Programska implementacija osnovnih algoritama teorije igara

Kedačić, Branimir

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

PROGRAMSKA IMPLEMENTACIJA OSNOVNIH ALGORITAMA TEORIJE IGARA

Diplomski rad

Branimir Kedačić

Osijek, 2019.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 22.09.2019.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Branimir Kedačić
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-932R, 24.09.2018.
OIB studenta:	72236296577
Mentor:	Doc.dr.sc. Tomislav Rudec
Sumentor:	Izv.prof.dr.sc. Tomislav Keser
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Izv. prof. dr. sc. Davor Vinko
Član Povjerenstva:	Doc.dr.sc. Anita Katić
Naslov diplomskog rada:	Programska implementacija osnovnih algoritama teorije igara
Znanstvena grana rada:	**Procesno računarstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	Student će opisati i zapisati u programskom jeziku osnovne algoritme iz teorije igara. Tema je zauzeta - Branimir Kedačić
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Vrlo dobar (4)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 1 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	22.09.2019.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 30.09.2019.

Ime i prezime studenta:	Branimir Kedačić
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-932R, 24.09.2018.
Ephorus podudaranje [%]:	8

Ovom izjavom izjavljujem da je rad pod nazivom**: Programska implementacija osnovnih algoritama teorije igara**

izrađen pod vodstvom mentora Doc.dr.sc. Tomislav Rudec

i sumentora Izv.prof.dr.sc. Tomislav Keser

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA

Ja, Branimir Kedačić, OIB: 72236296577, student/ica na studiju: Diplomski sveučilišni studij Računarstvo, dajem suglasnost Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek da pohrani i javno objavi moj **diplomski rad**:

Programska implementacija osnovnih algoritama teorije igara

u javno dostupnom fakultetskom, sveučilišnom i nacionalnom repozitoriju.

Osijek, 03.10.2019.

_____

potpis

Sadržaj

1. UVOD

Teorija igara je matematička disciplina koja se pojavljuje u prvoj polovici 20. stoljeća i bavi se preciznom analizom igara. Temelji se na proučavanju matematičkih modela interakcije i sukoba između dva ili više sudionika. Jedna od glavnih pretpostavki u teoriji igara je da su igrači racionalni i kao takvi će se proučavati u radu. U igrama strategije ishod ne ovisi samo o postupcima jednog sudionika, on ovisi i o tome kako će drugi sudionici reagirati na određenu akciju. Svaki sudionik za cilj može imati maksimiziranje dobitka, minimiziranje rizika, gubitka ili nanošenje što veće štete drugim sudionicima. Uz teoriju igara je usko vezan i proces odlučivanja, tj. kako odabrati akciju koja će dovesti do željenog stanja, te kako ona utječe na druge sudionike. Teorija igara ima širok spektar primjena u raznim granama, kao što su psihologija, informacijske tehnologije, filozofija, politika, ekonomija, vojne i računalne znanosti.

1.1. Zadatak diplomskog rada

Zadatak diplomskog rada je proučiti područje teorije igara. Potrebno je istražiti i opisati osnovne algoritme teorije igara na primjerima. Zatim na temelju proučenih materijala o igrama u programskom jeziku implementirati navedene igre s traženim funkcionalnostima. Nakon toga testirati njihovu ispravnost kako bi se osiguralo pravilno funkcioniranje aplikacije.

2. TEORIJSKA PODLOGA

U ovom poglavlju će se opisati razvoj teorije igara kroz povijest, reći ponešto o tome gdje se danas nalazi teorija igara, navesti osnovni primjer teorije, a zatim proučiti koji svi tipovi igara i strategija postoje.

2.1. Razvoj teorije igara

Temelji teorije igara datiraju još od 5. stoljeća nove ere, ali je, prema [1] tek 1713. godine James Waldegrave formulirao pojam minimalne strategije za dvije osobe, za igru s kartama zvanu Le Her. U narednim godinama još znanstvenika je u svojim radovima spominjalo pojam teorije igara, ali je prema [2] tek 1928. godine matematičar John von Neumann dokazao minimax teorem u svom članku „Zur Theorie der Gesellschaftsspiele“. 1944. godine John von Neumann i ekonomist Oskar Morgenstern objavljuju knjigu „Theory of Games and Economic Behavior“ u kojoj se povezuje ekonomija i teorija igara. Svoj doprinos razvoju daje i John Nash, koji između 1950. i 1953. godine u svoja dva rada „Equilibrium Points in N-Person Games“ i „Non-cooperative Games“ obrađuje spomenuto područje. U njima je dokazao da za nekooperativne igre postoji strateška ravnoteža, koja dobiva ime po njemu *Nash equilibrium*, te predlaže izučavanje kooperativnih igara na način da se one svedu na oblik ne-kooperativnih igara. 1966. godine John Harsanyi u svom radu „A General Theory of Rational Behavior in Game Situations“ daje detaljne definicije za razlikovanje kooperativnih i nekoonoperativnih igara. Reinhard Selten 1975. godine u članku „Reexamination of the Perfectness Concept for Equilibrium Points in Extensive Games“ detaljnije proučava i precizira ideju savršene ravnoteže. Prema [3], o neprestanom rastu i razvoju teorije igara, te njezinoj primjeni u raznim područjima znanosti svjedoči i 12 Nobelovih nagrada dodijeljenih znanstvenicima koju su kroz svoje radove postavili temelje i poboljšali razumijevanje onoga što danas nazivamo teorijom igara.

Definicije teorije igara različitih autora:

„Proučavanje matematičkih modela sukoba i suradnje inteligentnih donositelja odluka.“ - Roger B. Myerson

„Teorija igara bavi se analizom kako racionalni pojedinci donose odluke kada su međusobno ovisni.“ – Graham Romp

„To je opći okvir koji pomaže u donošenju odluka kada čvrste isplate ovise o aktivnostima koje poduzimaju druge tvrtke“ – McGraw Hill

2.1.1. Trenutno stanje teorije igara

Teorija igara se posljednjih godina poprilično razvila te je prisutna u širokom spektru grana znanosti. Najveću primjenu ipak ima u ekonomiji i političkim znanostima. U području ekonomije, ima neizostavnu ulogu zbog konstantnog rasta i razvoja postojećih tvrtki i pojave novih. Tu se primarno bavi odnosima konkurentskih ponašanja, stanjima na tržištu i određivanju cijena, koje se mogu modelirati teorijom igara. U političkim znanostima bavi se proučavanjem poštene podjele, te područja javnog mišljenja. Prema [4] u zadnje vrijeme je također zanimljiva primjena teorije igara u računarstvu gdje se počela razvijati sa strojnim učenjem, i umjetnom inteligencijom. Neke od primjena su u višeagencijskom potpomognutom učenju (MARL), i generativnim mrežama (GANs).

2.2. Reprezentacija igara

Za lakše razumijevanje samih igara potrebno je definirati nekoliko standardnih pojmova iz teorije igara. Prema [5], oni se mogu opisati na sljedeći način:

- **Igra:** svaki skup okolnosti čiji rezultat ovisi o postupcima dva ili više igrača, najčešće su to suprotstavljene strane. Svaka igra je određena pravilima.
- **Igrači:** donositelji strateških odluka s aspekta igre, oni su sastavni dio svake igre.
- **Strategija:** plan akcije koji igrač može poduzeti uzimajući u obzir okolnosti koje se mogu pojaviti u igri. Svaki igrač pokušava primijeniti strategiju koja će mu osigurati najveći dobitak.
- **Dobitak:** iznos koji prikazuje koliko igrač može dobiti ili izgubiti na kraju igre. Dobitak može biti prikazana u svim oblicima koje je moguće kvantificirati.
- **Skup informacija:** informacije koje su na raspolaganju igračima u određenom trenutku u igri.
- **Ravnoteža:** trenutak u igri kada je postignut neki rezultat na osnovi odluka koje su igrači donijeli.

2.2.1. Normalni forma

Normalna forma igre se ostvaruje kroz matricu koja prikazuje igrače, strategije i dobitke.

Tablica 2.1. Normalna forma igre.

	Igrač 2 bira lijevi stupac	Igrač 2 bira desni stupac
Igrač 1 bira prvi red	50 , 25	-10 , -10
Igrač 1 bira drugi red	0 , 0	25 , 50

U prikazanoj matrici prikazana su dva igrača, Igrač 1 i Igrač 2. Svaki od njih može odabrati između dvije strategije, Igrač 1 bira red, a Igrač 2 bira stupac. Mogući dobitci su prikazani u unutrašnjosti matrice. Prvi broj označava dobitak/gubitak Igrača 1, a drugi Igrača 2.

Primjer 1. Igrač 1 odabire prvi red, Igrač 2 odabire prvi stupac, te im se slijedno dodjeljuje dobitak u iznosu od 50 i 25.

Primjer 2. Igrač 1 odabire drugi red, Igrač 2 odabire drugi stupac, te im se slijedno dodjeljuje dobitak u iznosu od 25 i 50.

2.2.2. Zatvorenikova dilema

Prema [6] zatvorenikova dilema je najpoznatiji primjer teorije igara kreirana od strane znanstvenika Merrilla Flloda i Melvina Dreshera, a kasnije ju je usavršio Albert William Tucker. Igra se koristi za analizu odluka, te je pomoću nje opisan odnos između suradnje i/ili sukoba. Ilustrira kako gledanje vlastitog probitka, ne znači uvijek i optimalan ishod.

Scenarij: Dvije osobe, osumnjičenje za pljačku su odvedene u zatvor. Međutim, policija nema dovoljno dokaza kako bi ih osudila za taj zločin, već ih mogu teretiti samo za novce pronađene kod njih. Svaki zatvorenik se nalazi u zasebnoj sobi za ispitivanje, te nemaju mogućnost međusobne komunikacije. Obojici zatvorenika se daje do znanja da se sumnja da je pljačkaš i da se ispituju zasebno. Policija pokušava uvjeriti svakog zatvorenika da prizna zločin, te im se daje do znanja da ako:

a) Oba zatvorenika priznaju zločin, svaki će biti osuđen na 5 godine zatvora.
b) Samo jedan zatvorenik prizna zločin on će biti oslobođen, a drugi će biti osuđen na 10 godina zatvora.
c) Oba zatvorenika ne priznaju, svaki će biti osuđen na 2 godine zatvora.

