Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01661/2010
Datum: 05.04.2010

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

Kandidat: **MITJA DREU**

Naslov: **RAZVOJ IGRE SPACE PONG ZA IPHONE OS**
SPACE PONG GAME DEVELOPMENT FOR IPHONE OS

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

V diplomski nalogi najprej predstavite razvojno okolje za platformo iPhone OS, nato pa opišite cevovod razvoja iger. Po korakih cevovoda implementirajte lastno igro in jo objavite v trgovini aplikacij App Store.

Mentor:
doc. dr. Peter Peer

Dekan:
prof. dr. Franc Solina

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Mitja Dreu

Razvoj igre Space Pong za iPhone OS

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: doc. dr. Peter Peer

Ljubljana, 2010

Besedilo je bilo lektorirano s strani prof. Lidije Štrucl in oblikovano z urejevalnikom besedil LaTeX.

Namesto te strani **vstavite** original izdane teme diplomskega dela s podpisom mentorja in dekana ter žigom fakultete, ki ga diplomant dvigne v študentskem referatu, preden odda izdelek v vezavo!

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Mitja Dreu,

z vpisno številko 63040028,

sem avtor diplomskega dela z naslovom:

Razvoj igre Space Pong za iPhone OS

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
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 20. 6. 2010 Podpis avtorja:

Zahvala

Najprej bi se želel zahvaliti staršem, ki so me vzpodbujali in podpirali ne samo v času pisanja diplomske naloge, ampak v času celotnega študija na Fakulteti za računalništvo in informatiko. Ovir na poti do cilja ni bilo malo, niti niso bile enostavne. Brez vajine pomoči in podpore mi ne bi uspelo. Najlepša hvala.

Posebno zahvalo bi namenil mentorju doc. dr. Petru Peeru za vse predloge, nasvete in napotke. Z dodatnim gradivom ste mi pomagali pri razumevanju problema in uspešnem zaključku diplomskega dela.

Nenazadnje gre zahvala vsem prijateljem, sošolcem, ki ste me spremljali skozi študijska leta. Hvala, ker ste mi pomagali reševati probleme na fakulteti in zunaj nje.

Kazalo

Seznam uporabljenih kratic in simbolov

2D	Two Dimensional (2-dimenzionalen prostor)
3D	Three Dimensional (3-dimenzionalen prostor)
3G	3rd Generation (mišljen iPhone druge generacije)
3GS	3rd Generation Speed (mišljen iPhone tretje generacije)
CPU	Central Processing Unit (centralno procesna enota ali CPE)
CSS	Cascading Style Sheets (predloge, ki definirajo izgled spletnih strani)
EAGL	Embedded Apple Graphics Library (grafična knjižnica mobilnih naprav)
EDGE	Enhanced Data rates for Global Evolution (tehn. za prenos podatkov)
FTP	File Transfer Protocol (protokol za prenos podatkov)
GB	Gigabyte (oznaka za kapaciteto pomnilnika)
GCC	GNU Compiler Collection (zbirka prevajalnikov)
GHZ	Gigahertz (enota za frekvenco ure procesorja)
GNU	GNU's Not Unix (rekurzivni akronim)
GPS	Global Positioning System (sistem globalnega določanja položaja)
GUI	Graphical User Interface (grafični uporabniški vmesnik)
H.264/AVC	H.264 Advanced Video Coding (standard za kodiranje videa)
HTML	Hyper Text Markup Language (jezik za označevanje nadbesedila)
iOS	iPhone Operating System (operacijski sistem telefona iPhone)
LCD	Liquid Crystal Display (posebna vrsta zaslona)
MB	Megabyte (oznaka za kapaciteto pomnilnika)
Mb/s	Megabit per second (oznaka za prenos podatkov)
MHZ	Megahertz (enota za frekvenco ure procesorja)

MPEG-4	Moving Picture Experts Group 4 (standard za predvajanje filmov)
OS	Operating System (Operacijski Sistem)
PC	Personal Computer (Osebni računalnik)
SDK	Software Development Kit (orodje za razvoj programske opreme)
SMS	Short Message Service (storitev kratkih sporočil)
SSL	Secure Sockets Layer (varovan prenos podatkov)
UMTS	Universal Mobile Telecommunications System (tehn. za večji prenos pod.)
WAP	Wireless Application Protocol (tehn. za dostop do internetnih vsebin)
XML	eXtensible Markup Language (razširljiv označevalni jezik)

Povzetek

V nalogi obravnavamo področje razvoja iger za iPhone OS. V uvodnem poglavju je predstavljena potrebna strojna in programska oprema, kjer je posebni poudarek na uporabniškem vmesniku in arhitekturi operacijskega sistema iOS. Dodatno so opisana razvojna orodja in tehnologije, ki omogočajo razvoj aplikacij. Na koncu uvodnega dela so predstavljene omejitve manjših elektronskih naprav, ki jih mora razvijalec aplikacij upoštevati.

