Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Declan McPartlin

Asinhron večigralski način igranja mobilne igre preko medmrežja

diplomsko delo
visokošolskega strokovnega študija prve stopnje

Mentor: doc. dr. Peter Peer

Ljubljana, 2013

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00427/2013
Datum: 09.04.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **DECLAN STEWART MCPARTLIN**

Naslov: **ASINHRON VEČIGRALSKI NAČIN IGRANJA MOBILNE IGRE PREKO MEDMREŽJA**

ASYNCHRONOUS MULTIPLAYER MODE FOR PLAYING MOBILE GAME OVER THE INTERNET

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Z večanjem hitrosti in lahkega dostopa medmrežja lahko s prijatelji preko velikih razdalj igramo igre. Vendar se še vedno lahko pojavi težava z zakasnitvami pri sinhronizaciji, včasih igralci težko uskladijo skupni prosti čas, še posebej, če so na različnih koncih sveta. Tako še vedno ostajajo aktualne igre, ki jih lahko igramo s časovnim zamikom. Preglejte področje različnih načinov igranja iger s časovnim zamikom. Implementirajte mobilno demonstracijsko igro, ki se bo lahko igrala s časovnim zamikom in omogočala vsakemu igralcu igranje ob želenem času. Zaradi potencialnih daljših časovnih zamikov, naj del igranja po potrebi prevzame umetna inteligenca. Za povezovanje prijateljev si pomagajte z uporabo storitev portalov socialnih omrežij. Opredelite ustrezno arhitekturo, ki bo omogočala varno in zanesljivo izmenjavo podatkov o stanju igre.

Mentor:
doc. dr. Peter Peer

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Declan McPartlin, z vpisno številko **63090286**, sem avtor diplomskega dela z naslovom:

Asinhron večigralski način igranja mobilne igre preko medmrežja

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Petra Peera,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela in
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 1. avgusta 2013 Podpis avtorja:

Zahvaljujem se mentorju doc. dr. Petru Peer in as. Bojanu Klemencu, za pomoč in prijaznost pri izdelavi diplome.

Zahvaljujem se tudi celotni ekipi na Kliki za pomoč pri izdelavi diplome in za urjenje mojih sposobnosti.

Tudi družini in prijateljem se zahvaljujem za podporo.

Kazalo

Povzetek

Abstract

1 Uvod **1**

2 Pregled področja **5**

2.1 Začetki večigralskega načina 6

2.2 Nogometne igre 8

2.3 Večigralnost na mobilnih telefonih 10

3 Programska Tehnologija **13**

3.1 Knoxball 13

3.2 Večigralski vmesnik 14

3.3 Strežnik 14

4 Vmesnik za večigralski način igre **17**

4.1 Facebook avtentikacija 17

4.2 Funkcionalnosti vmesnika 18

4.3 Posnetki iger 22

4.4 Komunikacija s strežnikom 23

5 Strežnik za pošiljanje in shranjevanje podatkov o igri **25**

5.1 Sporočila z vmesnikom 26

5.2 Vpisovanje v podatkovno bazo 27

5.3 *Google App Engine Datastore* in poizvedbe GQL 28
5.4 Nastali problemi s sistemom 29

6 Umetna Inteligenca **31**
6.1 Pozicije in stanja igralcev 31
6.2 Lastnosti igralcev 34

7 Zaključne ugotovitve **37**

Kazalo slik

1.1 Začetni meni igre Knoxball. Igralec lahko izbira med enoigralskim načinom in večigralskim načinom. 2

3.1 Arhitektura odjemalec strežnik igre Knoxball. 15

4.1 Overitev s pomočjo Facebook-aplikacije. 18
4.2 Po prejetju zahteve Facebook aplikacije. 19
4.3 Izgled vmesnika takoj po vpisu na Facebook. 19
4.4 Seznam prijateljev. 20
4.5 Seznam igralcev. 21
4.6 Seznam vhodnih in izhodnih zahtev. 21
4.7 Seznam vseh posnetkov iger. 22

6.1 Razhroščevanje. 32
6.2 Prikaz algoritma za računanje nevarnosti. 33
6.3 Primer načina preigravanja. 34

Povzetek

Diplomsko delo opisuje način implementacije večigralnosti na asinhron način v mobilnih telefonskih igrah. Težava pri večigralskem načinu je v sinhronizaciji vseh udeleženih igralcev, oziroma zakasnitvi prenosa podatkov med napravami, ki so pri mobilnih napravah kljub vse hitrejši povezavi na medmrežje, še vedno problematične. Ena od možnih rešitev je, da se večigralnost izvaja asinhrono, torej da ni potrebno, da so vsi igralci sočasno prisotni v igri.

V okviru diplomskega dela smo implementirali asinhron večigralski način na primeru nogometne igre na mobilni platformi. Igra Knoxball je simulacija nogometa, pri katerem je uporabnik tako igralec na igrišču, kot trener ekipe. Pri večigralskem načinu pa ima samo vlogo trenerja. Večigralski način igre deluje tako, da prvi igralec pošlje, poleg zahteve po igri, podatke o svoji izbrani vrsti preko strežnika drugemu igralcu in nato prejme tudi nasprotnikove podatke. Na koncu se ustvari posnetek igre, ki si ga lahko igralca ogledata kadarkoli ter kolikorkrat hočeta. Posamezen igralec se overi s pomočjo svojega profila na Facebooku. Delo opisuje tehnologije, potrebne za implementacijo takšnih funkcionalnosti na mobilni platformi iOS ter arhitekturo umetne inteligence, ki je v igri uporabljena za samodejno izvajanje dela igre.

Abstract

This thesis describes an implementation of an asynchronous multiplayer mode in mobile platform games. One of the problems with real time multiplayer games is synchronising all the players involved, because the latency between devices is still a problem even though mobile devices have ever faster Internet connections.

We implemented an asynchronous multiplayer mode in a football themed game as part of the thesis. Knoxball is a basic football simulation game. The user can assume the role of a player on the field as well as the role of the coach of the team. While the user is in asynchronous multiplayer mode, he only has the role of the team coach. The asynchronous multiplayer is performed in a series of steps: the first player sends a request via the server to the second player along with his team data, then the second player accepts the request and sends his team data to the first player. Finally, a game replay is created. This replay can be viewed by players without any access time or replay count limitations. Each player is authorised by using their Facebook account. This thesis describes the technology needed to implement such functionality into game on mobile platforms, with emphasis on the platform iOS and artificial intelligence that is used for executing and automating certain parts of the game.

Poglavje 1

Uvod

Mobilni telefoni dajejo igram zaradi povezanosti na medmrežje, ne glede na lokacijo, vse več možnosti za večigralski način igre. Postajajo tudi zmogljivejši. Poleg tega ima skoraj vsak posameznik svoj mobilni telefon, s čimer je zagotovljenih tudi več igralcev. Toda ne glede na zmogljivosti trenutnih mobilnih telefonov je razlika v povprečni hitrosti povezave do medmrežja med namiznimi računalniki in mobilnimi telefoni še vedno izrazita, zato možnost večigralskega načina v realnem času zaradi zakasnitev ni najbolj izvedljiva. Zaradi tega so še vedno aktualne tudi druge vrste večigralskih načinov.

Eden izmed teh je tudi večigralnost na asinhron način. Ko prvi igralec pošlje zahtevo po igri ali svojo odigrano igro, ni potrebno, da je drugi igralec dostopen. Ta lahko sprejme ali odigra svoj del kadarkoli v prihodnosti. Cilj diplomske naloge je bilo implementirati tovrsten večigralski način v igro na mobilni platformi s pomočjo overitve preko Facebooka.

Tovrsten način večigralnosti je primeren za igre na mobilnih platformah. Veliko težav, ki lahko nastanejo pri večigralskem načinu igre v realnem času, kot so recimo slaba odzivnost, izguba povezave in netočni ali izgubljeni podatki, niso prisotni pri asinhronemu večigralskem načinu. Čeprav se igra ne dogaja v realnem času, ima igralec vseeno občutek, kot da v resnici tekmuje proti prijatelju s Facebooka.

Za demonstacijo takšnega načina igranja smo naredili igro – simulacijo nogometa – imenovano Knoxball (slika 1.1). V igri se igralec znajde v vlogi trenerja ekipe, kot

Slika 1.1: Začetni meni igre Knoxball. Igralec lahko izbira med enoigralskim načinom in večigralskim načinom.

tudi v vlogi posameznih igralcev na zelenici. Vendar pa je večigralnost virtualnega nogometa na asinhron način, pri katerem so na igrišču tudi človeški igralci v realnem času, skoraj nemogoča. Zato so lahko v tem večigralskem načinu vsi igralci samo agenti z umetno inteligenco. Agenti imajo stanja, ki vplivajo na njihovo premikanje. Vsak agent ima tudi svojo vlogo znotraj ekipe, ki določa, v katerem stanju je. Vloga agenta se lahko tekom igre spreminja. V igri se na nivoju ekipe določa, kdo ima kakšno vlogo. Tudi sama ekipa ima stanja, kar prav tako vpliva na stanja posameznih igralcev.

Knoxball gradi na knjižnici XNI[1], toda za vmesnik za večigralski način smo uporabili običajen iOSov *Interface Builder*. Medtem, ko se uporabnik overi s pomočjo Facebooka, se dejanski podatki shranijo na našem strežniku. Na strežniku se shranjujejo tako zahteve, kot celotne odigrane tekme.

