Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko

David Bombek

IGRA REVERSI NA TELEFONSKEM APARATU CISCO IP

Diplomska naloga

Maribor, september 2008

UNIVERZA V MARIBORU

FAKULTETA ZA ELEKTROTEHNIKO,
RAČUNALNIŠTVO IN INFORMATIKO
2000 Maribor, Smetanova ul. 17

Diplomska naloga visokošolskega strokovnega
študijskega programa

IGRA REVERSI NA TELEFONSKEM APARATU CISCO IP

Študent: David BOMBEK
Študijski program: visokošolski strokovni, Računalništvo in informatika
Smer: Programska oprema

Mentor: izred. prof. dr. Janez BREST

Maribor, september 2008

ZAHVALA

Zahvaljujem se mentorju izred. prof. dr. Janezu Brestu in asistentu mag. Alešu Zamudi za strokovno pomoč in usmerjanje pri opravljanju diplomske naloge.

Posebna zahvala velja očetu Mirku, ki me je med študijem vzpodbujal. Zahvala gre tudi punci Mateji, ki mi je nudila moralno podporo.

IGRA REVERSI NA TELEFONSKEM APARATU CISCO IP

Ključne besede: **igra reversi, aplikacija, umetna inteligenca, telefonski aparat Cisco IP, spletni strežnik, podatkovni strežnik**

UDK: 004.72:621.396(043.2)

Povzetek

V diplomski nalogi smo pokazali, da je možno igranje igre reversi na IP telefonskem aparatu. Aplikacijo smo izdelali s pomočjo različnih tehnologij. Pomembno vlogo pri našem programu ima tudi razširljivi označevalni jezik XML. Igra reversi spada med igre s popolno informacijo. V diplomski nalogi smo opisali strategije za učinkovito igranje igre reversi in iskalne algoritme.

Osrednji del diplomske naloge sestavljajo trije segmenti naše aplikacije ReversiOnCisco. Ti so Cisco IP telefonski aparat, spletni in podatkovni strežnik. Med drugim predstavlja pomembno vlogo spletnega strežnika vizualizacija igre. Sledi opis delovanja motorja programa, ki poganja igro reversi in podroben opis metode shranjevanja in pridobivanja statističnih podatkov. Zadnji del obsega rezultate in možnosti za nadaljnje raziskave.

THE REVERSI GAME ON CISCO IP PHONE

Keywords: **reversi game, application, artificial intelligence, Cisco IP Phone, web server, database server**

UDK: 004.72:621.396(043.2)

ABSTRACT

We have demonstrated that, with the help of various technologies, it is possible to play the reversi game on an IP telephone set. On the other hand, XML Extensible Markup Languages played an important role in our program as well. The reversi game belongs to a group of games that provide complete information. In the diploma work, we have described the strategies of how to play reversi successfully and search algorithms.

The central part of the diploma work is composed of three segments of our program ReversiOnCisco. These are Cisco IP telephone set, web and database server. A relevant role of the web server is, among other things, the visualization of the game. The next stage will be the description of a game engine and a detailed description of the method of saving and acquiring statistical data. The final part is wrapped up by our results and possibilities for future research.

Seznam uporabljenih simbolov

x, y	koordinate
p	poteza igralca
s	stanje pozicije igralca na igralni plošči
r	igralec z rdečimi žetoni
g	igralec z zelenimi žetoni
w	širina igralne plošče
h	višina igralne plošče
$eval$	ocenitvena funkcija
$-\infty$	ocenitev, pri kateri igralec min izgubi
∞	ocenitev, pri kateri igralec maks zmaga
d	globina iskalnega drevesa
b	vejitveni faktor iskalnega drevesa
n	končno število vozlišč iskalnega drevesa
O	časovna zahtevnost algoritma
α	najboljša vrednost igralca maks
β	najboljša vrednost igralca min

Seznam uporabljenih kratic

IP	Internet Protocol
XML	Extensible Markup Language
PBX	Private Branch eXchange
VoIP	Voice Over Internet Protocol
HTTP	Hypertext Transfer Protocol
URL	Uniform Resource Locator
ASCII	American Standard Code for Information Interchange
ISO	International Organization for Standardization
LCD	Liquid Crystal Display
PNG	Portable Network Graphics
ID	Identifikator

Slike

Seznam algoritmov

Kazalo

Poglavje 1

Uvod

IP telefonija je dandanes najbolj tehnološko napredna rešitev telefonije. Je usmerjanje glasovnih pogovorov preko interneta ali skozi katerokoli drugo omrežje. Lahko jo uporabljamo na istem omrežju kot računalnike, kar je prednost in razlika med klasično telefonijo. To pomeni, da lahko med samim telefoniranjem uporabljamo podatke, ki so dostopni v omrežju. Tako lahko s telefonskim aparatom dostopamo tudi do posebej pripravljenih spletnih vsebin tako imenovanih servisov. Vse to pa so nam omogočili različni proizvajalci IP telefonskih aparatov, kot so: 3Com, Aastra, Alcatel, Mitel, Polycom itd. Ti se nenehno ukvarjajo z razvojem inovativnih IP telefonskih aparatov. Večina od naštetih ponudnikov v sklopu IP telefonije ne ponuja le telefonov, temveč tudi stikala, usmerjevalnike ter druge proizvode znotraj sklopa. Med največjimi je Cisco Systems, ki nam ponuja rešitev telefonskega aparata z zaslonom občutljivim na dotik (angl. touch screen), kar nam bo pri nadaljnjih raziskavah v pomoč.

Ker so nas dandanes že množično poplavili mobilni telefoni, ki nudijo veliko možnosti zabave in razvedrila, smo tudi mi poskušali dobiti idejo o tem, kako ustvariti IP telefon, ki bo za posameznika koristen in zabaven hkrati. Tako smo prišli na idejo za izdelavo igre, ki bi se lahko igrala preko zaslona občutljivega na dotik. Idejo smo realizirali in tako smo ustvarili igro reversi. Reversi je igra dveh igralcev s popolno informacijo in ničelno vsoto. Igralcu so v vsakem trenutku znane vse možne poteze, ki jih ima nasprotnik. Med tovrstne igre spadajo šah, dama, go, reversi itd. Med njih ne štejemo igre s kartami.

Diplomsko delo vsebuje devet poglavij. Po uvodu v drugem poglavju se v grobem seznanimo z IP telefonskim aparatom, protokolom, ki je potreben za prenos, govora ter z etiketami (TAG) razširljivega označevalnega jezika (XML), ki so bistvenega pomena našega dela. Nato se seznanimo z igrami s popolno in nepopolno informacijo. Med igre s popolno informacijo in ničelno vsoto sodi tudi igra reversi. Opišemo strategije učinkovite igre reversi, iskalne algoritme ter prikažemo njihovo

delovanje. V tretjem poglavju prikažemo strukturo našega programa ReversiOn-Cisco in posamezne segmente potrebne za realizacijo le te. V četrtem poglavju utemeljimo, zakaj smo izbrali Cisco IP telefonski aparat in zakaj ta razume ukaze spletnega strežnika. Peto poglavje je namenjeno zgolj spletnemu strežniku. V njem opišemo posamezne spletne strani in njihove naloge. Prikažemo tudi podatkovni tok med njimi in dinamičnima knjižnicama. Opišemo postopek priprave podatkov ter potrebne postopke in elemente za vizualizacijo igre na telefonskem aparatu. Poleg postopkov opišemo generator dinamičnih slik, animiran prikaz izvajanja potez ter definiranje območij občutljivih na dotik. Naslednje, šesto poglavje je namenjeno motorju. Motor predstavlja program, ki vsebuje umetno inteligenco za igranje igre. Tukaj opisujemo samo delovanje igre reversi, način izvajanja potez, iskanje možnih smeri zavzemanja s pomočjo matrične šablone ter umetno inteligenco in kombinirano ocenitveno funkcijo. Sedmo poglavje je zadnje izmed treh segmentov, ki zaznamujejo sestavne dele našega programa. Predstavlja podatkovni strežnik kamor po končani igri shranjujemo oz. ob želji po pregledu statističnih podatkov le te prikažemo. Prikažemo povezavo med spletnim in podatkovnim strežnikom, strukturo podatkovne baze ter shranjene procedure za obdelavo statističnih podatkov. Na koncu podrobno opišemo metodi shranjevanja in pridobivanja statističnih podatkov. V osmem poglavju podamo rezultate odigrane igre in komentiramo posamezne poteze. Deveto poglavje poda zaključek k diplomskemu delu in možnosti za nadaljnje raziskave. Na koncu je navedena še literatura.

Poglavje 2

Pregled IP telefonskih aparatov in iger s popolno informacijo

2.1 IP telefonski aparat

IP telefonski aparat je naprava priključena na omrežje, ki lahko izmenjuje informacije z drugimi omrežnimi napravami. Osnovna funkcija je prenos zvoka in govora, vključuje pa vse funkcije standardnih telefonskih sistemov PBX (angl. Private Branch eXchange) in še mnogo več. Za prenos glasu ali zvoka uporablja VoIP (angl. Voice Over Internet Protocol), ki spreminja zvok v IP-pakete, ti paketi pa potujejo preko IP-infrastrukture, kar ponuja širok spekter telefonskih storitev ter aplikacij.

IP telefonski aparati so lahko v obliki samostojne naprave (telefon) ali kot aplikacija na računalniku t.i. SoftPhone. Poznamo pa več vrst SoftPhonov in eden izmed njih je tudi Cisco IP SoftPhone, ki je predhodnik današnjega Cisco IP Communicator-ja, ki je izšel leta 2004. IP Communicator predstavlja aplikativno obliko IP telefonskega aparata.

Prednosti Cisco IP Communicator-ja so:

- podpora za razširljivi označevalni jezik (XML),
- povpraševanje po spletnih vsebinah (HTTP),
- samodejno posodabljanje programske opreme,
- prikaz ozadja na zaslonu,
- datum in čas,

- servisi
- itd.

Med zgoraj naštetimi je tudi Cisco Systems IP Phone Services, ki omogoča prikaz spletnih vsebin na Cisco IP telefonskem aparatu. Da se spletna vsebina pravilno prikaže, mora vsebovati posebej pripravljene objekte v obliki razširljivega označevalnega jezika [1].

Cisco za razliko od drugih proizvajalcev ponuja obsežnejšo rešitev. Rešitev pri tem daje poudarek na novih storitvah, ki jih omogoča združevanje omrežij.

2.1.1 Razširljivi označevalni jezik *XML*

XML ali Extensible Markup Language je jezik, ki je zelo uporaben za računalniške komunikacije in je zelo preprosto zgrajen. Je format za opisovanje strukturiranih podatkov. Kljub svoji preprostosti omogoča prenos podatkov in njihovo izmenjavo med več omrežji. Je razširljiv jezik, katerega se da tudi razširiti tako, da si sami izmislimo imena etiket (TAG). Pogosto ga srečujemo kadar brskamo po internetu [2].

Sledi pregled Cisco IP Phone XML [1] objektov in njihov opis:

- CiscoIPPhoneMenu,
- CiscoIPPhoneText,
- CiscoIPPhoneInput,
- CiscoIPPhoneDirectory,
- CiscoIPPhoneImage,
- CiscoIPPhoneImageFile,
- CiscoIPPhoneGraphicMenu,
- CiscoIPPhoneGraphicFileMenu,
- CiscoIPPhoneIconMenu,
- CiscoIPPhoneIconFileMenu,
- CiscoIPPhoneStatus,
- CiscoIPPhoneStatusFile,

- CiscoIPPhoneExecute,
- CiscoIPPhoneResponse in
- CiscoIPPhoneError.

CiscoIPPhoneMenu

CiscoIPPhoneMenu ima funkcijo, da na zaslonu prikaže meni. Meni je sestavljen iz naziva, pozivne vrstice ter menijskih vrstic. Menijska vrstica sestoji iz imena in pripadajočega naslova URL (angl. Uniform Resource Locator), kamor smo ob izbiri menijske vrstice preusmerjeni.

Naziv menijske vrstice lahko sestavlja maksimalno 64 znakov. Omejeni smo na 100 menijskih vrstic.

CiscoIPPhoneText

Objekt CiscoIPPhoneText prikazuje besedilo v obliki 8-bitnih ASCII znakov. To pomeni, da ne podpira šumnikov, temveč podpira ISO 8859-1 (Latin 1) in ShiftJIS nabor znakov.

CiscoIPPhoneInput

Objekt CiscoIPPhoneInput na zaslonu ustvari vnosno polje, kjer uporabnik lahko vnaša predefinirane nabore znakov.

CiscoIPPhoneDirectory

CiscoIPPhoneDirectory oz. telefonski imenik omogoča vnos naziva ter telefonske številke. Naenkrat lahko izpišemo le 32 zapisov.

CiscoIPPhoneImage

Objekt CiscoIPPhoneImage prikaže bitno sliko velikosti 133×65 pikslov. Telefonski aparat uporablja LCD zaslon, pri čemer je potrebno paleto obrniti.

CiscoIPPhoneImageFile

Novejše različice IP telefonskih aparatov omogočajo zaslone z večjo barvno globino. Tako ima telefonski aparat zaslon velikosti 298×168 pikslov in 12-bitno barvno globino. Možen je prikaz slike formata PNG.

CiscoIPPhoneGraphicMenu

Ta objekt služi enakemu namenu kot CiscoIPPhoneText. Uporablja se, če gre za kompleksne zapise nazivov, katere s sliko boljše ponazorimo.

CiscoIPPhoneGraphicFileMenu

Nekateri izmed najnovejših IP aparatov imajo vgrajen LCD zaslon občutljiv na dotik. Za uporabo tega zaslona potrebujemo objekt CiscoIPPhoneGraphicFileMenu. Sliki nastavimo lokacijo na x in y-osi ter območje delovanja.

CiscoIPPhoneIconMenu

CiscoIPPhoneIconMenu ima enako funkcijo kot CiscoIPPhoneMenu z majhno razliko, da temu pred naziv lahko dodamo sličice oz. ikone velikosti 16×10 pikslov.

CiscoIPPhoneIconFileMenu

Tudi ta objekt je podoben CiscoIPPhoneMenu z razliko, da lahko temu pred naziv dodamo barvne sličice formata PNG.

CiscoIPPhoneStatus

Objekt CiscoIPPhoneStatus se razlikuje glede na ostale tako, da ga je možno prikazati med klicanjem. Običajno se uporablja za prikaz stanja aplikacij.

CiscoIPPhoneStatusFile

Temelji na objektu CiscoIPPhoneStatus z razliko, da prikaže PNG slike, velikosti 262×50 pikslov.

CiscoIPPhoneExecute

Objekt CiscoIPPhoneExecute se razlikuje od ostalih. Ni ga možno prikazati. Namen objekta je razreševanje večkratnega povpraševanja namenjenega aparatu.

CiscoIPPhoneResponse

Objekt CiscoIPPhoneResponse priskrbi sporočila in informacije izhajajoče iz objekta CiscoIPPhoneExecute.

CiscoIPPhoneError

Slednji upravlja z možnimi stanji napak, ki jih vrne telefonski aparat:

- napaka 1 - napaka pri razčlenjevanju objekta CiscoIPPhoneExecute,
- napaka 2 - napaka pri sestavljanju objekta CiscoIPPhoneResponse,
- napaka 3 - notranja napaka datoteke in
- napaka 4 - napaka verodostojnosti.

2.2 Igre s popolno in igre z nepopolno informacijo

Igra je razvedrilna dejavnost, ki že od nekdaj navdušuje človeka. Poznamo več vrst iger in njihove lastnosti: simetričnost, igre z ničelno vsoto, s popolno ali nepopolno informacijo, statične in zaporedne oziroma dinamične itd. Poznamo tudi igre, ki vključujejo enega ali več igralcev in imajo nek cilj, ki ga igralci poskušajo doseči ter pravila, ki opredelijo, kako igralci igrajo.

Igranje iger spada tudi v področje umetne inteligence, ki je interdisciplinarnega značaja. Omogoča nam igranje, pri katerih je lahko računalnik nasprotnik človeškemu igralcu. V grobem lahko igre dveh igralcev delimo glede na informacijo o igri, te so:

- popolna informacija in
- nepopolna informacija.