Osrednji del zajema proces razvoja iger. Opisali smo njegovo spreminjanje skozi čas, analizirali sestavne dele igre ter njihov cevovodni razvoj, ki predstavlja optimizacijo osnovnega procesa razvoja. Ob razvoju igre Space Pong smo uporabili osnovni proces, ki smo ga razdelili na predprodukcijsko in produkcijsko fazo ter fazo izdaje igre. Produkcijsko fazo smo razdelili še na posamezne faze in jih podrobno opisali.

Na koncu sledi še pogled na napredne grafične tehnologije in možnosti izboljšave razvite igre.

Ključne besede:

igra, razvoj, proces, grafični elementi, fizikalni sistem, umetna inteligenca, zaslon na dotik, iPhone, iOS

Abstract

The work describes game development for iPhone OS. In the introduction we look at needed software and hardware, especially at user intefrace and architecture of the operating system iOS. The introductionary chapter also describes important development tools and technologies which enable development of the applications. At the end of the chapter we present some limitations of small electronic devices with a major effect on the development.

The core of the work explaines the game development process. We describe the basic parts of a game, changes in its development process through the time and its pipeline development. The game pipeline development represents the optimization of the development process. In developed game Space Pong we used the basic development process, which was split into the preproduction, production and release phase. The production phase was split further into phases that are described in detail.

At the end we describe advanced graphics technologies and possibilities to make the developed game Space Pong even better.

Key words:

game, development, process, graphic elements, physical system, artificial intelligence, touchscreen, iPhone, iOS

Poglavje 1

Uvod

1.1 Motivacija in cilji

Mobilni telefon je elektronska telekomunikacijska naprava, ki se v omrežje povezuje s pomočjo oddajanja in sprejemanja radijskih valov. Komunicira z omrežjem baznih postaj, ki so povezane z običajnim telefonskim sistemom. V slabih štirih desetletjih razvoja mobilne telefonije so napravo, ki je sprva podpirala le zvočni pogovor, nadgradili s sporočili SMS (storitev, ki omogoča pošiljanje kratkih sporočil), shranjevanjem kontaktnih številk in brskanjem po prilagojenih straneh za internetne vsebine po protokolu WAP. Eno večjih sprememb je naznanil prihod pametnih telefonov (angl. smartphone), ki poleg osnovnih funkcij omogočajo namestitev bolj kompleksnih aplikacij. Čeprav je bil prvi pametni telefon predstavljen že leta 1992, je njihova prisotnost na svetovnem trgu sunkovito zrasla šele v zadnjih nekaj letih.

Trenutno smo v obdobju, ko pametni telefoni prevzemajo zmogljivosti osebnih računalnikov (angl. PC) izpred parih let. Pisanje elektronske pošte (angl. e-mail), kratkih dopisov, zapiskov, snemanje, obdelovanje videoposnetkov, zajem, oblikovanje fotografij in igranje iger, ne predstavlja nobenega pravega izziva za te naprave. Tako je razvijalcem strojne in programske opreme uspelo ustvariti telefon, ki nas spremlja pri vsakodnevnih opravilih in postaja vse bolj zmogljiv.

Ob sunkovitem razvoju strojne in programske opreme je podjetje Apple Inc.[1] 27. junija 2007 predstavilo pameten telefon z imenom iPhone, ki po karakteristikah strojne opreme ni predstavljal konkurence telefonom dugih podjetij. To, kar ga je poslalo v sam prodajni vrh, ni bila njegova zmogljivost,

[1] http://www.apple.com

ampak popolnoma drugo področje računalništva, uporabniški vmesnik. Gre za visoko kvaliteten zaslon z možnostjo dotika z večimi prsti hkrati (angl. multi-touch display). S tem so začrtali novo obdobje razvoja pametnih telefonov. Vse nadaljne poteze podjetja Apple Inc., spletna trgovina aplikacij (angl. App Store), poensotavljen razvoj aplikacij, prost dostop do orodij za razvoj (angl. software development kit ali SDK), so imele vpliv na znana podjetja v tej industriji, npr. Nokia[2], Sony Erricsson[3], Motorola[4].

Velik delež v Applovi spletni trgovini aplikacij (angl. App Store) predstavljajo igre. Igre, ki so v zadnjih 15. letih zaznamovale razvoj računalniške opreme za osebne namene, prav tako razvoj igralnih konzol. Predstavljajo trg, kjer velika podjetja, npr. Microsoft[5] iščejo svoje priložnosti. Govorimo torej o področju računalništva, ki podobno kot filmska industrija sprošča ob uporabi elektronskih naprav. Igre se razlikujejo po kompleksnosti, vsebini, ciljni platformi, kljub temu pa imajo vse razvojne faze nek skupen pristop. Srečamo se s pojmom cevovodni razvoj iger (angl. pipeline development), kjer s pomočjo vzporednega izvajanja faz optimiziramo proces razvoja iger.

