištenjem pygame-a, implementirano je grafičko sučelje koje olakšava praćenje tijeka igre.

- Funkcionalnost Poteza: Potezi igrača i umjetne inteligencije su implementirani prema pravilima igre, uz primjenu minimax algoritma za dinamičko odlučivanje umjetne inteligencije.

4.4. Primjena

- Razvoj Umjetne Inteligencije: Projekt pruža primjer primjene umjetne inteligencije u jednostavnom igračem okruženju, koristeći minimax algoritam.

5. Programski kod

Riješenje igre koju igramo protiv AI u python-u napisan je ispod te se nalazi i opis implementacije.

```

1  import random
2  import sys
3  import numpy as np
4  import math
5  import pygame
6  def postavi_parametre():
7      global ljubicast, tamno_plava, Red, Stupac, duzina_niza, roza, UmjetnaInt_zeton,
8          ↪ Prazno, covjek_zeton, narancasta, tamno_zelena, bijela
9      ljubicast=(128, 0, 1285)
10     tamno_plava=(0, 0, 139)
11     Red=6
12     Stupac =7
13     duzina_niza = 4
14     roza =(255, 192, 203)
15     UmjetnaInt_zeton= 2
16     Prazno =0
17     covjek_zeton =1
18     narancasta=(255, 165, 0)
19     tamno_zelena=(0, 128, 0)
20     bijela=(255, 255, 255)

21 postavi_parametre()

22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

Isječak koda 1: Isječak koda

```

1  def procjeni_poziciju(ploca, zeton):

2      ## zgoditak center stupacumn
3      srednje_brojanje= np.count_nonzero(ploca[:, Stupac // 2]== zeton)
4      zgoditak=0
5      zgoditak += srednje_brojanje * 3

6      ##Horizontalo
7      r = 0
8      while r<Red:
9          red_polja= [int(i) for i in list(ploca[r, :])]
10         c= 0
11         while c< Stupac - 3:
12             rupa= red_polja[c:c + duzina_niza]
13             zgoditak += procjeni(rupa, zeton)
14             c += 1
15         r += 1

16     ##Vertikalno
17     c= 0
18     while c <Stupac:
19         stupac_polja =[int(i) for i in list(ploca[:, c])]
20         r =0
21         while r < Red -3:
22             rupa = stupac_polja[r:r + duzina_niza]
23             zgoditak +=procjeni(rupa, zeton)
24             r +=1
25         c +=1

26     ##dijagonalu prema gore
27     r =0
28     while r<Red -3:
29         c= 0
30         while c < Stupac - 3:
31             rupa = [ploca[r + i][c + i] for i in range(duzina_niza)]
32             zgoditak += procjeni(rupa, zeton)
33             c +=1
34         r+=1

35     ##dijagonalu prema gore
36     r=0
37     while r< Red -3:
38         c =0
39         while c< Stupac -3:
40             rupa = [ploca[r+3- i][c+i] for i in range(duzina_niza)]
41             zgoditak+= procjeni(rupa, zeton)
42             c+= 1
43         r +=1

44     return zgoditak

```

Isječak koda 2: Isječak koda

```

1 def postavi_na_vrh(ploca, stupac):
2     r=0
3     while r < Red:
4         if ploca[r][stupac] == 0:
5             return r
6         r +=1

7 def pobjednicki_potez(ploca, zeton):
8     #horizontalo
9     c =0
10    while c < Stupac - 3:
11        r = 0
12        while r < Red:
13            if ploca[r][c + 1] == zeton and ploca[r][c] == zeton:
14                if ploca[r][c + 2] == zeton and ploca[r][c + 3] == zeton:
15                    return True
16            r += 1
17        c += 1

18    #vertikalno
19    c = 0
20    while c < Stupac:
21        r = 0
22        while r < Red - 3:
23            if ploca[r][c]== zeton and ploca[r + 1][c] == zeton:
24                if ploca[r+ 2][c] ==zeton and ploca[r +3][c]== zeton:
25                    return True
26            r += 1
27        c += 1

28    #diagnoalno gore
29    c = 0
30    while c <Stupac -3:
31        r =0
32        while r < Red -3:
33            if ploca[r][c]== zeton and ploca[r +1][c +1] == zeton:
34                if ploca[r + 2][c + 2] == zeton and ploca[r +3][c +3] == zeton:
35                    return True
36            r +=1
37        c +=1