Tablica 2.2. Matrični prikaz zatvornikove dileme

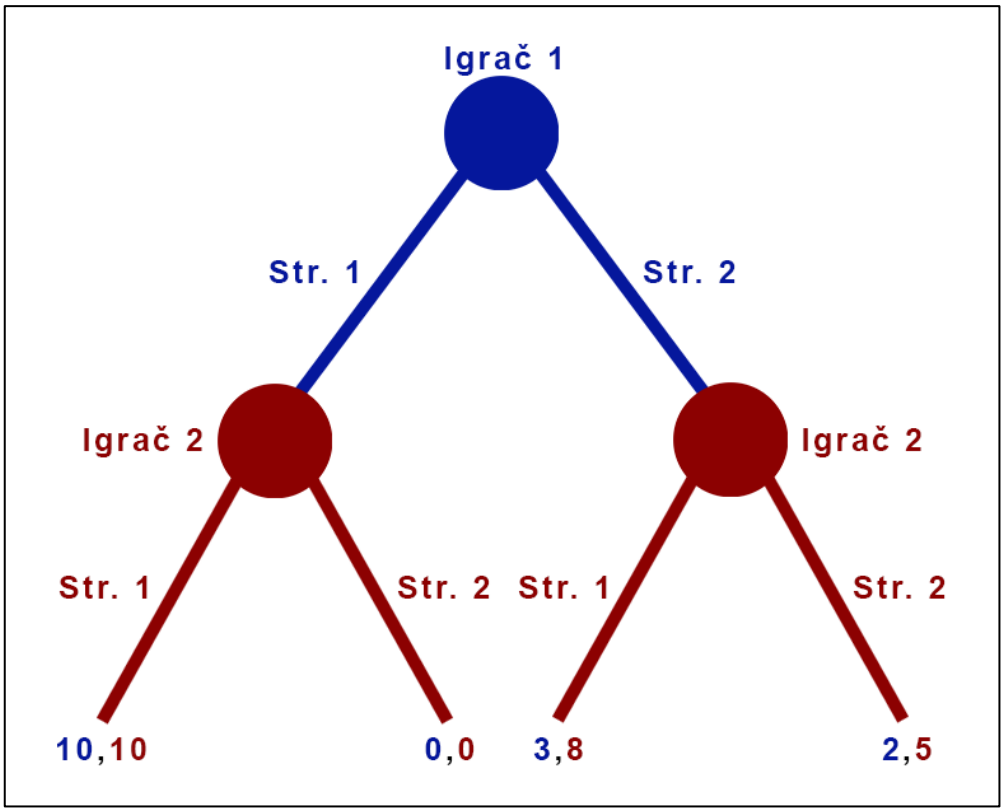
	Zatvorenik 2 šuti	Zatvorenik 2 priznaje
Zatvorenik 1 šuti	2 , 2	10 , 0
Zatvorenik 1 priznaje	0 , 10	5 , 5

Glavno pitanje je zatvorenikove dileme je, surađivati ili ne surađivati? Gledajući ponuđene opcije, vidi se da postoje četiri ishoda. Najbolja opcija bi bila kada bi se oba zatvorenika branila šutnjom i tako dobila zatvorsku kaznu od 2 godine. Ali pošto niti jedan od zatvorenika ne može sa sigurnošću znati što će onaj drugi odlučiti, najpovoljnija opcija bi bila priznati zločin i tako odbaciti mogućnost dobivanja zatvorske kazne od 10 godina, ali završavajući sa zatvorskom kaznom od 5 godina.

Prema [7] zatvorenikova dilema je važna zbog svoje primjene u stvarnom životu. Ona je podsjetnik da suradnja nije uvijek najbolja opcija, jer može dovesti do loših posljedica za pojedinca ako suprotna strana u vidu ima samo vlastiti interes, i nema namjeru surađivati. Zato je bitno da svaki igrač ima razvijenu strategiju, o kojoj će se više govoriti u sljedećem poglavlju.

2.2.3. Ekstenzivni oblik

Ekstenzivni oblik, koristi se za prikaz igara u kojima postoji određeni redoslijed. Opis igre se prikazuje stablom odlučivanja, koje se sastoji od čvorova i grana. Čvorovi predstavljaju igrače, a grane predstavljaju poteze koje igrači mogu odigrati.



Slika 2.3. Stablo prikaz ekstenzivnog oblika

Na slici 1. je prikazana igra između dva igrača u ekstenzivnom obliku. Prvi broj označava dobitak/gubitak Igrača 1, a drugi Igrača 2.

Primjer 1. Igrač 1 je prvi na potezu i odabire Strategiju 1, zatim na red dolazi Igrač 2 koji odabire također odabire Strategiju 1, što rezultira dobitkom od 10 za svakog igrača.

Primjer 2. Igrač 1 odabire Strategiju 2, zatim na red dolazi Igrač 2, koji odabire Strategiju 1, što rezultira dobitkom od 3 za Igrača 1, i dobitkom od 8 za Igrača 2.

2.3. Vrste strategija

Šire objašnjenje strategije može se opisati kao bilo koja opcija koju igrač može odabrati u okruženju gdje rezultat ne ovisi samo o njegovom odabiru, nego i o odabiru njegovog protivnika. Strategija se ponekad miješa s potezom, iako su to dva različita koncepta. Potez je radnja koju igrač može odabrati u trenutku kada je njegov red za igranje, dok je strategija algoritam za igranje igre koji igraču govori što može učiniti za bilo koju situaciju koja se može ostvariti tijekom igre. Prema [8] igrač se može odlučiti na korištenje jedinstvene strategije koja mu pruža najveći dobitak ili najmanji gubitak i primjenjivati ju kroz cijelu igru, ili može koristiti više različitih strategija

koje smatra povoljnima. Postroje različite vrste strategija koje će biti opisane u sljedećim poglavljima, a neke od njih su čista, mješovita, maximin, minimax, dominanta i dominirana strategija.

2.3.1. Čista i mještovita strategija

Čista strategija je definirani odabir koji igrač koristi tijekom igre. Igrač može na raspolaganju imati više čistih strategija, te se odlučiti na praćenje samo jedne. Prema [8], to je praćenje one strategije koja pruža maksimalni dobitak. U mješovitoj strategiji, igrač odabire između različitih strategija kako bi osigurao najbolji ishod za sebe. U osnovi, mješovita strategija je skup čistih strategija kojima se dodjeljuje vjerojatnost. Na taj način igrač koji koristi takvu strategiju izabire neku od ponuđenih čistih strategija nasumično ili ih izabire na osnovu definiranih vjerojatnosti.

2.3.2. Dominantna i dominirana strategija

Prema [9], dominantna strategija je najbolja strategija koju igrač može izabrati u skupu svih strategija, bez obzira na to koju strategiju će odabrati protivnik. Gledajući sa strateškog stajališta ona igraču osigurava najbolji dobitak uzimajući u obzir sve dostupne informacije. Postoje dvije vrste dominantne strategije, strogo dominantna i slabo dominantna. Dominirana strategija je ona koja igraču nudi manji dobitak u usporedbi s drugim, i također se ne obazire na to koju će strategiju odabrati protivnik. Svrha ove strategije je detekcija najlošije opcije kako bi se ona mogla izbaciti. Kao i kod dominantne, postoje strogo i slabo dominirana strategija.

2.3.3. Maximin i minimax strategija

Prema [10], Maximin je strategija maksimiziranja minimalnog dobitka. Koristi se u slučajevima kada postoji određena doza neizvjesnosti, te se tako pokušava smanjiti stupanj rizika. Svoju primjenu nalazi u natjecanju za tržište, i koristi se kada se smatra da druga strana neće igrati racionalno. Ova strategija se još naziva i konzervativna. Minimax strategija se oslanja na korištenje koncepta matrice dobitka, koja prikazuje odnos mogućeg gubitka i dobitka. Ovom strategijom se pokušava minimizirati najgori mogući gubitak.

2.4. Kategorizacija igara

Prema [11] u teoriji igra različiti tipovi igara pomažu u obradi različitih problema. Tipovi igara formiraju se na osnovi broja sudionika, suradnji između članova, simetriji.

2.4.1. Kooperativne i nekooperativne igre

Podjela igara u teoriji se dijeli na dvije osnovne skupine, kooperativne i nekooperativne. U kooperativnim igrama igrači ili sudionici dogovorom i pregovaranjem dolaze do sporazuma i zajedničke strategije, dok u nekooperativnim nema pregovaranja i svaka strana primjenjuje svoju strategiju kako bi sebi osigurala najveći dobitak. Najbolji primjer ovakve vrste igre je tržišni oligopol, gdje se više poduzeća natječu za udio na tržištu. Poduzeća se mogu odlučiti na suradnju i zajedničkim dogovorom odrediti cijenu nekog proizvoda i jednako konkurirati na tržištu. Kontradiktorno tome, jedno od poduzeća se može odlučiti da ne želi surađivati i sniziti cijenu proizvoda, čime će zaokupiti veći postotak tržišta, ali tako također može pokrenuti cjenovni rat sa svojim konkurentima.

2.4.2. Slijedne i simultane igre

Prema [11], igre također možemo podijeliti prema vrsti poteza. Simultane igre su one u kojima igrači donose odluke istovremeno. U ovakvoj vrsti igre, niti jedan igrač nema saznanja o tome što će onaj drugi učiniti ili kakvu će strategiju primijeniti. Simultane igre se opisuju matricom, tj. normalnom formom. Primjer simultane igre je već navedena zatvorenikova dilema. Suprotno simultanom, u slijednom obliku igrači igraju jedan za drugim. Igrač koji je sljedeći na redu, uvijek ima određenu količinu informacija o akciji prethodnog igrača. To ne mora biti potpuna informacija o svim akcijama koje je napravio, ali može npr. znati da igrač ne koristi samo jednu strategiju. U ovoj vrsti igre postoji takozvana, prednost prvog igrača jer on može prvi odabrati strategiju. Primjer slijedne igre je šah, gdje sljedeći igrač zna koji potez je odigrao prethodni i na temelju toga može odrediti svoju strategiju. Slijedne igre se prikazuju u ekstenzivnom obliku.

2.4.3. Simetrične i asimetrične igre

Prema [11], simetričnom igrom se naziva ona u kojoj su uloge igrača zamjenjive. U takvim igrama, dobitci ovise samo o strategijama primijenjenim od strane igrača, i nijedan igrač nema prednost nad drugim. Kada bi se dva igrača zamijenila za pozicije, a ishod igre ostao isti, tada se može reći

da je igra simetrična. Primjer ovakve igre je škare-papir-kamen, koja će biti obrađena u sljedećim poglavljima, jer za bilo koju strategiju odabranu od strane jednog igrača, drugi može odabrati istu, te nijedan ne može računati na to da će uzastopno pobjeđivati. S druge strane, u asimetričnim igrama, strategije koje igrači koriste nisu jednake za oba igrača. Strategija koja pruža najbolji dobitak za jednog igrača, ne znači da će i drugom igraču pružiti najbolji dobitak. Jedan od primjera ovakve igre je Bombardiranje [10], koja će također biti detaljnije proučena.

2.4.4. Igre s nultom i promjenjivom sumom

Prema [12], igra s nultom sumom se definira kao igra u kojoj je zbroj rezultata jednog ili više igrača jednak nuli. U ovakvoj vrsti igre postoje samo dobitnici i gubitnici, jer zbroj svih dobitaka u igri je jednak zbroju svih gubitaka za svaki potencijalni ishod, što znači da jedan igrač može ostvariti korist samo na štetu drugih. Bez obzira kakvu strategiju igrači primijenili, nemaju mogućnost utjecanja na resurse pa tako ni na promjenu konačnog ishoda. Ovakve igre su gotovo uvijek natjecateljske, primjer takve igre opet je šah, iz čega se može zaključiti da jedna igra može sadržavati karakteristike više tipova igara. U igrama s promjenjivom sumom zbroj rezultata tijekom igre može biti i različit od nule. Dobitak jednog igrača ne mora nužno značiti gubitak za drugog, zato je ovo malo složeniji oblik igre. Za razliku od igara s nultom sumom, u ovoj vrsti igre igrači se ne moraju samo međusobno natjecati već mogu i surađivati, te u ovisnosti kolika im je količina informacija dostupna, primijenjene strategije mogu varirati. Valja napomenuti da se svaka igra s promjenjivom sumom može pretvoriti u igru s nultom sumom na način da se doda još jedan imaginarni igrač čiji će gubitci kompenzirati zaradu drugih igrača. Primjer ovakve igre je zatvorenikova dilema.