Tako smo prišli do problematike, ki jo to diplomsko delo obravnava: predstavitev operacijskega sistema iPhone OS, ki omogoča delovanje telefonu iPhone in drugim multimedijskim napravam podjetja Apple (iPod Touch in iPad), predstaviti potek razvoja igre za iPhone OS od samega ogrodja dodajanja grafičnih elementov, dizajna logike igre, testiranja končne aplikacije in vse do objave igre. iPhone OS predvsem s poenostavljenim razvojem in preprostim uporabniškim vmesnikom, omogoča dobro izhodišče za vsakega razvijalca iger, naj bo začetnik ali veteran na tem področju. Applov sistem za trženje novih aplikacij pa omogoča dodatno prepoznavnost razvijalca na svetovnem spletu.

V sklopu naloge je bila izdelana igra *Space Pong* za iPhone OS.

1.2 Pregled vsebine

Prvo poglavje ima namen bralca seznaniti s širšim področjem mobilne telefonije in motivi za delo.

V drugem poglavju je predstavljen telefon iPhone in njegov OS, predvsem strojna in programska oprema, ter razvojno okolje xCode .

V tretjem poglavju je predstavljen proces razvoja igre. Od ogrodja do

[2]http://www.nokia.com
[3]http://www.sonyericsson.com/cws
[4]http://www.motorola.com
[5]http://www.microsoft.com

testne faze in kasneje do faze izdaje igre. Predstavljena je pomembnost same ideje igre, identifikacija ključnih elementov, posamezni koraki razvoja, grafični razvoj objektov s pomočjo orodja *Adobe Photoshop CS4*[6] in dodajanje zvoka.

V četrtem poglavju so predstavljene napredne grafične tehnologije iPhone OS, ki omogočajo grafično bogate 2D ali 3D aplikacije.

Peto poglavje predstavlja zaključek dela, kjer so strnjene ugotovitve in podane smernice nadaljnega dela.

Dodatek je namenjen podrobni predstavitvi realizirane igre *Space Pong*.

[6]http://www.adobe.com

Poglavje 2

iPhone OS in okolje xCode

2.1 Telefon iPhone

Mobilni telefon iPhone, ki spada v kategorijo pametnih telefonov, je leta 2007 oblikovalo in razvilo podjetje Apple Inc. Telefon ima uporabniški vmesnik, ki je v tistem obdobju revolucinoral uporabo mobilnih telefonov. Unikatna oblika, preprosta uporaba in enostaven razvoj novih aplikacij so lastnosti, ki so naredile telefon zelo uporaben. Osnovne funkcije, ki jih je omogočala prva generacija telefona, so bile možnost slikanja z integriranim digitalnim fotoaparatom, intuitivna aplikacija za pošiljanje kratkih sporočil, možnost predvajanja zvočnih in video datotek, pošiljanje elektronske pošte, brskanje po internetnih vsebinah in možnost povezave na brezžična omrežja.

Uporabniški vmesnik je bil zasnovan na osnovi visokokvalitetnega zaslona LCD, velikosti 320×480 točk, ki je omogočal sočasni dotik več prstov. Telefon je brez fizične tipkovnice, kar je bila za tisto obdobje novost. Tipkovnica je popolnoma navidezna in se pojavi samo v tistih aplikacijah, kjer je potrebna.

Spletna trgovina aplikacij App Store zajema v treh letih svojega delovanja že več kot 225.000 (7. junij 2010) aplikacij. Govorimo o vseh mogočih funkcionalnostih, od aplikacij senzorja GPS, iger, do aplikacij specializiranih za vse bolj popularna socialna omrežja, npr. *Facebook* [1]. Sistem omogoča posamezniku, ki je del neke razvojne skupine (lahko je sam), za plačilo letne naročnine, razvoj poljubnih aplikacij, ki jih lahko objavijo in tržijo v spletni trgovini aplikacij.

Kot omenjeno v prejšnjih odstavkih je izšla prva generacija telefona leta 2007 in je podpirala tehnologijo EDGE za prenos podatkov. Naslednje leto je

[1] http://www.facebook.com

izšla različica 3G, ki je za prenos podatkov uporabljala tehnologijo UMTS s hitrostjo 3.6 Mb/s. Dodatno je telefon vseboval čip GPS, ki je omogočal lociranje posameznika preko satelita. S pomočjo ustrezne programske opreme se lahko preračuna in grafično prikaže naša trenutna pot, po kateri se premikamo. Leta 2009 so predstavili različico 3GS. Model je zaznamovala izboljšana strojna oprema ter vgrajena močnejša digitalna kamera, z vgrajeno možnostjo snemanja videa.