Igre s popolno informacijo imenujemo tako, ker so v poljubnem trenutku igranja znane vse možnosti, ki jih ima igralec oz. nasprotnik. Med njih štejemo igre kot so šah, dama, go, reversi itd. Kot primer lahko vzamemo igro reversi. Igra poteka med dvema igralcema, igralcem z rdečimi in igralcem z zelenimi žetoni. Oba igralca imata pregled nad celotno igro in vsemi možnimi potezami. Zato pravimo, da je to igra s popolno informacijo. Poleg tega lahko vsak igralčev element (polje igralne plošče) označimo z numerično vrednostjo. Torej bo imel igralec številka ena pozitivno numerično vrednost, igralec številka dve pa negativno numerično vrednost. V pravilih igre je določeno, da poteka igra z izmenjevanjem izbir potez med njima. Dodatna lastnost igre je ničelna vsota, katere izraz izhaja iz teorije iger [11]. Kot smo dejali, lahko vsak igralčev element predstavimo kot numerično vrednost. Legalne poteze igralcev ob mešanih strategijah igranja prinašajo prednost oziroma slabost za igralca, ki je na potezi. Torej lahko stanje pozicije s ocenimo za oba igralca. Igralca rdečih žetonov lahko označimo z r, igralca zelenih žetonov z g, ocenitveno funkcijo pa z $eval$.

Matematični model ničelne vsote lahko zapišemo:

$$eval_r(s) + eval_g(s) = 0. \tag{2.1}$$

Matematični model 2.1 pravi, da je vsota ovrednotenih igralčevih elementov enaka nič.

Za nasprotnika oziroma umetno inteligenco so strategije za igranje igre reversi lahko naučene brez strokovnega predznanja. Možno je učenje z uporabo evolucijskih algoritmov na osnovi nevronskih mrež in ocenitvena funkcija, kombinirana s standardnim minimaks iskalnim algoritmom [5]. Torej lahko igranje igre reversi poteka

brez predhodnega človeškega znanja, saj ga nadomešča povratno učenje. Dokazano je, da igranje igre reversi s povratnim učenjem deluje boljše, kot igranje z uporabo osnovne strategije [6]. Primerno opremljeno učenje s koevolucijo *"co-evolutionary learning"* se lahko nauči boljših igralnih strategij kot pa učenje z vmesnimi razlikami *"temporal difference learning"*, čeprav se sledne uči mnogo hitreje.

Pri učenju s koevolucijo je bistvenega pomena za doseganje dobrih rezultatov, da so starši in otroci iskalnega drevesa povprečno obteženi. Z uporabo te metode je bil razvit visoko kvaliteten števec obteževanja žetonov in prikazan pomen številnih standardnih hevrističnih uteži [7]. Poznamo posebno ogrodje za pol avtomatično razvijanje ocenjevalnih funkcij. Postopek poteka tako, da za iskanje prostih mest najde nove poteze ob oceni izvedljivih poti. Poleg tega poteka iskanje in izvajanje učnih potez za napredno izbiranje in prilagajanje teže v velikih linearnih sistemih [8]. Poznana je tudi aplikacija treh znanih statističnih metod na področju igralnih iskalnih dreves. Ta za ocenitveno funkcijo uporablja veliko število dobrih reversi pozicij in uteži ne glede na igralno fazo, kar pomeni, da ocenjuje po pomenu logične regresije [9].

Za igre z nepopolno informacijo je značilno, da v poljubnem danem trenutku niso znane vse možnosti, ki jih ima nasprotnik. Med tovrstne spada večina iger s kartami [12]. V nadaljevanju bomo govorili le o igrah z dvema igralcema in popolno informacijo.

Računalniške igre s popolno informacijo zahtevajo kompleksne igralne algoritme, ki ocenijo možne igralne poteze in se odločijo o nadaljnji potezi. Računalnik razvije veliko več potez vnaprej kot človek in z večanjem zmogljivosti računalnika se večajo tudi razlike v zmogljivosti med računalnikom in človekom. Ta prepad si je človek ustvaril sam in je razočaran, ko izgublja v igrah proti računalniku, za kar si je v bistvu kriv sam. Vendar se tehnologija vedno bolj razvija in s tem tudi računalništvo in tako je tudi pravilno. Igre bi bile nezanimive, če bi proti računalniku vedno zmagal človek. Obstaja še mnogo iger, ki čakajo, da bodo raziskane iz evolucijske perspektive in še mnogo več je čakajočih, da jih izumijo. Raziskovalno področje računalniških iger ima dolgo zgodovino in sega v korenine računalniške znanosti [4].

2.2.1 Reversi

O nastanku igre reversi oz. othello (reversibel (lat.) = povraten) obstaja več različnih zgodb. Prva trdi, da je avtor igre Goro Hasegawa leta 1971 na japonskem izdal igro reversi. Druga pa, da je igro reversi leta 1888 v Angliji izumil Lewis Waterman, ki naj bi jo videl že pri John W. Molletu. Med njima sta bili samo dve razliki. Prva je bila, da je pri igri reversi bilo potrebno tudi prve štiri žetone odigrati, druga pa,

da je vsak igralec dobil točno določeno število žetonov. V bodoče bomo obravnavali igro reversi z vnaprej pripravljeno ploščo. Torej so prvi štirje žetoni že pripravljeni.

Leta 1980 je program za igranje igre reversi po imenu The Moor premagal svetovnega prvaka. Leto 1997 je bilo zaznamovano kot leto igre othello. Tega leta je potekal boj v igranju igre othello med svetovnim prvakom Takeshi Murakami in računalniškim programom Logistello, katerega avtor je Michael Buro. Logistello je na osnovi strategije GLEM premagal svetovnega prvaka s 6:0. S tem rezultatom je Buro dokazal, da dandanes človek več ni kos računalniškemu programu [10].

Današnje tehnike dobrega računalniškega igralca tovrstne igre so plod predhodnega znanja dobrih strategij in uporaba naprednih iskalnih algoritmov za pridobivanje možnih potez.

Kljub temu, da je računalniški program premagal svetovnega prvaka, še zdaleč ne pomeni, da je ta inteligentne narave. Sicer so hitri in imajo ogromne zaloge pomnilnika, a vendar uporabljenih strategij niso razvili sami. Uporabljajo kombinacijo človeškega znanja ter računalniške zmogljivosti.

Leta 2005 sta Michiel Vuurboom in Bas Jacob pričela z raziskavo na temo igranje igre reversi. Začela sta brez predhodnega znanja in temeljila na nevronskih mrežah in povratnem učenju. Izhajajoč iz rezultatov je Michiel Vuurboom v znanstvenem delu na temo Othello in uporaba nevrorazvojnih tehnik primerjal zmožnosti učenja in hitrosti. Primerjal je tri tehnike SANE, ESP ter NEAT in prikazal, da je NEAT (Neuro-Evolution of Augmenting Topologies) dosegla najboljše rezultate, vendar je to le majhen del problematike v igri othello [13].

2.2.2 Strategije

Igro reversi lahko razdelimo v tri faze. Prva faza se imenuje faza odprtja (angl. opening game), druga faza je osrednja faza (angl. mid game), na koncu sledi še končna faza (angl. end game). Slednje niso ločene s točno definirano mejo, a vendar vsaka izmed njih zahteva drugačno strategijo [14]. Za primerjavo omenimo, da računalniški program Logistello pozna približno dvajset faz. Običajno ima igra šestdeset potez. Včasih manj, a nikoli več. Faza odprtja predstavlja začetno igralno ploščo z vsemi štirimi začetnimi žetoni. Vsak igralec ima po dva žetona svoje barve. Ta faza se lahko odigra ob uporabi seznama ugodnih položajev. Običajno predstavlja ta faza četrt celotne igre. V končni fazi je možnost izbire primerne strateške poteze za nadaljnjo igro manjša, saj je večina polj že zasedenih. Igralec skuša zavzeti zadnja prosta mesta in zbrati toliko žetonov, da bo zmaga njegova. Tudi ta faza predstavlja približno tolikšen del igre. Preostali del igre je namenjen osrednji fazi. V zadnjih letih je osrednja faza precej pridobila na pomembnosti, saj je ta del precej zahteven

za računalniški program. Pri tej fazi je izvedba dobre poteze precej pomembna, saj lahko le ena nepravilna poteza nasprotniku prinese potrebno prednost in s tem pot do zmage. Ena izmed preprostejših strategij je zavzemanje maksimalnega števila žetonov.

Osrednja faza ima dve bistveni strategiji igre. To sta položajna strategija (angl. positional strategy) in premičnostna strategija (angl. mobility strategy). Položajna strategija je preprostejša kot premičnostna strategija, vendar ji je slednja podrejena. Namen igralca, ki uporablja položajno strategijo, je zavzeti čim več žetonov v najkrajšem možnem času. Da bi to dosegel, mora svoje žetone postaviti na rob plošče in hkrati obkrožiti čim več nasprotnikovih. Igralec s tako strategijo vedno igra v smer oglišč, saj mu žetonov na ogliščih nasprotnik ne more več zavzeti. Raziskave so pokazale, da pri igranju igre reversi proti računski inteligenci le ta zmaga v 85% vseh iger. Igranje proti naključnim nasprotnikom so izvedli 10.000-krat. Da bi dobili realistično povprečje so postopek ponovili 10-krat [13]. Nekoliko zahtevnejša strategija je premičnostna strategija. Pri tej strategiji igralec nadzoruje središče igralne plošče in nasprotnika sili v obkrožitev lastnih žetonov. S tem igralec prisili nasprotnika, da opusti svoje strateške poteze, dokler se sam ne odloči za napad. Zato je nasprotnik v končni fazi igre prisiljen predati oglišča ter robove. Cilj te strategije v osrednji fazi igre je imeti čim manj svojih žetonov in čim več možnih potez. To za nasprotnika naj ne bi veljalo. Billman in Shaman sta pokazala, da je strategija premičnosti precej zahtevna [15].

2.2.3 Iskalni algoritmi

Pri tovrstnih igrah igralca igrata izmenoma. Vsak opravi po eno potezo naenkrat. Ob vsaki opravljeni potezi se stanje igre spremeni in s tem tudi možne nadaljnje poteze igranja. Igre s popolno informacijo lahko predstavimo v obliki drevesa. Koren ali vrh predstavlja trenutno stanje igre, ki je na potezi. Ostala vozlišča predstavljajo naslednje možne položaje glede na svojega očeta. Sinovi vozlišča predstavljajo položaje igre, do katerih pridemo z izvedbo le ene poteze. Povezave med očetom in sinovi oz. plastmi imenujemo poteze. Položaje na dnu drevesa imenujemo listi. Plasti (angl. ply) s položaji se izmenjujejo in tako pravimo, da smo v tej plasti na vrsti mi, v naslednji nižji plasti je na vrsti nasprotnik in tako naprej.

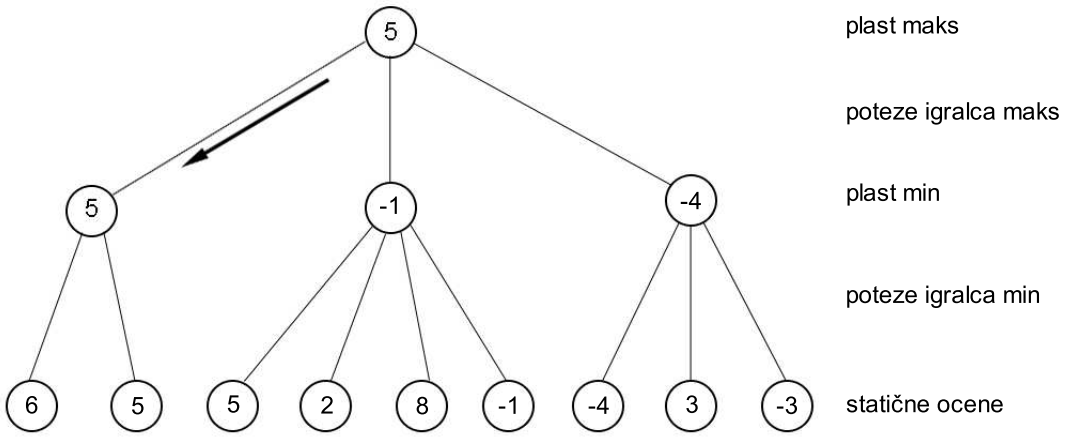
Da igralec dobi občutek igrati proti inteligentnemu računalniškemu programu, je programu potrebno liste drevesa z ocenitveno funkcijo oceniti, in se nato odločiti, katera poteza mu bo prinesla prednost pred nasprotnikom. Pri večini iger je zaradi števila možnih potez nemogoče drevo razviti do konca. Zaradi prevelike porabe pomnilnika in časovne zahtevnosti drevo raje razvijemo do vnaprej določene globine.

Nato z ocenitveno funkcijo ocenimo liste iskalnega drevesa. Zasnova ocenitvene funkcije je lahko od programa do programa drugačna.

2.2.4 Algoritem minimaks

Gonilo za razvoj postopka za izbiro potez pri večini iger z dvema igralcema je bila igra šah. John Von Neumann si je leta 1928 izmislil postopek za iskanje potez imenovan minimaks [11]. Ta postopek se uporablja pri večini iger z dvema igralcema. Če ponazorimo scenarij, da je računalnik na potezi, sebe označi kot igralca maks ali tako imenovani maksimizirajoč. Nas kot nasprotnika oziroma igralca min pa označimo kot minimizirajoč. Pogosto uporabljamo izraza maks in min, da označimo posamezne plasti drevesa. Pozitivna ocena položaja bi pomenila prednost za igralca maks negativna pa za igralca min. Prav tako bi pozitivna ocena pomenila slabši položaj za igralca min ter negativna ocena slabši položaj za maks. V primeru, ko je ocena ničelna, torej enaka nič, se položaj ovrednoti kot nevtralen, saj noben izmed igralcev ni v prednosti. Ocena položaja je v razponu med -∞ in ∞. Oznaka ∞ pomeni zmagovito oceno igralca maks, -∞ igralca min.

Na trivialnem primeru (slika 2.1) bomo poskusili prikazati delovanje iskalnega algoritma minimaks.



Slika 2.1: Drevesna struktura iskalnega algoritma minimaks.

Razvito je iskalno drevo z globino dva. Sestavljajo ga tri plasti. Prva plast predstavlja igralca maks in njegov trenutni položaj (koren). Druga plast predstavlja igralca min, zadnja, pa statične ocene pozicij. Dejali smo, da je drevo razvito do globine dva, kar ne pomeni, da sta v njem samo dva nivoja. Prve plasti oz. korena namreč ne štejemo kot potezo. Šteje se le trenutni položaj in tako razvito drevo prikazuje iskanje za dve potezi vnaprej.

Iskalni algoritem deluje tako, da v plasti igralca maks izbere maksimalno oceno poddrevesa. V plasti igralca min pa izbere minimalno oceno poddrevesa. Na zadnjem (najnižjem) nivoju priredimo statično oceno položaja. Algoritem bo za najboljšo potezo izbral tisto, katera bo igralcu v prvi plasti oziroma korenu najbolj koristila. V primeru, ko prva plast predstavlja igralca maks, bo izbral maksimalno vrednost, če pa le ta predstavlja igralca min, bo izbral minimalno vrednost. Ocene druge plasti so minimum ocen iz tretje plasti, ocena prve plasti je maksimum ocen iz druge. Seveda samo v našem primeru, saj je v prvi plasti igralec maks, sicer bi se ocene izbirale obratno.

Drevo opazujemo iz leve proti desni in od spodaj navzgor. Ker je v prvi plasti na potezi igralec maks, bi na prvi pogled lahko dejali, da bo zagotovo igral v smeri položaja 8, saj je tam maksimalna statična ocena. Vendar je med prvo in tretjo plastjo še ena plast, ki predstavlja igralca min. Če bi se odločili za izbiro poteze v smeri 8, nam to ne bi preveč koristilo. Bilo bi samo še slabše, saj bi s tem koristili nasprotniku. Ta bi v naslednji potezi igral v smeri -1, ki je zanj najboljša, za nas pa najslabša. Zato bi v prikazanem primeru najboljša poteza igranja bila poteza v smeri 5. Na sliki 2.1 je označena s puščico.