3. PRIMJERI IGARA

U ovom radu proučit će se četiri igre, Pismo - glava, Šakre - papir - kamen, Šije - šete i Bombardiranje. Kroz te igre će se opisati kojoj vrsti one pripadaju, kakve strategije igrači mogu primijeniti kako bi sebi osigurali najveću korist, te matematički prikazati vjerojatnosti. Ove igre će također služiti kao temelj za programski dio rada gdje će one biti implementirane.

3.1. Pismo – glava

Prema [10], Pismo – glava je najjednostavnija simetrična igra u teoriji igara u kojoj sudjeluju dva igrača. Ovo je igra s nultom sumom jer dobitak jednog igrača je ostvaren gubitkom drugog. Igra se može igrati u dvije varijante, okretanjem jednog novčića od strane jednog igrača ili svaki igrač okreće svoj novčić. Ovdje će se opisati drugi oblik, a programski implementirati prvi.

Igra: Dva igrača, Igrač 1 i Igrač 2, istovremeno pokazuju i stavljaju svoj novčić na stol. Pobjednik ovisi o tome da li se strane podudaraju ili razlikuju. Ako oba igrača pokažu pismo ili glavu, Igrač 1 je pobjednik, a ako pokažu različite strane, Igrač 2 je pobjednik.

Igra se prikazuje matricom dobitka.

Tablica 3.1. Matrica dobitka za igru Pismo – glava.

	Igrač 2		
Igrač 1	Novčić	Pismo	Glava
	Pismo	+1 , -1	-1 , +1
	Glava	-1 , +1	+1 , -1

Primjer 1: Igrač 1 je pokazao pismo, Igrač 2 je pokazao glavu. Ovakav ishod igre rezultira time da Igrač 1, gubi i daje svoj novčić Igraču 2.

Primjer 2: Igrač 1 je pokazao glavu, Igrač 2 je također pokazao glavu. Igra završava pobjedom Igrača 1 koji dobiva novčić Igrača 2.

U ovakvoj igri niti jedan igrač nema prednost na drugim, štoviše ako se igra igra samo jednom nijedan od igrača ne može primijeniti strategiju koja bi mogla poboljšati njegovu poziciju u igri. Čak i u slučaju da se igra više puta, opet ne postoji čista strategija koja garantira pobjedu. Ako su igrači racionalni, što znači da svaki igrač želi ostvariti najveću korist za sebe, najbolja opcija je da svaki igrač primjeni mješovitu strategiju i nasumično odabire između pisma i glave u omjeru 50%-50%. U tom slučaju, prema [13], govorimo o Nash-ovoj ravnoteži za mješovitu strategiju, koja kaže da ako oba igrača koriste mješovitu strategiju, niti jedan nema razloga mijenjati strategiju.

3.2. Škare – papir – kamen

Kao i prošla, i ovo je simultana igra s nultom sumom. Prema [10], svaki od igrača može odabrati jednu od tri ponuđene opcije, pa se postavlja pitanje da li se u ovoj igri može uspostaviti dominantna strategija koja bi osigurala prednost na drugim igračem.

Igra: Dva igrača, Igrač 1 i Igrač 2 simultano prikazuju ruke u jednom od tri ponuđena oblika. Svaki može prikazati „škare“ (ispruženi kažiprst i srednji prst), „papir“ (dlan i prsti okrenuti prema dolje), „kamen“ (sklopljena šaka). Bodovanje se vrši na temelju prikazanih oblika, u smislu da škare pobjeđuju papir, papir pobjeđuje kamen, a kamen pobjeđuje škare.

Igra se prikazuje matricom dobitka.

Tablica 3.2. Matrica dobitka za Škare – papir – kamen.

		Igrač 2		
Igrač 1		Škare	Papir	Kamen
	Škare	0 , 0	1 , -1	-1 , 1
	Papir	-1 , 1	0 , 0	1 , -1
	Kamen	1 , -1	-1 , 1	0 , 0

U ovoj igri postoje dva moguća ishoda. Pobjeda jednog igrača i gubitak drugog, ili neriješeni rezultat. Iz prikazane tablice može se vidjeti da ni u ovoj igri ne postoji Nash-ova ravnoteža za čistu strategiju jer ne postoji opcija u kojoj oba igrača imaju najbolji odgovor na opciju drugog igrača. To znači da nije moguće ostvariti prednost nam protivnikom koji doista nasumično bira svoj potez. Pojedini igrač se može odlučiti na korištenje strategije npr, prikazivanja samo „škara“, i iako će mu to možda donijeti pobjedu sljedećih nekoliko partija, nakon više odigranih drugi igrač će zasigurno shvatiti o čemu se radi, i početi pokazivati samo „kamen“.

Radi lakšeg razumijevanja matricom vjerojatnosti će se prikazati primjer kada Igrač 1 koristi strategiju (1/3, 1/3, 1/3), a Igrač 2 strategiju (1/2, 1/4, 1/4).

Tablica 3.3. Matrica vjerojatnosti za odabrane strategije.

		Igrač 2		
Igrač 1		Škare	Papir	Kamen
	Škare	$\frac{1}{6}$	$\frac{1}{12}$	$\frac{1}{12}$
	Papir	$\frac{1}{6}$	$\frac{1}{12}$	$\frac{1}{12}$
	Kamen	$\frac{1}{6}$	$\frac{1}{12}$	$\frac{1}{12}$

Također matricom dobitka će se prikazati dobitak i gubitak gledajući s strane Igrača 1.

Tablica 3.4. Matrica dobitka Igrača 1.

		Igrač 2		
Igrač 1		Škare	Papir	Kamen
	Škare	0	1	-1
	Papir	-1	0	1
	Kamen	1	-1	0

Kako bi izračunali koliko bodova u prosjeku će Igrač 1 osvojiti po rundi, potrebno je pomnožiti i zbrojiti vrijednosti iz predočenih tablica.

$$\frac{1}{6}*0+\frac{1}{12}*1+\frac{1}{12}*(-1)+\frac{1}{6}*(-1)+\frac{1}{12}*0+\frac{1}{12}*1+\frac{1}{6}*1+\frac{1}{12}*(-1)+\frac{1}{12}*0=0$$

Iz izračuna se vidi da će Igrač 1 u prosjeku osvojiti 0 bodova, što znači da će za primijenjene strategije ishod većinom biti neriješen. Ove strategije ne tvore Nash-ovu ravnotežu jer Igrač 1 može promijeniti svoju strategiju kako bi si osigurao veći postotak pobjede. To može ostvariti na način da primjeni strategiju (1/4, 1/2, 1/4).

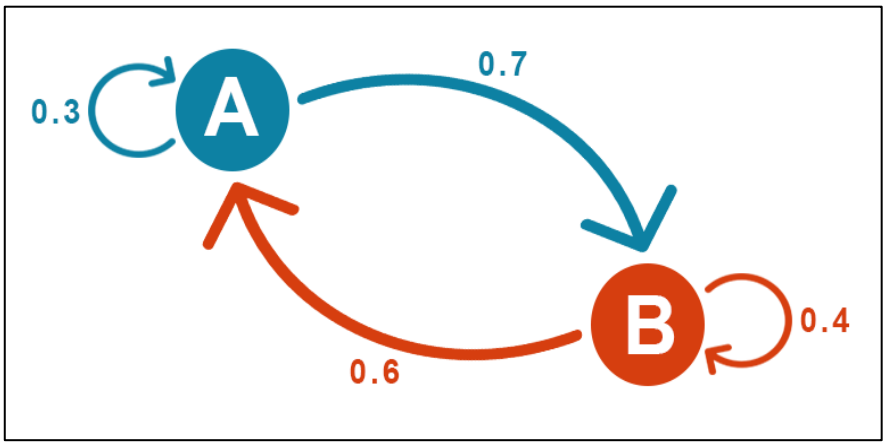
Tablica 3.5. Matrica vjerojatnosti odabrane strategije.

	Igrač 2			
Igrač 1		Škare	Papir	Kamen
	Škare	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{16}$
	Papir	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{8}$
	Kamen	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{16}$

$$\frac{1}{8}*0+\frac{1}{16}*1+\frac{1}{16}*(-1)+\frac{1}{4}*(-1)+\frac{1}{8}*0+\frac{1}{8}*1+\frac{1}{8}*1+\frac{1}{16}*(-1)+\frac{1}{16}*0=\frac{1}{16}$$

Izračunom se vidi da će u slučaju primjene ove strategije Igrač 1 na kraju svake runde zaraditi 1/16 boda. Ovo također ne tvori Nash-ovu ravnotežu jer sada Igrač 2 ima razloga promijeniti svoju strategiju. Matematički je zato kao i u prethodnoj igri najbolje odabrati strategiju nasumičnog pokazivanja svakog znaka u omjeru 1/3, te tako protivnik neće moći predvidjeti sljedeći potez. Pošto ljudi obično nisu 100% nasumični u svom odabiru, moguće je iskoristiti tu karakteristiku. Prema [14] postoje natjecanja gdje se pokušavaju razviti algoritmi koji osiguravaju najveći postotak pobjede. Iako se čini kao trivijalna igra, zapravo uključuje problem prepoznavanja uzoraka. Prepoznavanje uzoraka je jedan od problema s kojima se susreće strojno učenje i umjetna inteligencija. Na taj način algoritmi pokušavaju pretpostaviti koji će sljedeći potez igrač odigrati. Jedan od načina rješavanja tog prolema je korištenje Markovljevog lanca. To je sustav koji se

sastoji od više različitih stanja, i može prijeći iz jednog u drugo uz određenu vjerojatnost. Svako stanje ima dvije moguće tranzicije, može se vratiti u sebe ili prijeći u novo stanje, a suma tih vjerojatnosti mora biti jednaka 1. Pomoću njega se pokušava na temelju prošlog stanja predvidjeti sljedeće.



Slika 3.6. Prikaz Markovljevog lanca

Srž ovakve igre se često koristi i u razvoju računalnih igara, gdje se na takav kružni način kreiraju grupe od kojih svaka ima svoje prednosti i nedostatke. Na taj način se postiže ravnoteža i sprječava nadmoć jedne grupe nad drugom.

3.3. Šije – šete

Prema [10], Igra Šije - šete je također simetrična s nultom sumom. Kao i u prošlim igrama proučava se koja je strategija optimalna za igrača. Za potrebe proučavanja, objašnjenje će se temeljiti na jednostavnijoj verziji igre s dva prsta, iako će se u programskoj implementaciji kreirati igra s više opcija.

Igra: Dva igrača, Igrač 1 i Igrač 2, istovremeno pokazuju jedan ili dva prsta, i izgovaraju broj za koji misle da će biti zbroj pokazanih prstiju. Igrač koji pogodi zbroj odnosi bodova koliko je iznosio zbroj. U slučaju da nijedan ne pogodi točno, igra je neriješena.

Tablica 3.7. Matrica dobitka za igru Šije – šete.