2.1.1 Strojna oprema

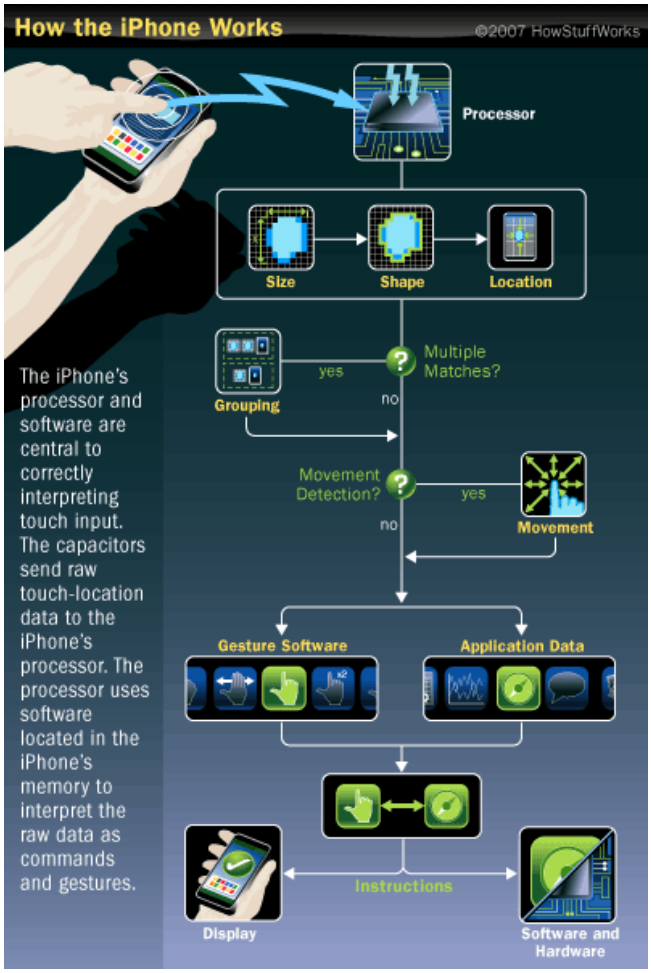
Zaslon in delovanje telefona iPhone

Steklen zaslon na dotik, ki hkrati predstavlja uporabniški vmesnik, po diagonali meri 9 cm in je sestavljen iz 320 $\times$ 480 točk [7]. Steklo je odporno na praske, ki jih lahko povzroči uporabnik ob normalni uporabi. Kapacitivni zaslon na dotik (angl. capacitive touchscreen) omogoča dotik z enim ali večimi prsti hkrati. Kapacitivni pomeni, da je zaslon sestavljen iz posebne plasti v obliki mreže. Ta plast vsebuje material, ki lahko ohranja električni naboj. Ob dotiku se identificira točka dotika na osnovi spremembe naboja. Plast, sestavljajo točke v obliki mreže. Vsaka točka ob dotiku (slika 2.1 [15]) generira svoj signal in ga posreduje procesorju telefona, kar omogoča detekcijo simultanega dotika prstov na različnih lokacijah. Procesor uporablja program s pomočjo katerega analizira dobljene podatke. Analiza poteka tako, da pri vsakem dotiku določi velikost, obliko (uporaba več prstov hkrati lahko tvori določene oblike) in pozicijo na zaslonu. Če je potrebno, se dotiki s podobnimi parametri grupirajo. Sledi uporaba posebnega progama, ki procesorju omogoča detekcijo uporabnikovega giba in aplikacije, ki je trenutno aktivna. Podatek, ki ga dobi o aplikaciji, je obogaten s stanjem aplikacije. Na osnovi teh podatkov procesor pošlje ukaze, ki so bodisi vidni na zaslonu, bodisi spremenijo delovanje naprave. Če je dotik neveljaven, ga ignorira.

Kapacitivni material omogoča detekcijo le ob dotiku roke (prenos energije) in ne drugih predmetov. Telefon ima vgrajene tri senzorje, ki so neposredno povezani z zaslonom.

Senzor za prilagajanje svetilnosti zaslona glede na svetlobo iz okolja (angl. ambient light sensor) je namenjen za povečanje zmogljivosti baterije.

Senzor za merjenje pospeška (angl. accelerometer) deluje na vse tri osi v prostoru (x , y in z os), kar omogoča zaznavanje položaja telefona (slika 2.2, vir: http://www.apple.com). Obstajata namreč dva načina uporabe, in sicer



Slika 2.1: Princip delovanja telefona iPhone ob dotiku

pokončna in ležeča. Oba načina izkoriščajo številne aplikacije, npr. internetni brskalnik, pregledovalnik elektronske pošte, pregledovalnik slik, in predvsem igre.