Iskalni algoritem zadostuje osnovnim potrebam. Zaradi preiskovanja celotnega drevesa pa ob iskanju z višjo globino ne doseže želenih rezultatov. Z rastjo globine iskalnega drevesa, se eksponentno veča velikost iskalnega drevesa, s tem pa tudi časovna in prostorska zahtevnost. Ozko grlo algoritma predstavlja ocenjevanje statičnih ocen v listih drevesa. Predvsem pri igrah s kompleksnimi ocenitvenimi funkcijami je njihovo izvajanje časovno zahtevno.

Če poznamo globino iskalnega drevesa d ter vejitveni faktor (povprečno število poddreves) b, potem lahko zapišemo časovno zahtevnost:

$$O(b^d). \tag{2.2}$$

Problem lahko rešimo, če se osredotočimo na pregledovanje le pomembnih delov razvitega drevesa (algoritem alfa-beta opisujemo v poglavju (2.2.6)).

2.2.5 Algoritem negamaks

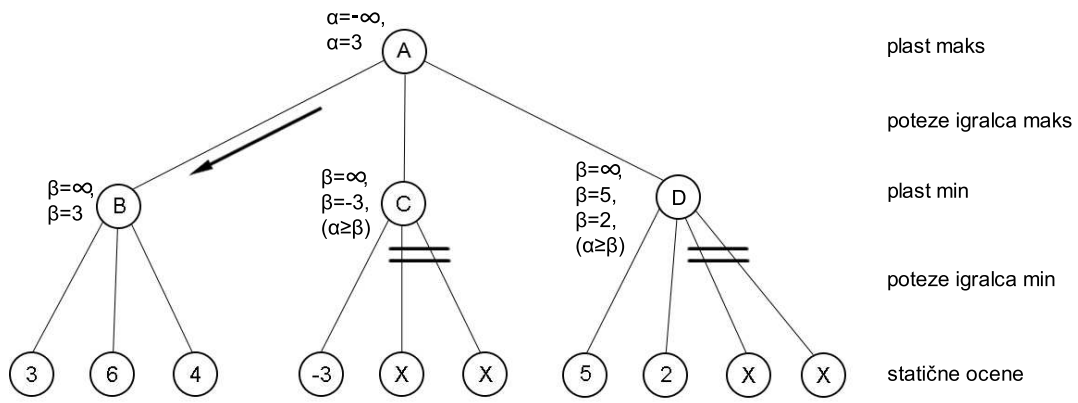
Iz enake družine izhaja algoritem negamaks, ki temelji na ideji minimaks. Bistvena razlika med njima je, da negamaks v vseh plasteh staršem dodeli nasprotno oceno igralca poddrevesa. Torej vrednostim pri vsakem prehodu med plastmi spremeni predznak. V primerjavi z iskalnim algoritmom minimaks je negamaks lažje implementirati in vzdrževati.

2.2.6 Algoritem alfa-beta

Algoritem alfa-beta temelji na enakem principu kot minimaks. Samo ime izhaja iz oznak α in β. Oznaka α predstavlja najboljšo vrednost za igralca maks, oznaka β pa najboljšo vrednost za igralca min. Pravimo, da α in β predstavljata okno iskalnega drevesa. Tudi pri tem algoritmu sta možna dva scenarija. Prvi je, da je koren drevesa označen kot maks, drugi pa da je koren drevesa plast min. Bistvo algoritma je klestenje poddreves. To pomeni, da poddreves, za katera predvidevamo, da niso obetajoča, ne pregledujemo. Če gre za prvo varianto, se opravlja tako imenovano alfa klestenje. Pri tem mora biti izpolnjen pogoj:

$$\alpha \geq \beta. \tag{2.3}$$

Pri drugem scenariju velja analogno. Beta klestenje se izvaja v plasteh maks, pogoj za klestenje je enak (glej neenačbo (2.3)).



Slika 2.2: Drevesna struktura iskalnega algoritma alfa-beta.

Na podanem primeru (slika 2.2) bomo poskusili prikazati delovanje algoritma alfa-beta. Prvo plast bomo zaznamovali kot igralca maks. Torej velja pravilo (2.3). Ob pregledovanju poddreves se lahko spremenljivki α in β spreminjata, zato smo položaje v višjih plasteh označili z A, B, C in D. Alfa klestenje se izvaja v plasteh min. Zgled prikazuje, da je vrednost α v prvi plasti, ki je doslej najvišja ocena izmed vseh poddreves. Vrednost β predstavlja doslej najnižjo oceno izmed pregledanih poddreves. Pregledovanje pričnemo pri najbolj levem poddrevesu. Po pregledu tega poddrevesa smo dobili vrednost 3. Zaenkrat igralcu na položaju A ta vrednost predstavlja najboljšo možno izbiro. Nadaljujemo s preiskovanjem drugega poddrevesa. Ob pregledu prve statične ocene smo dobili vrednost -3. Ker bo igralec min v položaju C težil k oceni β ali manj, igralec maks v položaju A k oceni α ali več, igralec α zagotovo ne bo igral v smeri vozlišča C, saj ima boljšo alternativo od prej. Pogoj (2.3) je izpolnjen, zato lahko opravimo alfa klestenje in končamo s

preiskovanjem tega poddrevesa. Podobna situacija nastane v zadnjem poddrevesu, kjer pri preiskovanju naletimo na statično oceno 2, ki je manjša od vrednosti α v vozlišču A. Izvede se klestenje. Igralec maks se bo odločil za izbiro poteze v smeri prvega poddrevesa, saj je v tej smeri možnost njegove zmaga najvišja.

Prečrtane poteze in listi z oznako X pomenijo, da teh zaradi klestenja ni bilo treba pregledati. Iz danega primera je razvidno, da je bilo pregledanih le šest od desetih listov. Zmogljivost tega algoritma je odvisna od tega, kdaj pride do klestenja. V idealnem primeru prvo pregledamo vozlišča, ki povzročajo klestenje, kar bi iskalno drevo enormno zmanjšalo. Pri taki prihranitvi na času lahko globino dvakrat povečamo. V najslabšem primeru ne pride do klestenja in algoritem posledično opravi enako delo kot algoritem minimaks opisan v poglavju 2.2.4.

Znano je, da ima drevo n listov in da je v najboljšem primeru najboljša poteza igralca na skrajni levi strani, potem je potrebno ovrednotiti le $\sqrt{n}$ listov. Na podlagi ugotovitve (2.2) lahko zapišemo:

$$O((b^d)^{\frac{1}{2}}) = O(\sqrt{b^d}). \tag{2.4}$$

Izboljšave algoritma so [16]:

- podaljševanje dominantnih položajev (angl. quiescence search),
- transpozicijske tabele (angl. transposition table),
- postopno poglabljanje (angl. progressive deepening)
- itd.

Poglavje 3

Struktura programa ReversiOnCisco

V prejšnjem poglavju smo naredili pregled IP telefonskih aparatov in iger s popolno in nepopolno informacijo. Opisali smo strategije in iskalne algoritme, ki so sestavni del igre reversi. V tem poglavju bomo razkrili, kaj nas je motiviralo za razvoj našega programa ReversiOnCisco in zakaj smo razdelili problem v tri segmente.

3.1 Motivacija

Motivirala nas je ideja igranja igre reversi na IP telefonskem aparatu. Privzeto so IP telefonski aparati namenjeni telefoniranju ter prikazu določenih informacij a ne igranju iger. Pri tem smo se soočili z omejitvami s strani IP telefonskega aparata, kako:

- prikažemo želene podatke na IP telefonskem aparatu,
- sestavimo sliko iz več elementov,
- prikažemo igralno ploščo, ki se ob vsakem vstavljenem žetonu spremeni,
- uporabniku omogočimo izbiro poteze in
- prikažemo izvajanje potez na animiran način.

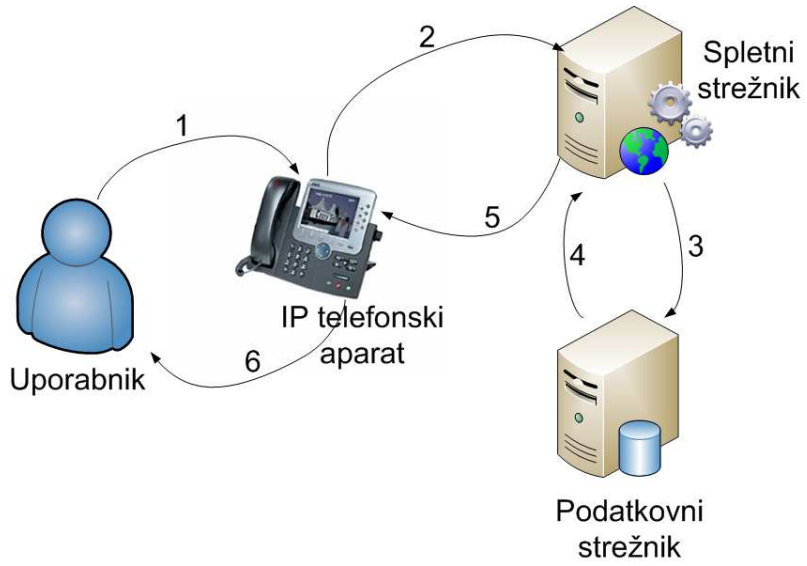
Ker gre za edinstven primer reševanja takega problema, smo se odločili problem rešiti po principu strategije deli in vladaj.

3.2 Posamezni segmenti kot celota

Po temeljiti raziskavi ugotovimo, da je najboljše problem razdeliti v tri glavne segmente, kar je prikazano na sliki 3.1:

- IP telefonski aparat,
- spletni strežnik in
- podatkovni strežnik.

Vsak izmed naštetih segmentov opravlja svoje delo. IP telefonski aparat služi interaktivnemu komuniciranju z uporabnikom. Podatkovni strežnik služi shranjevanju statističnih podatkov, bistveno vlogo pa ima spletni strežnik. Ta upravlja z ukazi oziroma zahtevami, ki jih dobi od telefonskega aparata. Odloči ali bo zahteva preusmerjena motorju (angl. game engine), dinamični knjižnici za delo s podatkovno bazo (angl. game statistics) ali gre zgolj za ukaz, katerega strežnik zna obdelati sam. Konec koncev tudi pripravi povratne podatke IP telefonskemu aparatu, ki jih bo le ta prikazal uporabniku. S tem se komunikacijski krog zaključi in uporabnik po pritisnjeni tipki dobi ustrezno povratno informacijo.

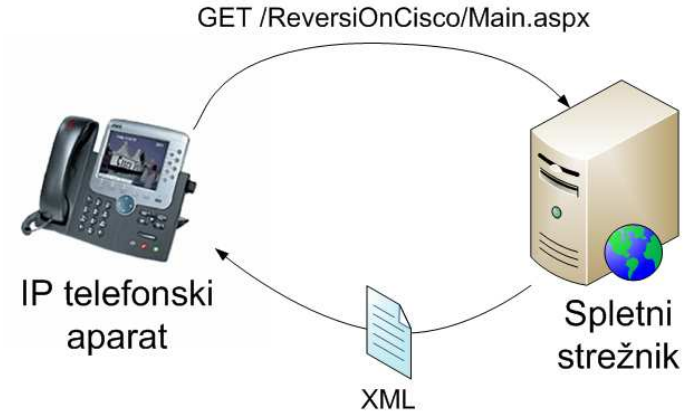


Slika 3.1: Celotno rešitev ReversiOnCisco sestavljajo posamezni segmenti.

Poglavje 4

Naš program in Cisco IP telefonski aparat

Dandanes obstaja širok izbor IP telefonskih aparatov, vendar smo se odločili za rešitev s Cisco telefonskim aparatom. Cisco s svojimi IP telefonski aparati nudi možnost komunikacije s spletnimi servisi (angl. web services). Spletni servisi vračajo podatke v posebej pripravljenem formatu. To je tako imenovani razširljivi označevalni jezik (poglavje 2.1). Za nas to pomeni prvi korak povezave IP telefona s spletnim strežnikom. Zaradi kompleksnosti problema in številnih drugih problemov smo se odločili, da bomo na spletnem strežniku zasnovali spletno stran, ki bo vračala podatke v popolnoma enakem formatu kot spletni servis (slika 4.1).



Slika 4.1: Povpraševanje po vsebini in odgovor kot XML format.

Poglavje 5

Spletni strežnik

Spletni strežnik si lahko predstavljamo kot imenik. Vsaka izmed njegovih strani ima določen pomen in nam ob pogledu nanjo vrača določene informacije. Bodisi v takšni ali drugačni obliki. V realnem življenju bi to lahko bila imena in številke v obliki alfanumeričnih znakov, v našem primeru v obliki XML (poglavje 2.1.1).

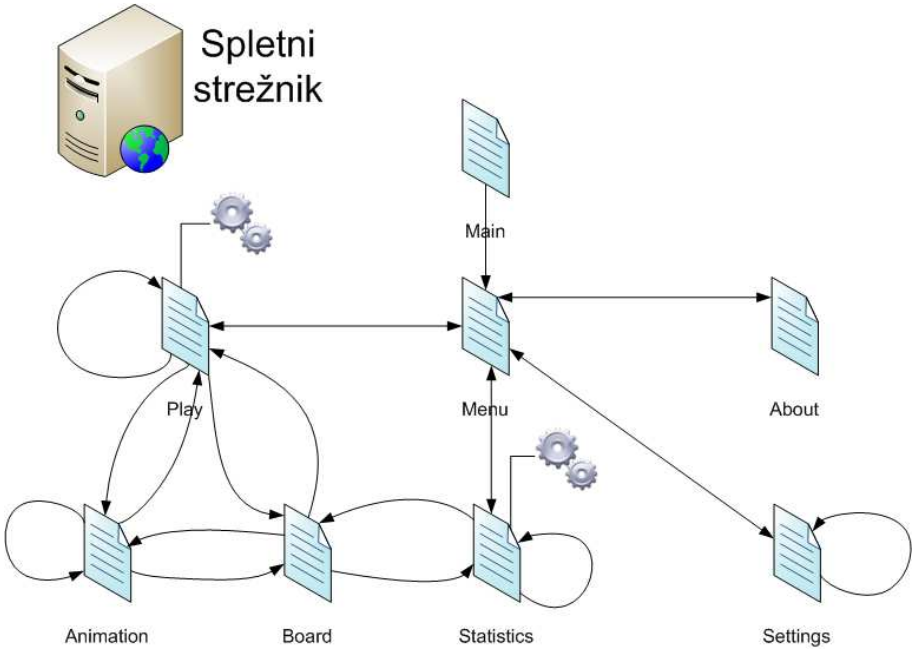
5.1 Podatkovni tok

Slika 5.1 prikazuje podatkovni tok med posameznimi spletnimi stranmi. Linearne povezave označujejo preusmeritve strani. Krivulje med različnimi povezanimi stranmi označujejo povpraševanje oz. vračanje določene vsebine. Krivulje, ki ne zapuščajo določene strani in kažejo na samega sebe, označujejo prenos oz. potrjevanje podatkov. Povezave iz strani do zobnika pomenijo dostop do dinamičnih knjižnic, do katerih spletni strežnik mora dostopati, da lahko opravi nadaljnje delo. Naša aplikacija vključuje dve dinamični knjižici. Prva je motor, druga dinamična knjižnica pa je za delo s podatkovno bazo. Delo na spletnem strežniku opravlja osem spletnih strani:

1. **Main**,
2. **Menu**,
3. **About**,
4. **Settings**,
5. **Statistics**,
6. **Play**,
7. **Board** in

8. **Animation**.