	Igrač 2				
Igrač 1	Prsti , izgovor	1 , 2	1 , 3	2 , 3	2 , 4
	1 , 2	0	2	-3	0
	1 , 3	-2	0	0	3
	2 , 3	3	0	0	-4
	2 , 4	0	-3	4	0

U tablici nije naveden slučaj kada igrač pokaže i izgovori isti broj, jer takav potez nema smisla. Smisleno je da igrač uvijek izgovori broj za bar jedan veći od pokazanog. Prema tome, svaki od igrača ima četiri opcije, i iako po opisu ista kao i prethodne, ova igra je ipak malo drugačija što se tiče strategije. U igrama dosad, ispostavilo se da je najbolja strategija nasumično odabiranje jedne od ponuđenih opcija, ali ovdje to nije slučaj, te odabir svake od opcije 1/4 puta nije optimalna strategija. Za primjer ćemo uzeti da Igrač 1, primjeni strategiju (1/4, 1/4, 1/4, 1/4), tada bi Igrač 2, za slučaj kada se igra više puta, mogao primjenom (0, 1, 0, 0) strategije osigurati veći dobitak. To se može dokazati izračunom.

$$\frac{1}{4} * 2 + \frac{1}{4} * 0 + \frac{1}{4} * 0 + \frac{1}{4} * (-3) = -\frac{1}{4}$$

Iz priloženog se vidi da će Igrač 2, ostvariti dobitak od 1/4 po rundi. Zato je ova igra drugačija od prethodnih, jer iako je igra simetrična, i svaki igrač primjenom dugoročne strategije očekuje najbolji ishod za sebe, teško je odrediti koja je to strategija. Za slučaj kada bi igrači primijenili navedene strategije, Igrač 1, nakon što shvati strategiju Igrača 2, bi promijenio strategiju u (1, 0 , 0, 0) i tako stavio sebe u vodeću poziciju i dobitak od 2. Igrač 2 bi na to opet promijenio svoju strategiju u (0, 0, 1, 0) za dobitak od 3 po rundi. Igrač 1 na to može odgovoriti sa (0, 0, 0, 1) za dobitak od 4, na što Igrač 2 opet mijenja svoju strategiju u (0, 1, 0, 0). Nakon ovog se može

ustanoviti da napravljen krug i igrači su se vratili u početno stanje. To se naziva logička petlja i jedan je od problem koje teorija igara nastoji riješiti.

3.4. Bombardiranje

Bombardiranje je igra s nultom sumom u kojoj sudjeluju dva igrača. U igri se, prema [10], proučava kakav ishod daju različite strategije, te kako maksimizirati dobitak ili minimizirati gubitak za različite slučajeve.

Igra: Igrač 1 i Igrač 2 su zapovjednici u vojskama koje vode rat. Svaki dan Igrač 1, šalje eskadrilu aviona da napadnu položaje Igrača 2. Eskadrila se sastoji od jednog bombardera i jednog manjeg pomoćnog aviona. Zadaća tih aviona je da bace jednu bombu i time unište vojsku Igrača 2. Kako bi to spriječio, Igrač 2 ima jedan borbeni avion, koji u zasjedi čeka priliku za napad. Ako borbeni avion napadne bombardera Igrača 1, bombarder ima 80% šanse da preživi napad, i ako ga preživi sigurno će baciti bombu na vojsku Igrača 2. Igrač 1 također ima mogućnost da bombu stavi na pomoćni avion. Ako se bomba nalazi na pomoćnom avionu, a borbeni avion napadne bombarder, zbog svoje slabije građe pomoćni avion se u napadu može oštetiti i eksplodirati, ali ima 90% šanse da preživi. U slučaju da se bomba nalazi na pomoćnom avionu i borbeni avion napadne njega, on ima 50% šanse da preživi taj napad, ali ako ga preživi također će sigurno baciti bombu na na vojsku Igrača 2.

Tablica 3.8. Matrica vjerojatnosti napada.

	Igrač 2		
Igrač 1	Napad \ Položaj bombe	Bombarder	Pomoćni avion
	Bombarder	80%	100%
	Pomoćni avion	90%	50%

U tablici su prikazane vjerojatnosti kako bi se igra lakše razumjela. Igrač 1 može zaključiti da ako bombu stavi na bombardera, može očekivati najmanje 80% uspješnosti. Igrač 2 bi brzo mogao shvatiti koju strategiju koristi, i usmjeriti svoj borbeni avion da uvijek napada bombardera. Na taj

način bi osigurao maksimalnu uspješnost Igrača 1 na 80% i ne više. Ipak, Igrač 1 može odlučiti ponekad staviti bombu na pomoćni avion. Za primjer ćemo uzeti slučaj kada bi to napravio 1/4 puta. Kada Igrač 2 sazna za strategiju protivnika, mora odlučiti kakvu protustrategiju primijeniti. On zna da bi bilo pametno napasti i pomoćni avion, te se može odlučiti da ga napadne 1/2 puta.

Tablica 3.9. Matrični prikaz učestalosti napada.

	Igrač 2			
Igrač 1		Učestalost napada	0.50	0.50
	Učestalost položaja bombe		Bombarder	Pomoćni avion
	0.75	Bombarder	80%	100%
	0.25	Pomoćni avion	90%	50%

Tablica 3.10. Matrica prikaza vjerojatnosti.

Događaj	Vjerojatnost događaja	Vjerojatnost uspješnosti
Bomba se nalazi na bombarderu, i borbeni avion napada bomberdera	$.75 * .50 = .375$	80%
Bomba se nalazi na bombarderu, a borbeni avion napada pomoćni avion	$.75 * .50 = .375$	100%
Bomba se nalazi na pomoćnom avionu, a borbeni avion napada bomberdera	$.25 * .50 = .125$	90%
Bomba se nalazi na pomoćnom avionu, i borbeni avion napada pomoćni avion	$.25 * .50 = .125$	50%

U ovim okolnostima, moguće je izračunati koliko će misija biti uspješna.

$$.375 * 80\% + .375 * 100\% + .125 * 90\% + .125 * 50 = 30\% + 37.5\% + 11.25\% + 6.25\% = 85\%$$

Iz izračuna se može vidjeti da će korištenjem ove strategije Igrač 1 ostvariti uspješnost od 85%, što je bolja opcija nego da bombu stavlja samo na bombardera gdje će uspješnost biti 80%. Postavlja se pitanje što Igrač 2 može učiniti kako bi smanjio tu brojku. Najlogičnija opcija je smanjiti napade na pomoćni avion. Za slučaj da se Igrač 2 odlučio napasti pomoćni avion samo 1/5 puta, ponovo će se izračunati uspješnost misije.

Tablica 3.11. Matrični prikaz učestalosti napada.

	Igrač 2			
Igrač 1		Učestalost napada	0.80	0.20
	Učestalost položaja bombe		Bombarder	Pomoćni avion
	0.75	Bombarder	80%	100%
	0.25	Pomoćni avion	90%	50%

Tablica 3.12. Matrica prikaza vjerojatnosti.

Događaj	Vjerojatnost događaja	Vjerojatnost uspješnosti
Bomba se nalazi na bombarderu, i borbeni avion napada bomberdera	$.75 * .80 = .60$	80%
Bomba se nalazi na bombarderu, a borbeni avion napada pomoćni avion	$.75 * .20 = .15$	100%
Bomba se nalazi na pomoćnom avionu, a borbeni avion napada bomberdera	$.25 * .80 = .20$	90%
Bomba se nalazi na pomoćnom avionu, i borbeni avion napada pomoćni avion	$.25 * .20 = .05$	50%

Za ovaj slučaj uspješnost misije iznosi:

$$.60 * 80\% + .15 * 100\% + .20 * 90\% + .05 * 50\% = 48\% + 15\% + 18\% + 2.5\% = 83.5\%$$

Iz navedenog izračuna Igrač 2 može zaključiti da smanjivanjem broja napada na pomoćni avion, smanjuje i uspješnost misije Igrača 1 na 83.5%, što je bolje nego 85%. Vođen tom logikom, Igrač 2 se može odlučiti u potpunosti prestati napadati pomoćni avion i fokusirati se samo na napadanje bombardera. Takva strategija rezultira sljedećim.

Tablica 3.13. Matrični prikaz učestalosti napada.

	Igrač 2			
Igrač 1		Učestalost napada	1	0
	Učestalost položaja bombe		Bombarder	Pomoćni avion
	0.75	Bombarder	80%	100%
	0.25	Pomoćni avion	90%	50%

Tablica 3.14. Matrica prikaza vjerojatnosti.

Događaj	Vjerojatnost događaja	Vjerojatnost uspješnosti
Bomba se nalazi na bombarderu, i borbeni avion napada bomberdera	$.75 * 1 = .,75$	80%
Bomba se nalazi na bombarderu, a borbeni avion napada pomoćni avion	$.75 * 0 = 0$	100%
Bomba se nalazi na pomoćnom avionu, a borbeni avion napada bomberdera	$.25 * 1 = .25$	90%
Bomba se nalazi na pomoćnom avionu, i borbeni avion napada pomoćni avion	$.25 * 0 = 0$	50%

$$.75 * 80\% + \ 0 * 100\% + \ .25 * 90\% + 0 * 50\% = 60\% + 0\% + 22.5\% + 0\% \ = 82.5\%$$

Iz priloženog se vidi da će ne napadanjem pomoćnog aviona, Igrač 2 smanjiti uspješnost misije Igrača 1 za još 1%. Za ovaj primjer kada Igrač 1 koristi strategiju (0.75, 0.25), najbolja strategija Igrača 2 je (1, 0) jer na taj način on minimizira svoje gubitke i smanjuje uspješnost misije Igrača 1. U ovakvom slučaju, Igrač 2 koristi čistu strategiju kao odgovor na fiksnu strategiju Igrača 1. Prema [10] to je potkrijepljeno teoremom „Ako jedan igrač koristi fiksnu strategiju, tada njegov protivnik ima optimalnu protustrategiju koja je čista“ (čista strategija je praćenje isključivo jedne strategije koja pruža maksimalni dobitak). Pronalazak protustrategije se može i matematički modelirati kako ne bi bilo potrebe za nagađanjem. Prethodno prikazani izračuni su dobiveni metodom koja će sada biti prikazana. Ova metoda vrijedi za izračun očekivanog dobitka u 2x2 igrama. Strategija Igrača 1 se definira kao (1 - p, p), gdje p predstavlja vjerojatno odabira Igrača 1. Igrač 2 tada može tražiti odgovarajuću protustrategiju (1 - q, q). gdje q predstavlja vjerojatnost odabira Igrača 2.

Tablica 3.15. Matrica za traženje protustrategije.

	1 - q	q
1 - p	a	b
p	c	d

Iz tablice se vidi da je vjerojatnost dobitka a za Igrača 1, $(1-p)*(1-q)$. Analogno tome se računaju iz ostali dobitci. Svi slučajevi dobitaka su međusobno isključivi, pa je očekivani dobitak Igrača 1 računa na sljedeći način.

$$(1-p)*(1-q)*a+(1-p)*q*b+p*(1-q)*c+p*q*d$$

Kao što je već spomenuto, ovo je način pronalaska optimalne strategije za određenu fiksnu mješovitu strategiju protivnika.