Senzor, ki zaznava prisotnost bližine predmetov brez fizičnega kontakta, pravijo mu tudi tipalo (angl. proximity sensor), se uporablja za izklop zaslona med klicem. Podobno tehnologijo uporabljamo pri avtomobilih, kjer s pomočjo parkirnih senzorjev avto z zvočno signalizacijo opozarja, da se nahajamo v neposredni bližini določene ovire. S tem se prepreči škoda na vozilu. Izklop zaslona je potreben za povečanje zmogljivosti baterije in preprečitev morebitnega kontakta obraza z zaslonom.

Dodatno bi omenil, da ima telefon vgrajene senzorje vlage (angl. mois-

Slika 2.2: Pokončen in ležeč način uporabe s senzorjem za merjenje pospeška

ture sensors), ki identificirajo prisotnost vode v napravi. Ti senzorji imajo velik pomen, kajti gre za indikator, ki pove, ali telefon deluje pravilno ali ne. Zaradi njihove izpostavljenosti (nahajajo se ob vhodu za slušalke in vhodu za polnjenje) je moč zaslediti določeno kritiko. Problem se pojavi, kadar telefon zazna prisotnost vode na napravi in ne v njej. Konkurenčni telefoni imajo te senzorje ob bateriji in ostali strojni opremi. Sprožijo se samo ob detekciji vode v notranjosti telefona.

Tehnične lastnosti

V tabeli 2.1 [2, 7, 15] so podane tehnične lastnosti telefonov iPhone od leta 2007 do 2009.

	iPhone 2G (prva generacija)	iPhone 3G (druga generacija)	iPhone 3GS (tretja generacija)
Procesor	Samsung[2] 32-bit ARM RISC[3] 412 MHZ	Samsung 32-bit ARM RISC 412 MHZ	Samsung 32-bit ARM RISC 600 MHZ
Pomnilnik RAM	128 MB eDRAM[4]	128 MB eDRAM	256 MB eDRAM
Flash Pomnilnik[5]	4, 8, 16 GB različice	8, 16 GB različice	16, 32 GB različice
Grafična kartica	PowerVR[6] MBX Lite 3D	PowerVR MBX Lite 3D	PowerVR SGX

Tabela 2.1: Okvirna predstavitev strojne opreme

24. junija 2010 je izšel telefon iPhone 4 (četrta generacija), s procesorjem Apple A4 ARM RISC 1 GHZ, pomnilnikom 512 MB eDRAM, flash pomnilnikom 16 GB ali 32 GB in prenovljenim zaslonom z resolucijo 960 × 640.

2.1.2 Programska oprema

Operacijski sistem

Operacijski sistem telefona iPhone (prav tako naprave iPad in iPod Touch) je znan pod imenom iPhone OS oziroma iOS, gre za mobilno različico bolj znanega opearcijskega sistema Mac OS X podjetja Apple Inc. [1]. Slednji se uporablja na prenosnih in osebnih računalnikih, osnova obeh je *Unix*[7].

V tabeli 2.1 sem omenil velikosti flash pomnilnika. Tukaj je potrebno upoštevati, da nekaj manj kot polovico GB zasede mobilni operacijski sistem iOS. Ta prostor je za uporabnika nedosegljiv. Operacijski sistem omgoča pravilno delovanje telefona, interno razvitih aplikacij v podjetju Apple Inc. in aplikacij posameznih razvijalcev širom sveta. Vse uradne aplikacije telefona iPhone se razvijajo s pomočjo orodij xCode na operacijskem sistemu Mac OS X. Razvitih aplikacij ni mogoče kopirati neposredno iz razvojnega okolja na napravo, ampak se je potrebno najprej registrirati na spletni strani http://developer.apple.com kot razvijalec in ustrezno pridobiti *Developer certificate*, ki je sicer pri Applu na voljo proti letnemu plačilu (enako velja za naprave iPad in iPod Touch) [2].

V prejšnjem odstavku sem uporabil besedno zvezo *uradne aplikacije*, razlog se skriva v tem, da je mogoče na telefon (tudi naprave iPad in iPod Touch) namestiti posebno obliko operacijskega sistema iOS, ki vsebuje aplikacijo z imenom *Cydia*. Postopek imenujemo *jailbreak* in s pomočjo *Cydie* omogoča prenos neuradnih aplikacij. Govorimo o aplikacijah, za katere Apple Inc. ne garantira pravilnega delovanja.

[2]http://www.samsung.com

[3]RISC je angleška kratica za *Redundand Instruction Set Computer* in predstavlja skupino mikroprocesorjev, ki imajo poenostavljen nabor ukazov.