V naslednjem poglavju si bomo pregledali zgodovino večigralskih iger. Osredotočili se bomo na igre, ki imajo podoben način večigralnosti, kot ga ima naša demonstracijska igra. V tretjem poglavju si bomo podrobneje pogledali uporabljeno programsko opremo. Četrto poglavje opisuje delovanje vmesnika, ki skrbi za povezovanje igralcev. Peto poglavje opisuje delovanja strežnika za sprejemanje in oddajanje zahtev po igri ter shranjevanje posnetkov iger. V šestem poglavju si bomo pogledali umetno inteligenco, ki pomaga odigrati tekme. V zadnjem poglavju smo strnili

[1] `http://code.google.com/p/xni/`

zaključne ugotovitve in možnosti nadaljnjega razvoja.

Poglavje 2

Pregled področja

Dandanes veliko ljudi igra igre. V Združenih državah Amerike naj bi Američani v 67% gospodinjstev igrali igre v povprečju osem ur na teden [26], zato je razumljivo, da je število aplikacij in s tem tudi število mobilnih telefonskih iger v zadnjih letih skokovito naraslo. Eden od pokazateljev je tudi število različnih aplikacij objavljenih na AppStore [12]. V okviru te diplomske naloge smo implementirali asinhron večigralski način na primeru nogometne igre. Razlog za takšno izbiro je primernost za implementacijo asinhronega večigralskega načina, hkrati pa popularnost športnih iger, predvsem nogometa, ki ima precej oboževalcev po celem svetu. Na primer, zadnje nogometno svetovno prvenstvo leta 2010 v Južnoafriški Republiki je glede na statistične podatke gledalo skoraj pol sveta (točneje 46%) [11]. Zaradi ogromnih napredkov v mobilni tehnologiji sedaj obstaja tudi široka izbira nogometnih iger za mobilne platforme.

Veliko računalniških nogometnih iger podpira tudi večigralski način. Nekatere igre, na primer Top Eleven, omogočajo interakcijo in občutek tekmovanja z drugimi ljudmi v vlogi nogometnega trenerja, toda brez dodanih vrednosti, kot je ogled odigrane tekme. Drugi primer je X2 Football 2010, ki je prava simulacija nogometa v smislu, da uporabnik upravlja z igralci. V praksi na žalost prihaja do neugodnosti pri igranju v realnem času: počasnost, neodzivnost in celo sesutje aplikacije predstavljajo precej resne težave [28]. Igra Knoxball je namenjena tistim, ki niso zadovoljni le z napisom, da so zmagali ali zgubili, ampak jim omogoča ogled in analizo celotne

igre. Namenjena je tudi tistim, ki sicer uživajo v večigralskem načinu v realnem času, ampak niso pripravljeni igrati neoptimalne igre.

2.1 Začetki večigralskega načina

Začetek večigralskega načina v računalniških igrah je bil v leta 1958 pri igri Tennis For Two, ki je bila hkrati prva računalniška igra z grafičnim prikazom. Igra se je igrala na osciloskopu, interakcijo pa so omogočali gumbi. Predstavljala je zelo poenostavljen prikaz simulacije tenisa, ki vsebuje mrežo, žogo in igrišče [3].

Ena izmed prvih komercialno uspešnih iger z večigralskim načinom je bila Atarijev Pong [4]. Pong, ki je bil izdan leta 1972 in je verjetno ena najbolj prepoznavnih iger na svetu, je predstavljala enostavno simulacijo namiznega tenisa s ptičje perspektive. Igra je bila prisotna v igralnicah, kot tudi doma na televizijskih konzolah. Vsak igralec ima svojo ploščico, s katero odbija žogo, ki se premika po igrišču. Ko igralec zgreši žogo, se drugemu igralcu prišteje točka. Igralec z največ točkami zmaga. Prva potezna igra z večigralskim načinom je bila Walter Brightov Empire, izdana leta 1977. Delovanje igre Empire je podobno kot pri družabni igri Risk. Na začetku je igra bila v tekstovni obliki, po nekaj letih pa je dobila tudi grafično podobo [14].

Ena izmed prvih iger, ki je delovala preko računalniškega omrežja (takrat ARPAnet) z večigralskim načinom, je bila igra Mazewarm, ki jo ustvaril Jim Guyton leta 1977. Igralci Mazewara so se lahko streljali in hodili po hodnikih v labirintu. Komunikacija znotraj igre ni bila mogoča [6].

Po Mazewaru je kmalu sledila igra, ki je postala izhodišče za skoraj vse današnje MMOG (angl. Massively multiplayer online game) igre: MUD (angl. Multi-User Dungeon). MUD je igra, ki sprejme povezave od več igralcev preko omrežja (telefonske linije ali medmrežja itd.). Vsakemu igralcu priskrbi dostop do skupne podatkovne baze, sob, izhodov in drugih objektov. Vsak igralec brska in upravlja s podatkovno bazo znotraj ene sobe, v kateri vidi samo tiste objekte, ki so v isti sobi. Iz sobe v sobo se premika načeloma preko izhodov, ki jih povezujejo. MUD je neke vrste virtualna resničnost, toda ne takšna, kot si jo ljudje ponavadi predstavljajo.

Uporabniški vmesnik je zgolj neformatirano besedilo. Vsi ukazi so v tekstovni obliki, ki jih uporabnik vpiše v terminal. Tudi vsi povratni podatki iz podatkovne baze so prikazani v terminalu v obliki neformatiranega besedila. Podobni igri, glede na vmesnik, sta še Adventure ter Zork [6].

Naprednejše MMOG igre so se pojavile proti koncu osemdesetih. Ena igra, ki je prvič uporabila grafičen prikaz, (kasneje tudi v 3D) je Kesmaijev Air Warrior. Igralec je lahko izbiral med več letali iz druge svetovne vojne in z njimi upravljal in streljal. Igralci so igrali preko modemov, dostop do igre je omogočala medmrežna storitev GEnie [10]. Kasneje so se pojavile druge igre MMOG, ki so uporabljale podobne posredniške storitve kot je GEnie, na primer GemStone (leta 1988) in Neverwinter Nights (leta 1991). V primeru Neverwinter Nights je bilo zaradi omejitev tehnologije največje število igralcev na začetku petdeset, leta 1995 so število povečali na petsto. Podoben trend pa je bil opazen tudi pri drugih igrah MMOG. Sredi devetdesetih se je zvrst igre MMOG razcepila na več podzvrsti kot so: MMORPG (angl. Massively Multiplayer Online Role Playing Game), MMORTS (angl. Massively Multiplayer Online Real Time Strategy), itd.

Okoli leta 1991 je prvič izšla Sid Meierjeva potezna igra Civilization. Igra deluje na principu, da igralec gradi imperij in premaga nasprotnike s pomočjo svoje vojske. Gradi tudi mesta in se mora odločati, katere tehnologije bo raziskal itd. Kasneje je izšla predelana različica za operacijski sistem Windows 3.1: Civnet. Posebnost te različice je bila, da je omogočala igro preko lokalne mreže in preko medmrežja. Igra je imela veliko nadaljevanj in dodatkov. Isto leto kot Civnet je izšla tudi igra Worms razvijalca Team17. Pri tej igri prvi igralec najprej opravi svoje poteze, nato šele pride na vrsto drugi igralec. Ko oba opravita poteze, se začne nov krog. V letu 2000 je izšla Creative Assemblyjev Shogun: Total War. Igra je strateška večigralska igra. Igralec upravlja s svojo vojsko in strateško premaguje nasprotnike, cilj pa je dobiti oblast nad celotno Japonsko [2]. Igra je sestavljena iz dveh igralskih sklopov: v prvem igralci med svojo potezo premaknejo vojsko po zemljevidu; v drugem pa poteka boj v realnem času.

Ena najbolj igranih plačljivih iger MMOG je WoW (angl. World of Warcraft), ki je izšla leta 2004. Leta 2009 je igra imela več kot 11 miljonov naročnikov in 62% ce-

lotnega tržnega deleža [27]. Ena najbolj igranih iger na Facebooku je bila Farmville, izdana leta 2009. Igralci lahko asinhrono in neposredno medsebojno vplivajo tako, da drug drugega obiščejo na virtualni kmetiji. V obdobju, ko so imeli prijavljenih največ igralcev, je za virtualne kmetije skrbelo kar 62 milijonov igralcev [15].

Od leta 2000 naprej je število večigralskih iger močno naraslo, vključno z MMOSG (angl. Massively Multiplayer Online Sports Game), kot tudi drugih oblik, npr. športnih in nogometnih iger. Implementacija asinhronega večigralskega načina, ki je nastala v okviru te diplomske naloge, je v osnovi nogometna igra, zato si bomo v nadaljevanju bolj podrobno pogledali nogometne igre in večigralnost v tej podzvrsti.

2.2 Nogometne igre

Ena izmed prvih nogometnih iger, ki je dejansko izgledala kot nogomet, je bila Intellivision Soccer, ki je izšla leta 1979. Je zelo osnovna igra, igralec lahko preigrava, podaja in strelja, toda nima nobene statistike in podobnih dodatkov. Igre, kot so Atari 2600 RealSports Soccer (leta 1983) in NES Soccer (leta 1985), so napredovale v igralnosti in v grafiki. Velik korak v nogometni zvrsti je naredila igra Sensible World of Soccer zaradi možnosti kariernega načina igre. Igra je izšla leta 1994. Igralec je lahko prevzel vlogo trenerja ali pa bil tako igralec na zelenici, kot tudi trener ekipe hkrati. Igralec je lahko kupoval in prodajal nogometne igralce v igri.

Isto leto je EA prvič izdala igro FIFA. Največja novost, ki jo je prinesla FIFA International Soccer, je zorni kot, s katerega igralec gleda igro. Igralec gleda igro približno tako, kot je na televiziji; prejšnje igre pa so bile skoraj vse s ptičje perspektive. V kasnejših izdajah je FIFA postajala ena najbolj igranih nogometnih iger, na primer 4,5 milijonov izvodov Fife 13 je bilo prodanih v prvih petih dnevih prodaje [17]. Nove izdaje igre FIFA prihajajo skladno z začetkom posamezne nogometne sezone.