38    # diagonalno dolje
39    c =0
40    while c < Stupac -3:
41        r =3
42        while r <Red:
43            if ploca[r][c] == zeton and ploca[r - 1][c + 1] == zeton:
44                if ploca[r - 2][c + 2] ==zeton and ploca[r -3][c +3] == zeton:
45                    return True
46            r +=1
47        c +=1

48    return False

```

Isječak koda 3: Isječak koda

```
1 def terminal_node(ploca):
2     return pobjednicki_potez(ploca, covjek_zeton) or pobjednicki_potez(ploca,
    ↪ UmjetnaInt_zeton) or len(get_odobrena_lokacija(ploca)) == 0

3 def polje_postavljanja(ploca, red, stupac, zeton):
4     ploca[red][stupac] = zeton
```

Isječak koda 4: Isječak koda

```

1  def minimax(ploca, dubina, a, b, igracc):
2
3      def zadavanje_vrijednsoti(stupac, new_zgoditak):
4          nonlocal vrijednost
5          nonlocal stupacumn
6          vrijednost = new_zgoditak
7          stupacumn = stupac
8
9      terminal = terminal_node(ploca)
10     odobrena_lokacija = get_odobrena_lokacija(ploca)
11
12     if terminal or dubina == 0:
13         return (
14             (None, 1000000000) if pobjednicki_potez(ploca, UmjetnaInt_zeton)
15             else (None, -1000000000) if pobjednicki_potez(ploca, covjek_zeton)
16             else (None, procjeni_poziciju(ploca, UmjetnaInt_zeton)) if dubina == 0
17             else (None, 0)
18         )
19
20     if igracc:
21         vrijednost = -math.inf
22         stupacumn = random.choice(odobrena_lokacija)
23     else:
24         vrijednost = math.inf
25         stupacumn = random.choice(odobrena_lokacija)
26
27     for stupac in odobrena_lokacija:
28
29         red = postavi_na_vrh(ploca, stupac)
30         B = ploca.copy()
31         if igracc:
32             polje_postavljanja(B, red, stupac, UmjetnaInt_zeton)
33         else:
34             polje_postavljanja(B, red, stupac, covjek_zeton)
35         if igracc:
36             new_zgoditak = minimax(B, dubina - 1, a, b, False)[1]
37             if new_zgoditak > vrijednost:
38                 zadavanje_vrijednsoti(stupac, new_zgoditak)
39         else:
40             new_zgoditak = minimax(B, dubina - 1, a, b, True)[1]
41             if new_zgoditak < vrijednost:
42                 zadavanje_vrijednsoti(stupac, new_zgoditak)
43
44         if igracc:
45             a = max(a, vrijednost)
46         else:
47             b = min(b, vrijednost)
48
49     if a >= b:
50         break
51     return stupacumn, vrijednost

```

Isječak koda 5: Isječak koda

```

1  def provjera_polja(ploca, stupac):
2      return ploca[Red -1][stupac] == 0

3  def get_odobrena_lokacija(ploca):
4      return [stupac for stupac in range(Stupac) if provjera_polja(ploca, stupac)]

5  def dizajn_ploce(ploca):
6      c =0
7      while c < Stupac:
8          r =0
9          while r < Red:
10             pygame.draw.rect(sucelje, bijela, (c * dimenzije, r * dimenzije +
11             ↪ dimenzije, dimenzije, dimenzije))
12             pygame.draw.circle(sucelje, tamno_plava, (c * dimenzije + dimenzije //
13             ↪ 2, r * dimenzije + 3 * dimenzije // 2), RADIUS)
14             pygame.draw.rect(sucelje, tamno_plava, (c * dimenzije, r * dimenzije +
15             ↪ dimenzije, dimenzije, dimenzije), 2)
16             r +=1
17             c +=1

18      c =0

19      while c < Stupac:
20          r= 0
21          while r < Red:
22              if ploca[r][c] == covjek_zeton:
23                  pygame.draw.circle(sucelje, narancasta, (c * dimenzije + dimenzije
24                  ↪ // 2, height - r * dimenzije - dimenzije // 2), RADIUS)
25              elif ploca[r][c] == UmjetnaInt_zeton:
26                  pygame.draw.circle(sucelje, roza, (c * dimenzije + dimenzije // 2,
27                  ↪ height - r * dimenzije - dimenzije // 2), RADIUS)
28              r +=1
29              c +=1