[4]eDRAM je vrsta pomnilnika, integrirana v modul procesorja.

[5]Flash pomnilnik (angl. flash memory) je a vrsta pomnilnika, ki omogoča večkratno brisanje in zapisovanje brez dodatnega vezja.

[6]http://www.imgtec.com/powervr/powervr-graphics.asp

[7]UNIX je operacijski sistem razvit leta 1969. Imel je velik vpliv na razvoj drugih operacijskih sistemov 20. stoletja [9].

Ob priklopu telefona na osebni računalnik, se operacijski sistem iOS poveže s progrmaskim paketom *iTunes*. Upravljanje operacijskega sistema poteka preko računalnika. Sam paket nam omogoča prenos datotek (zvočnih, video), prenos aplikacij, prenos posodobitev in popravkov.

Pri novejših operacijskih sistemih, npr. Windows ali Mac OS X, srečamo situacijo, kjer kompleksne aplikacije zahtevajo veliko pomnilniškega prostora. Operacijski sistem deluje tako, da ugotovi, kateri bloki[8] pomnilnika niso bili uporabljeni že dlje časa. Ti bloki se prenesejo v izmenjevalno datoteko (angl. swap file) na trdem disku in dodatno sprostijo pomnilniški prostor aplikacijam, ki ga potrebujejo [4, 10, 14]. Ob zahtevi aplikacije po podatkih se ti prenesejo nazaj iz trdega diska v pomnilnik. Ta postopek imenujemo izmenjava blokov (angl. swapping). Govorimo o sposobnosti računalnika, ki omogoča naslavljanje večjega pomnilniškega prostora od dejansko prisotnega - koncept, ki je znan pod imenom navidezni pomnilnik. Operacijski sistem iOS ga žal ne pozna. Tako je razpoložljiv pomnilnik, ki ga ima aktivna aplikacija na voljo, omejen z neuporabljenim sistemskim pomnilnikom telefona (v tabeli 2.1 omenjen kot pomnilnik RAM).

Na tem mestu bi želel bralca opozoriti, da je opis v prejšnjem odstavku le groba predstavitev tega, kar se dejansko dogaja pri uporabi navideznega pomnilnika. Za več informacij mu svetujem pogled v predložene vire.

Arhitektura operacijskega sistema iOS

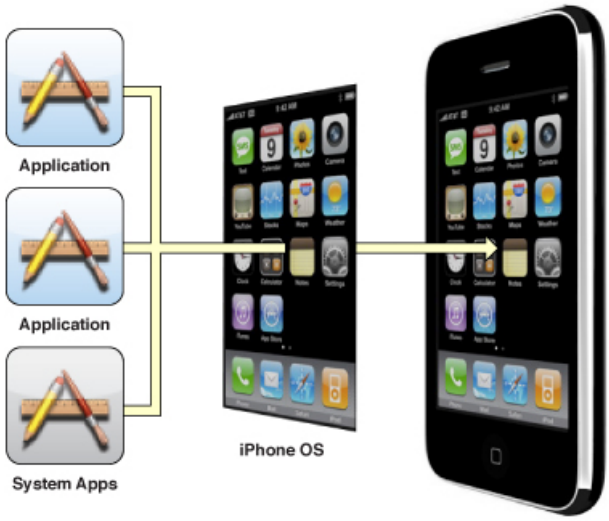
Poenostavljena oblika arhitekture operacijskega sistema Mac OS X je identična arhitekturi operacijskega sistema iOS. Razvijalci širom sveta lahko razvijajo aplikacije in jih ponujajo v spletni trgovini aplikacij (App Store). Te aplikacije, ki jih razlikujemo od sistemsko integriranih aplikacij, ne komunicirajo neposredno s strojno opremo, ampak preko določenih vmesnikov, povezanih z gonilniki strojne opreme. Ta pristop zaščiti aplikacijo pred morebitnimi spremembami strojne opreme.

Operacijski sitem iOS torej deluje kot posrednik med aplikacijami in strojno opremo (slika 2.3 [1]).

Zavedati se moramo, da operacijski sistem za pravilno delovanje telefona uporablja enostavno programsko opremo. Razlog za enostavnost se skriva v omejeni zmogljivosti telefonov v primerjavi s sodobnimi računalniki.

V prejšnjem odstavku omenjena programska oprema je organizirana v obliki sklada. Sklad v računalništvu predstavlja podatkovno strukturo, kjer so podatki shranjeni v neki hierarhiji, eden na drugem. Do njih dostopamo po

[8] Procesna slika nekega programa v pomnilniku je navadno sestavljena iz več blokov [14].