Danes je ena izmed največjih tekmecev FIFE igra Pro Evolution Soccer (skrajšano PES). Prvič izdana leta 1995 z imenom Goal Storm, se ta igra v nasprotju s FIFA bolj osredotoča na igralnost, kot na licence za imena igralcev. Od prve izdaje naprej je Konami vsako leto izdal novo verzijo Pro Evolution Soccerja. Če seštejemo

vse verzije skupaj, je Konami prodal okoli 76 miljonov izvodov iger Pro Evolution Soccerja [16].

Igre v katerih ima igralec izključno vlogo trenerja nogometne ekipe igralcu ne omogočajo neposrednega vpliva nad tekmami, ki jih odigra. Igralec ima le posreden vpliv, kot je sestavljanje ekip, nakup in prodaja igralcev, določanje cene vstopnic, itd. Eden izmed prvih primerov takšnih iger je Championship Manager, izdan leta 1992. Grafični prikaz je zelo osnoven, igralci so izmišljeni in prikaz tekme v živo je sestavljen samo iz kazalcev posesti žoge, časovnice dogodkov in komentarjev v tekstovni obliki. Kompleksnejše igre iste zvrsti so se pojavile kasneje. Football manager 2005 je izšel leta 2004. Ena izmed novosti, ki jih je prinesla ta igra, je način prikaza tekem v živo. Namesto uporabe kazalcev kot način prikaza stanja na igrišču, je prikaz v obliki pogleda na stadion; na igrišču stadiona pa so krogi, ki predstavljajo igralce in žogo. V naslednjih letih je bila z vsako sezono izdana nova verzija Football Managerja. Najnovejši, Football Manager 2013, ima skoraj milijon prodanih izvodov [18] in je prejel visoke ocene [22]. Prikaz odigranih tekem se je izboljševal z vsako izdajo, pri Football Manager 2013 pa že skoraj izgleda kot prava nogometna tekma.

Poleg plačljivih iger se je pojavilo veliko zastonjskih iger, ki so se igrale preko brskalnika. Glede na grafiko in igralnost so načeloma slabše kot plačljive igre, so pa vseeno popularne zaradi dostopnosti.

Leta 1997 je izšla nogometna trenerska igra Hattrick. Igra se igra preko brskalnika. V njej ima igralec vlogo trenerja nogometne ekipe in sestavlja taktiko, izbira in kupuje igralce, nadgrajuje svoj domači stadion itd. Vsak igralec predstavlja eno ekipo v ligi. Lig je veliko, so razdeljene tudi glede na države. Obstajajo še druge podobne nogometne trenerske igre, ki bazirajo na igranju preko brskalnika. Nekatere uporabljajo obstoječe igralce in ekipe, kot so PitchSide Manager in Soccer Manager. Druge nogometne trenerske igre se osredotočijo na kakovosten prikaz odigrane tekme, kot recimo pri Striker Manager, pri katerem je osnoven prikaz odigrane tekme na igrišču z igralci [25].

Obstaja tudi nekaj nogometnih iger preko brskalnika, pri katerih ima igralec vlogo nogometnega igralca. Ena najbolj znanih je Power Soccer, pri kateri je igralec

nogometni igralec na igrišču s 3D grafiko. Lahko strelja, podaja, preigrava, podobno kot pri igrah FIFA in PES.

Navdih za nekaj osnovnih konceptov naše igre (Knoxball) smo dobili iz igre Haxball. Igra deluje samo na večigralski način v realnem času in ne vsebuje umetne inteligence. Vsi igralci so ljudje. Igra je mešanica nogometa in zračnega hokeja [21]. V zadnjih letih se je pojavilo tudi kar nekaj iger narejenih za mobilne telefone z večigralskim načinom.

2.3 Večigralnost na mobilnih telefonih

Vse do leta 1997 je bila ideja igric na mobilnem telefonu neizkoriščena. Toda leta 1997 je Nokia začela prodajati mobilni telefon Nokia 6110, ki je vsebovala igro Kača (angl. Snake) [7]. Kasneje je Nokia izdala mobilni telefon Nokia 7110, ki je imela prvo igro z večigralskim načinom. To je bila igra Kača v novejši obliki. Večigralski način igre se vršil v realnem času s pomočjo infrardeče povezave med mobilnima telefonoma [24]. Ena izmed prvih simulacij nogometa na mobilnem telefonu je bila Nokiina igra Soccer League.

Še ena nogometna igra je Tournament Arena Soccer 3D. Igra, ki je bila narejena v programskem jeziku Java, je delovala na platformi Symbian. Z zelo osnovno 3D grafiko je igralec lahko izbral eno izmed držav kot ekipo in igral v pokalu.

Dandanes obstaja ogromno število iger za mobilne telefone, ki so dosegljive preko Appstorea in imajo večigralski način igranja. Nekatere imajo precej podobnosti z našo igro.

Padkick je primer igre za mobilne telefone s platformo iOS, ki je v osnovi podobna spletni igri Haxball. Tako kot Haxball, ima igra samo večigralski način igre v realnem času. Pogosta težava iger, ki imajo večigralski način igre v realnem času, je zakasnitev. Pri igri Padkick rešujejo težave z zakasnitvijo tako, da se pri vsakem igralcu ves čas meri zakasnitev. Če igralec prestopa določeno mejo zakasnitve za daljši čas, je izključen iz igre. Prednost takšne rešitve je, da igra skoraj vedno teče tako tekoče, kot je bilo zamišljeno. Največja slabost te rešitve je, da se število možnih igralcev drastično zmanjša, saj veliko uporabnikov nima dostopa do dovolj

hitrih povezav na medmrežje preko svojega mobilnega telefona ali pa niso dovolj blizu strežnikom igre.

Še en nogometno-zračnohokejski hibrid je Soctics. Tako kot Haxball in Padkick, nima enoigralskega načina. Od Haxballa in Padkicka se razlikuje v tem, da ni v realnem času. Vsak igralec ima svojo potezo, ki določi, kako in s kakšno močjo se bo premikal vsak krogec v njegovi ekipi. Igra ima večigralski način s potezami v realnem času. Igralec nima omejenega časa za odločitev, toda če predolgo čaka, lahko nasprotnik zaradi dolgočasenja odide. Dobra plat tega načina je, da ni tipičnih težav, ki se pojavijo pri večigralskem načinu v realnem času. Morebitna slabost pa je, da ne spominja več na nogomet, ampak je bolj kot hibridna verzija igre med šahom in zračnim hokejem.

Verjetno najbolj znana igra z večigralskim načinom s časovnim zamikom na mobilnih telefonih je Real Racing 3. Smisel tega načina igre je, da lahko uporabniki tekmujejo s svoji prijatelji ali z neznanci, čeprav ne igrajo ob istem času. Igra vključuje celo trčenja med igralci, toda to ima svoje posledice. Posnetki igralcev niso točni, samo končni čas je enak. Kar se dejansko zgodi je, da igra naredi igralca z umetno inteligenco, ki cilja na končni rezultat igralca, ki je že odigral progo. Umetno-inteligenčni igralec tako omogoča igro, ki naj bi dajala občutek, da bi bila v realnem času [23].

Igra, pri kateri deluje večigralski način podobno kot pri Knoxballu, je Bike Race Pro. Bike Race Pro je igra, osnovana na podlagi fizike, pri kateri igralec poskuša priti po progi do cilja v čim hitrejšem času. Pri večigralskem načinu se je potrebno vpisati preko Facebooka kot načina overitve. Prvi igralec pošlje zahtevo po igri drugemu igralcu, skupaj z odigrano igro in doseženim časom, drugi igralec pa ima tri poizkuse, da premaga njegov rezultat. Ko drugi igralec odigra svoj del tekme, pošlje svoj posnetek igre skupaj s svojim rezultatom prvemu igralcu, ki lahko potem pogleda posnetek obeh igralcev skupaj.

Knoxball je poenostaljena simulacija nogometa za mobilne naprave z operacijskim sistemom iOS. Posebnost igre je, da je lahko med igralci v enoigralskem načinu poleg igralcev z umetno inteligenco tudi človeški igralec. Vendar pa v večigralskem načinu s časovnim zamikom ni človeških igralcev na igrišču. Igralca igrata v igri

vlogo trenerja posamezne ekipe. Na koncu si oba lahko ogledata odigrano igro. V naslednjem poglavju so opisane uporabljene tehnologije pri implementaciji asinhronega večigralskega načina igre.

Poglavje 3

Programska Tehnologija

Celoten projekt se sestoji iz treh delov:

- Igra Knoxball
- Vmesnik za asinhronski večigralski način
- Strežnik za pošiljanje in shranjevanje podatkov o igri

3.1 Knoxball

Knoxball je mobilna telefonska igra, narejena za mobilne telefone, ki imajo operacijski sistem iOS. V celoti je napisana v programskem jeziku Objective-C. Igra poleg drugih standardnih knjižnic iOSa uporablja knjižnico XNI. Podpira iOS od inačice 4.3 naprej. Deluje tudi na tabličnih računalnikih z operacijskim sistemom iOS. Za platformo iOS smo se (namesto npr. za platformo Android ali Windows) odločili zaradi programske opreme, ki je bila na razpolago, ter trenutne razširjenosti platforme iOS. Potrebovali smo računalnik z operacijskim sistemom OS X 10.8 in z Xcode 4.5 ter mobilni telefon z operacijskim sistemom iOS, ki ima inačico, višjo od 4.3. Za testiranje aplikacije na mobilnem telefonu smo potrebovali tudi univerzitetno licenco.