30      pygame.display.update()

31  def postavi_sucelje(dimenzije):
32      global RADIUS ,sucelje, myfont
33      RADIUS = int(dimenzije / 2-2)
34      sucelje = pygame.display.set_mode(size)
35      myfont = pygame.font.SysFont("monospace", 45)

```

Isječak koda 6: Isječak koda

```
1 def polje_igranja():
2     ploca = np.zeros((Red, Stupac))
3     return ploca

4 def print_ploca(ploca):
5     print(np.flip(ploca, 0))

6 ploca = polje_igranja()
7 print_ploca(ploca)
8 game_over= False

9 pygame.init()
10 dimenzije =89

11 width, height = Stupac *dimenzije, (Red + 1) *dimenzije
12 size = (width, height)
13 postavi_sučelje(dimenzije)
14 covjek =0
15 UmjetnaInt =1

16 dizajn_ploce(ploca)
17 pygame.display.update()

18 turn = random.randint(covjek, UmjetnaInt)
```

Isječak koda 7: Isječak koda

```

1  while not game_over:

2      def update_game_state():
3          global turn
4          turn += 1
5          turn %= 2
6          print_ploca(ploca)
7          dizajn_ploce(ploca)

8      for event in pygame.event.get():
9          if event.type == pygame.QUIT:
10             pygame.quit()
11             sys.exit()

12         if event.type == pygame.MOUSEMOTION or (turn == covjek and event.type ==
↪ pygame.MOUSEBUTTONDOWN):
13             pygame.draw.rect(sucelje, tamno_zelena, (0, 0, width, dimenzije))
14             pygame.display.update()

15             if turn == covjek and event.type == pygame.MOUSEBUTTONDOWN:
16                 posx = event.pos[0]
17                 stupac = posx // dimenzije

18                 if provjera_polja(ploca, stupac):
19                     red = postavi_na_vrh(ploca, stupac)
20                     polje_postavljanja(ploca, red, stupac, covjek_zeton)

21                     if pobjednicki_potez(ploca, covjek_zeton):
22                         sucelje.blit(myfont.render("Pobjedio s!!!!!!!", 1,
↪ narancasta), (40, 10))
23                         game_over = True

24                     update_game_state()

25         if turn == UmjetnaInt and not game_over:
26             stupac, minimax_zgoditak = minimax(ploca, 5, -math.inf, math.inf, True)

27             if provjera_polja(ploca, stupac):
28                 red = postavi_na_vrh(ploca, stupac)
29                 polje_postavljanja(ploca, red, stupac, UmjetnaInt_zeton)

30                 if pobjednicki_potez(ploca, UmjetnaInt_zeton):
31                     sucelje.blit(myfont.render("Pobjedio je AI !!!!!!!!", 1,
↪ narancasta), (40, 10))
32                     game_over = True

33                 update_game_state()