Slika 2.3: iOS kot posrednik med aplikacijami in strojno opremo

sistemu, zadnji noter prvi ven. Na dnu sklada se nahaja mikrojedro operacijskega sistema (angl. microkernel) z imenom *Mach* in gonilniki strojne opreme. Jedro upravlja izvajanje aplikacij, pomnilnik, niti, datotečni sistem, omrežje in komuniciranje med procesi. Na vrhu jedra se nahaja dodatna programska oprema, ki vsebuje osnovne tehnologije (angl. core technologies) in vmesnike za razvoj aplikacij. O osnovnih tehnologijah mikrojedra bomo več povedali v naslednjih poglavjih.

Uporabniški vmesnik

Uporabniški vmesnik je zasnovan na osnovi koncepta direktne manipulacije (angl. direct manipulation). Namen koncepta je, uporabnika prisiliti do gibov iz realnega sveta pri uporabi elektronskih naprav. Vsebina na zaslonu telefona je predstavljena tako, da uporabnik za dosego cilja instinktivno uporabi gib, brez dodatnega razmišljanja, kako bo to storil. To nas pripelje do nove stopnje interakcije človek računalnik, v tem primeru telefon. Zaradi enostavnega vmesnika, uporabnik naredi manj napak in opravi več opravil v krajšem času.

Vmesnik sestavljajo: gumbi (angl. buttons), drsniki (angl. sliders) in stikala (angl. switches). Odzivnost na uporabnikove zahteve je takojšnja, kar predstavlja tekoče delovanje, brez zatikanja. V prejšnjem poglavju omenjen zaslon na dotik zahteva od uporabnika gibanje prstov na različne načine. Prav tako omogoča naprava dva načina delovanja, pokončno in ležeče. Uporabnik

lahko telefon obrača na vse mogoče načine.

Ob pritisku na gumb domov (angl. home button), ki je mimogrede edini fizično obstoječi gumb na napravi, se prikaže namizje. Namizje operacijskega sistema upravlja aplikacija z imenom *SpringBoard*, ki omogoča prikaz ikon vseh na telefonu (tudi druge naprave iPad in iPod Touch) nameščenih aplikacij, ki jih lahko poljubno premikamo, nekatere tudi brišemo. Na vrhu zaslona lahko opazimo vrstico stanja, ki prikazuje uro, signal omrežja in stanje baterije. Koncept odpiranja in zapiranja aplikacij, kot ga poznamo iz drugih operacijskih sistemov, je tukaj neznanka. Aplikacijo odpremo tako, da se s prstom dotaknemo ikone, za izhod pa uporabimo prej omenjeni gumb domov.

Ena izmed pomanjkljivosti operacijskega sistema je večopravilnost (iOS verzije 1 do 3.1.3 ne podpirajo). Večopravilnost pomeni, da uporabnik lahko preklaplja med različnimi aplikacijami, ne da bi izgubil trenutno delo. V najnovejši različici iOS verzija 4.0 je ta funkcija implementirana.

2.2 Orodje za razvoj aplikacij

2.2.1 Osnove

Vmesniki, orodja in viri za razvoj aplikacij na računalnikih *Macintosh* podjetja Apple Inc., so vsebovani v programskem paketu za razvoj aplikacij (angl. software development kit ali SDK). Računalniki morajo biti opremeljeni s procesorji podjetja Intel[9](kar nas pripelje do zaključka, da bi aplikacije bilo teoretično možno razvijati tudi na računalnikih drugih proizvajalcev, npr. PC).

Večina sistemskih vmesnikov je predstavljena v obliki ogrodja (angl. framework), ki vsebuje knjižnice in njihove vire (slike, podpora in (angl. header) datoteke[10]). Ogrodja in njihove značilnosti lahko uporabljamo tako, da jih povežemo z obstoječim projektom aplikacije. S tem omogočimo dostop do header datotek, slik ter ostalih lastnosti, hkrati pa orodju za razvoj aplikacij povemo, kje se nahajajo.

Dodatno ponuja programski paket tehnologijo v obliki posebne vrste knjižnic. Zavedati se moramo, da je osnova operacijskega sistema iOS Unix. Mnogo tehnologij, ki tvori spodnjo plast operacijskega sistema (jedro in gonilniki) je odprtokodnih[11]. Knjižnice vsebujejo vmesnike, s katerimi lahko dostopamo do teh tehnologij.

[9]http://www.intel.com

[10]Predstavljajo posebne datoteke, ki vsebujejo deklaracijo metod, lastnosti itd.

[11]Prost dostop do programske kode.

Ostale pomembne komponente programskega paketa: razvojni paket xCode, iPhone Simulator in razvojna dokumentacija (angl. iPhone Reference Library).