3.2 Večigralski vmesnik

Za izgradnjo vmesnika asinhronega večigralskega načina igre smo uporabili Interface Builder za iOS in Facebook SDK. Poleg tega smo uporabili standardne knjižnice iOS: `AddressBook`, `CoreLocation`, `Accounts`, `Security`, `Social` in `libsqllite3`. Večina je bila narejena v Interface Builderju, kar se ni dalo, pa smo programsko dodali z Objective-C. Za delovanje vmesnika in za dostop do podatkov uporabnikov aplikacije smo morali ustvariti tudi Facebook-aplikacijo.

Facebook smo izbrali kot način overitve zaradi razširjenosti. Tako bi bilo treba mnogo uporabnikom za prijavo pritisniti le na en gumb. Izbrali ga smo tudi zaradi enostavnosti uporabe knjižnice Facebook ter zaradi veliko dokumentacije.

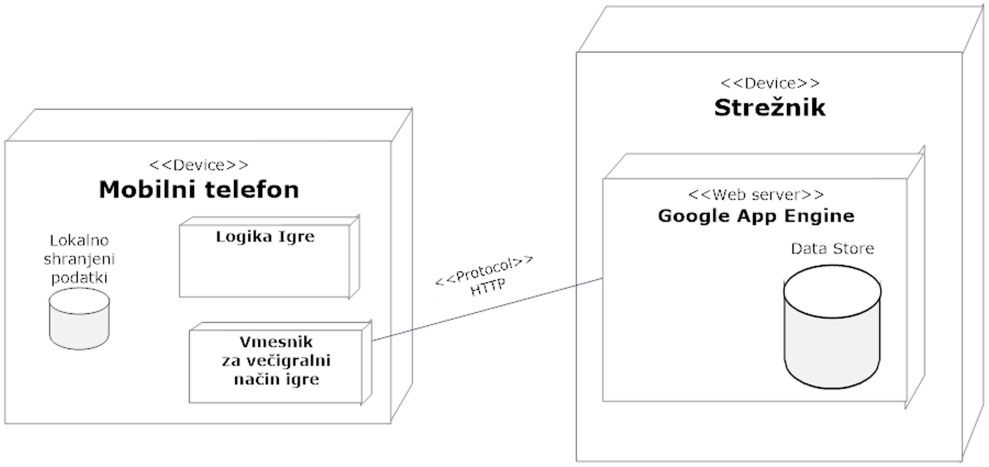
Sporočila, ki se pošiljajo med vmesnikom in strežnikom, so v obliki JSON (angl. Javascript Object Notation). Za JSON smo se, namesto oblike XML, odločili zaradi manjših sporočil z enako količino informacij. V vmesniku ima iOS vgrajene metode za razčlembo sporočil JSON v obliki slovarja. Za pošiljanje sporočil JSON preko HTTP smo uporabili vgrajene metode.

3.3 Strežnik

Pri asinhronskem večigralskem načinu igre potrebujemo strežnik za shranjevanje in vračanje zahtev po igri ter posnetkov odigranih iger. Strežnik smo razvijali v okolju Eclipse. Za strežnik je bilo veliko možnosti glede na razpoložljivo programsko opremo, kot tudi ponudnike. Odločali smo se med Amazonom [1], Google App Engine[2] ter Heroku[3]. Iz finančnega vidika smo izločili Amazon, saj po enem letu ni več zastonj, ne glede na promet. Glede na to, da je samo Google App Engine ponujal naložitev aplikacije v jeziku Java, smo se tako odločili za Google App Engine. Aplikacijo smo želeli v Javi zaradi široke izbire med knjižnicami in pa tudi zaradi predhodnih izkušenj s programiranjem v Javi.

Sporočila preko HTTP z metodo POST v obliki JSON smo pretvorili s pomočjo

[1] http://aws.amazon.com/
[2] https://developers.google.com/appengine/
[3] https://www.heroku.com/



Slika 3.1: Arhitektura odjemalec strežnik igre Knoxball.

knjižnice GSON (Google Gson), ki spremeni JSON v objekt Java. Za shranjevanje podatkov smo uporabili *Google App Engine Datastore*, za sam postopek shranjevanja pa smo uporabili JDO (angl. Java Data Objects) in JPA (angl. Java Persistence API) v implementaciji DataNucleus, ki objekte shrani neposredno v bazo in ne uporablja SQLa ampak jezik podoben SQLu, GQL (angl. Google Query Language). Pri poizvedbah vrača sezname objektov, ki so nato pretvorjeni v obliko JSON s pomočjo GSON. Struktura igre je vidna na sliki 3.1. V naslednjem poglavju si bomo podrobneje pogledali delovanje vmesnika za večigralski način igre s časovnim zamikom.

Poglavje 4

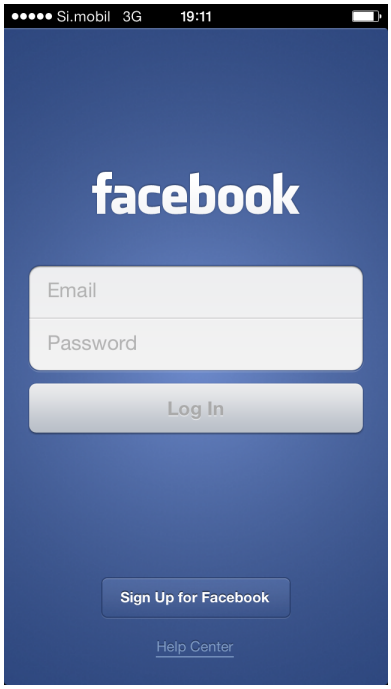
Vmesnik za večigralski način igre

Smisel vmesnika je, da se uporabnik lahko overi na intuitiven način. Zahteve po igri pošilja pregledno, da lahko vidi zahteve drugih prijateljev uporabnika, ter vidi tudi svoje zahteve, ki jih je poslal drugim. Uporabnik lahko vidi tudi seznam že odigranih tekem ter lahko izbere kakšno že odigrano tekmo in lahko pogleda posnetek.

4.1 Facebook avtentikacija

S pomočjo Facebook SDKja ter Facebook-aplikacije se uporabnik lahko overi. Prvič, ko uporabnik izbere asinhronski večigralski način igre, ga vmesnik napoti na stran za prijavo s pomočjo Facebooka. Ko uporabnik klikne gumb za prijavo, se sproži aplikacija Facebook oziroma brskalnik, če uporabnik na mobilnem telefonu nima aplikacije Facebook (slika 4.1). Knoxball postane proces v ozadju in druga aplikacija se prikaže na ekranu. Ko se uporabnik overi preko Facebooka, se prikaže obvestilo, če se uporabnik strinja s pogoji aplikacije Knoxball, vidno na sliki 4.2. Tudi na Facebooku mora obstajati Facebook-aplikacija Knoxball. Omenjeno aplikacijo uporabi Facebook za overitev same mobilne telefonske aplikacije Knoxball, saj ko enkrat uporabnik sprejme zahteve po njegovih podatkih preko Facebooka, se ta odločitev shrani na Facebooku, da ni potrebe po vnovičnem spraševanju za odobritev pravice.

Da druge aplikacije ne pridejo do informacij uporabnikov brez njihove odobritve,

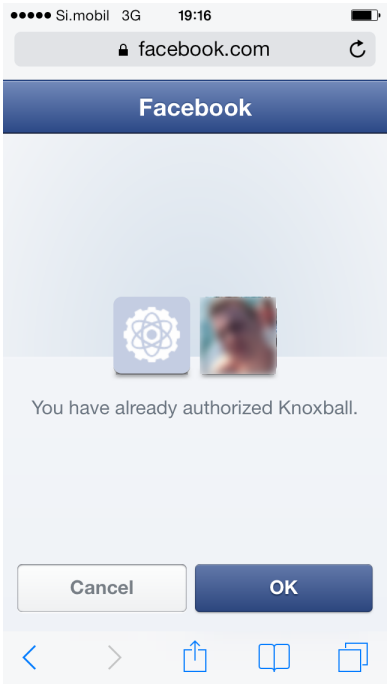


Slika 4.1: Overitev s pomočjo Facebook-aplikacije.

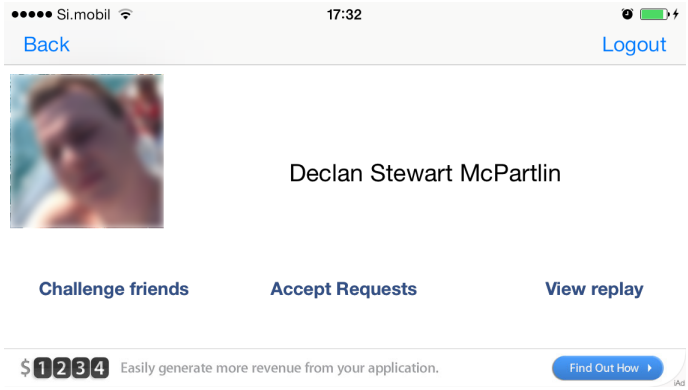
mora imeti vsaka mobilna telefonska aplikacija, ki uporablja Facebook, tudi ustrezno aplikacijo na Facebooku. V tej aplikaciji lahko tudi določimo katere podatke bo aplikacija zahtevala od uporabnika. Ko uporabnik sprejme zahtevo po njegovih podatkih, postane Facebook ali brskalnik proces v ozadju in v ospredje se ponovno vrne igra Knoxball. V tem trenutku se prikaže prva stran za prijavljene uporabnike, ki prikazuje podatke o igralcu ter meni (slika 4.3).