34         if game_over:
35             pygame.time.wait(5000)

```

Isječak koda 8: Isječak koda

6. Objašnjenje koda

6.1. Bibiloteke i moduli

Na početku koda su navedeni određeni moduli i biblioteke koje koristi program.

1. `random`: Koristi se za generiranje nasumičnih brojeva.
2. `sys`: Omogućuje pristup funkcionalnostima povezanim sa sustavom.
3. `numpy (np)`: Koristi se za rad s višedimenzionalnim poljima.
4. `math`: Koristi se za korištenje matematičkih funkcija.
5. `pygame`: Biblioteka za izradu igara i grafičkih korisničkih sučelja.

6.2. Funkcije

6.2.1. Organizacija funkcija

S ciljem poboljšanja čitljivosti i održivosti implementacije igre "Connect Four", kod je logički organiziran u dvije glavne sekcije: **Funkcije Grafičkog Sučelja**, **Logika Igre** i **Parametri**.

6.2.1.1. Funkcije Grafičkog Sučelja

Funkcije Grafičkog Sučelja obuhvaćaju kod odgovoran za grafičko korisničko sučelje (GUI) igre "Connect Four". Ove funkcije rješavaju prikaz igre, igračkih žetona i interakcija s korisnikom. Glavni cilj ove sekcije je osigurati vizualno privlačno i korisnički prihvatljivo iskustvo igranja.

- `dizajn_ploce`: Crtanje vizualnog prikaza ploče igre pomoću Pygame biblioteke, uključujući boje i položaje žetona.
- `postavi_sucelje`: Postavljanje sučelja igre pomoću Pygame-a, uključujući dimenzije prozora i font koji se koristi za prikazivanje poruka tijekom igre.
- `polje_igranja`: Inicijalizacija igre, postavljanje početne ploče i drugih parametara potrebnih za igru.
- `print_ploca`: Ispis trenutnog stanja ploče u konzoli. Koristi se uglavnom u svrhu debugiranja.
- `update_game_state`: Ažuriranje stanja igre nakon svakog poteza, uključujući provjeru pobjednika, ažuriranje vizualnog prikaza i promjenu redoslijeda igrača.
- `zadavanje_vrijednosti`: Pomoćna funkcija koja postavlja vrijednost i stupac na temelju zadane vrijednosti. Koristi se unutar minimax algoritma.

6.2.1.2. Logika igre