Vstopno točko predstavlja spletna stran **Main**. Ta pripravi podatke za nadaljnje delo. Priskrbi ime telefonskega aparata, katero predstavlja edinstveni ključ. Po njem ločimo statistične podatke glede na telefonske aparate. Ime telefonskega aparata pridobimo s preprostim trikom. Ker je to edina direktna povezava med telefonskim aparatom ter spletnim strežnikom in nam telefonski aparat ob le takih povezavah vrača svoje ime, preprosto takoj povprašamo po imenu. Pripravi tudi privzete nastavitve potrebne za igranje. To so na primer privzeta globina igranja, prikaz možnih potez, animiran prikaz izvajanja potez, igralec ena in igralec dva itd. Vse podatke shrani v sejo, kjer so dostopni tudi iz vseh ostalih strani. Po opravljeni pripravi podatkov nas preusmeri na stran **Menu**. **Menu** oz. meni je prva stran, ki se uporabniku prikaže ob zagonu naše aplikacije. Sestavljena je iz štirih menijskih vrstic, katere uporabniku omogočajo vstop v igro (**Play**), nastavljanje posameznih parametrov za igranje igre (**Settings**), pregled preteklih iger in izidov **Statistics** in pregled ostalih informaciji o igri **About**. **Board** opravlja funkcijo generatorja dinamičnih slik (glej poglavje 5.3.2), **Animation** pa simulira animiran prikaz izvajanja potez (glej poglavje 5.3.3).



Slika 5.1: Struktura in podatkovni tok spletnih strani na spletnem strežniku.

Zraven že omenjenih funkcij strežnika velja še omeniti druge njegove naloge:

- priprava podatkov,
- prenos podatkov med stranmi,

- shranjevanje vmesnih podatkov,
- opravljanje z motorjem,
- prikaz animiranega izvajanja potez,
- vizualizacija elementov in
- komunikacija s podatkovnim strežnikom.

Nekatere izmed naštetih nalog strežnika nimajo tako pomembne vloge kot ostale. Vendar brez njih delovanje ne bi bilo popolno, zato bomo tudi te opisali.

5.2 Upravljanje s podatki

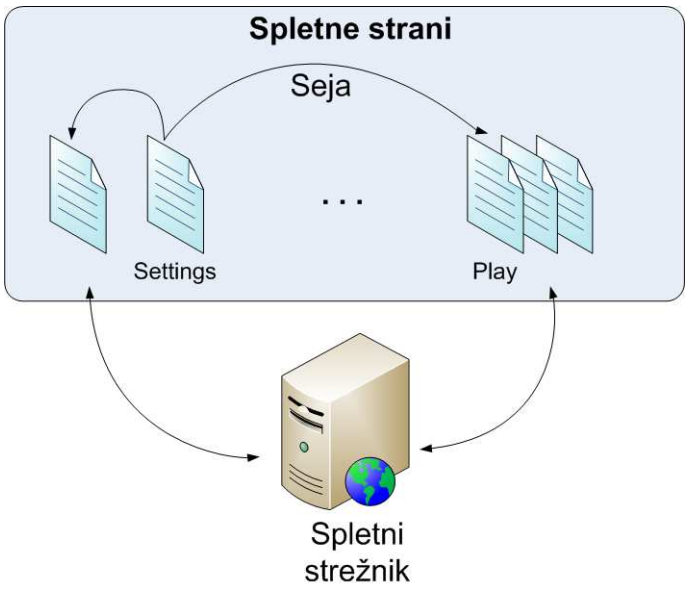
5.2.1 Priprava podatkov

Za IP telefonski aparat je priprava podatkov najpomembnejši del. Sam aparat s podatki ne bi znal operirati, če le teh ne bi že vnaprej poznal. Govorimo o XML formatu. Spletna stran v osnovi vrača podatke v obliki enostavnega besedila (angl. plain text). V osnovi je to dovolj, vendar pa ne za potrebe telefonskega aparata.

```
Response.ContentType = "text/xml";    //nastavi XML kot tip vsebine strani
Response.Expires = -1;                //ne dovolimo shranjevanja zgodovine
```

Z dvema vrsticama izvorne kode v jeziku C# ogrodja ASP.NET [17] lahko nastavimo željen tip vsebine. S tem se izognemo še drugi nevšečnosti, kot je shranjevanje obiskanih strani. Telefonski aparat podobno kot internetni brskalnik hrani zgodovino obiskanih strani. Slabost tega je, da se med njimi ne moramo poljubno premikati naprej oziroma nazaj, temveč z zapiranjem trenutne strani samo nazaj. Zamisel, da bi po končani igri vse izvedene poteze korak za korakom lahko pregledali je lepa, vendar postane nadležna, če bi želeli igro predčasno zapustiti in bi za vsako potezo morali stran znova in znova zapreti.

5.2.2 Prenos podatkov med stranmi



Slika 5.2: S sejo prenašamo podatke iz ene do druge strani.

Za prenos podatkov iz ene do druge spletne strani obstajajo preproste metode

kot sta GET in POST. Ti dve metodi lahko uporabimo, če želimo določene podatke prenesti iz strani A do strani B, vendar podatkov ne moremo prenašati naprej. Z določenimi metodami je tudi to možno, a je preprostejša rešitev uporaba seje (angl. session). Seja se prične, kadar nov uporabnik vstopi na prvo stran nekega sistema. Vsak uporabnik ima svojo sejo, ki ob mirovanju uporabnika poteče po določenem času. Vanjo lahko shranimo poljubne podatke. Te lahko prenašamo med različnimi stranmi, kar je naš cilj.

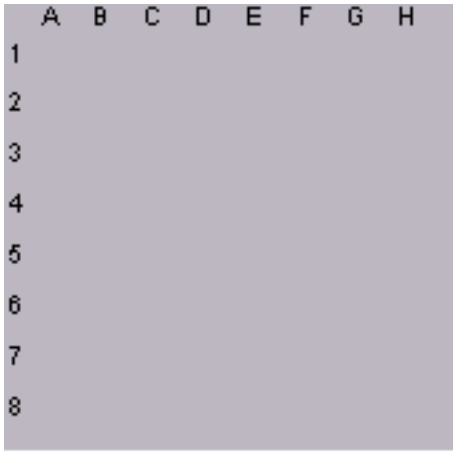
Na sliki 5.2 vidimo, da je seja dostopna iz vseh strani. Lahko bi se zgodilo, da uporabnik želi spremeniti nastavitve (angl. settings) igre. Torej bi vpisal svoje ime in izbral globino igre. V tem primeru bi za shranjevanje teh dveh podatkov potrebovali sejo z dvema različnima spremenljivkama. To bi bilo nadležno, zato ustvarimo razred, ki ima vse potrebne lastnosti in ga priredimo seji. Tako imamo vse nastavitve in ostale potrebne podatke na enem mestu.

5.3 Vizualizacija

Pod pojmom vizualizacija razumemo prikaz ali izris slike. V osnovi nam Cisco IP telefonski aparat omogoča prikaz statične slike. Vendar nam je pri realizaciji naše aplikacije le ta predstavljala velik problem. Za igranje igre reversi potrebujemo ploščo, na kateri so žetoni igralcev. Z vsako potezo se stanje plošče in število žetonov na njej spremeni. Torej bi za igranje potrebovali mnogo sličic plošče z vsemi možnimi stanji na njej. Zaradi porabe virov nam ta rešitev ni primerna. Rešitev je generator dinamičnih slik.

5.3.1 Elementi

Igralna plošča (slika 5.3) je kvadratna, dimenzije 8×8 igralnih polj, skupaj štiriinšestdesetih celic. Osnovo oziroma podlago predstavlja prazna plošča velikosti 168×168 pikslov, kar je znotraj dovoljenega območja zaslona (poglavje 2.1.1).



Slika 5.3: Podlaga predstavlja prazno igralno ploščo.

Na njej so igralna polja velikosti 19×19 pikslov. Igralno polje lahko zavzame eno izmed sedmih osnovnih oblik:

1. **prazno polje** označuje del igralne plošče, kamor ni možnosti vstavljanja žetona, vendar je del igralne plošče (na sliki 5.4(a) je zaradi boljše ponazoritve uokvirjeno s črno barvo),

2. **rdeči žeton** predstavlja trenutni zavzet položaj rdečega igralca (slika 5.4(b)),

3. **rdeči animacijski žeton** je viden samo po izvedbi poteze rdečega igralca in zaznamuje polja, ki jih je zavzel zelenemu igralcu (slika 5.4(c)),

4. **prosti rdeči žeton** predstavlja polje, ki ga lahko zaseda rdeči igralec s svojim žetonom (slika 5.4(d)),

5. **zeleni žeton** predstavlja trenutni zavzeti položaj zelenega igralca (slika 5.4(e)),
6. **zeleni animacijski žeton** je možno viden samo po izvedbi poteze zelenega igralca in zaznamuje polja, ki jih je zavzel rdečemu igralcu (slika 5.4(f)) in
7. **prosti zeleni žeton** predstavlja polje, ki ga lahko zaseda zelen igralec s svojim žetonom (slika 5.4(g)).

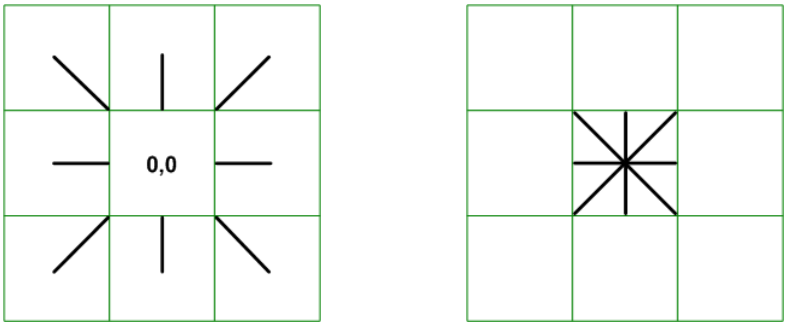
(a) (b) (c) (d) (e) (f) (g)

Slika 5.4: Osnovni sestavni elementi igralne plošče.

Ostali elementi

Vključili smo dodatno možnost opazovanja smeri zavzemanja območij. Ta možnost je razširitev osnovnega animiranega prikaza izvajanja potez. V primeru, da je animiran prikaz izvajanja potez onemogočen, potem tudi opazovanje smeri zavzemanja območij ni možna. Elemente ločimo glede na:

- začetni ali končni element (slika 5.5(a)) in
- vmesni oziroma povezovalni element (slika 5.5(b)).



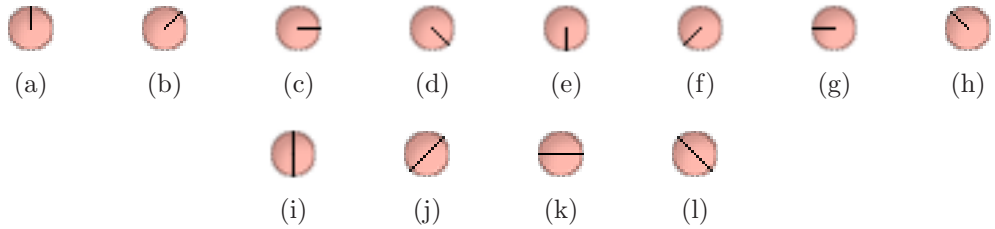
(a) Začetni ali končni element.

(b) Vmesni ali povezovalni element.

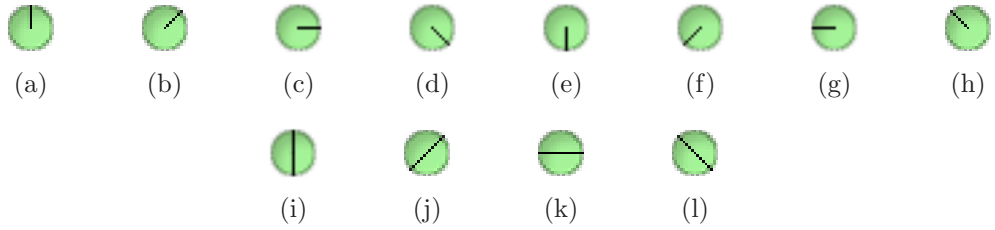
Slika 5.5: Razporejenost začetnih, končnih in povezovalnih elementov.

Začetni ali končni elementi imajo le polovico velikosti povezovalnih elementov. Razlog temu je večja preglednost nad pričetkom in koncem zavzete linije. Izbira elementa se določi glede na orientacijo zavzemanja (poglavje 6.2.1) in ima glede na igralca eno izmed dvanajstih oblik (slika 5.3.1 ali slika 5.3.1).

Ideja je, da bi se hkrati ob iskanju možnih potez zavzemanja shranjevale tudi informacije o orientaciji. Izhajamo iz matrične šablone, točke $(0, 0)$. Podali bomo



Slika 5.6: Elementi za animiran prikaz smeri zavzemanja območij zelenega igralca na igralni plošči.



Slika 5.7: Elementi za animiran prikaz smeri zavzemanja območij rdečega igralca na igralni plošči.

primer zavzemanja žetonov iz izhodišče točke navzgor. Izbira elementa za zavzemanje navzgor poteka iz izhodiščne točke glede na njeno orientacijo. Opazujemo celico nad izhodiščno točko (slika 5.5(a)). Ker je prikazana linija v spodnji polovici celice in le ta pomeni končni element linije zavzemanja bi izbrali element 5.6(a). To nam ne ustreza, zato pomnožimo njegovo vrednost z -1. S tem smo spremenili orientacijo in dobili nasprotni element (slika 5.6(e)), ki označuje začetek linije zavzemanja. Dokler ponovno ne najdemo svojega žetona, vstavljamo povezovalni element (slika 5.6(i)). Ko najdemo svoj žeton, to pomeni konec linije zavzemanja. Nato vstavimo element s prvotno orientacijo, torej končni element (slika 5.6(a)).

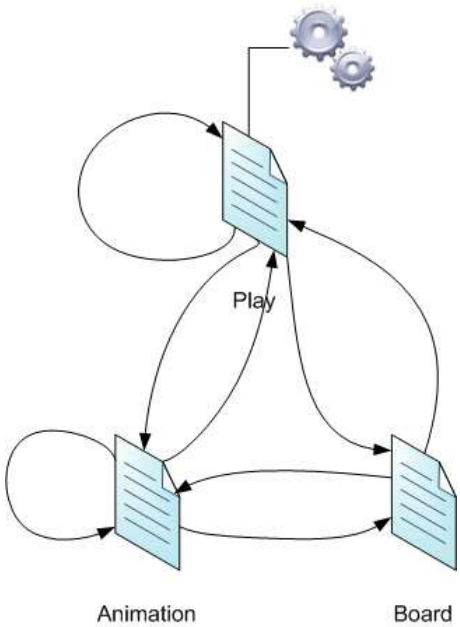
5.3.2 Generator dinamičnih slik

Operiranje s spletnimi stranmi se v tem primeru ponovno obrestuje. Tako kot lahko tip vsebine spletne strani nastavimo na obliko razširljivega označevalnega jezika, lahko tip vsebine spletne strani nastavimo tudi na sliko. Torej bo spletna stran **Board** vračala sliko v formatu PNG. Podlago (slika 5.3) vzamemo kot osnovo. Sledi izračun koordinat elementov. Glede na vhodni parameter *Board*, ki nosi numerične informacije o igralni plošči v dvodimenzionalnem polju se odločimo za enega izmed sedmih vnaprej definiranih elementov. Glede na nove koordinate obstoječo podlago prerišemo. Opisan postopek podaja algoritem 1.

Algoritem 1 Vizualizacija igralne plošče.

Vhod: $Board$ - 2D igralna plošča z numeričnimi informacijami o poziciji elementov
Vhod: W - širina
Vhod: H - višina
Vhod: D - dimenzija igralne plošče
Vhod: G_B - slika igralne plošče
Vhod: G_{E_i} - slika igralnega elementa
Vhod: G_{E_W} - širina elementa
Vhod: G_{E_H} - višina elementa
Vhod: E_o - orientacija usmerjenega elementa
Izhod: igralna plošča $NewBoard$

```
W = 0; {začetni koordinati elementov na igralni plošči}
H = 0;
NewBoard = G_B; {novi plošči priredimo osnovno obliko}
for column = 0 to D - 1 do
  for row = 0 to D - 1 do
    {nova pozicija igralnega elementa na osnovni plošči}
    W = (column * G_{E_W}) + Border_{Width};
    H = (row * G_{E_H}) + Border_{Height};
    switch (Board[column, row]) {na lokacijo W, H vstavimo element}
        case (default): Draw(NewBoard, G_{E_a}, W, H); {5.4(a)}
        case (player1, 2): Draw(NewBoard, G_{E_b}, W, H); {5.4(b) ali 5.4(e)}
        case (player1, 2_{Animation}):
            if Enable_{Direction} then {omogočen prikaz možnih potez}
                Draw(NewBoard, G_{E_o}, W, H); {element z ustrezno orientacijo}
            else
                Draw(NewBoard, G_{E_c,f}, W, H); {5.4(c) ali 5.4(f)}
            end if
        case (player1, 2_{PossibleMove}):
            if Enable_{PossibleMoves} then {omogočen prikaz možnih potez}
                Draw(NewBoard, G_{E_d,g}, W, H); {5.4(d) ali 5.4(g)}
            else
                Draw(NewBoard, G_{E_b,e}, W, H); {5.4(b) ali 5.4(e)}
            end if
  end for
end for
return NewBoard; {nova igralna plošča z vsemi potrebnimi elementi}
```



Slika 5.8: Potrebne komponente za animirano izvajanje potez.