4. PROGRAMSKO RJEŠENJE

Programsko rješenje ove aplikacije se temelji na prethodnim opisima igara, te će se njihova implementacija prikazati u ovom poglavlju. Također opisat će se tehnologije i alati koji su bili potrebni za izradu aplikacije.

4.1. Android OS

Prema [15], Android operacijski sustav dizajniran za uređaje sa zaslonom osjetljivim na dodir (tableti, mobilni telefoni, pametne satovi, netbook računala). Uređaji trebaju imati ekran veličine dovoljne za ugodno korištenje, te moraju rabiti 2D ili 3D grafičku knjižnicu temeljenu na OpenGL ES 2.0 specifikacijama. Operacijski sustav se temelji na Linux 2.6. jezgri napisanoj u C/C++ programskom jeziku, ali sadrži određene arhitekturalne izmjene po kojima se razlikuje od ostalih standardnih Linux sustava. Razvoj aplikacija za Android operacijski sustav obavlja se većinom u programskom jeziku Java, u Android Studio programskom paketu uz pomoć Android Software Development Kit-a (SDK). Od 2017. godine, Google također počinje pružati službenu podršku Kotlin programskom jeziku za razvoj aplikacija kao alternativu Javi. Dosad je Google izdao 17 verzija operacijskog sustava od kojih su svi bili imenovani po desertima osim zadnje koja nosi službeni naziv Android 10. Od nedavno Google je počeo s razvojem različitih inačica Android operacijskog sustava, pa se tako sada može naći Android TV koji je namijenjen korištenju u pametnim televizorima, Android Wear za pametne satove i Android Auto koji se koristi u automobilskim sustavima. Prema [16], arhitektura Android-a sastoji od nekoliko slojeva, koji su prikazani na slici 4.1. Na samom dnu se nalazi Linux jezgra koja upravlja procesima, memorijom, kamerom uređaja, zvukom itd. Sljedeći sloj čine biblioteke poput Surface Manager (nadzire iscrtavanje grafičkog sučelja), OpenGL | ES (za sklopovsko ubrzanje 3D prikaza), SSL (Secure Sockets Layer – za sigurnu komunikaciju putem interneta) i slične. Paralelno njima, nalazi se Dalvik Virtual Machine koji služi pokretanje aplikacija, i omogućava višenitnost na Android uređajima. Na njima se nalazi aplikacijski okvir i on pruža usluge više razine, poput upravljanja resursima, aktivnostima, programskim paketima. Najviši sloj u Androidu čini aplikacijski sloj koji se sastoji od samih aplikacija koje se instaliraju na uređaj.

Slika 4.1. Arhitektura Android operacijskog sustava

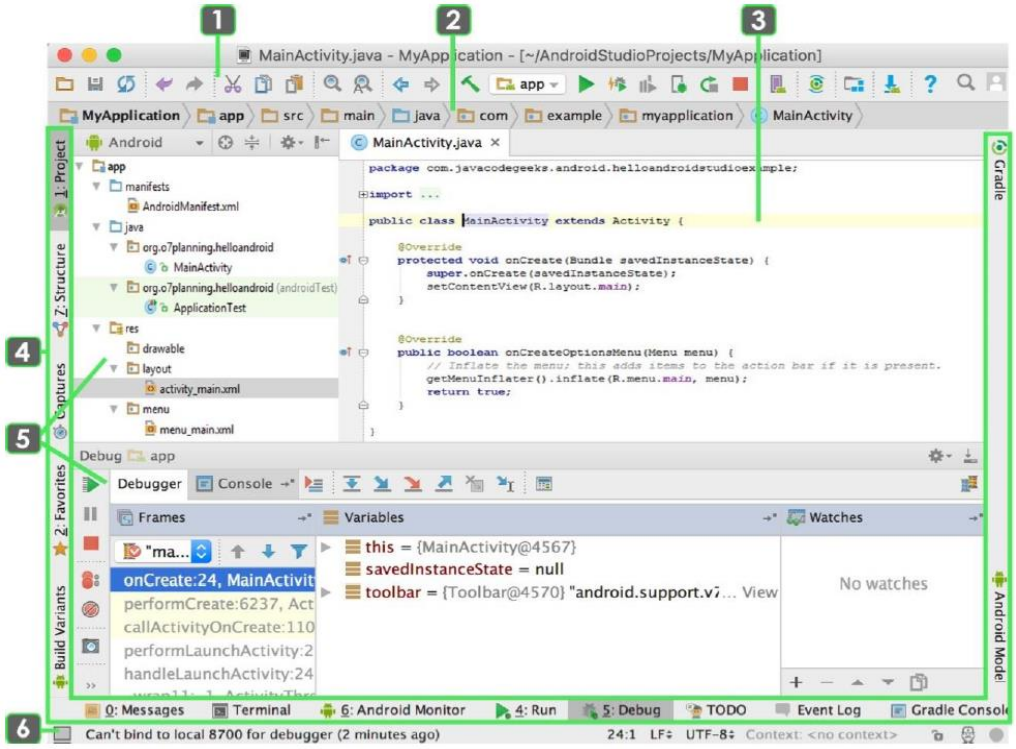
4.1.1. Povijest Androida

Prema [17], 2003. godine četvorica prijatelja Rich Miner, Nick Sears, Chris White i Andy Rubin osnivaju tvrtku pod nazivom Android Inc. Inicijalni cilj tvrtke je bio razvoj softvera za mobilne uređaje koji će pratiti lokaciju korisnika i njegove preferencije. Nakon dvije godine razvoja bez značajnog pomaka, odlučuju prodati tvrtku Google-u koji se želi uključiti na tržište pametnih telefona. 2007. godine Google okuplja 34 tvrtke iz različitih područja mobilne industrije i osniva Open Handset Alliance (OHA) s ciljem razvoja standarda za mobilne uređaje. Ubrzo nakon toga Open Handset Alliance na tržište lansira Android operacijski sustav temeljen na Linux kernelu, i tako se priključuje tada već poznatim platformama poput iOS-a, Symbian-a, Palm-a, Windows Phone-a. Prva inačica operacijskog sustava je predstavljena na HTC Dream uređaju u rujnu 2008. godine. Kasnije te godine, izvorni kod Androida je pušten u javnost pod Apache licencom, te je tako postao dostupan svima koji se žele baviti njegovim razvojem. Taj potez je uveliko doprinio popularnosti Androida jer je njegova zajednica je počela razvijati nadogradnje za starije uređaje koji više nisu podržani od strane Google-a, a također često razvijaju i nadogradnje za nove uređaje s raznim preinakama u odnosu na službene verzije.

4.2. Razvojni alati

4.2.1. Android Studio

Prema [18], Android studio je službeno integrirano razvojno okruženje (IDE) za razvoj Android aplikacija. Prva verzija je izašla 2014. godine kao zamjena za Eclipse ADT koji je do onda služio istoj svrsi. Zasnovan je na IntelliJ IDEA softveru i osim njegovih paketa za razvoj, proširen je snažnim emulatorom, integracijom za Github, podrškom za Google Cloud platformu i sličnim nadogradnjama. Kako bi korisnik mogao programirati aplikacije u Android Studiu, osim njega potrebno je instalirati i Android SDK i Android JDK pakete. Sučelje Android Studia se sastoji od nekoliko komponenti koje su prikazane na Slici 4.2.



Slika 4.2. Komponente Android Studia

Komponente prikazane na slici su:

[1] Alatna traka za pokretanje aplikacije

[2] Navigacijska traka za lakše snalaženje u projektu

[3] Prozor za kreiranje i uređivanje koda

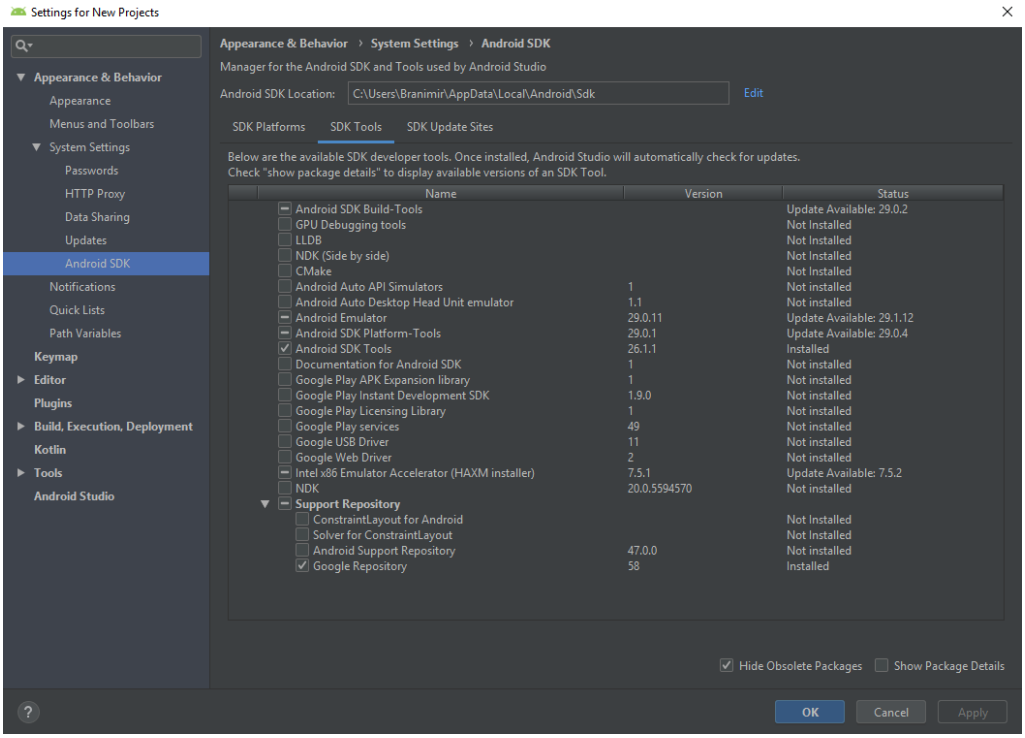
[4] Traka alatnog prozora za proširivanje ili smanjivanje određenih prozora alata

[5] Alatni prozor za pristup određenim zadacima

[6] Statusna traka za prikazivanje statusa projekta, upozorenja i grešaka

4.2.2. Android SDK

Prema [19], Android SDK (paket za razvoj softvera) je skup razvojnih alata za razvoj Android aplikacija. On sadrži sve potrebne alate za razvoj i omogućuje preuzimanje svih ili samo onih alata koji su korisniku potrebni. Android SDK se može instalirati na Windows, MacOs i Linux operacijskim sustavima. Izlaskom svake nove verzije Androida, SDK se mora nadograditi kako bi podržao i tu novu verziju. To ujedno znači i da se aplikacije mogu razvijati za bilo koji stariju verziju. Na slici 4.3 su prikazani neki od alata dostupnih u Android SDK-u.



Slika 4.3. Android SDK alati

4.2.3. Java programski jezik

Prema [20], programski jezik Java je kreiran 1995. godine u tvrtki Sun Mycrosystems od strane Jamesa Goslinga. Kasnije tu tvrtku preuzima Oracle Corporation koja je i danas vlasnik Jave. To je objektno orijentirani programski jezik temeljen na klasama. Osmišljen je s ciljem da omogući programerima WORA pristup (Write once, run anywhere), što znači da kada je jednom napisan može se izvoditi na bilo kojem Java virtualnom (JVM) stroju neovisno o platformi.