Funkcije Igre obuhvaćaju osnovnu logiku igre "Connect Four". Ove funkcije bave se aspektima kao što su evaluacija poteza, određivanje pobjednika i izvođenje optimalnih poteza umjetne inteligencije (AI). Glavni cilj ove sekcije je implementirati osnovna pravila i mehanike igre.

- `procjeni`: Ova funkcija ocjenjuje trenutno stanje niza na ploči dodjeljujući bodove prema određenim pravilima. Na primjer, dodaju se bodovi za redove, stupce ili dijagonale s više žetona istog igrača.
- `procjeni_poziciju`: Sumira zgoditke za određenu poziciju na ploči, uzimajući u obzir horizontalne, vertikalne i dijagonalne linije. Koristi se kako bi se odredila vrijednost određene pozicije tijekom izvođenja minimax algoritma.
- `postavi_na_vrh`: Pronalazi prvo slobodno mjesto na odabranom stupcu i postavlja žeton igrača na vrh tog mjesta. Ova funkcija je ključna za implementaciju poteza igrača.
- `pobjednicki_potez`: Ova funkcija provjerava je li trenutni potez pobjednički, odnosno je li igrač postigao niz od četiri žetona u bilo kojem smjeru - vodoravno, vertikalno ili dijagonalno.
- `terminal_node`: Provjerava je li trenutno stanje ploče terminalno, odnosno je li igra gotova jer je jedan od igrača pobijedio ili nema više slobodnih mjesta za poteze.
- `polje_postavljanja`: Postavlja žeton na određeno mjesto na ploči, a koristi se nakon što se odabere stupac za potez.
- `minimax`: Ovo je implementacija minimax algoritma koji se koristi za izračun optimalnog poteza umjetne inteligencije. Algoritam traži najbolji potez koji minimizira ili maksimizira vrijednost ovisno o tome je li trenutni potez za umjetnu inteligenciju ili igrača.

6.2.1.3. Parametri

- `postavi_parametre`: Ova funkcija postavlja globalne parametre igre kao što su boje, broj redova i stupaca, duljina niza za pobjedu te druge vrijednosti koje će se koristiti tijekom izvođenja igre..

Ove jasno odvojene sekcije doprinose modularnosti i razumljivosti koda, čineći ga lakšim za održavanje, otklanjanje pogrešaka i proširenje igre "Connect Four".