5.3.3 Animiran prikaz izvajanja potez

Splošen problem Cisco IP telefonskih aparatov je animacija. Kljub temu v našo aplikacijo želimo vključiti animiran prikaz izvajanja potez. Razlog za to je sunkovito spreminjaje igralne plošče. To pomeni, da se ob uporabnikovi izbiri poteze in tik za njim še računalnikove poteze, z dvema potezama precejšen del igralne plošče lahko spremeni. Uporabnik bi videl le stanje plošče pred in po izvedeni potezi, kar bi ga lahko zmedlo. Zato smo razvili prefinjen postopek za animirano izvajanje potez. Potrebujemo tri strani in motor (slika 5.8)

Prva stran **Play** ima več funkcij. Skrbi za pravilno odvijanje igre. Možna sta dva scenarija:

1. **navadna igra** (scenarij 1) in
2. **igra z animiranim prikazom izvajanja potez** (scenarij 2).

V primeru prvega scenarija poteka komunikacija neposredno med spletno stranjo **Play** in spletno stranjo **Board**. Splatna stran **Play** pokliče motor, kateri na igralni plošči ($Current_{Board}$) izvede uporabnikovo potezo, poišče najboljšo računalnikovo potezo in to tudi izvede. Nato poišče vse možne poteze uporabnika in jih priredi igralni plošči. Po opravljenih korakih je igralna plošča pripravljena za izris. Podatki se shranijo v sejo, nadaljnje delo pa opravi generator dinamičnih slik (poglavje 5.3.2).

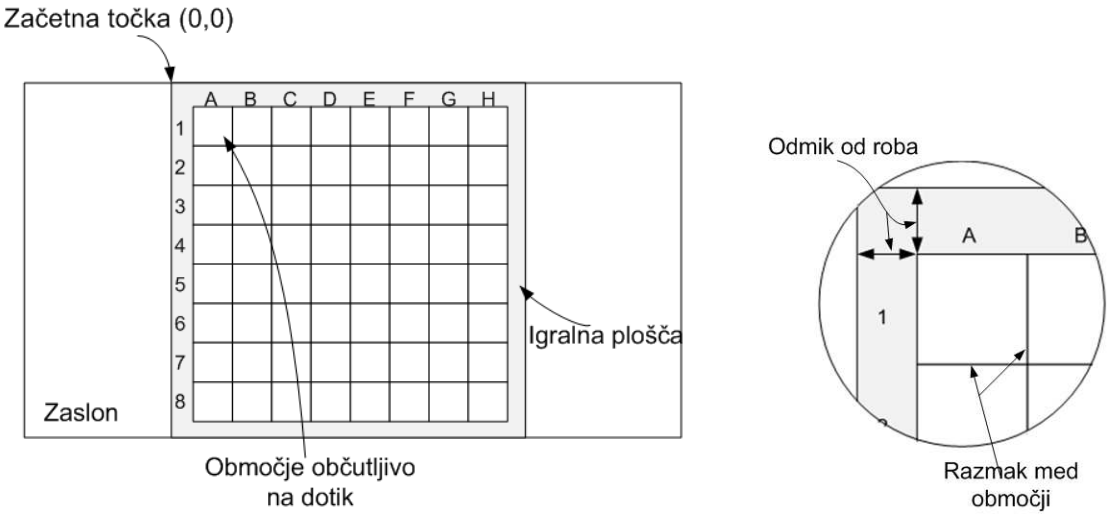
Pri drugem scenariju je nekoliko drugače. Če spletna stran **Play** ugotovi, da je parameter $Enable_{Animation}$ vključen, se pred izvedbo poteze uporabnika pobrišejo

vse možne poteze in naredi kopija trenutne igralne plošče. Vsa stanja se preslikajo v začasno igralno ploščo $Animation_{Board}$, z razliko, da se ob vsaki izvedeni potezi le te in njihove pozicije shranjujejo v začasni seznam $Animation_{List}$. Nato je potek enak kot pri scenariju 1. Sedaj imamo dve igralni plošči in seznam animiranih potez. Stran **Play** ponovno preveri ali je parameter $Enable_{Animation}$ vključen in ali je v $Animation_{List}$ katera poteza. Če sta pogoja izpolnjena, nas preusmeri na spletno stran **Animation**. Ponovno se preveri, ali seznam $Animation_{List}$ vsebuje katero potezo. Če jo, potem iz njega vzamemo prvo potezo in jo v seznamu $Animation_{Board}$ shranimo na ustrezno pozicijo. Dokler ni seznam $Animation_{List}$ prazen, torej še vsebuje poteze za simuliranje izvajanja, se v glavo spletne strani zapiše meta podatek za osveževanje. Ta podatek spletni strani pove, da naj le to po določenem času preusmeri. V našem primeru na samo sebe. Ta možnost nam omogoča ponovno nalaganje iste spletne strani. Torej bi lahko dejali, da osvežujemo stran **Animation**. Dokler seznam ni prazen, je izhod spletne strani v XML formatu, ki nosi pot do **Board**. Ta je dejanski generator dinamičnih slik in ob vsakem izrisu začasne plošče pobriše pravkar dodano potezo. S tem se seznam potez manjša, dokler ni prazen. Ko je prazen, nas stran **Animation** preusmeri nazaj na **Play**, ki nam posledično izriše trenutno igralno ploščo $Current_{Board}$.

5.3.4 Območje občutljivo na dotik

Za igranje igre in izbiro potez potrebujemo vhodno napravo. To nalogo opravlja Cisco IP telefonski aparat s svojim zaslonom občutljivim na dotik (poglavje 2.1.1). Preden stran **Play** telefonskemu aparatu vrne XML zapis, definira območja občutljiva na dotik. Definira jih tako, da glede na vse možne poteze igranja (poglavje 6.2.1) in njihove koordinate dodatno še določi, kje točno so te koordinate na zaslonu telefonskega aparata. Za pridobivanje ustreznih koordinat najprej poišče začetni koordinati $Board_{Scr_X}$ in $Board_{Scr_Y}$, ki predstavljata začetno točko $(0, 0)$ igralne plošče na zaslonu. Ta proces je potreben, saj je zaslon pravokotne in ne kvadratne oblike, kot naša igralna plošča (poglavje 2.1.1). Del igralne plošče so tudi številke in črke. Z njihovim kombiniranjem omogočamo lažje odčitavanje potez (slika 5.9(a)).

Tudi ta del igralne plošče zavzema nek prostor $Border_S$, zato je to vrednost pri vsaki novi definiciji potrebno prišteti. Da se območja občutljiva na dotik ne stikajo, imajo vmesni nedefinirani prostor. Zato vsaki začetni poziciji oz. koordinati (X_1, Y_1) elementa prištejemo to vrednost, vsaki končni koordinati (X_2, Y_2) elementa pa odštejemo to vrednost (slika 5.9(b)). Glede na stolpec in vrstico pozicije njeno vrednost pomnožimo s širino oziroma višino elementa, ki grafično zaznamuje območje občutljivo na dotik.



(a) (b)

Slika 5.9: Igralna plošča na zaslonu ter njena povečava.

Dodatno mora določiti, kaj se bo zgodilo, ko bo uporabnik pritisnil območje na telefonskem aparatu. To stori z naslovom URL, ki ga preusmeri nazaj na spletno stran **Play** in ga opremi z dodatno informacijo o območju oziroma izbrani poziciji. Ko bo Cisco IP telefonski aparat prejel XML zapis, ga bo obdelal in najprej izrisal sliko, potem pa označil območja občutljiva na dotik. Postopek definiranja območij občutljivih na dotik podaja algoritem 2.

Algoritem 2 Definiranje območij občutljivih na dotik.

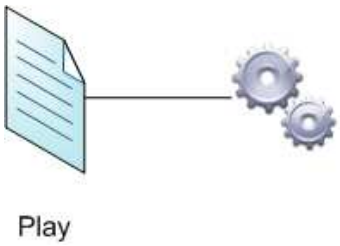
Vhod: X_1 - začetna koordinata x
Vhod: Y_1 - začetna koordinata y
Vhod: X_2 - končna koordinata x
Vhod: Y_2 - končna koordinata y
Vhod: $Possible_{Moves}$ - seznam možnih potez
Vhod: $Board_{Scr_X}$ - začetna točka igralne plošče x, na zaslonu telefonskega aparata
Vhod: $Board_{Scr_Y}$ - začetna točka y
Vhod: $Border_S$ - odmik elementov od roba igralne plošče
Vhod: $Elem_W$ - širina elementa
Vhod: $Elem_H$ - višina elementa
Vhod: $Grid_S$ - razmik med elementi
Izhod: XML z definiranimi območji občutljivimi na dotik

$X_1, Y_1, X_2, Y_2 = 0$;
$Board_{Screen_X} = (Scr_W/2) - (Board_W/2)$; {od polovice širine zaslona odštejemo polovico širine plošče}
$Board_{Screen_Y} = (Scr_H/2) - (Board_H/2)$; {od polovice višine zaslona odštejemo polovico višine plošče}
foreach ($Position\ p$ **in** $Possible_{Moves}$)
    $X_1 = Board_{Scr_X} + (Elem_W * p_{Column}) + Grid_S + Border_S$;
    $Y_1 = Board_{Scr_Y} + (Elem_H * p_{Row}) + Grid_S + Border_S$;
    $X_2 = Board_{Scr_X} + ((Elem_W * p_{Column}) + Elem_W)$ - $Grid_S + Border_S$;
    $Y_2 = Board_{Scr_Y} + ((Elem_H * p_{Row}) + Elem_H)$ - $Grid_S + Border_S$;
    $TouchItem = Touch_{Postion}(X_1, Y_1\ X_2, Y_2)$;
    $TouchURL = "Play?moveTo = " + p_{Column} + "," + p_{Row}$;
    $TouchScreen.Add(TouchItem, TouchURL)$;
end for
return *TouchScreen*; {XML z vsemi definicijami}

Poglavje 6

Motor

V tem poglavju bomo opisali motor (angl. game engine). Ta predstavlja program, ki vsebuje umetno inteligenco, katera je osnovna celica naše igre reversi. Vse, kar je povezano z delovanjem igre, kot je na primer izvajanje potez, iskanje možnih potez, zavzemanje žetonov, igranje nasprotnega igralca itd., je del motorja. Zasnovan je v obliki dinamične knjižnice (angl. dynamic-link library) ali kar DLL. Slika 6.1 prikazuje povezavo med dinamično knjižnico in spletno stranjo **Play**.



Slika 6.1: Povezava med motorjem in spletno stranjo.

6.1 Pričetek igre

Vstopno točko motorja predstavlja pričetek igre. Ko govorimo o pričetku igre, imamo v mislih inicializacijo vseh potrebnih spremenljivk in pripravo položajev igralcev na igralni plošči. Igro prične igralec ena z izbiro svoje poteze. Potezo vrne Cisco IP telefon, ki je ob prejšnjem izrisu igralne plošče na zaslonu dodatno označil območja občutljiva na dotik (poglavje 5.3.4). V primeru, ko poteza ni podana, gre za pričetek igre s pripravljenimi začetnimi položaji igralcev. V nasprotnem primeru gre za tekočo igro, kjer je že odigrana najmanj ena poteza. Glede na parametre potrebne za animiran prikaz, se ustvari kopija igralne plošče (poglavje 5.3.3). Sledi izvedba poteze trenutnega igralca (poglavje 6.2), temu pa poteza igralca dva, ki je v našem primeru računalnik. Tudi on bo glede na globino igre poiskal njemu najboljšo

igralno potezo (poglavje 6.3). Nato poiščemo vse možne poteze obeh igralcev. Če nobeden od igralcev nima možnosti nadaljevanja igre, torej nobeden izmed njiju nima možnosti poteze, potem to zaznamuje konec igre.

6.2 Izvajanje potez

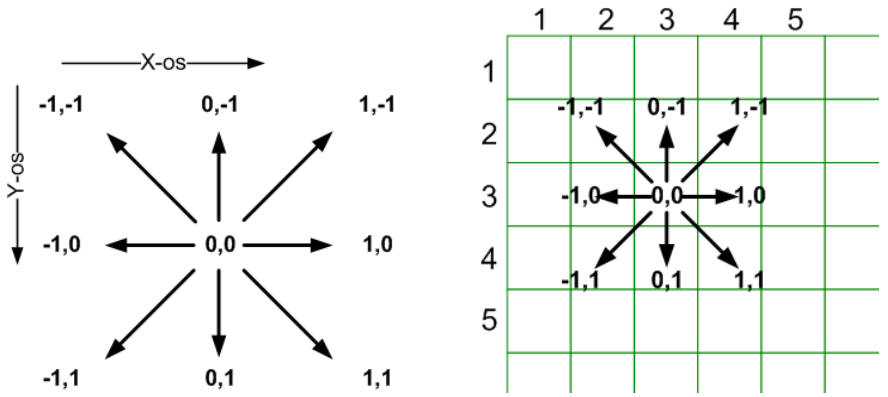
Ko uporabnik na zaslonu izbere potezo, izgleda kot, da bi potezo izvedel on sam, v resnici pa jo izvede motor. Za izvedbo ene poteze potrebujemo:

- potezo igranja,
- podatek o igralcu, ki želi izvesti potezo,
- postopek za obhod ostalih pozicij,
- postopek za preverjanje veljavnosti poteze in
- postopek za zavzemanje nasprotnikovih žetonov.

Torej potezo igranja že poznamo. Hkrati poznamo tudi igralca, ki jo želi izvesti. Na pozicijo poteze vstavimo žeton tega igralca in opravimo obhod v vse smeri igralne plošče. S tem smo izvedli potezo in zavzeli vse nasprotnikove žetone v vseh smereh.

6.2.1 Iskanje možnih smeri zavzemanja

Iskanje možnih smeri zavzemanja v vseh smereh igralne plošče, si lahko predstavljamo kot matrično šablono (slika 6.2(a)).



(a) Matrična šablona.

(b) Matrična šablona položena na želeno potezo igranja.

Slika 6.2: Uporaba matrične šablone.

Izhodiščna točka matrične šablone je točka $(0, 0)$. Orientacija matrične šablone določa smer iskanja. Tako bi se pri orientaciji $-1, -1$ pomikali v smer levo navzgor. Pri orientaciji $0, -1$ navpično navzgor itn. Šablono položimo na igralno ploščo, kjer izhodiščno točko matrične šablone postavimo na želeno potezo igranja. Slika 6.2(b) predstavlja odsek igralne plošče s potezo igranja $(3, 3)$, na kateri je položena matrična šablona s svojo izhodiščno točko $(0, 0)$. S prištevanjem orientacije, ki je celoštevilčni element iz intervala $[-1,\ 1] \times [-1,\ 1]$, dobimo novo začasno pozicijo. Sledi preverjanje veljavnosti poteze.

Veljavnost poteze preverimo tako, da prvo pogledamo, ali je nova začasna poteza znotraj območja igralne plošče. Temu sledi preverjanje, ali je trenutni lastnik nove pozicije nasprotnik. Če je pogoj izpolnjen, nadaljujemo iskanje v tej smeri. V trenutni smeri iskanja se pomikamo z izbrano orientacijo, dokler ne naletimo spet na svoj žeton. V tem primeru pomeni, da je ta orientacija veljavna in lahko zavzamemo celotno linijo (poglavje 6.2.2). Sicer spremenimo orientacijo in postopke ponavljamo, dokler niso preverjene vse smeri igralne plošče.