4.2 Funkcionalnosti vmesnika

Prva možnost, ki jo ima uporabnik, je da pošlje zahtevo prijatelju. To naredi tako, da pritisne na gumb, pri čemer se prikaže stran z vsemi njegovimi prijatelji na Facebooku. Omenjena stran prihaja iz Facebook-SDKja in jo je potrebno samo prikazati, izgled je viden na sliki 4.4. Ko uporabnik klikne na prijatelja, se označi kot izbran. V osnovi lahko igralec izbere več igralcev. Ko je uporabnik zadovoljen in zapre stran, aplikacija sprejme seznam izbranih prijateljev v posebni obliki. Informacija



Slika 4.2: Po prejetju zahteve Facebook aplikacije.



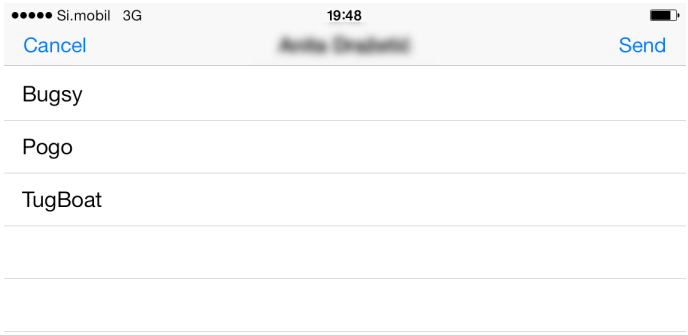
Slika 4.3: Izgled vmesnika takoj po vpisu na Facebook. Gumbi so spodaj vidni v obliki besedila, gumbi z možnosti izhoda so na vrhu.

Slika 4.4: Tu uporabnik izbere kateremu prijatelju bi poslal zahtevo. Ko se odloči za prijatelja, klikne na gumb Done in ga vmesnik preusmeri na stran za izbiranje igralcev iz ekipe.

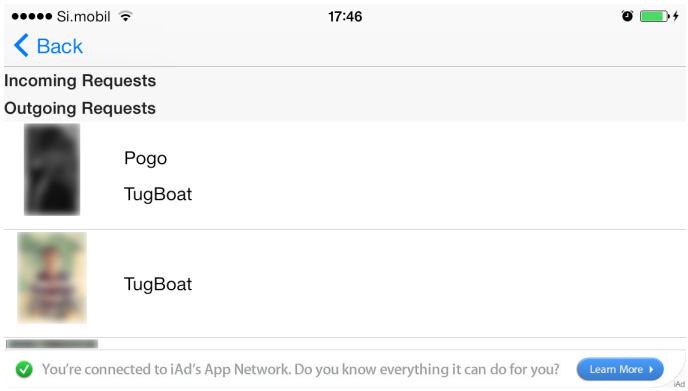
prihaja iz Facebookove baze v obliki objektov `UserGraph`. Objekt `UserGraph` vsebuje podatke o uporabnikih s Facebooka. Vsi podatki, pridobljeni iz Facebooka o uporabniku, prihajajo v obliki `GraphObject`. `GraphObject` je osnovni protokol o tem, kakšno obliko mora imeti objekt iz Facebooka. Če uporabnik izbere več kot enega prijatelja, ga vmesnik opozori, da je bil izbran samo prvi.

Po izbiri prijatelja vmesnik prikaže stran za izbiro njegovih igralcev, primer na sliki 4.5. Seznam igralcev predstavlja uporabnikovo ekipo. Uporabnik trenira, prodaja in kupuje igralce v enoigralskem načinu igre, kjer tekmuje v virtualnih ligah in virtualnih prvenstvih. Število igralcev, ki jih izbere uporabnik, določi koliko jih mora izbrati njegov nasprotnik. Uporabnik ne more izbrati več kot treh igralcev. Ko se uporabnik dokončno odloči, pošlje zahtevo s pomočjo gumba 'pošlji'. Vmesnik generira sporočilo JSON in ga pošlje na strežnik.

Med drugim lahko uporabnik pregleda vhodne in izhodne zahteve po igri, kot na sliki 4.6. Vhodne zahteve lahko igralec tudi sprejme. Vmesnik pošlje zahtevo strežniku po seznamu iger ali seznamu zahtev po igrah, v katerih je uporabnik udeležen. Nato vmesnik določi, ali gre za vhodno ali izhodno zahtevo po igri. Izhodne zahteve po igri prikazujejo zahteve po igri, ki jih je uporabnik poslal drugim prijateljem in še niso dobili povratnega odgovora. Vhodne zahteve po igri so pa tiste zahteve po igri, ki jih je uporabnik prejel od prijateljev. Ko uporabnik sprejme



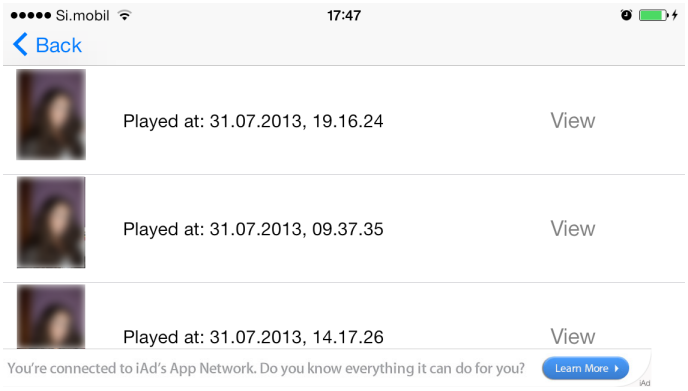
Slika 4.5: Stran prikaže seznam igralcev, ki jih ima uporabnik v enoigralskem načinu. Ko uporabnik izbere do tri igralce, pošlje zahtevo s pomočjo gumba Send.



Slika 4.6: Tu se prikaže seznam vhodnih in izhodnih zahtev. Pri vhodnih zahtevah je prisoten gumb Accept za sprejem zahtevo po igri.

zahtevo po igri od prijatelja, vmesnik prikaže stran za izbiro njegovih igralcev. Podobno kot pri pošiljanju zahteve po igri, se pri sprejemu zahteve po igri takoj po izbiri igralcev, shrani izbiro na strežniku kot posnetek igre. Takoj po sprejemu zahteve po igri lahko oba udeležena uporabnika pogledata posnetek igre. Medtem, ko imata dva uporabnika nedokončano zahtevo za igro, ne moreta začeti nove zahteve, dokler se ni končala prva.

Še zadnja funkcionalnost vmesnika je seznam odigranih tekem in posnetki tekem. Vmesnik pošlje zahtevo strežniku za seznam posnetkov tekem, v katerih je bil uporabnik udeležen. V seznamu se tudi prikaže informacija o času in datumu tekme. Dva uporabnika imata lahko več odigranih tekem med seboj. Ko uporabnik



Slika 4.7: Prikaz seznama vseh posnetkov iger z drugimi prijatelji s Facebooka. Če pritisnemo na gumb View, se začne predvajanje tekme.

klikne za ogled izbrane tekme, se zapre vmesnik in se odpre stran od igre. Igra se začne takoj po pritisku gumba. Primer izgleda je prikazan na sliki 4.7.

4.3 Posnetki iger

Dejanski posnetki se ne shranjujejo, ampak se shranjujejo samo podatki o igralcih. V algoritmih igralcev je prisotno nekaj šuma, da ne bi prišlo do enakih potekov tekem. Uporabljali smo psevdonaključno funkcijo `rand()`. Deluje na osnovi semen in za določeno seme zmeraj vrne isto zaporedje vrednosti. Pri funkciji `rand()` se določi seme s pomočjo funkcije `srand(int seed)`. Zaradi tega se ob shranjevanju igralcev na strežnik poleg osnovnih podatkov o igralcu vsakemu igralcu dodeli dejansko naključno seme s pomočjo funkcije `arc4random()`, ki ni psevdonaključna. Nato med igro seme povečamo vsak krog. Vsakič, ko uporabnik pogleda posnetek igre, bo posnetek popolnoma isti, čeprav pozicije igralcev niso shranjene za vsako obhod zanke igre. Na ta način se naložijo 'posnetki' tekem skoraj takoj, namesto daljšega nalaganja posnetkov tekem, ki bi imeli velikost tudi do več MB. V primeru dveh različnih tekem s popolnoma enakimi igralci je verjetnost, da bosta tudi posnetka iger enaka, blizu ničle. Toda način ima nekaj slabosti, ni mogoče spreminjati hitrosti predvajanja ali preskoka po časovnici. Za ta način smo se odločili zaradi hitrosti in sposobnosti strežnika.

4.4 Komunikacija s strežnikom

Vsa komunikacija med vmesnikom in strežnikom se vrši s protokolom HTTP z metodo POST ter v obliki JSON. Ko vmesnik pošilja podatke na strežnik, jih preslika najprej v `NSDictionary`, nato v JSON s pomočjo vgrajene funkcije `NSJSONSerialization`, ki spremeni slovar v obliko JSON, pripravljeno za pošiljanje. Objekti, ki se pošiljajo preko vmesnika, na primer igralec, imajo metodo za preslikavo njihovih lastnosti v obliki slovarja. Ko je zahteva JSON pripravljena za pošiljanje, se jo pošlje na določen URL glede na to, ali gre za: pošiljanje zahteve po igri, sprejem zahteve po igri ali zahtevo po zahtevku iger uporabnikov oziroma posnetek tekme. Pošlje se s pomočjo vgrajene funkcije `NSURLConnection`. Pri sprejemu se pogleda, ali strežnikovo sporočilo JSON vsebuje ključ 'status'. Pri zahtevah, kot so pošiljanje zahteve po igri in sprejem zahtev po igri uporabnikov, strežnik samo vrne odgovor, ali se je uspešno vpisalo v bazo ali ne. Pri zahtevah za seznam zahtevkov po igri ali po posnetkih načeloma ni ključa 'status', edino če je prišlo do napake. V tem primeru vmesnik pokaže razlog v obliki sporočila iz baze. Kadar ni napak, se sprejet JSON pretvori v objekt oblike `KnoxballGameRequestObject`. Omenjeni objekt vsebuje identifikacijo obeh igralcev, morebiten datum, če je objekt posnetek, ter seznam igralcev obeh ekip. Znotraj tega objekta se instancirajo igralci ekip s pomočjo konstruktorja z vhodnim podatkom JSON. Če gre za sprejem zahteve po igri, se `KnoxballGameRequestObject` preslika v JSON in se tako neposredno pošlje. V primeru posnetka se instancirani igralci dodajo v svojo ekipo pripravljeni za igro. V naslednjem poglavju si bomo bolj podrobno pogledali še delovanje strežnika.