4.3. Programski kod

Prilikom kreiranja projekta u Android Studiu bilo je potrebno definirati za koju verziju Androida će se praviti aplikacija. Pogledom na Build Gradle na slici 4.4 može se vidjeti da je „targetSdkVersion 29“, a „minSdkVersion 23“. Verzija 29 označava Api razinu, a ona nam govori o kojoj verziji Androida se radi. Api razina 29 predstavlja najnoviju inačicu Androida, Android 10. Verzija 23 označava minimalnu verziju koju mobilni uređaj mora koristiti, i predstavlja Android Marshmallow (7.0).

```
android {
    compileSdkVersion 29
    buildToolsVersion "29.0.0"
    defaultConfig {
        applicationId "hr.ferit.branimirkedacic.diplomskirad"
        minSdkVersion 23
        targetSdkVersion 29
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }
```

Slika 4.4. Sdk verzije

Zatim je kreiran izgleda početnog zaslona aplikacije vidljivog na slici 3.5. Za kreiranje je korišten Constraint Layout koji omogućuje međusobno povezivanje elemenata, te na taj način određuje položaje gdje se koji nalazi. Zatim je potrebno u pripadajućoj aktivnosti definirati koji su to elementi i pronaći ih pomoću findViewById() funkcije.


```
public class MainActivity extends AppCompatActivity {

    Button PismoGlavaButton;
    Button buttonSPK;
    Button SijeSete;
    Button buttonBombardiranjeNapad;
    Button buttonBombardiranjeObrana;
    Button buttonSKPvsIGRAC;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        PismoGlavaButton = findViewById(R.id.pismoGlavaButton);
        PismoGlavaButton.setOnClickListener((view) -> {
                Intent intent = new Intent(getApplicationContext(), PismoGlavaActivity.class);
                startActivity(intent);
        });

        buttonSPK = findViewById(R.id.buttonSPK);
        buttonSPK.setOnClickListener((view) -> {
                Intent intent = new Intent(getApplicationContext(), SPKActivity.class);
                intent.putExtra( name: "vsIgrac",  value: 1);
                startActivity(intent);
        });

        SijeSete = findViewById(R.id.buttonSijeSete);
        SijeSete.setOnClickListener((view) -> {
                Intent intent = new Intent(getApplicationContext(), SijeSeteActivity.class);
                startActivity(intent);
        });
```

Slike 4.5. Izgled i inicijaliziranje početne aktivnosti

4.3.1. Pismo – glava

Za igru Pismo – glava je napravljena verzija kada se igra samo protiv računala. Za taj slučaj bilo je potrebno generirati slučajnu vrijednost pomoću Random() funkcije, koja će predstavljati odabir računala. Glavna funkcionalnost se odvija u funkcijama getPismo() prikazanoj na slici 4.6 i getGlava(), gdje se vrše provjere odabira igrača i računala i na osnovu njih se ispisuju rezultati funkcijom setText(). OdabirKorisnika i strana su varijable za uspoređivanje ishoda igre, broj 1 označava pismo, a broj 2 označava glavu. Varijable rezKorisnik i rezPC su brojači koji služe da praćenje trenutnog rezultata.

```
public void getPismo() {
    if (odabirKorisnika == 1 && strana == 1) {
        imageViewCoin.setImageResource(R.drawable.pismo);
        Animation();
        textViewRezultat.setText("Izabrali ste pismo, i zavrtili pismo");
        rezKorisnik++;
        textViewKorisnik.setText("Vi: " + rezKorisnik);
        textViewPismo.setText("Pismo: " + rezPismo);
    } else if (odabirKorisnika == 2 && strana == 1) {
        imageViewCoin.setImageResource(R.drawable.pismo);
        Animation();
        textViewRezultat.setText("Izabrali ste glavu, a zavrtili pismo");
        rezPC++;
        textViewPC.setText("PC: " + rezPC);
        textViewPismo.setText("Pismo: " + rezPismo);
    }
    odabranaStrana = 0;
}
```

Slika 4.6. Funkcija za provjeru Pisma

Animacija se poziva funkcijom Animate(), čije su karakteristike definirane u animate.xml datoteci i prikazane na slici 4.7.

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <rotate
        android:duration="1000"
        android:fromDegrees="0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:toDegrees="360">
    </rotate>

    <alpha
        android:fromAlpha="0"
        android:toAlpha="1"
        android:duration="1000" >
    </alpha>

    <scale
        android:fromXScale="0"
        android:toXScale="1"
        android:fromYScale="0"
        android:toYScale="1"
        android:duration="1000"
        android:pivotX="50%"
        android:pivotY="50%" >
    </scale>
</set>
```

Slika 4.7. Karakteristike Animate() funkcije

4.3.2. Škare – papir – kamen

Za igru Škare – papir – kamen implementirana su dva slučaja. Prvi slučaj je kada se igra protiv računala, a drugi slučaj se odnosi na igranje dva igrača. Za slučaj protiv računala, glavni dio logike se nalazi u funkciji checkVsPC() prikazanoj na slici 4.8. Pomoću varijable odabraniZnak vrši se provjera da li je igrač prethodno izabrao neki od ponuđenih znakova. Zatim se na osnovu odabira uvećava jedan od brojača za praćenje rezultata, te se nakon njih generira Random vrijednost za odabir računala. Nakon toga se u Timer() funkciji, koja traje jednu sekundu, pozivaju funkcije za provjeru rezultata getSkare(), getPapir() i getKamen() u kojima se nalazi logika ispitivanja rezultata.

```
public void checkVsPC(){
    if (odabraniZnak == 0) {
        textViewRezultatSKP.setText("Odaberite znak za pokazati!");
    } else {

        if (odabirKorisnika == 1) {
            rezSkare++;
        } else if (odabirKorisnika == 2) {
            rezPapir++;
        } else if (odabirKorisnika == 3) {
            rezKamen++;
        }
        randomZnak = randVrijednost.nextInt( bound: (3 - 1) + 1) + 1;
        new Timer().schedule(() → {
                new Handler(Looper.getMainLooper()).post(() → {
                        getSkare();
                        getPapir();
                        getKamen();

                });
        }, delay: 1000);
    }
}
```

Slika 4.8. Funkcija checkPC() za igru Škare – papir – kamen

Praćenje odabira tipa igre korisnika, vrši se pomoću flag varijabli koje se prenose iz MainActivity.java pomoću .putExtra() funkcije. Kada igraju dva igrača potrebno je uvesti novu varijablu IgracNaRedu vidljivu na slici 4.9, koja prati koji je igrač na potezu. Ako neki od igrača pokuša odabrati bilo što drugo osim dopuštene radnje, na zaslonu će mu se ispisati odgovarajuća poruka s uputom. Ovdje je također definirana i setTimer() funkcija, koja služi za prikaz brojača na zaslonu uređaja. Duljina trajanja je predefinirana na tri sekunde, te se nakon isteka vremena prikazuje rezultat na zaslonu.

Slika 4.9. Funkcija checkPlayers() i buttonIgraj za igru Škare – papir – kamen

4.3.3. Šije – šete

Kao i prošla, igra Šije – šete koristi sličnu logiku. Glavna razlika je što za razliku od prošle igre, ovdje korisnik mora unositi svoj izbor u polja. Funkcijom checkVsPC() prikazanoj na slici 4.10, vrši se provjera unosa. Prikupljanje unesene vrijednosti u polje, vrši se pomoću getText().toString() funkcija, te se odmah nakon toga provjera ispunjenost polja. Nakon toga se prikupljeni String funkcijom Integer.parseInt() pretvara u integer kako bi se mogao koristiti za daljnje provjere. Sljedeći korak je provjeravanje vrijednosti unosa kako bi se osigurao unos samo dopuštenih vrijednosti. Ako je sve u redu, poziva se funkcija checkResultPC(), koja generira i uspoređuje rezultate računala s vrijednostima unesenim od strane korisnika te ih ispisuje na zaslonu uređaja.

```
public void checkVsPC() {
    StrKorisnikPokaz = editTextPokaziKor.getText().toString();
    StrKorisnikGovor = editTextIzgovorKor.getText().toString();

    if (StrKorisnikPokaz.isEmpty() && StrKorisnikGovor.isEmpty()) {
        Toast.makeText(getApplicationContext(), text: "Unesite brojeve u polja!", Toast.LENGTH_LONG).show();
    } else if (StrKorisnikPokaz.isEmpty()) {
        Toast.makeText(getApplicationContext(), text: "Unesite broj za pokazati!", Toast.LENGTH_LONG).show();
    } else if (StrKorisnikGovor.isEmpty()) {
        Toast.makeText(getApplicationContext(), text: "Unesite broj za izgovoriti!", Toast.LENGTH_LONG).show();
    } else {
        KorisnikPokaz = Integer.parseInt(StrKorisnikPokaz);
        KorisnikGovor = Integer.parseInt(StrKorisnikGovor);

        if (KorisnikPokaz < 1 || KorisnikPokaz > 4 || KorisnikGovor < 2 || KorisnikGovor > 8) {
            Toast.makeText(getApplicationContext(), text: "Unesite broj između 1 i 4 za pokazati, i između 1 i 8 za izgovoriti", Toast.LENGTH_LONG).show();
        } else if (KorisnikPokaz >= KorisnikGovor) {
            Toast.makeText(getApplicationContext(), text: "Broj za pokazati mora biti manji od broja za izgovoriti!", Toast.LENGTH_LONG).show();
        } else if (KorisnikGovor > KorisnikPokaz + 4) {
            Toast.makeText(getApplicationContext(), text: "Broj za izgovoriti može biti najviše za četiri veći od broja za pokazati!", Toast.LENGTH_LONG).show();
        } else {
            checkResultPC();
        }
    }
}
```

Slika 4.10. Funkcija checkVsPC() za igru Šije – šete

Kada igraju dva igrača, analogno prethodnoj provjeri, vrše se provjere za unose oba igrača. Funkcija checkResultPVP() se poziva nakon provjera i u njoj se odvija logika provjere rezultata i ispisivanja na zaslon, što je vidljivo na slici 4.11.