6.2.2 Zavzemanje nasprotnikovih žetonov

Zavzemanje nasprotnikovih žetonov je dokaj preprost postopek. Vendar brez zavzemanja igranje ne bi bilo možno. Ko ugotovimo, da je smer igranja veljavna (poglavje 6.2.1), opravimo enak postopek kot pri samem iskanju. Tokrat v izbrani smeri iščemo svoj žeton in hkrati spreminjamo nasprotnikove žetone v naše. To naredimo tako, da na igralni plošči numerično vrednost nasprotnika zamenjamo z našo numerično vrednostjo.

6.3 Umetna inteligenca

Umetna inteligenca (angl. artificial intelligence)[1] je najzahtevnejše opravilo motorja. Z umetno inteligenco simuliramo nasprotnika, ki je do neke mere "inteligenten". To mero podamo z globino (angl. depth). Ko govorimo o globini igre, imamo v mislih, koliko potez (poteza v igri) vnaprej bo računalnik razvil iskalno drevo. Večja kot je globina, večja je možnost računalnikove zmage. Uporabnik je naredil potezo in čaka na izvedbo poteze računalnika. Za iskanje najboljše poteze smo implementirali iskalni algoritem alfa-beta. Naloga računalnik je (v tem primeru je mišljena umetna inteligenca) zgraditi iskalno drevo igre. Glede na izračunano oceno se bo odločil, v katero smer bo igral. Postopke za iskanje najboljše poteze prikazuje algoritem 3.

Algoritem 3 Algoritem alfa-beta.

Vhod: *depth* - globina igre
Vhod: *board* - igralna plošča
Vhod: *alpha* - vrednosti igralca maks
Vhod: *beta* - vrednosti igralca min
Izhod: statična ocena

```
eval = 0;
if depth == 0 then
  return Evaluate(board); {ocenitvena funkcija}
end if
Possible_Moves = GetPossible_Moves(); {iskanje možnih potez}
foreach (Position p in Possible_Moves)
    if alpha >= beta then
        break; {klestenje drevesa}
    end if
    board = Make_Move(p) {izvedba poteze}
    {rekurzivni klic}
    eval = −alphaBeta(board, depth − 1, −beta, −alpha) {ocena poteze}
    if eval > alpha then
        alpha = eval; {nova vrednost}
    end if
end fo
return alpha; {statična ocena}
```

Podroben postopek delovanja algoritma 3 (alfa-beta) je opisan v poglavju 2.2.6.

[1] Bralec lahko najde opis o umetni oz. računski inteligenci v knjigi[3].

6.4 Ocenitvena funkcija

Ocenitvena funkcija se izvede šele, ko je iskalno drevo zgrajeno. Zato je snovanje le te zelo pomembno. Odloča o tem, katera poteza bo izbrana. Ocenitveno funkcijo je možno zasnovati na več načinov. Vsaka ima svoje prednosti in slabosti. Odločili smo se za ocenitveno funkcijo vsote. Matematični model 6.1 predstavlja vsoto vseh žetonov igralne plošče. Oznaka w zaznamuje širino igralne plošče, h višino igralne plošče, i vrstico, j stolpec in *Board* igralno ploščo z numeričnimi vrednostmi. Ocenitveno fukncijo f izračunamo:

$$f = \sum_{i=0}^{w-1} \sum_{j=0}^{h-1} Board[i, j]. \tag{6.1}$$

Naknadno smo ocenitveno funkcijo ojačali. Pravimo, da smo povečali moč ocenitvene funkcije. Ocenitveno funkcijo vsote smo kombinirali s položajno strategijo (poglavje 2.2.2). Smisel položajne strategije je zavzeti robna območja, saj so le ta strateško gledano najboljša. Zato pripravimo matriko enake dimenzije, kot je igralna plošča in jo zapolnimo. Posamezna vrednost celice pomeni tolikšno ojačanje položaja, kot je numerična vrednost položaja v matriki položajne strategije.

$$PositionalMatrix = \begin{bmatrix} 100 & -20 & 10 & 5 & 5 & 10 & -20 & 100 \\ -20 & -50 & -2 & -2 & -2 & -2 & -50 & -20 \\ 10 & -2 & -1 & -1 & -1 & -1 & -2 & 10 \\ 5 & -2 & -1 & -1 & -1 & -1 & -2 & 5 \\ 5 & -2 & -1 & -1 & -1 & -1 & -2 & 5 \\ 10 & -2 & -1 & -1 & -1 & -1 & -2 & 10 \\ -20 & -50 & -2 & -2 & -2 & -2 & -50 & -20 \\ 100 & -20 & 10 & 5 & 5 & 10 & -20 & 100 \end{bmatrix} \tag{6.2}$$

Matriko položajne strategije $PositionalMatrix$ lahko razdelimo na štiri kvartale, ki se med seboj preslikavajo. Kotne pozicije imajo močno ojačanje, ob kotne pa šibko oz. negativno. Razlog je ta, če vstavimo svoj žeton tik pred kotno pozicijo, bo nasprotnik zavzel želeno kotno, kar seveda ne želimo. Ojačano ocenitveno funkcijo f_m izračunamo:

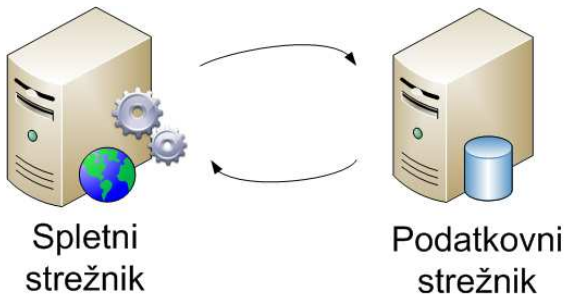
$$f_m = \sum_{i=0}^{w-1} \sum_{j=0}^{h-1} Board[i, j] \cdot Positional_{Matrix}[i, j]. \tag{6.3}$$

Poglavje 7

Podatkovni strežnik

To poglavje je namenjeno podatkovnemu strežniku. Za realizacijo naše rešitve ReversiOnCisco smo uporabili podatkovni strežnik *SQL Server 2005*. Na njem je shranjena podatkovna baza, s katero operira naša aplikacija. Podatkovni strežnik služi shranjevanju statističnih podatkov v podatkovno bazo, ki nam ob poljubnem trenutku dajo informacijo o preteklih igrah in njenih izidih. Pisali bomo o njegovih nalogah:

- zakaj in kdaj ga potrebujemo,
- dostop do njega,
- zasnova podatkovne baze,
- shranjenih procedurah in
- obdelavi podatkov.



Slika 7.1: Povezava med spletnim in podatkovnim strežnikom.

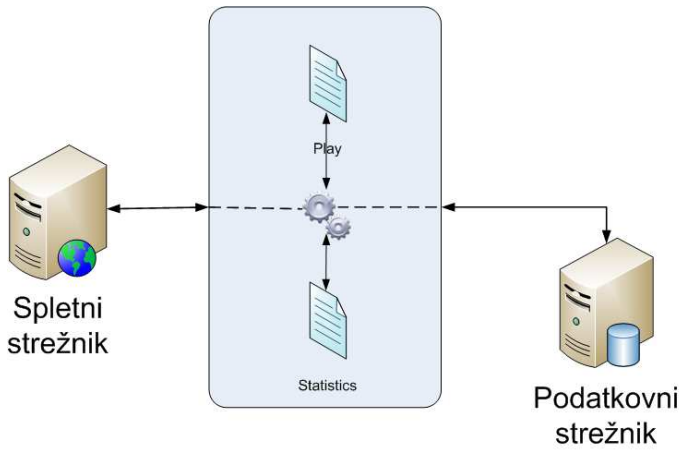
Za dostop do podatkovne baze shranjene na podatkovnem strežniku smo uporabili rešitev *SubSonic*. Slika 7.1 prikazuje povezavo med spletnim in podatkovnim strežnikom. Dostop do podatkovnega strežnika je možen le preko spletnega strežnika.

Cisco IP telefon nima neposrednega dostopa do podatkovnega strežnika in podatkovne baze shranjene na njem.

7.1 Dostop do podatkovnega strežnika

Do podatkovnega strežnika dostopamo z dinamično knjižnico, ki smo jo poimenovali *GameStatistics*, samo v dveh primerih:

- končanje igre (primer 1) in
- zahteva po prikazu statističnih podatkov (primer 2).



Slika 7.2: Podatkovni tok med spletnim in podatkovnim strežnikom.

V primeru 1 gre za shranjevanje podatkov. Ne glede na izid igre se shranijo statistično pomembni podatki. Izhajamo iz spletne stran **Play**. Izid igre je lahko:

- zmagovalec je igralec ena (uporabnik),
- zmagovalec je igralec dva (umetna inteligenca) ali
- neodločeno.

V primeru 2 gre za pridobivanje že shranjenih statističnih podatkov. Izhajamo iz spletne strani **Statistics**. Slika 7.2 prikazuje podatkovni tok med spletnim in podatkovnim strežnikom. Za upravljanje uporabljamo dinamično knjižnico.

7.2 Podatkovna baza

Podatkovna baza (angl. database) je prostor, kamor shranjujemo statistične podatke. Poleg shranjenih podatkov predstavljajo shranjene procedure (angl. stored procedures) postopke za obdelavo le teh.

Za nas relevantvni statistični podatki so:

- naziv telefonskega aparata ($\mathbf{Device_{Name}}$),
- ime igralca 1 ($\mathbf{Player1_{Name}}$),
- ime igralca 2 ($\mathbf{Player2_{Name}}$),
- globina igre ($\mathbf{Game_{Depth}}$),
- število doseženih točk igralca 1 ($\mathbf{Player1_{Score}}$),
- število doseženih točk igralca 2 ($\mathbf{Player2_{Score}}$),
- končni zmagovalec ($\mathbf{Game_{Winner}}$),
- čas pričetka igranja ($\mathbf{Game_{Start}}$),
- čas trajanja igranja ($\mathbf{Game_{Duration}}$) in
- stanje igralne plošče po končani igri ($\mathbf{Board_{State}}$).

Za shranjevanje in vračanje statističnih podatkov uporabljamo tri shranjene procedure:

- shranjevanje podatkov (**SetStatistic**),
- vračanje podatkov glede na ime naprave (**GetStatisticByDeviceName**) in
- vračanje podatkov glede na identifikator (**GetStatisticByID**).

7.3 Obdelava statističnih podatkov

Ko govorimo o obdelavi statističnih podatkov, imamo v mislih shranjevanje in pridobivanje le teh. Za izvedbo teh akcij uporabljamo vnaprej shranjene procedure, s katerimi povečamo varnost dostopa do podatkovnega strežnika.

7.3.1 Shranjevanje podatkov

Izvedba shranjevanja statističnih podatkov poteka na spletne strani **Play**. Vsak skupek shranjenih statističnih podatkov oz. zapis ima svoj edinstven ključ ali identifikator (ID). Običajen postopek shranjevanja podatkov bi lahko bil podoben slednjemu: pričetek igre s predčasno določenimi nastavitvami. Shranjevanje novega zapisa z nastavljenimi nastavitvami in časom pričetka igre. Po končani igri poišče ID trenutne igre, izračuna čas igranja, določi zmagovalca in vstavi končno stanje igre. V prikazanem primeru bi za eno odigrano igro potrebovali dva dostopa do podatkovne baze, kar prikazuje postopek 4. Shranjevanje ali vračanje podatkov pa stane nekaj časa. Večja kot je časovna rezina opravila, bolj bo uporabnik to občutil.

Algoritem 4 Odigrana igra z dvema dostopoma do podatkovne baze.

Vhod: *settings* - nastavitve
Izhod: odigrana igra s shranjenimi statističnimi podatki

```
saveSettingsIntoDB; {shranjevanje začetnih nastavitev (prvi dostop)}
startGame; {pričetek igre}
WHILE !gameOver DO {dokler igra ni končana}
    play; {igraj}
ENDWHILE
editStatisticIntoDB; {dopolnjevanje statističnih podatkov (drugi dostop)}
```

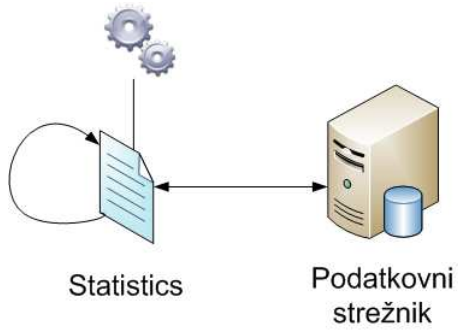
Na problem smo pogledali iz druge strani. Če uporabnik igro prične, nato nekaj časa igra in igro predčasno prekine, potem tako igro ne štejemo kot veljavno. Torej smo postopek shranjevanja podatkov prilagodili: pričetek igre s predčasno določenimi nastavitvami in trenutnim časom, nemoteno odigranje igre in konec igre. Klic shranjene procedure za shranjevanje podatkov se izvede šele po končani igri. Proceduro smo zasnovali tako, da prejme vse podatke razen časa trajanja igre. Ta se tik pred shranjevanjem izračuna glede na trenutni čas, od katerega smo odšteli čas pričetka igre. S tem smo dobili zadnji manjkajoči podatek in opravili zapis s samo enim dostopom. Postopek opisuje algoritem 5.

Algoritem 5 Odigrana igra z enim dostopom do podatkovne baze.

Vhod: *settings* - nastavitve
Izhod: odigrana igra s shranjenimi statističnimi podatki
  startGame; {pričetek igre}
  WHILE !gameOver **DO** {dokler igra ni končana}
    play; {igraj}
  ENDWHILE
  saveStatisticIntoDB; {shranjevanje statističnih podatkov z enim dostopom}

7.3.2 Pridobivanje podatkov

Kot smo že omenili, poteka operiranje s statističnimi podatki na spletni strani **Play** in **Statistics**. Povedali smo tudi, da na spletni strani **Play** izvajamo shranjevanje, na spletni strani **Statistics** pa vračanje oz. pridobivanje le teh.



Slika 7.3: Pridobivanje statističnih podatkov.

Spletna stran **Statistics** ima tri naloge:

- vračanja vseh shranjenih statističnih podatkov (Pregled),
- vračanje igralne plošče glede na ID zapisa (Igralna plošča) in
- vračanje podrobnosti o poteku igre glede na ID zapisa (Podrobnosti).

Smiselno bi bilo vračanje statističnih podatkov po vrstnem redu pregled, podrobnosti in nato igralna plošča, vendar zaradi omejitve Cisco IP telefona tak načni ni možen. Za podrobnosti uporabljamo menijske vrstice objekta *CiscoIPPhoneMenu*. Ta nam omogoča vračanje maksimalno 100 shranjenih zapisov dolžine do 64 znakov. Ta objekt nam omogoča dodajanje naslova URL, kamor nas telefonski aparat po izbrani menijski vrstici preusmeri. Naslednji objekt je *CiscoIPPhoneGraphicFileMenu*. Uporabljamo ga za prikaz igralne plošče. Tokrat generatorju dinamičnih slik pošljemo sekvenco numeričnih vrednosti. Ta zaznamuje stanje zavzetih celic na igralni plošči. Območje občutljivo na dotik definiramo podobno kot v poglavju 5.3.4 (Območje občutljivo na dotik), vendar tokrat čez celotno sliko igralne plošče.

Definirano območje tokrat ne pomeni izvedbo nove poteze, temveč preusmeritev na samega sebe z zahtevo po podrobnostih te igre. Podrobnosti prikažemo z objektom *CiscoIPPhoneText*. Ker je zmožnost tega objekta le prikaz znakov in ne omogoča nadaljnje preusmeritve, smo na tem mestu prisiljeni vrniti vse preostale statistične podatke. Lastnosti omenjenih objektov smo opisali v poglavju 2.1.1. V vseh treh primerih je preusmeritev na spletno stran **Statistics** vendar z različnimi zahtevami.