6.2.1.4. Minimax algoritam

Inicijalizacija Terminalnog Čvora (`terminal_node`): Funkcija `terminal_node` provjerava je li trenutno stanje igre terminalno, odnosno je li došlo do pobjede nekog igrača ili je

ploča popunjena. Vraća `True` ako je igra završena, inače `False`.

Odobrena Lokacija (`get_odobrena_lokacija`): Funkcija `get_odobrena_lokacija` vraća listu stupaca na kojima se može postaviti žeton, tj. stupce koji nisu potpuno popunjeni.

Procjena Pozicije (`procjeni_poziciju`): Funkcija `procjeni_poziciju` ocjenjuje trenutno stanje igre za određenog igrača. Prati zgoditke na horizontali, vertikali i dijagonalama, dodjeljujući ocjene ovisno o prisutnosti određenih uzoraka.

Postavljanje Žetona na Vrh (`postavi_na_vrh`): Funkcija `postavi_na_vrh` pronalazi prvi slobodan redak u odabranom stupcu i postavlja žeton tog igrača.

Pobjednički Potez (`pobjednicki_potez`): Funkcija `pobjednicki_potez` provjerava je li trenutno postavljeni žeton rezultirao pobjedom (četiri u nizu) na horizontali, vertikali ili dijagonalama.

Zadavanje Vrijednosti (`zadavanje_vrijednosti`): Unutarnja funkcija `zadavanje_vrijednosti` služi za postavljanje nove vrijednosti i odgovarajućeg stupca u slučaju da se pronade bolji potez tijekom pretrage.

Minimax Algoritam (`minimax`): Glavna funkcija `minimax` implementira minimax algoritam. Rekurzivno istražuje stablo igre, odabire najbolji potez za umjetnu inteligenciju i vraća odgovarajući stupac i pripadajuću vrijednost.

6.2.2. Glavna Petlja Igre

Glavna petlja igre se izvršava sve dok igra nije završena. Unutar petlje se obrađuju događaji, ažurira stanje igre i provjeravaju potezi igrača i umjetne inteligencije.

Provjera Događaja:

- Ako korisnik zatvori prozor, igra se završava.
- Ako je red na potezu čovjek i klikne na ekran, igra reagira na odabir stupca i postavlja žeton.

Potez Umjetne Inteligencije (AI):