Poglavje 5

Strežnik za pošiljanje in shranjevanje podatkov o igri

Strežnik ima funkcijo shranjevanja zahtev in posnetkov tekem, obenem pa mora biti ves čas dostopen. Facebookov SDK je vir velikega števila pomembnih podatkov. Medtem ko podatke vrača, jih ne sprejema in ne shranjuje, zato smo morali ustvariti svoj strežnik. Naš strežnik je osnovan na Google App Engine in ima svoj URL: `knoxballbackend.appspot.com`. Ima tudi tri posebne URLje za tri različne funkcionalnosti, opisane v prejšnjem poglavju. Podatki se shranjujejo v Googleovo bazo s pomočjo programskega vmesnika Google Datastore. Poizvedbe se za omenjeno bazo pišejo v obliki GQL. Po obliki je podoben SQLu, toda ne podpira stika. Objekte Java zapiše v *Google App Engine Datastore* s pomočjo knjižnic JDO in JPA. Tudi pri rezultatih poizvedb *Google App Engine Datastore* sezname objektov Java vrača neposredno. Strežnik sprejema samo sporočila JSON po protokolu HTTP ali HTTPS na način POST. Sporočila v obliki JSON pretvori v objekte Java s pomočjo knjižnice GSON.

Resna težava, ki se lahko zgodi, je prestrežba podatkov in spreminjanje podatkov o igri. Večja težava je možnost vrinjanja lažnih iger direktno na strežnik brez uporabe mobilnega telefona. Rešitev je v kodiranju podatkov. Ena izmed implementacij, ki je tu uporabna, je uporaba zasebnih in javnih ključev. Tako strežnik kot odjemalec imata svoj zasebni in javni ključ. Odjemalec pošilja sporočila za-

kriptirana s pomočjo strežnikovega javnega ključa, nato že zakriptirano sporočilo zakriptira še z zasebnim ključem, kot način overitve. Strežnik prejeto sporočilo odkriptira s pomočjo odjemalčevega javnega ključa, nastalo sporočilo pa odkriptira s svojim zasebnim ključem. Princip velja tudi pri sporočilih v obratnem primeru [8].

5.1 Sporočila z vmesnikom

GSON je knjižnica v Javi, ki je uporabljamo pri pretvorbi objektov Java v obliki JSON. Lahko se tudi uporablja v obratni smeri, torej pretvorba JSON sporočila v Javine objekte. GSON naj bi deloval tako, da naj ne bi bilo treba imeti nobenih zaznamb napisanih v tistih objektih Jave, ki jih želimo pretvoriti. Zato je mogoče pretvoriti tudi objekte, katerih izvorne kode nimamo, recimo Javine objekte iz določenih knjižnic. Takšnih objektov se ne bi dalo pretvoriti, če bi bilo nujno dodati zaznambe v kodo. GSON naj bi skoraj popolnoma podpiral Javine generične tipe in objekte [20]. GSON pretvori JSON iz oblike niza neposredno v obliko objekta Java s pomočjo funkcije `fromJSON`. GSON pretvori tudi Javanske objekte v obliko JSON neposredno s funkcijo `toJSON`. GSON podpira tudi pretvorbo poljubno kompleksnih objektov, torej objekte z globoko podedovano hierarhijo ali večjo uporabo Javinih generičnih tipov [20].

Na sestavo naših objektov je imel velik vpliv GSON. Osnovni razred za zahteve in posnetke tekem je `GameInformation`, ki vsebuje identifikatorje obeh igralcev, datum ter sezname igralcev. Razred za igralce, `GamePlayerCircle`, vsebuje vse lastnosti igralcev. Poleg teh razredov ima strežnik še razred za pošiljanje sporočila iz baze – `GameResponse` – za lažje pretvorbe in pošiljanje sporočil JSON. Objekt za razčlembo zahtev za igro in posnetke iger pa je `RequestForIncomingRequestsOrReplay`. Na koncu smo se raje izognili uporabi generičnih tipov za zanesljivejše delovanje celotnega sistema.

5.2 Vpisovanje v podatkovno bazo

Pri izbiri podatkovnih baz smo si želeli delovanje, ki bi bilo čim bolj generično. Pri pisanju v podatkovne baze, ki niso objektno orientirane oziroma ne vpisujejo 'entitet' in uporabljajo SQL, bi si bilo treba celotno arhitekturo baze načrtovati, jo zgraditi in nato še dodati na strežnik. Imeli smo lažjo rešitev. Uporabili smo Googleov Datastore API, ki podpira takšno objektno orientirano zapisovanje v bazo. Toda za delovanje pri vpisovanju pravih, osnovnih objektov Java oz. POJO (angl. Plain old Java object), je potrebna uporaba knjižnice JDO ter uporaba specifikacij programskega jezika JPA.

JDO je vrsta shranjevanja trajnih objektov Java (angl. Java object persistence). Trajni objekti JDO so osnovni objekti Java, POJO [9]. Pri uporabi JDO ni potreb po implementaciji vmesnikov ali dedovanja iz določenih razredov v omenjenih objektih Java. Poleg tega, da je JDO standard za mapiranje iz objektnega v relacijski model, je tudi standard za transparentno shranjevanje trajnih objektov. Iz vidika programskega vmesnika naj bi bil JDO neodvisen od vrste implementacije tehnologije, ki jo uporablja podatkovna baza, s katerim ima opravka. Prej sta bila JPA in JDO popolnoma ločena, saj je bil JPA del EJB3 (angl. Enterprise Java Beans 3) in imel podobne funkcionalnosti kot JDO. Kasneje so JPA odstranili iz EJB3 ter zanj ustvarili lastno specifikacijo JPA. Zdaj vodilne komercialne implementacije JDOja ponujajo implementacijo, ki vsebuje JPA kot alternativen način dostopanja do temeljne trajne hrambe [9]. Implementacijo, ki smo jo izbrali mi, je DataNucleus. DataNucleus smo izbrali zato, ker dodatek DataNucleus za Google App Engine že obstaja. DataNucleus uporablja tako JDO kot JPA [9].

JPA opisuje upravljanje z relacijskimi podatki v aplikacijah, ki uporabljajo plaftormo Java s standardno ali poslovno izdajo. JPA se uporablja za dodajanje anotacij v razred Java, da se lahko pretvori v trajno entiteto. Trajna entiteta je načeloma lažja oblika razreda Java, katerega stanje je shranjeno kot tabela v relacijski podatkovni bazi. Instance teh entitet se prikažejo v podatkovni bazi kot vrstice v tabelah. Entitete so lahko povezane z drugimi entitetami, povezave so opisane znotraj objektno/relacijskih metapodatkov. Omenjeni metapodatki so lahko določeni že neposredno znotraj razredov Java, kot je bilo omenjeno prej, v obliki zaznamb.

Metapodatki so lahko drugod tudi določeni, recimo v obliki XML v opisani datoteki, ki se razpečuje skupaj s celotno aplikacijo [5].

V praksi smo uporabili iste razrede kot pri pretvorbi sporočila JSON v objekte. Objektoma `GameInformation` in `GamePlayerCircle` smo morali določiti ključe ter anotacije napisati za vsakega izmed polj, ki naj bi se shranili v podatkovno bazo. Ker tako `GameInformation` kot `GamePlayerCircle` nimata nobenega polja, s katerim bi lahko enolično identificirali eno instanco, je bilo treba dodati polje `Key`, ki prihaja iz Google Datastore APIja. Pri zapisovanju in vpisovanju v podatkovno bazo potrebujemo upravljalni razred `PersistanceManager`. S pomočjo tega upravljalca se vračajo poizvedbe iz *Google App Engine Datastore* že samo z eno upravljalčevo funkcijo `newQuery`, pri kateri je edini vhodni podatek razred, ki nas zanima. To velja samo v primeru, ko želimo celoten seznam vseh instanc določenega razreda. Isto velja pri zapisovanju entitet v podatkovno bazo. Kličemo upravljalčevo funkcijo `makePersistent`; instanca, ki je edini vhodni podatek, pa se zapiše v podatkovno bazo. Na koncu uporabljanja upravljalca je potrebno njegovo instanco nujno zapreti.

5.3 *Google App Engine Datastore* in poizvedbe GQL

Google App Engine Datastore je brezshemno objektno orientirano podatkovno skladišče, ki omogoča skalabilno skladiščenje [13]. V primerjavi s tradicionalnimi relacijskimi podatkovnimi bazami je nekaj razlik. *Google App Engine Datastore* uporablja distribuirano arhitekturo za avtomatsko upravljanje s skaliranjem na zelo velike podatkovne zbirke. Še ena velika razlika je v načinu, kako opisuje povezave med podatkovnimi objekti. Entitete istega tipa imajo lahko različne lastnosti, hkrati pa imajo lahko različne entitete lastnosti z istim imenom, toda z različnimi vrednostnimi tipi. Zaradi teh karakteristik je potreben drugačen način upravljanja s podatki, da se lahko izkoristi zmožnost skaliranja avtomatično. *Google App Engine Datastore* je oblikovan tako, da skalira. To omogoča aplikacijam, da obdržijo visoko učinkovitost, medtem ko sprejemajo povečan promet. Druga večja razlika je ta, da *Google App Engine Datastore* ne zahteva, da imajo entitete istega tipa enak nabor lastnosti.