Poglavje 8

Rezultati programa ReversiOnCisco

Program ReversiOnCisco združuje tri komponente. Cisco IP telefonski aparat, ki predstavlja sprednji konec (angl. front-end) ter spletni in podatkovni strežnik, ki predstavljata zadnji konec (angl. back-end) naše aplikacije. Sprednji konec služi komunikaciji med uporabnikom in telefonskim aparatom, zadnji konec pa skrbi za procesiranje oziroma vračanje podatkov.

V nadaljevanju je prikazano delovanje našega programa. Prikazane slike so rezultat izvajanja posameznih akcij našega programa na Cisco IP Communicator-ju. Identične rezultate vračajo novejše različice Cisco IP telefonskih aparatov, kot je Cisco IP telefonski aparat 7970. Ta že v osnovi vključuje zaslon občutljiv na dotik. Pri tem omenimo, da uporabljena različica Cisco IP Communicator-ja ne podpira šumnikov.

Delovanje našega programa bomo razdelili v več korakov. Prvo bomo opisali začetni ali vstopni meni. Nato opišemo posamezno menijsko vrstico in njene zmožnosti. Sledi prikaz igranja igre z različnimi nastavitvami. Tukaj bomo opisali razlog za posamezne izbrane poteze umetne inteligence. Nato bomo spremenili posamezne nastavitve igranja in opisali njihovo posledico v ponovni igri. Primerjali bomo moč igranja umetne inteligence pri različni globini igranja z (ali brez) uporabe položajne strategije. Opisali bomo razlike med igranjem z omogočeno (ali onemogočeno) nastavitvijo prikaza možnih potez, animiranega prikaza izvajanja potez in dodatno možnost animiranega prikaza izvajanja potez, ki je usmeritev zavzemanja žetonov. Za prikaz različnega delovanja igre z različnimi nastavitvami je potreben ponovni zagon igre. Zato spreminjamo nastavitve in igro začenjamo vedno znova. Za konec opišemo pregled statističnih podatkov odigranih iger in s tem končamo poglavje z rezultati našega programa ReversiOnCisco.

8.1 Primer igre

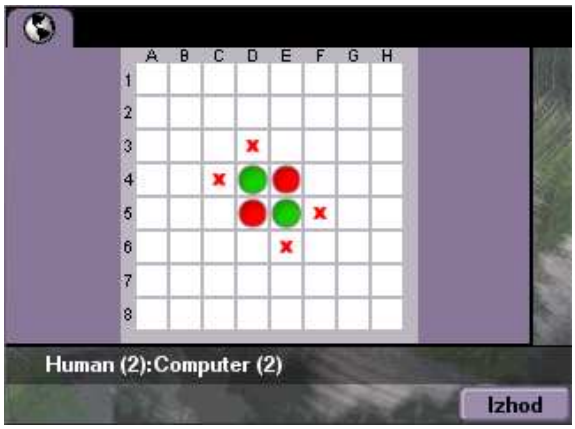
Slika 8.1: Vstopni meni.

Slika 8.1 prikazuje vstopni meni. Vstopni meni se uporabniku prikaže, kadar požene naš programa. Na izbiro imamo več menijskih vrstic. Vsaka izmed njih nas vodi v svoj odsek igre, od koder je možno njihovo nadaljnje upravljanje.

Slika 8.2: Prikaz možnih nastavitev.

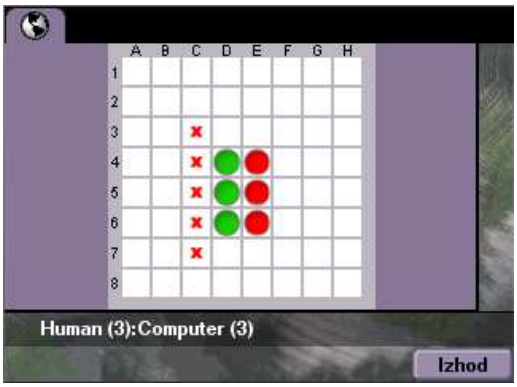
Kot je razvidno iz slike 8.2, smo pred pričetkom igre izbrali menijsko vrstico *nastavitve*. Prikazana je stran, ki jo kot odgovor na izbiro menijske vrstice *nastavitve* vrne spletni strežnik. Vidimo, da nam je dana možnost spreminjanja privzetih nastavitev. Poleg spreminjanja imena lahko spremenimo tudi globino igre. V našem primeru smo globino igre omejili na maksimalno šest potez vnaprej. Število šest predstavlja zgornjo sprejemljivo mejo, saj se z večanjem globine eksponentno veča tudi velikost iskalnega drevesa. Za prikaz trivialne igre smo izbrali globino ena. Animacijo in usmeritev smo v tem primeru onemogočili, prikaz možnih potez pa omogočili. Po potrditvi izbranih nastavitev se ponovno prikaže vstopni meni (slika 8.1), kjer za igranje igre z izbranimi nastavitvami izberemo menijsko vrstico *pričetek igre*.

Prikazali bomo primer igranja umetne inteligence pri globini ena z (ali brez) položajne strategije.

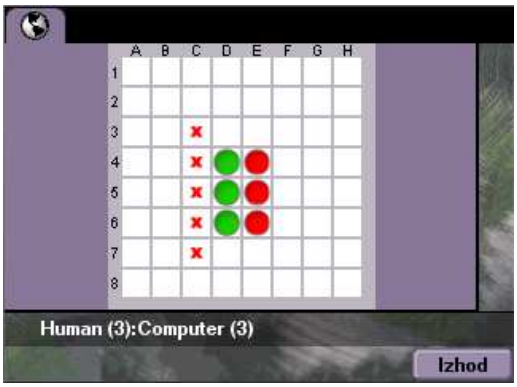


Slika 8.3: Začetna postavitev žetonov z možnimi potezami.

Pričetek igre z izbranimi nastavitvami predstavlja igralno ploščo z vnaprej določenimi začetnimi položaji. Pri nastavitvah smo omogočili možne poteze, zato so pri tej igri možne poteze igranja označene z rdečimi križci. Rdeči so, ker smo lastnik rdečih žetonov. Na dnu zaslona vidimo imena igralcev s pripadajočim številom zavzetih žetonov.



(a) Brez položajne strategije.

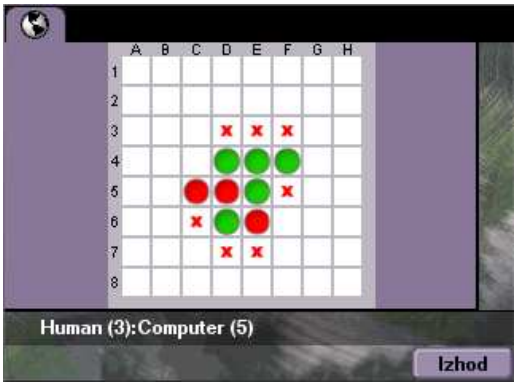


(b) S položajno strategijo.

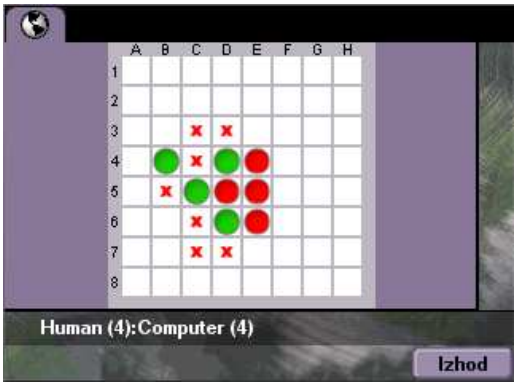
Slika 8.4: Stanje igralne plošče po prvi potezi *E,6*.

Slika 8.4 prikazuje, da se umetna inteligenca po naši izbrani potezi odloči za enako potezo *D,6* z (ali brez) položajne strategije.

Po izbrani potezi *C,5* je razlika v moči položajne strategije umetne inteligence očitna, kar prikazuje slika 8.5. Iz slike 8.5(a) je razvidno, da se umetna inteligenca odloči za potezo *F,4*. Vzrok temu je, da nasprotnik pri izbrani globini razvije iskalno drevo samo za eno potezo naprej. Torej v primeru izbrane poteze zavzame dva naša žetona, v vseh ostalih pa samo enega. Iz slike 8.5(b) je razvidno, da se umetna



(a) Brez položajne strategije.



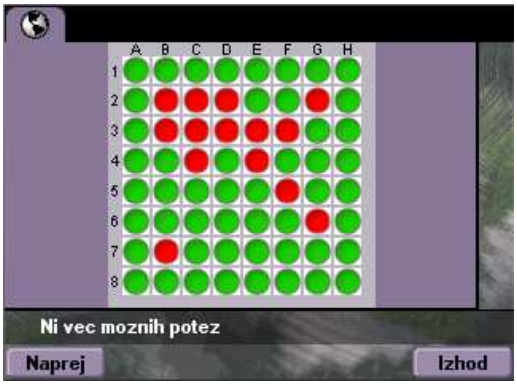
(b) S položajno strategijo.

Slika 8.5: Stanje igralne plošče po izbrani potezi *C,5*.

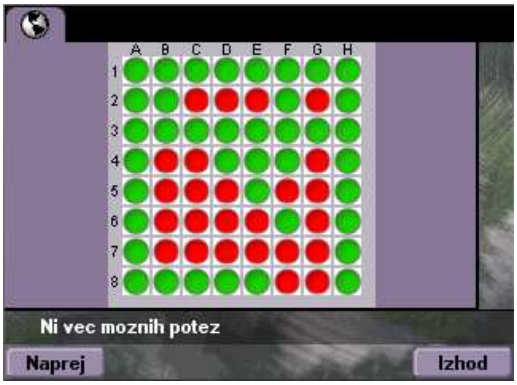
inteligenca z uporabo položajne strategije odloči za potezo *B,4*. Vzrok temu je matrika položajne strategije. Ta poveča možnost izbire pozicije bližje ležečim robovom, kajti bolj kot se bližamo središču igralne plošče slabše so možnosti nadaljnje igre. Sicer je nasprotnikovo stanje zavzetih žetonov v primeru brez položajne strategije višje, s položajno pa nižje, vendar se igranje v smeri robnih položajev obrestuje in konec koncev bi ob neprevidni izbiri potez umetna inteligenca z večjo verjetnostjo zmagala.

8.1.1 Primerjava moči dveh umetnih inteligenc

Za primerjavo moči umetnih inteligenc smo pripravili dva primerka umetne inteligence. Prvi primerek bo simuliral izbiro naših najboljših potez, drugi pa izbiro nasprotnikovih najboljših potez. Primerek ena igra z globino tri, primerek dva pa z globino šest. Odigrala bosta dve igri, prvo brez, drugo pa z uporabo položajne strategije. Prikazali bomo rezultate igranja dveh umetnih inteligenc.



(a)



(b)

Slika 8.6: Primerjava rezultatov umetnih inteligenc.

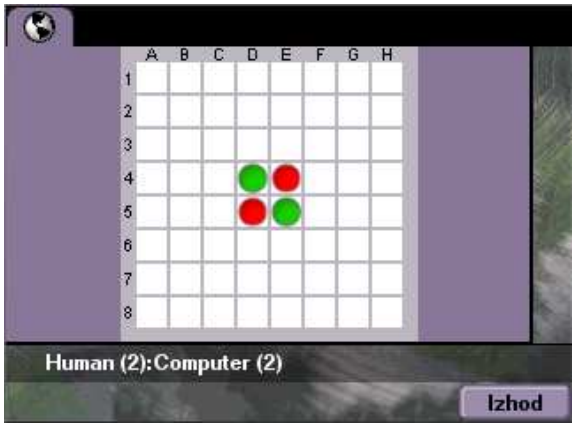
Že vnaprej lahko pričakujemo, da bo primerek dva v obeh primerih zmagal, saj igra z višjo stopnjo inteligence kot primerek ena. Razliko poda položajna strategija. Na sliki 8.6(b) vidimo več rdečih žetonov, saj je v odigrani igri igralec rdečih žetonov uporabljal položajno strategijo, pri sliki 8.6(a) pa ne.

8.1.2 Igranje igre brez možnih potez

Igranje igre brez možnih potez je nekoliko zahtevnejše kot z možnimi potezami. Za takšno igro je potrebno v nastavitvah možnost *možne poteze* onemogočiti. Izvedbo tega prikazuje slika 8.7.

Slika 8.7: Nastavitev brez možnih potez.

Slika 8.8 prikazuje igralno ploščo brez označenih možnih potez igranja.



Slika 8.8: Pričetek igre brez možnih potez.

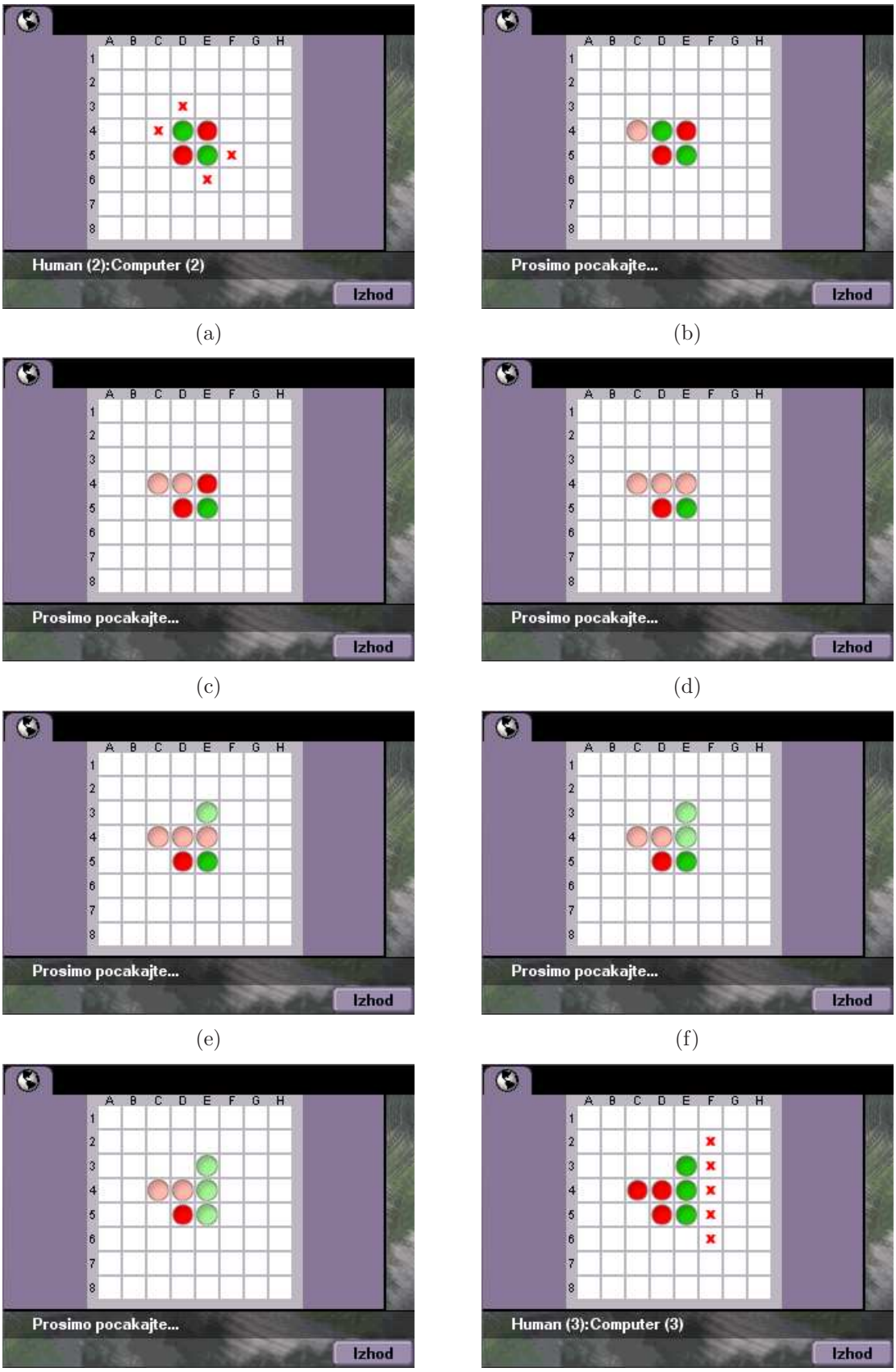
V primeru enakih izbir potez kot pri igranju z možnimi potezami, bi igra potekala analogno, saj možne poteze ne vplivajo na odločitev umetne inteligence. Uporabniku le olajšajo pot do prave izbire.