- Ako je red na potezu umjetna inteligencija, koristi se algoritam minimax za odabir najboljeg poteza.

Ažuriranje Stanja Igre:

- Ažurira se red na potezu, prikazuje trenutno stanje ploče i osvježava sučelje igre.

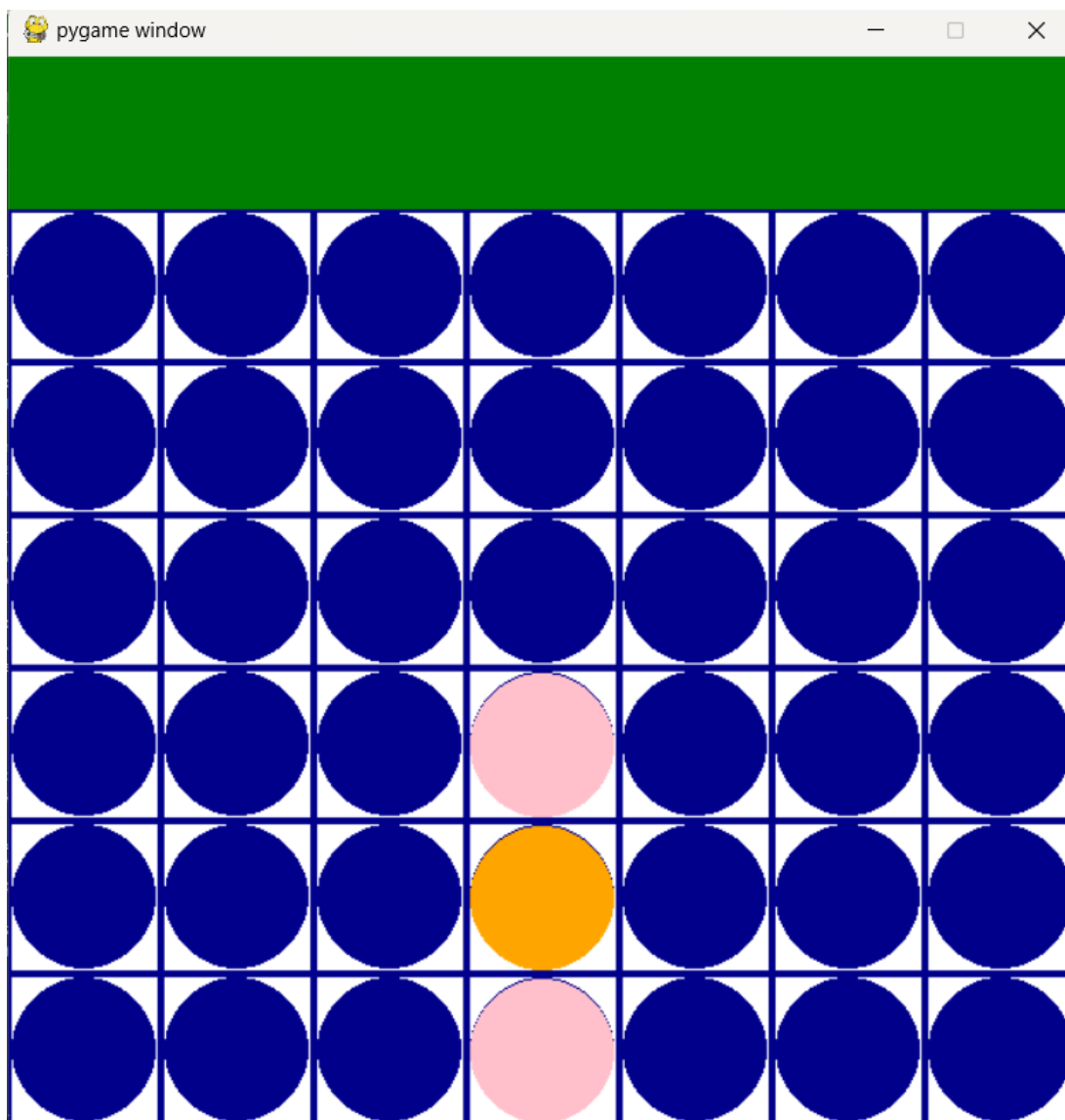
Završetak Igre:

- Ako je igra završena, program čeka 3 sekunde prije zatvaranja prozora pygame.

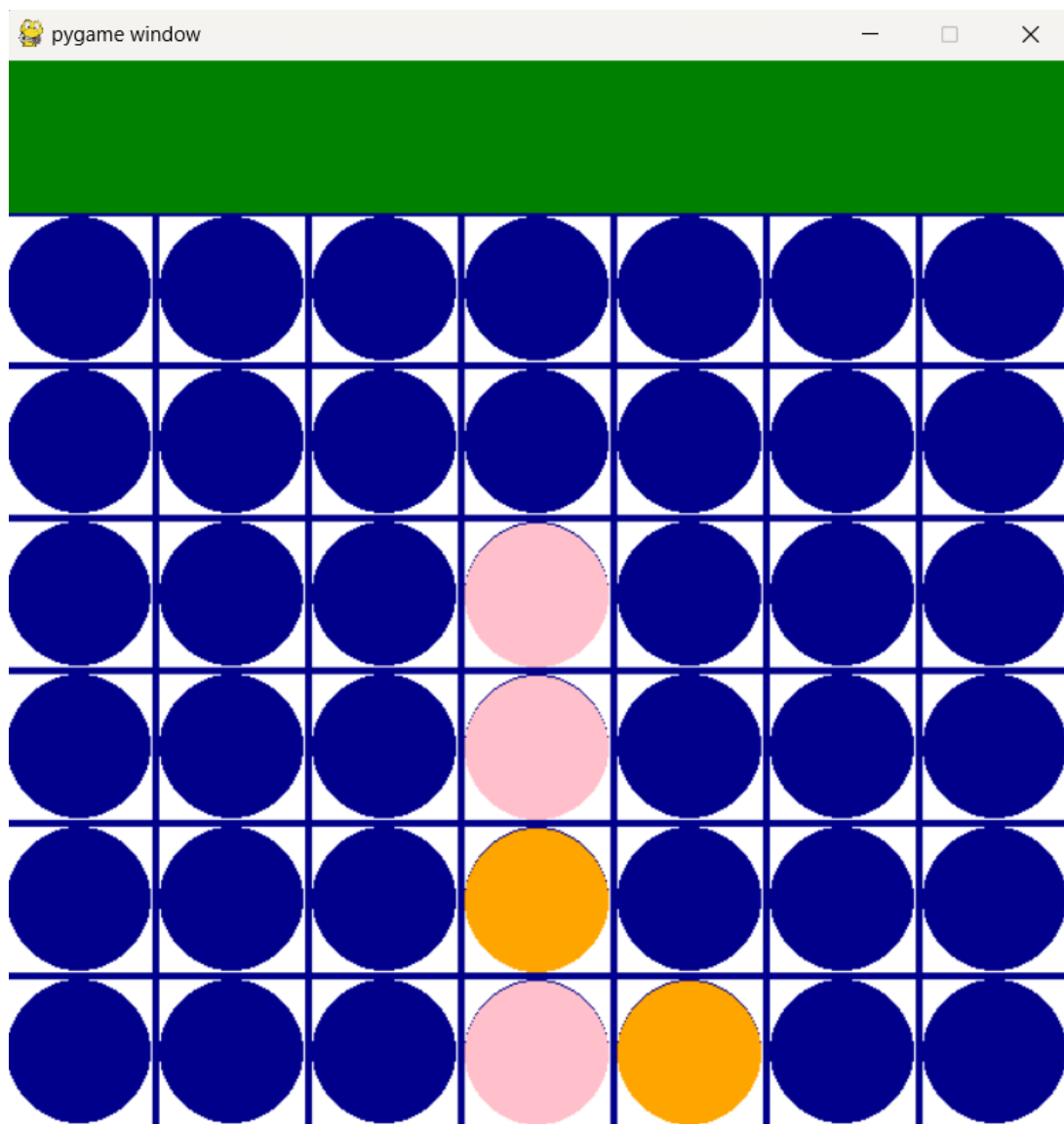
Ova petlja osigurava nesmetan tijek igre, interakciju s korisnikom i poteze umjetne inteligencije.

7. Prikaz rada aplikacije

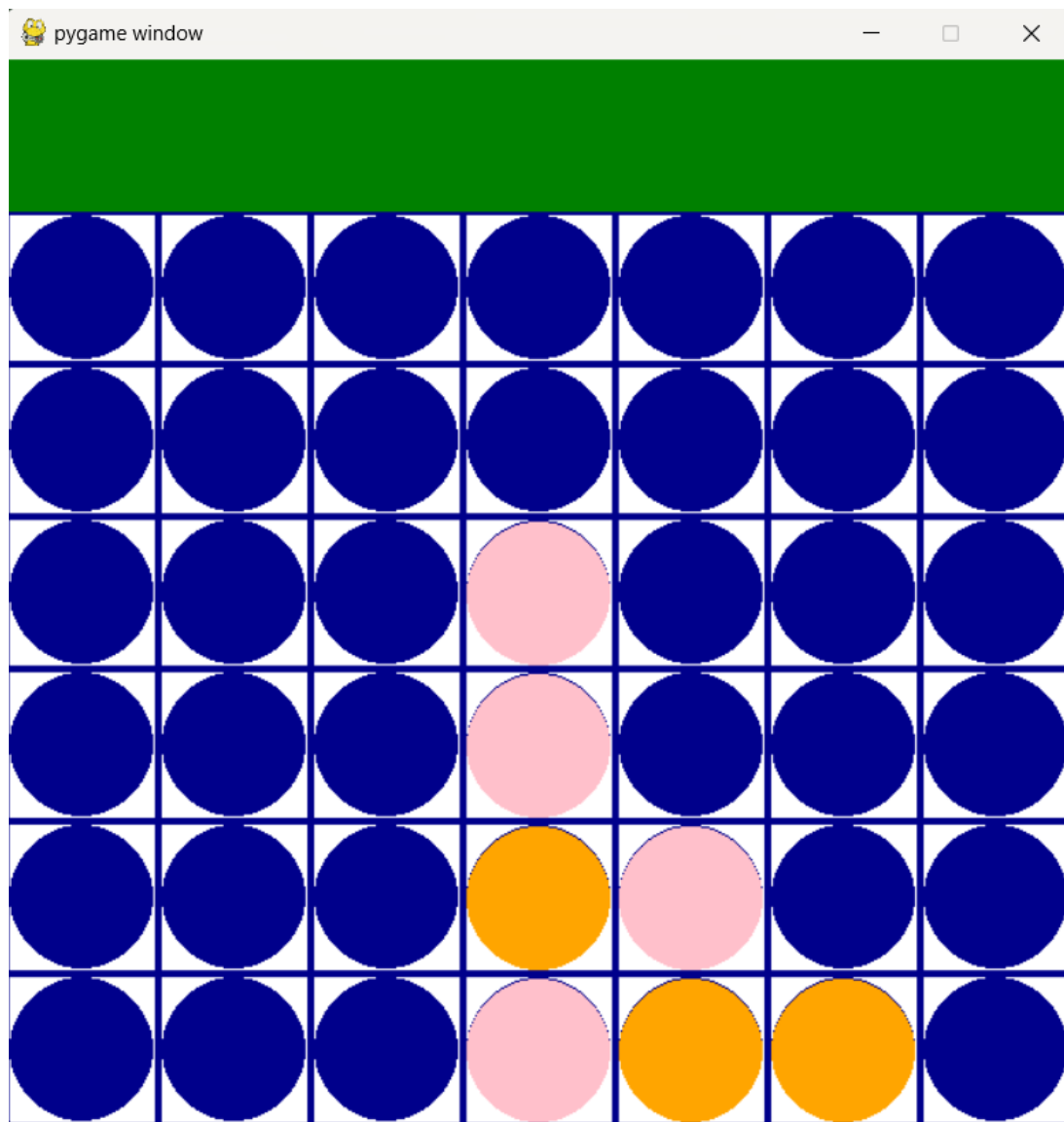
U ovom dijelu rada ćemo prikazati u par slika rad aplikacije.



Slika 1: Prvi potez je imao igrač te zatim odmah AI radi svoj potez



Slika 2: drugi potez je napravio igrač te zatim odmah AI radi svoj potez



Slika 3: Nakon 3 poteza igrača AI radi svoj potez



Slika 4: Nakon par poteza igrača AI radi svoj potez i pobjeđuje

8. Zaključak

U ovom istraživačkom radu istražena je implementacija Connect Four igre s naglaskom na upotrebu umjetne inteligencije. Kroz analizu algoritma minimax, razvijena je simulacija igre koristeći programski jezik Python, uz primjenu pygame biblioteke za grafičko sučelje. Glavne teze rada obuhvaćaju jasnu strukturu koda, praktičnu primjenu minimax algoritma te implementaciju grafičkog sučelja radi poboljšanja korisničkog iskustva.

Rad se fokusira na ključna pravila i logiku igre Connect Four, gdje igrači ciljaju postavljanje četiri svoja žetona u nizu, bilo vodoravno, okomito ili dijagonalno. Prikazane su strategije igre, uključujući potrebu za taktičkim razmišljanjem, blokiranjem protivničkih poteza te razvojem dugoročne strategije.

Kroz kritički osvrt na implementaciju u Pythonu, rad prepoznaje važnost korištenja algoritma minimax s Alpha-Beta podrezivanjem za odlučivanje poteza umjetne inteligencije. Navedene su korištene komponente poput NumPy-a i Pygame-a te istaknuta je praktična izvedivost projekta, uz jednostavnu implementaciju, jasnu strukturu koda te grafičko sučelje koje olakšava praćenje tijeka igre.