Tretja večja razlika je ta, da so vse poizvedbe postrežene s pomočjo prej zgrajenih indeksov. Zaradi tega so tipi poizvedb, ki so lahko izvedeni, bolj omejeni kot v relacijskih podatkovnih bazah s poizvedbami SQL [13].

Google App Engine Datastore zaradi tega ne more uporabljati SQLa temveč uporablja poizvedovalni jezik imenovan GQL. GQL v primerjavi z SQL nudi precej skromnejši nabor funkcionalnosti. GQL ne more izvesti stika, filtriranja glede na neenakost na več poljih, ter filtriranja podatkov, ki bazirajo na rezultatih podpoizvedb [13]. GQL se uporablja za pridobivanje entitet in ključev iz skalabilnega *Google App Engine Datastore*. GQL ima sicer sintakso podobno SQL. Ena od zanimivosti je, da če uporabnik želi preveriti, če dana entiteta deduje od določenega razreda, lahko to stori pomočjo operatorja `ANCESTOR IS` [19].

5.4 Nastali problemi s sistemom

Na koncu nam je uspelo vse tehnologije uspešno uporabiti, toda soočili smo se z nekaj ovirami pri vzpostavljanju pravilno delujočega strežnika. Nekatere tehnologije so dokaj nove in še ne brez napak. Odvečne dokumentacije tudi ni bilo. Sledi nekaj problemov, s katerimi smo se srečali.

Pri sprejemu zahtev za igro strežnik ni imel težav. Dobil je sporočilo, ga pretvoril v Javanski objekt in ga vpisal v *Google App Engine Datastore*. Težava je nastala, ko je strežnik dobil posnetek igre v sporočilu in je pri pretvorbi v Javanski objekt prišlo do napake. Izkazalo se je, da je pri pošiljanju seznamov zahtev za igro strežnik skupaj z uporabnimi podatki pretvoril tudi ključe in jih dodal v JSON. Vmesnik je nazaj poslal posnetek skupaj s prej določenimi ključi in pri pretvorbi je prišlo do napake, saj se ključ sam generira. Rešitev je uporaba GSON zaznambe `transient` v razredu Java.

Pri poizvedbi seznamov zahtev za igro ali posnetkov tekem dobimo samo seznam vseh shranjenih instanc določenega razreda, če v poizvedbi ne določimo ničesar drugega. Zanimajo nas samo posnetki vseh tekem, v katerem je bil en igralec udeležen. Če poskusimo izvesti poizvedbo, ki pregleduje identifikator obeh igralcev, bo potrebno uporabiti stik. Pri takšnih podatkovnih bazah se tovrstnih poizvedb ne da

izvesti. Takšno težavo je treba rešiti na drugačen način. Ko smo naredili poizvedbo na *Google App Engine Datastore*, smo nazaj dobili seznam zahtev ali posnetke tekem. Sprva izgledajo rezultati poizvedbe sprejemljivi, toda ko so se rezultati pretvorili v obliko JSON in jih poslali vmesniku, se je razkrila težava. Objekti igralcev so bili zmeraj prazni. *Google App Engine Datastore* uporablja sistem "lenih rezultatov" (angl. Lazy Results) za hitrejše poizvedbe. Sistem je v mnogih primerih načeloma zelo uporaben, toda v našem primeru se je izkazal kot precejšno težavo. Dokler se povezane entitete ne kličejo, se ne napolnijo s podatki. Tako smo našli in rešili težavo. Objekt `GameInformation` je prej imel tako strukturo, da je imel dva seznama obeh ekip. Neposredna uporaba generičnega tipa `ArrayList<GamePlayerCircle>` se je izkazala kot neuporabna, saj je GSON ni znal pretvoriti v generično obliko, čeprav so tako zatrjevali [20]. Težavo smo rešili tako, da generičnih tipov nismo uporabili, saj je vidno, da ne deluje točno tako, kot bi si želeli.

Poglavje 6

Umetna Inteligenca

Igra je odigrana na igrišču, ki je lahko različnih velikosti. Na igrišču sta dva gola in ena žoga. V primerjavi z nogometom tu avta ni, tudi prepovedanega položaja ni. Igralci se lahko premikajo tudi malo izven igrišča. Standardna igra traja 3 minute.

V vsaki igri sta dve ekipi, v kateri imamo od enega in treh igralcev, med katerimi je lahko tudi človeški igralec. Ekipe so končni avtomati (angl. FSM - Finite State Machines) [1] in so lahko v stanju napada ali obrambe. Med igralci ni komunikacije, zato ima pravico za komunikacijo z igralci dodeljeno "ekipa", ki med igro dodeljuje položaje igralcem v ekipi. Glede na število igralcev ekipa dodeljuje različne položaje. V izjemnih primerih lahko ekipa igralcem stanje tudi ponastavi.

6.1 Pozicije in stanja igralcev

Ko je v ekipi samo en igralec, mu je lahko dodeljena samo pozicija vratarja. Ko sta v ekipi dva igralca, sta možni poziciji vratar in napadalec. Ko so trije igralci, je možna še pozicija asistenta. Glede na pozicijo ima vsak igralec različen nabor stanj. Primer položajev in stanj je viden na sliki 6.1. Na nabor stanj vpliva tudi stanje ekipe. Ko je vratar v napadu, so njegova stanja sledeča: drži žogo, nima žoge, preigravanje, podaja, strel, zataknjen. Ko je vratar v obrambi ima stanja: nima žoge, drži napadalca, prestrežba, rešuje gol, zataknjen. Napadalec ima podoben nabor, toda ne rešuje gola. Ko je ekipa v stanju napada, ima asistent samo stanje 'nima

Slika 6.1: Ko je razhroščevanje vklopljeno, se vidi številke, ki nakazujejo trenutno stanje igralca, ter barva, ki kaže trenutno pozicijo igralca.

žoge'. V trenutku, ko asistent dobi žogo, postane napadalec, prejšnji napadalec pa postane asistent. Ko je ekipa v stanju obrambe, ima asistent enak nabor kot druge pozicije igralcev, razen reševanja gola.

Igralce vodijo agenti, ki se samostojno premikajo po igrišču. Agenti so tudi končni avtomati [1]. Različni cilji krmilijo agente po igrišču, ki so določeni znotraj vsakega stanja. Vsako stanje sestoji iz enega ali več ciljev. Večina ciljev je osnovnih, kot so na primer: premik proti žogi, prestrežba, vrtenje okoli žoge.

Nekatera stanja postanejo precej kompleksna s kombinacijo večih ciljev, ocenjevanj in izračunov. Pri podaji se najprej proti drugemu igralcu izračuna ocena o tem, kako smiselna je ta podaja. Ocena sestoji iz treh delov: čas vrtenja okoli žoge, nevarnost prestrežbe igralcev nasprotne ekipe ter lega soigralca. Vsak del ima svojo utež, ki določa, kako pomemben je omenjeni del. Igralec nato oceni svoje soigralce in če je vredno podati kateremu izmed njih. Če drži, se odloči za preigravanje.

Precej zanimiv algoritem v tem primeru je izračun nevarnosti prestrežbe igralcev nasprotne ekipe. Igralec, ki ocenjuje soigralce za podajo, preslikamo v lokalen koordinatni sistem, glede na premico med žogo in tarčo. Ostale pomembne dejavnike preslikamo tudi v svoj lokalen koordinatni sistem, kot so tarča (ki ima x-koordinato 0) ter vsi igralci iz nasprotne ekipe. Večja kot je y-koordinata pri nasprotnih igralcih, večji obseg kroga nevarnosti ima. Obseg kroga predstavlja čas, ki ga ima igralec nasprotne ekipe na razpolago za prestrežbo žoge. Igralci nasprotne ekipe z nega-

Slika 6.2: Prikaz algoritma za računanje nevarnosti. V tem primeru sta možna dva igralca nasprotne ekipe, ki bi lahko prestregla žogo. Modra črta prikazuje os x lokalnega koordinatnega sistema igralca, rdeča pa os y. Velikost rumenih krogov okoli nasprotnikov prikazuje območje, kamor bi se nasprotnik lahko premaknil glede na čas, ki ga ima, preden gre žoga mimo njega.

tivno vrednostjo y-koordinate niso pomembni, saj so v trenutku podaje postavljeni za igralcem z žogo. Po oceni za nevarnost vsakega igralca nasprotne ekipe posebej se izračuna skupno nevarnost podaje. Grafičen izgled izračuna je viden na sliki 6.2.

Na podoben način se igralec odloči za strel. Določi si seznam točk na črti gola in oceni vsako točko posebej, če predstavlja ugodno tarčo. Toda tu na oceno vpliva samo čas vrtenja okoli žoge ter nevarnost strela. Nevarnost strela na enak način računamo kot pri podaji.

Preigravanje deluje na drugačen način. Načinov preigravanja nasprotnika je več vrst. V trenutku, ko igralec hoče preigrati nasprotnika, si naključno izbere en način preigravanja, glede na to, katere pozna. Stanje preigravanja se največkrat sestoji iz več ciljev, ki se uresničujejo po vrsti. V primeru odboja od zida in nazaj je igralcu prvi cilj pripeljati žogo do bližine zida. Naslednji cilj je brcniti žogo tako, da se odbije od zida in nazaj k igralcu. Zadnji cilj je ponovno prejeti žogo. Primer enega izmed načinov preigravanja je na sliki6.3.