8.1.3 Animiran prikaz izvajanja potez

Za animiran prikaz izvajanja potez je pri nastavitvah potrebno omogočiti animacijo. Zaradi boljše ponazoritve izbranih potez smo omogočili tudi možne poteze. Usmeritev je v tem primeru onemogočena in globina igre je tri. Opisano podaja slika 8.9.

Slika 8.9: Nastavitve za igranje igre z animiranim prikazom izvajanja potez.

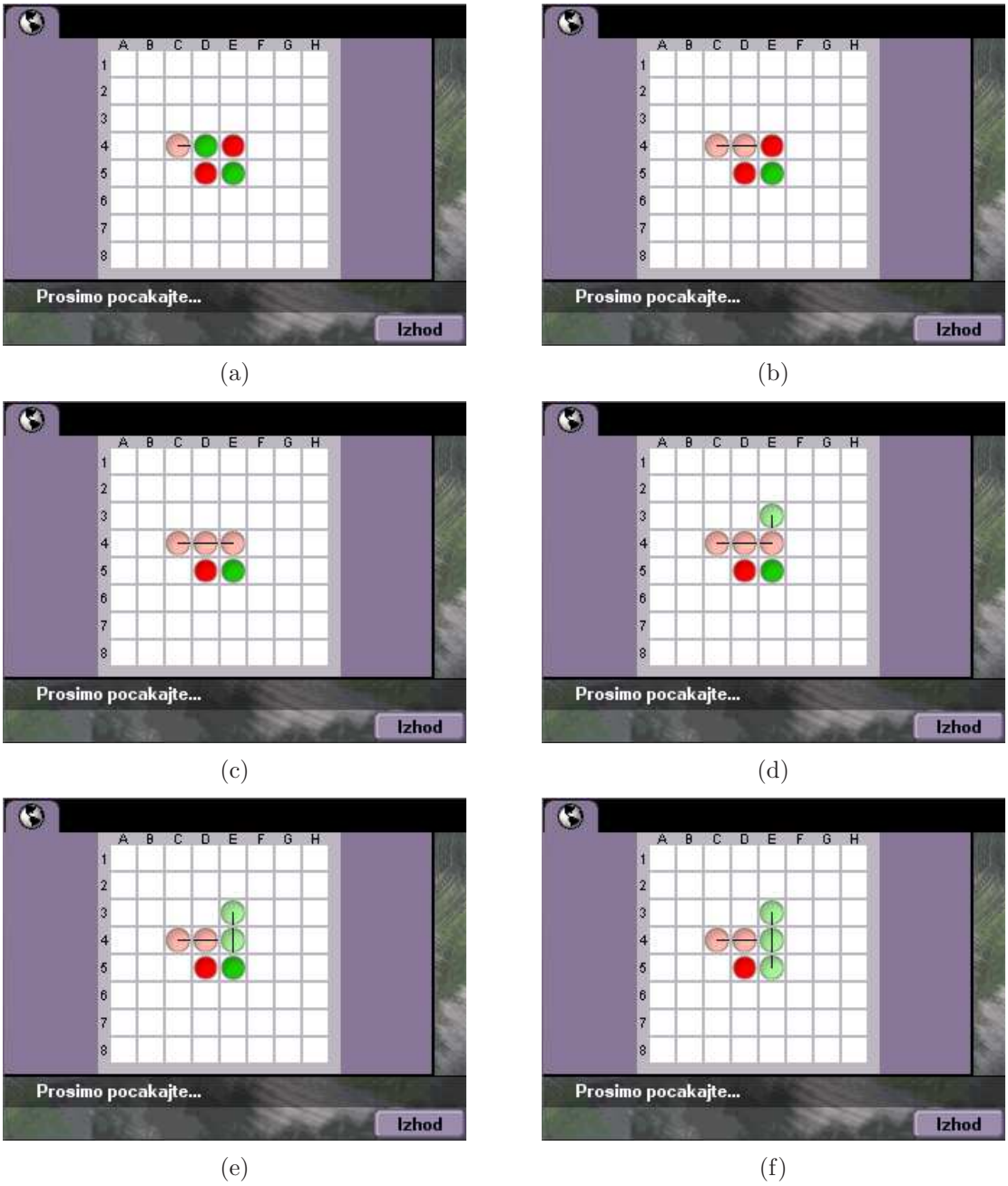
Slika 8.10 prikazuje animiran prikaz izvedbe dveh potez in posledično zavzemanje žetonov. Prvo potezo izvedemo mi, naslednjo pa nasprotnik. Zaporedje zavzemanja poteka v zaporedju izbranih potez. Osveževanje poteka v časovnem intervalu ene sekunde.



Slika 8.10: Animiran prikaz izvajanja potez.

8.1.4 Animiran prikaz izvajanja potez z usmeritvijo

Kadar gre igra v fazo zaprtja, je ob številnih žetonih na igralni plošči težko slediti animiranemu izvajanju potez. Obstaja možnost zavzemanja žetonov v več smeri hkrati. V takem primeru nam koristi znanje usmerjenosti zavzemanja. Ob enakih nastavitvah kot v poglavju 8.1.3, z dodatno omogočeno nastavitvijo usmeritev, je rezultat slika 8.11.

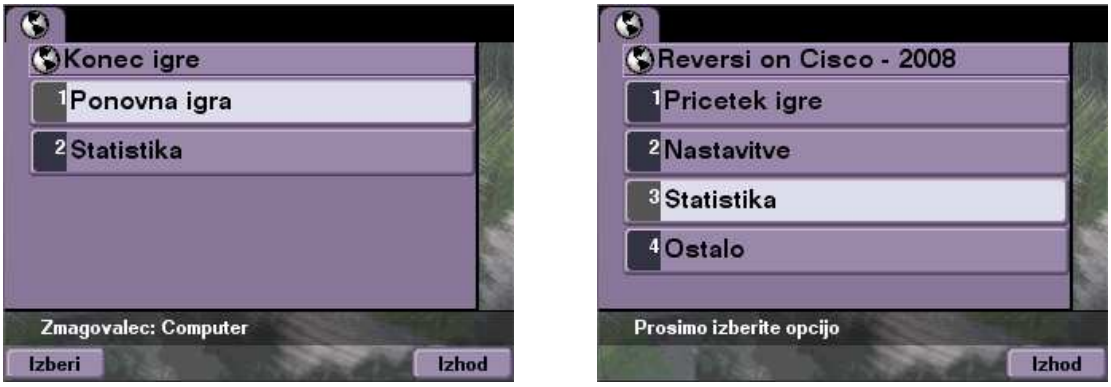


Slika 8.11: Animiran prikaz izvajanja potez z usmeritvijo.

Slika 8.12: Nastavitve za igranje igre z animiranim prikazom izvajanja potez z usmeritvijo.

8.2 Statistični podatki

Po vsaki končani igri se shranijo podatki o igri. Prikaže se končno stanje igre z napisom *"Ni več možnih potez"*, od koder lahko nadaljujemo ali se vrnemo na vstopno stran. V primeru nadaljevanja nam program na dnu zaslona izpiše zmagovalca igre in daje možnost ponovne igre ali pregleda statističnih podatkov (slika 8.13(a)).



(a) (b)

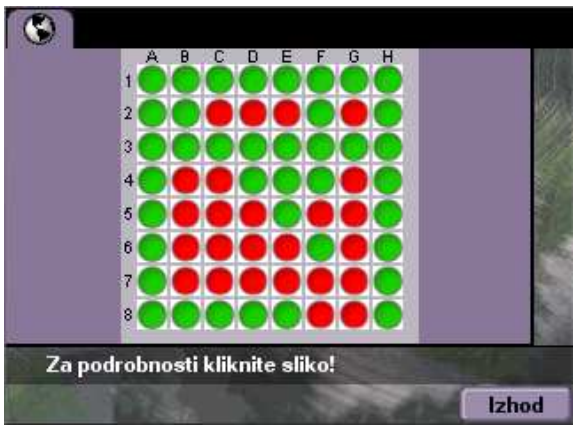
Slika 8.13: Možnost ponovne igre ali pregleda statističnih podatkov.

Če želimo pregledati statističnih podatkov, izberemo menijsko vrstico *statistika*. Enake rezultate vrne menijska vrstica *statistika* na vstopni strani (slika 8.13(b)).

Slika 8.14(a) prikazuje izpis zadnjih sto odigranih iger na tem telefonskem aparatu. Odigrane igre so v obliki časovnih podatkov. Sestavljajo jih datum igranja, čas pričetka igre ter čas trajanja igranja. Kot primer bomo izbrali zadnjo odigrano igro. Ta je na prvem mestu.

Slika 8.14(b) se prikaže po izbrani igri in prikazuje končno stanje odigrane igre. S klikom na igro se prikažejo še ostale podrobnosti o tej igri (slika 8.14(c)).

(a)



(b)

(c)

Slika 8.14: Pregled statističnih podatkov.

8.3 Ostalo

Zadnja menijska vrstica je *ostalo*. Njen pomen pa razkrije slika 8.15.

Slika 8.15: Naša vizitka.

Poglavje 9

Zaključek

V diplomski nalogi smo predstavili igranje igre reversi na Cisco IP telefonskem aparatu. Za realizacijo naše rešitve smo izbrali rešitev telefonskega aparata Cisco IP, saj je bila najbolj primerna za naše potrebe. Pri tem smo uporabljali razširljivi označevalni jezik XML, ki ga razume izbrani telefonski aparat. Problema smo se lotili po strategiji deli in vladaj, katere osnovni rezultati so bili trije segmenti: Cisco IP telefonski aparat, spletni in podatkovni strežnik. Bistveno vlogo ima spletni strežnik, ki komunicira z uporabnikom preko telefonskega aparata Cisco IP in podatkovnim strežnikom za shranjevanje statističnih podatkov. Poleg tega skrbi za pravilen tok igre.

Zaradi dinamike igre smo razvili generator dinamičnih slik. Ta ustvari igralno ploščo z žetoni kot izhodno sliko. Z izhodno sliko in območji občutljivimi na dotik smo omogočili prikaz aktualne igralne plošče na telefonskem aparatu, kjer območja občutljiva na dotik predstavljajo možne poteze uporabnika. Dodatna pomanjkljivost telefonskega aparata je animacija. S kombinacijo večih spletnih strani, tehnike osveževanja in vodenja dodatnega seznama izvedenih potez nam je uspelo simulirati animiran prikaz izvajanja potez. Dodatno smo glede na orientacijo implementirali možnost prikaza usmeritve zavzemanja žetonov.

Nasprotnik oz. umetna inteligence v igri je del motorja. Umetna inteligenca za izbiro potez uporablja iskalni algoritem alfa-beta v kombinaciji z ojačano ocenitveno funkcijo. Osnovno ocenitveno funkcijo smo križali s položajno strategijo. To doprinese večjo možnost odločitve za izbiro robnih kot obrobnih pozicij. S tem smo ojačali moč umetne inteligence, saj nasprotniku skrajno robnih žetonov ni možno zavzeti.

Po končani igri se končno stanje igre z vsemi nastavitvami in rezultati zapiše v podatkovno bazo. Po želji lahko v poljubnem trenutku prikličemo podrobnosti o odigrani igri in si ogledamo končno stanje žetonov na igralni plošči.

Možne nadaljnje raziskave bi bile na področju umetne inteligence. Za iskanje

možnih potez bi lahko uporabili nevronske mreže. Prav tako bi lahko osnovno ocenitveno funkcijo križali z drugimi strategijami. Na področju telefonije bi lahko obstoječo rešitev razširili tako, da bi omogočili sočasno igranje uporabnikov na različnih telefonskih aparatih. Dodatna možnost bi bila, da bi bilo možno igranje igre začasno prekiniti in jo pozneje nadaljevati. Potrebno bi bilo odgovoriti na vprašanja, kako reagira naš program ob dohodnem klicu. V primeru odhodnega klica bi lahko uporabili prejšnjo omenjeno rešitev začasne prekinitve igre.

Podali smo nekaj možnosti nadaljnjih raziskav in zaključili diplomsko delo, v katerem smo prikazali, da je možno igranje igre reversi na telefonskem aparatu za IP telefonijo.

Literatura

[1] "Cisco unified IP phone services application development notes release 6.0(1)," Technical Report, 2007, [Internet http://www.cisco.com/en/US/docs/voice_ip_comm/cuipph/all_models/xsi/6_0/english/programming/guide/xsi.pdf; dostop 28. junij 2008].

[2] "Xml syntax quick reference," 2006, [Internet http://www.mulberrytech.com/quickref/XMLquickref.pdf; dostop 2. junij 2008].

[3] D. B. Fogel, ''Evolutionary Computation: Toward a New Philosophy of Machine Intelligence,'' 3rd izd., Wiley-IEEE Press, 2006.

[4] S. Lucas in G. Kendall, "Evolutionary computation and games," *Computational Intelligence Magazine, IEEE*, letn. 1, št. 1, str. 10–18, 2006.

[5] S. Chong, M. Tan, in J. White, "Observing the evolution of neural networks learning to play the game of Othello," *Evolutionary Computation, IEEE Transactions on*, letn. 9, št. 3, str. 240–251, 2005.

[6] N. van Eck in M. van Wezel, "Application of reinforcement learning to the game of Othello," *Computers and Operations Research*, letn. 35, št. 6, str. 1999–2017, 2008.

[7] S. Lucas in T. Runarsson, "Temporal Difference Learning Versus Co-Evolution for Acquiring Othello Position Evaluation," *IEEE Symposium on Computational Intelligence and Games*, 2006, [Internet http://www3.hi.is/~tpr/papers/SiRu06.pdf; dostop 22. avgust 2008].

[8] M. Buro, "From simple features to sophisticated evaluation functions," v *Computers and Games, Proceedings of CG98, LNCS 1558.* Springer-Verlag, 1998, str. 126–145.

[9] M. Buro, "Statistical Feature Combination for the Evaluation of Game Positions," *Journal of Artificial Intelligence Research*, letn. 3, str. 373–382, 1995,

[Internet http://arxiv.org/PS_cache/cs/pdf/9512/9512106v1.pdf; dostop 23. avgust 2008].

[10] M. Buro, "The Othello match of the year: Takeshi Murakami vs. Logistello," *ICCA Journal*, letn. 20, št. 3, str. 189–193, 1997.

[11] J. V. Neumann, "Zur theorie der gesellschaftsspiele," *Mathematische Annalen*, letn. 100, št. 1, str. 295–320, 1928.

[12] M. Luštrek, "Računalniško igranje iger s kartami," Diplomska naloga, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2002.

[13] M. Vuurboom, "Learning othello using neuroevolution and temporal difference learning," Master thesis, 2008, [Internet http://people.cs.uu.nl/marco/thesis_vuurboom.pdf; dostop 14. avgust 2008].

[14] T. Andersen, K. Stanley, in R. Miikkulainen, "Neuro-Evolution Through Augmenting Topologies Applied To Evolving Neural Networks To Play Othello," Department of Computer Sciences, University of Texas at Austin, Technical Report, Technical Report, 2002, [Internet http://www.cs.utexas.edu/ftp/pub/techreports/tr02-32.pdf; dostop 16. julij 2008].

[15] D. Billman in D. Shaman, "Strategy knowledge and strategy change in skilled performance: A study of the game othello," *American Journal of Psychology*, letn. 103, št. 2, str. 145–166, 1990.

[16] T. Marsland, "A review of game-tree pruning," *ICCA Journal*, letn. 9, št. 1, str. 3–19, 1986, [Internet http://www.cs.ualberta.ca/~tony/OldPapers/1986review.pdf; dostop 15. julij 2008].

[17] S. Worley, *Inside ASP.Net*. Sams Publishing, 2002.

Priloge

– Zgoščenka z diplomsko nalogo v elektronski obliki.