Zaključci ukazuju na uspješnost implementacije i pružaju uvid u primjenu umjetne inteligencije u kontekstu jednostavne igre poput Connect Four. Rad se smatra korisnim resursom za razumijevanje umjetne inteligencije u igrama te može poslužiti kao temelj za daljnje istraživanje i edukaciju u području AI.

Popis literature

- [1] L. V. Allis, „A Knowledge-Based Approach of Connect-Four,” *J. Int. Comput. Games Assoc.*, sv. 11, br. 4, str. 165, 1988.
- [2] P. Borovska i M. Lazarova, „Efficiency of parallel minimax algorithm for game tree search,” *Proceedings of the 2007 international conference on Computer systems and technologies*, 2007., str. 1–6.
- [3] pygame. „pygame documentation.” Pristupljeno: 4.1.2024. (2024.), adresa: <https://www.pygame.org/docs/>.
- [4] A. L. „Minimax Algorithm in Game Theory | Set 1 (Introduction).” Pristupljeno: 4.1.2024. (2022.), adresa: <https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/>.
- [5] TheSharperDev. „Implementing Minimax Tree Search.” Pristupljeno: 3.1.2024. (), adresa: <https://thesharperdev.com/implementing-minimax-tree-search/>.
- [6] techwithtim. „Basic Movement i Key Presses.” Pristupljeno: 28.12.2023. (), adresa: <https://www.techwithtim.net/tutorials/game-development-with-python/pygame-tutorial/pygame-tutorial-movement>.
- [7] ChatGPT. „Large language model.” Pristupljeno: 3.12.2023. (), adresa: <https://chat.openai.com/chat>.

Popis slika

1. Prvi potez je imao igrač te zatim odmah AI radi svoj potez 20
2. drugi potez je napravio igrač te zatim odmah AI radi svoj potez 21
3. Nakon 3 poteza igrača AI radi svoj potez 22
4. Nakon par poteza igrača AI radi svoj potez i pobjeđuje 23

Popis isječaka koda

1.	Isječak koda	8
2.	Isječak koda	9
3.	Isječak koda	10
4.	Isječak koda	11
5.	Isječak koda	12
6.	Isječak koda	13
7.	Isječak koda	14
8.	Isječak koda	15