Slika 6.3: Primer načina preigravanja. Tu igralec skuša pretentati nasprotnika tako, da žogo odbije od stene po poti rdeče puščice, sam pa potuje po poti modre puščice, kamor z malo sreče kasneje pripotuje tudi žoga.

6.2 Lastnosti igralcev

Igralci si med seboj niso enaki. Vsak ima vnaprej določene lastnosti, ki dodajo novo dimenzijo igri. Obstajajo univerzalne lastnosti, ki veljajo tudi za človeške igralce. Te so: moč brca, pospešek, maksimalna hitrost in masa. Obstajajo še lastnosti, ki jih imajo samo igralci z umetno inteligenco: velikost območja nevarnosti, nadzor žoge, natančnost pri podaji in strelu, meja tveganja pri podaji, meja tveganja pri strelu, napadalnost in seznam poznanih preigravanj.

Velikost območja nevarnosti določi, kako blizu mora biti najbližji igralec nasprotne ekipe, da se počuti ogroženega. Ko se igralec počuti ogroženega, poskusi žogo dati stran od sebe, toda ne nasprotni ekipi. Najprej preveri, če je vredno streljati na gol, če ne, poskusi najti soigralca, kateremu lahko poda. Če ni vredno podati nobenemu igralcu, poskusi igralca nasprotne ekipe preigrati.

Kontrola žoge in natančnost pri podaji in strelu sta si dokaj podobni lastnosti. Ko igralec vodi žogo proti tarči, gre lahko po najbolj optimalni poti ali po manj optimalni poti. Glede na svoj nadzor se določi, kako bo optimalno prišel do tarče. Natančnost pri podaji odloča, kako natančno bo zadel določeno tarčo. Igralci z visoko natančnostjo bodo zmeraj natančno brcali proti tarči, medtem ko bodo manj natančni igralci kdaj natančni, včasih pa tudi ne.

Meji tveganja pri podaji in strelu ima direktno povezavo z oceno ugodnih podaj

in strelov. Glede na oceno podaje ali strela v primerjavi z njegovo najnižjo še sprejemljivo mejo tveganja, se odloči za strel oziroma podajo. Na primer, igralec z zelo nizko mejo tveganja pri podaji, bo skoraj vedno podal, čeprav bo najverjetneje igralec iz nasprotne ekipe dobil žogo. Igralec z visoko mejo tveganja pri podaji, skoraj nikoli ne bo podal žoge, saj bo zmeraj kakšna nevarnost ali pa bo soigralec v neugodnem položaju.

Igralec v ekipi, ki je v obrambnem stanju, v osnovi drži enega od igralcev nasprotne ekipe. Igralec se za prestrežbo odloči na podlagi oddaljenosti od žoge in oddaljenosti igralca nasprotne ekipe od žoge ter ti dve razliki primerja. Če se igralec odloči za prestrežbo, je od tega odvisno še kakšno vrednost agresivnosti ima. Igralec z visoko vrednostjo agresivnosti bo poskusil prestreči skoraj vsako podajo, medtem ko bo igralec z nizko vrednost agresivnosti prestregal samo, če bo zagotovo uspešen.

Na igralce in žogo deluje fizika. Vsak igralec ima svojo gibalno količino. Tudi žoga ima svojo gibalno količino, kar spremeni delovanje igre in malo spominja na zračni hokej. Masa, pospešek in najvišja hitrost določajo, kako hitro se lahko igralec obrne in začne potovati v nasprotno smer. Te lastnosti igralec tudi upošteva, ko se odloča za svoj naslednji cilj. Žoga zna pozicije žoge v prihodnosti izračunati glede na določen čas.

Poglavje 7

Zaključne ugotovitve

V našo igro smo implementirali delujoč, odziven in zabaven asinhronski večigralski način igre. Občutek tekmovalnosti s prijateljem je še zmeraj prisoten, čeprav se igra ne dogaja v realnem času. Odzivnost je sprejemljiva tudi v prostorih, kjer je povezava na omrežje šibka. Igra tako ostane zabavna, brez velikih časovnih zamikov in s tem povezanega nezadovoljstva, še vedno pa igramo s prijatelji. Opisan vmesnik za večigralski način na pregleden način prikazuje podatke o prijateljih. Glede na vse našteto, je cilj projekta dosežen.

Možnosti za izboljšave je sicer veliko. Pregled zmag in porazov med prijatelji še manjka. Manjka tudi prikaz končnega rezultata v seznamu posnetkov tekem. Te izboljšave so s postavljenim sistemom precej preproste za implementacijo. Komunikacijo med strežnikom in vmesnikom bomo v kasnejši verziji zakriptirali. Izboljšava, ki v času testiranja ni najbolj očitna, je vračanje zahtev in posnetkov. Trenutno se posnetki ne odstranjujejo in s časoma bi lahko bilo preveč posnetkov (odvisno od števila igralcev) in bi sistem postal neodziven. Potrebno bo dodati funkcijo, ki pregleduje posnetke v sistemu in bo, na primer, izbrisala vse posnetke, ki so starejši od dveh tednov.

Pokrili smo tudi veliko različnih tehnologij in konceptov, ki bodo morda prav prišli razvijalcem, ki želijo ustvariti nekaj podobnega. Opisali smo tudi možne težave in rešitve, ki smo jih srečali v tem delu. Tako se bodo lahko lažje odločili za eno ali drugo tehnologijo, ki jim bo omogočala globlji vpogled v delovanje takšnih sistemov.

Literatura

[1] Matt Buckland. *Programming Game AI by Example*. Wordware publishing inc., 2005.

[2] Dean Evans. *Shogun: Total War: Prima's Official Strategy Guide*. Premier Press, 2000.

[3] Eitan M. Glinert. *The Human Controller: Usability and Accessibility in Video Game Interfaces*. MIT, 2008.

[4] Steven L. Kent. *The Ultimate History of Video Games*. Random House LLC, 2010.

[5] Rok Logonder. Uporaba orodji za objektno-relacijsko preslikovanje pri razvoju javanskih aplikacij. Master's thesis, FRI, 2013.

[6] Peter Ludlow. *High Noon on the Electronic Frontier: Conceptual Issues in Cyberspace*, chapter 26. MIT, 1996.

[7] Martin Schumann Sandra Schon. Mobile gaming communities: State of the art analysis and business implications. *22nd Central European Conference on Information and Intelligent Systems*, 2011.

[8] Kenneth G. Paterson Sattam S. Al Riyami. *Certificateless Public Key Cryptography*. University of London, 2003.

[9] Jonas Vetterick Thomas Mundt. *Network Topology Analysis in the Cloud*. University of Rostock, 2011.

[10] Air warrior. Dostopno na:
http://massively.joystiq.com/2012/03/13/the-game-archaeologist-and-the-kesmai-legacy/, (Avgust) 2013.

[11] Almost half the world tuned in at home to watch 2010 fifa world cup south africa. Dostopno na:
http://www.fifa.com/worldcup/archive/southafrica2010/organisation/media/newsid=1473143/index.html, (Avgust) 2013.

[12] App store. Dostopno na:
http://ipod.about.com/od/iphonesoftwareterms/qt/apps-in-app-store.htm, (Avgust) 2013.

[13] Data store. Dostopno na:
https://developers.google.com/appengine/docs/java/datastore/, (Avgust) 2013.

[14] Empire. Dostopno na: http://www.classicempire.com/history.html, (Avgust) 2013.

[15] Farmville. Dostopno na:
http://www.omg-facts.com/Fun+Facts/At-The-Peak-Of-Its-Popularity-Farmville/57301, (Avgust) 2013.

[16] Fifa 13 / pes 2013. Dostopno na:
http://www.jeuxvideo.fr/jeux/fifa-13/fifa-2013-pes-2013-infographie-grand-match-actu-510103.html, (Avgust) 2013.

[17] Fifa 13 breaks records with 4.5 million sales in 5 days. Dostopno na:
http://metro.co.uk/2012/10/04/fifa-13-breaks-records-with-4-5-million-sales-in-5-days-598522/, (Avgust) 2013.

[18] Football manager 2013. Dostopno na:
http://www.eurogamer.net/articles/2013-05-10-aliens-colonial-marines-managed-1-31-million-sales, (Avgust) 2013.

[19] Gql. Dostopno na:
`https://developers.google.com/appengine/docs/python/datastore/gqlreference/`, (Avgust) 2013.

[20] Gson. Dostopno na: `https://code.google.com/p/google-gson/`, (Avgust) 2013.

[21] Haxball. Dostopno na: `http://www.haxball.com/about.html`, (Avgust) 2013.

[22] Metacritic football manager 2013. Dostopno na:
`http://www.metacritic.com/game/pc/football-manager-2013`, (Avgust) 2013.

[23] Real racing 3. Dostopno na:
`http://www.polygon.com/2013/2/15/3989990/real-racing-3-preview-time-shifted-multiplayer-tech`, (Avgust) 2013.

[24] Snake, classic phone game, turns 15. Dostopno na:
`http://www.webpronews.com/snake-phone-game-15-2012-02`, (Avgust) 2013.

[25] Top 10 best online football management games. Dostopno na:
`http://www.jumptogamer.com/top-10-best-online-football-management-games/`, (Avgust) 2013.

[26] Video game industry statistics. Dostopno na:
`http://www.esrb.org/about/video-game-industry-statistics.jsp`, (Avgust) 2013.

[27] World of warcraft. Dostopno na:
`http://www.telegraph.co.uk/technology/video-games/6081496/Video-Backstage-at-BlizzCon-2009.html`, (Avgust) 2013.

[28] X2 football 2010. Dostopno na:
http://www.pocketgamer.co.uk/r/iPhone/X2+Football+2010/review.asp?c=20706, (Avgust) 2013.