```
public void checkResultPVP() {
    zbrojIgraca = Igrac1Pokazuje + Igrac2Pokazuje;

    if (zbrojIgraca == Igrac1Pokazuje && zbrojIgraca == Igrac2Pokazuje) {
        textViewRez.setText("Neriješeno!");
        textViewRez.setTextColor(Color.parseColor( colorString: "#000000"));
    } else if (zbrojIgraca == Igrac1Izgovara) {
        textViewRez.setText("Igrač 1 je pobjedio!");
        textViewRez.setTextColor(Color.parseColor( colorString: "#0A26C7"));
        KorRez++;
        textViewKORRez.setText("Igrač 1: " + KorRez);
    } else if (zbrojIgraca == Igrac2Izgovara) {
        textViewRez.setText("Igrač 2 je pobjedio!");
        textViewRez.setTextColor(Color.parseColor( colorString: "#970404"));
        PcRez++;
        textViewPCRez.setText("Igrač 2: " + PcRez);
    } else {
        textViewRez.setText("Neriješeno!");
        textViewRez.setTextColor(Color.parseColor( colorString: "#000000"));
    }
    textViewIgrac1P.setText("Igrač 1 pokazuje: " + Igrac1Pokazuje);
    textViewIgrac1I.setText("Igrač 1 izgovara: " + Igrac1Izgovara);
    textViewIgrac2P.setText("Igrač 2 pokazuje: " + Igrac2Pokazuje);
    textViewIgrac2I.setText("Igrač 2 izgovara: " + Igrac2Izgovara);
    igracNaRedu = 1;

    editTextPokaziKor.setText("");
    editTextIzgovorKor.setText("");
    textViewKorisnik.setText("Igrač 1 je na redu!");
}
```

Slika 4.11. Funkcija checkVsPC() za igru Šije – šete

4.3.4. Bombardiranje

Igra Bombardiranje se igra protiv računala, a razvijena su dva scenarija. Prvi scenarij je kada korisnik napada i odabire gdje želi postaviti bombu, a drugi je kada se korisnik brani i bira koji avion želi napasti. Kao i u ostalim igrama koristi se Random() funkcija za kreiranje odabira računala, a odabir korisnika se prikuplja putem Buttona. Nakon što korisnik izabere avion, izvršavaju se funkcije Pcrandom() za generiranje slučajne vrijednosti, PCizbor() za prikaz odabira računala na zaslonu uređaja i funkcije NapadniBombarder() i napadniPomocni(). Ovisno o njegovom odabiru prikazuju se poruke s rezultatima na zaslonu uređaja. Kod koji se izvršava prilikom pritiska na buttonNapadni prikazan je na slici 4.12. Analogno napadanju, napravljen je kod i za slučaj branjenje.

```
buttonNapadni.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        new Timer().schedule(new TimerTask() {
            @Override
            public void run() {
                new Handler(Looper.getMainLooper()).post(new Runnable() {
                    @Override
                    public void run() {
                        if (odabranAvion == 0) {
                            textviewPoruka.setText("Morate prvo odabrati avion!");
                            textViewPC.setText("PC je postavio bombu na ");
                            textViewRezultat.setText("Rezultat");
                        } else {
                            textviewPoruka.setText("");

                            if (odabirKor == 1) {
                                textviewPoruka.setText("Vi ste napali bombardera");
                            } else if (odabirKor == 2) {
                                textviewPoruka.setText("Vi ste napali pomoćni avion");
                            }
                            PCrandom();
                            PCizbor();
                            napadniBombarder();
                            napadniPomocni();
                        }
                    }
                });
            }
        }, delay: 1000);
    }
});
```

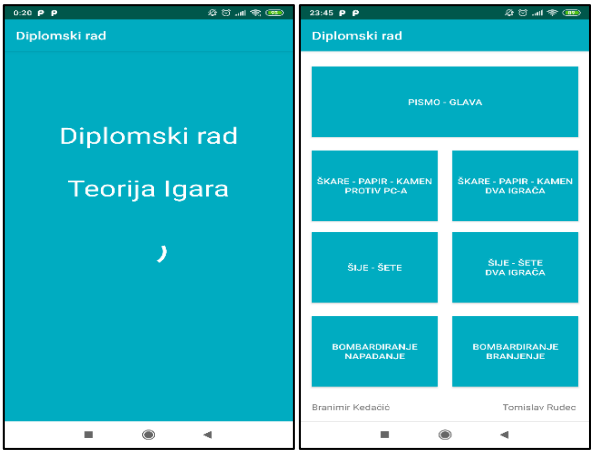
Slika 4.12. Kod za buttonNapadni

4.4. Korištenje aplikacije

Kako bi se moglo pristupiti aplikaciji, na pametni telefon s operacijskim sustavom Android potrebno je instalirati .apk datoteku ili izvršiti instalaciju putem Android Studia. Nakon uspješne instalacije korisnik ju može pokrenuti pritiskom na ikonu aplikacije Diplomski rad (Sl. 4.13).

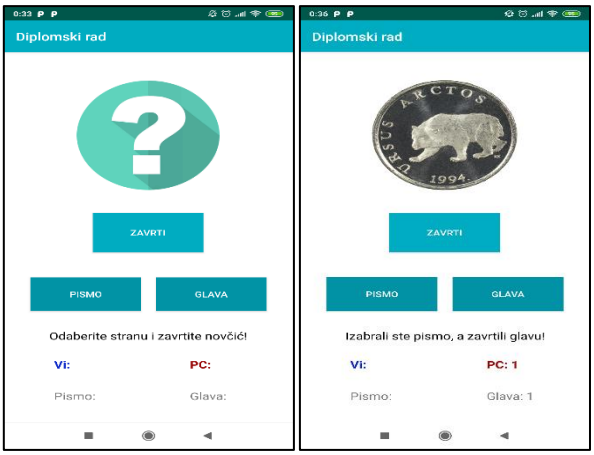
Slika 4.13. Pokretanje aplikacije

Nakon toga korisnik ulazi u aplikaciju gdje mu se prvo prikazuje pozdravni zaslon, a nakon toga otvara mu se početni zaslon (Sl. 4.14). Na njemu su prikazane sve igre i korisnik može odabrati bilo koju ponuđenih.



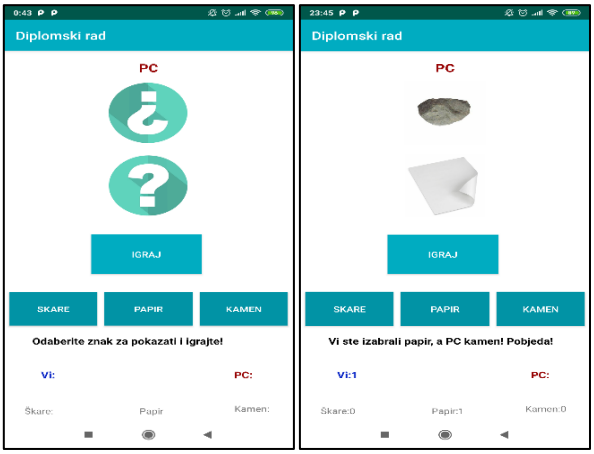
Slika 4.14. Prikaz pozdravnog i početnog zaslona

Odabirom na igru Pismo – glava, korisniku se otvara novi prozor s detaljima igre (Sl. 4.15). Početni prikaz govori korisniku da odabere stranu i zavrti novčić. Nakon te radnje prikazuje se rezultat igre, broj pobjeda i broj pojave pisma i glave, te korisnik može ponovno igrati.



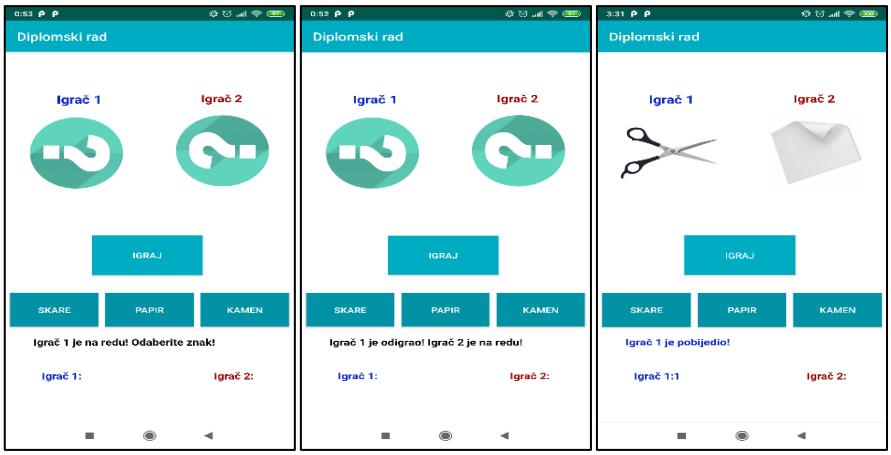
Slika 4.15. Prikaz igre Pismo – glava

Pritiskom na sistemsku tipku za povratak, vraća se na početni zaslon aplikacije i može se odabrati druga igra. Sljedeća igra je Škare – papir – kamen, koja ima dvije verzije. Prva verzija protiv računala (Sl. 4.16), nakon odabira prema uputama prikazuje rezultat, broj pobjeda i broj pojave pojedinog izbora od strane korisnika.



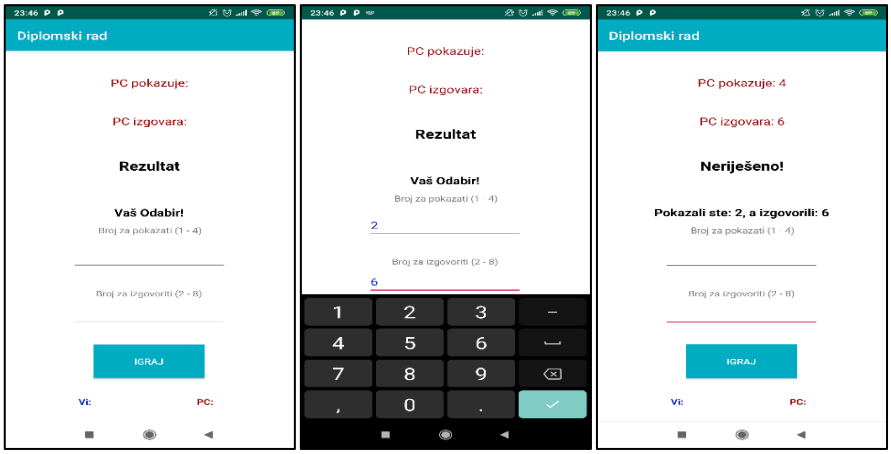
Slika 4.16. Prikaz igre Škare – papir – kamen protiv računala

Ukoliko korisnik odabere opciju igre protiv drugog igrača (Sl. 4.17), prikazuje mu se malo drugačije sučelje u odnosu na prethodno. Igra je zamišljena na način da prvi igrač odigra svoj potez, te da pametni telefon drugom igraču koji će odigrati svoj potez. Igrači bi trebali igrati pošteno i ne gledati odabir drugog igrača. Tri sekunde nakon toga prikazuje se rezultat i povećava brojač igrača koji je pobijedio.



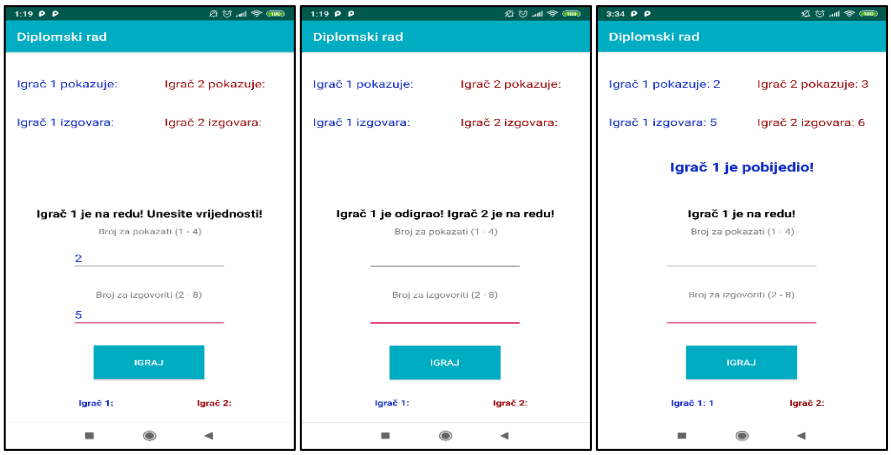
Slika 4.17. Prikaz igre Škare – papir – kamen za dva igrača

Sljedeća igra je Šije – šete protiv računala (Sl. 4.18). Kao što je vidljivo na sučelju, potrebno je unijeti jedan broj za pokazati i jedan za izgovoriti. Pritiskom na polje za unos, korisniku se otvara tipkovnica pomoću koje unosi vrijednosti. Nakon unosa i pritiska na tipku Igraj, prikazuje se ispis odabira, rezultat, i uvećavaju se brojači pobjeda.



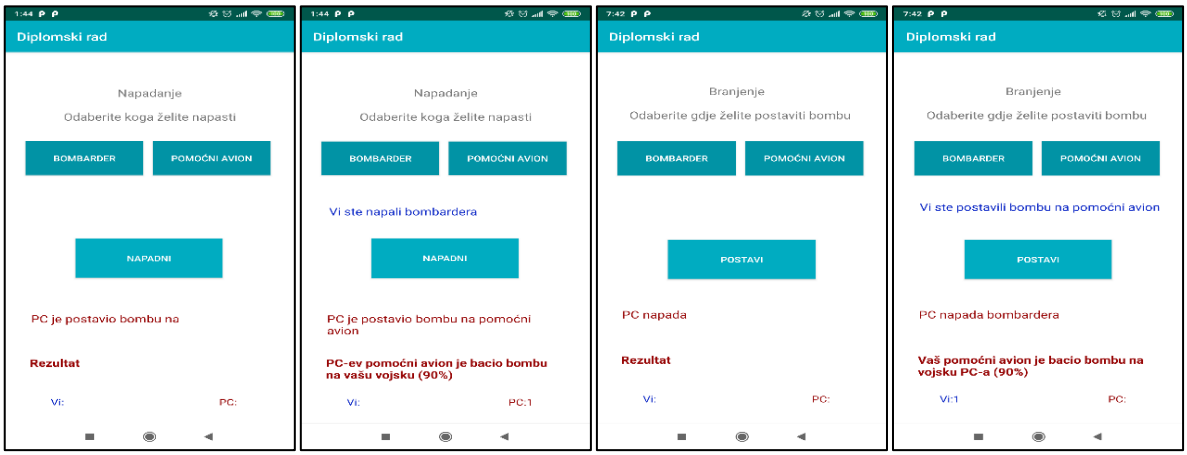
Slika 4.18. Prikaz igre Šije – šete protiv računala

Igra Šije – šete za dva igrača (Sl. 4.19), prvo zahtijeva unos za prvog igrača i pritiskom na tipku Igraj na red dolazi drugi igrač. Kao i u igri Škare – papir – kamen, igrači trebaju igrati pošteno. Nakon što i drugi igrač potvrdi svoj odabir, nakon tri sekunde se ispisuju informacije o tome što je koji igrač odabrao kao i rezultat, te se povećava brojač pobjeda za igrača koji je pobijedio.



Slika 4.19. Prikaz igre Šije – šete za drva igrača

Zadnja igra je Bombardiranje (Sl. 4.20), koja se kao i prva može igrati samo protiv računala, ali ima dva slučaja. Kada korisnik napada, mora prvo odabrati avion koji želi napasti i zatim pritisnuti tipku Napadni. Nakon toga se generira prikaz rezultata na zaslonu. Kada se korisnik brani, mora odabrati na koji avion želi postaviti bombu te pritisnuti tipku Postavi, a nakon toga mu se kao i u prvom slučaju prikazuje rezultata na zaslonu.



Slika 4.20. Prikaz igre Bombardiranje

5. ZAKLJUČAK

Teorija igara je područje s kojim se skoro svakodnevno susrećemo u svakodnevnom životu, iako ponekad toga nismo ni svjesni. Cilj ovog diplomskog rada bio je istražiti i proučiti područje teorije igara, te na temelju teorijske podloge, realizirati mobilnu aplikaciju za igranje igara navedenih u radu. U prvom dijelu rada je proučena teorija, gdje je pokriveno upoznavanje s osnovnim pojmovima korištenim u teoriji igara. Zatim su opisane strategije i vrste igara koje postoje u ovom području. Na teorijskim primjerima obrađene su četiri igre, te su one služile kao podloga za kreiranje aplikacije. Programski dio rada izveden je u obliku mobilne aplikacije za Android operacijski sustav. Programski kod je napisan u programskom jeziku Java, a pomoću XML jezika za označavanje kreiran je vizualni izgled i intuitivno korisničko sučelje. Aplikacija je izrađena u skladu s traženim zahtjevima, ali je njezino izvršavanje potaknulo na razmišljanje kako bi se ona još mogla poboljšati. Implementacija nekog oblika strojnog učenja, umjesto nasumičnog generiranja odabira računala i prikaz mogućih strategija igračima su neka od njih. Teorija igara se ne razvija brzo, ali polako širi svoju primjenu u različitim grana znanosti, od kojih je jedna od zanimljivijih je u računarstvo gdje se pojavljuje u poljima poput strojnog učenja i umjetne inteligencije.

LITERATURA

[1] An outline of the history of game theory. [Na internetu]. Dostupno na: http://euler.fd.cvut.cz/predmety/teorie_her/histf.html [Posjećeno: 07.06.2019]

[2] Başar T. (2013) Game Theory: Historical Overview. In: Baillieul J., Samad T. (eds) Encyclopedia of Systems and Control. Springer, London [Posjećeno 07.06.2019]

[3] Nobel prizes in game theory field. [Na internetu]. Dostupno na : http://lcm.csa.iisc.ernet.in/gametheory/nobel.html [Posjećeno: 10.06.2019]

[4] https://towardsdatascience.com/game-theory-in-artificial-intelligence-57a7937e1b88 [Posjećeno 11.06.2019]

[5] Game Theory in Artificial Intelligence. [Na internetu]. Dostupno na: https://www.investopedia.com/terms/g/gametheory.asp [Posjećeno: 15.06.2019]

[6] Levent Kockesen, Efe A. Ok, An Introduction to Game Theory, Koc University, New York University, 8. lipnja, 2007., str. 10. – 13. [Posjećeno: 15.06.2019]

[7] Stucke, M. E., Is competition always good? Legal Studies Research Paper Series, 2013. str. 8. – 10. [Posjećeno: 16.06.2019]

[8] 4 Strategies of the Game Theory. [Na internetu]. Dostupno na: http://www.economicsdiscussion.net/game-theory/4-strategies-of-the-game-theory-explained/3825 [Posjećeno: 19.06.2019]

[9] Dominant and Dominated Strategies. [Na internetu]. Dostupno na: https://effectiviology.com/strategic-dominance/#Dominant_strategies [Posjećeno: 19.06.2019]

[10] Saul Stahl, A gentle introduction to game theory, American Mathematical Soc., 2012. [Posjećeno: 19.06.2019]

[11] 5 Types of Games in Game Theory. [Na internetu]. Dostupno na: http://www.economicsdiscussion.net/game-theory/5-types-of-games-in-game-theory-with-diagram/3827 [Posjećeno: 19.06.2019]

[12] Non-Zero-Sum Games vs. Zero Sum Games. [Na internetu]. Dostupno na: https://www.brighthubpm.com/risk-management/61459-comparing-zero-sum-and-non-zero-sum-games/ [Posjećeno: 19.06.2019]

[13] Nash Equilibrium. [Na internetu]. Dostupno na: http://math.ucr.edu/home/baez/games/games_13.html [Posjećeno: 20.06.2019]

[14] Rock Paper Scissors Programming Competition. [Na internetu]. Dostupno na:

http://www.rpscontest.com/ [Posjećeno: 20.06.2019]

[15] Android operacijski sustav. [Na internetu]. Dostupno na: https://www.informatika.buzdo.com/pojmovi/mobile-3.htm [Posjećeno: 14.07.2019]

[16] Android architecture. [Na internetu]. Dostupno na: https://www.elprocus.com/what-is-android-introduction-features-applications/ [Posjećeno: 14.07.2019]

[17] Android History. [Na internetu]. Dostupno na: https://www.androidcentral.com/android-history [Posjećeno: 14.07.2019]

[18] Android Studio. [Na internetu]. Dostupno na: https://developer.android.com/studio/intro [Posjećeno: 25.07.2019]

[19] Android SDK. [Na internetu]. Dostupno na: https://www.techopedia.com/definition/4220/android-sdk [Posjećeno: 25.07.2019]

[20] Java programming language. [Na internetu]. Dostupno na: https://howtodoinjava.com/java/basics/what-is-java-programming-language/ [Posjećeno: 25.07.2019]

SAŽETAK

U diplomskom radu je obrađeno je područje teorije igara i ostvarena mobilna aplikacija za pametne telefone. U teorijskom dijelu prvo je bilo potrebno istražiti gdje se sve pojavljuje teorija igara koje su njezine primjene. Nakon toga su proučene karakteristike različitih tipova igara koje se pojavljuju u ovom području, a nakon toga je bilo potrebno detaljno se upoznati s vrstama strategija koje se mogu primjenjivati u igrama. Zatim je trebalo modelirati i prikazati nekoliko različitih igara i na osnovu njih ostvariti mobilnu aplikaciju. Mobilna aplikacija je kreirana za Android operacijski sustav koristeći sve potrebne alate za izradu, a orijentirana je na slučajeve kada igrač igra protiv računala i kada dva igrača igraju jedna protiv drugog. Sastoji se od četiri igre navedene u radu (Pismo – glava, Škare – Papir – Kamen, Šije – šete i Bombardiranje). Nakon izrade aplikacije je testirana kako bi se osigurao ispravan rad aplikacije.

Ključne riječi: Android, Java, odlučivanje, strategija, teorija igara, vrste igara, vjerojatnost.

ABSTRACT

The thesis deals with the field of game theory and creating of a mobile application for smartphones. In the theoretical part it was first necessary to investigate where the theory of games and its possible applications are used. After that, the characteristics of the different types of games that emerge in this area were studied, and after that it was necessary to get acquainted in detail with the types of strategies that could be implemented in games. Then several different games had to be modeled and displayed and a mobile app built on them. The mobile application was created for the Android operating system using all the necessary creation tools, and is oriented to the cases when a player is playing against a computer and when two players are playing against each other. It consists of four games listed in the paper (Penny Matching, Rock – paper – scissors, Four finger Morra and Bombing sorties). Once created, the application has been tested to ensure that the application works properly.

Keywords: Android, Java, decision making, strategy, game theory, game types, probability

ŽIVOTOPIS

Branimir Kedačić rođen je u Osijeku 12.06.1993. Nakon pohađanja i završetka osnovne škole u Batini, 2008. upisuje srednju školu u Belom Manastiru gdje se obrazuje za smijer Tehničar za računarstvo. Nakon završetka srednjoškolskog obrazovanja, godine 2012. upisuje preddiplomski studij računarstva na Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Po završetku preddiplomskog studija 2017. godine upisuje diplomski studij programskog inženjerstva na spomenutom fakultetu. U travnju 2019. počinje s odrađivanjem stručne prakse u Ericsson Nikola Tesla d.d. Osijek, te ju završava u svibnju iste godine.

Branimir Kedačić