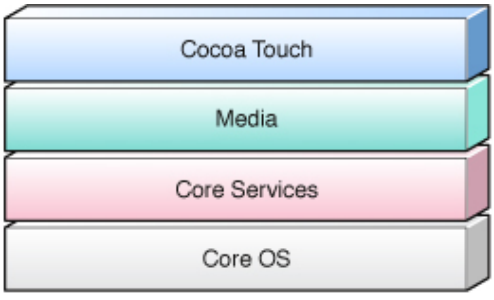
Razvojni paket xCode s pomočjo simulatorja omogočajo razvoj in testiranje aplikacij v razvojnem okolju na računalniku [2]. Ob vključitvi v razvojni progam (angl. iPhone developer program) podjetja Apple Inc. nam omogoča dodatno testiranje aplikacij neposredno na napravi. Ta program je brezplačen le za univerze.

Razvijalci imajo možnost razvoja dveh vrst aplikacij, spletne in standardne [2]. Spletne aplikacije uporabljajo kombinacijo jezika HTML, Java skript in posebne oblike opisnega jezika za določanje lastnosti objektom jezika HTML (angl. cascading style sheets ali CSS). Aplikacije shranimo na spletni strežnik. Zagon je možen samo preko spletnega brskalnika telefona (prav tako naprave iPad in iPod Touch) z imenom *Safari* in povezave na internet. Standardne aplikacije na drugi strani shranjujemo na telefon in omogočajo neodvisen zagon.

Opozoriti je treba, da razvijalci z orodji xCode ne morejo razvijati gonilnikov ali aplikacij, ki tečejo v ozadju. Obstajajo sicer ukazi, ki omogočajo standardni aplikaciji izvajanje v ozadju, vendar na omejen način.

2.2.2 Tehnologije iOS

Tehnologije operacijskega sistema iOS si lahko predstavljamo kot množico plasti (slika 2.4 [1]). Na nižjih plasteh govorimo o osnovnih storitvah, od katerih so vse aplikacije odvisne, na višjih plasteh srečamo bolj kompleksne storitve in tehnologije. Dobro je vedeti, da nam te tehnologije pri razvoju aplikacij omogočajo dodatne možnosti.



Slika 2.4: Plasti tehnologij iOS

Plast Cocoa Touch

Najpomembnejšo plast operacijskega sistema predstavlja *Cocoa Touch*. Obsega ključna ogrodja za zagotavitev infrastrukture pri razvoju aplikacij. Vsak razvijalec začne na tej plasti in se premika navzdol po potrebi. V naslednjih razdelkih bom na kratko opisal sestavne dele plasti *Cocoa Touch*.

- Storitev pošiljanja opozoril (angl. push notification service). Storitev omogoča uporabnikom pošiljanje informacij o aplikaciji, kljub temu, da aplikacija ni aktivna. Informacije lahko zajemajo tekstovna sporočila ali opozorila. Služijo kot pomoč tako razvijalcem kot uporabnikom. Storitev opozarja uporabnika, da je prišlo do morebitne spremembe aplikacije.

- Ogrodje kontaktnih podatkov (angl. adress book UI framework). Govorimo o ogrodju, ki predstavlja vmesnike za dostop in urejanje kontaktnih podatkov. Če razvijalec želi v svoji aplikaciji uporabiti kontaktne podatke, mu to ogrodje delo poenostavi. Dodatno omogoča, da vse aplikacije uporabljajo enake vmesnike, kar pripelje do usklajenosti med aplikacijami.

- Elektronska pošta (angl. E-mail). Ogrodje, ki omogoča pregled in kreiranje nove pošte. Razvijalec lahko uporabi poseben vmesnik, ki omogoča pisanje in odpošiljanje elektronske pošte v njegovi aplikaciji.

- Ogrodje zemljepisnih kart (angl. map kit interface). Ogrodje, ki razvijalcem omogoča vgradnjo zemljepisne karte v aplikacijo.

- Podpora konceptu vsak z vsakim (angl. peer to peer support). Ogrodje, ki s pomočjo tehnologije Bluetooth omogoča komunikacijo z drugimi napravami v bližini. Namenjeno je predvsem razvijalcem iger (igre, ki ponujajo možnost igranja več igralcev naenkrat), čeprav se lahko implementira v druge vrste aplikacij.

Ostali vmesniki so namenjeni implementaciji grafičnih, dogodkovno vodenih aplikacij, npr. možnosti upravljanja aplikacij; kopiranju, rezanju in lepljenju teksta (angl. copy, cut, paste); grafični podpori; podpori dotikov zaslona; stanju baterije itd.

Bralca bi opozoril, da je bila večina storitev in ogrodij implementiranih v verziji 3.0 operaciijskega sistema iOS.

Multimedijska plast

Ta plast predstavlja zlitje video, grafičnih in zvočnih tehnologij s ciljem ustvariti najboljšo možno multimedijsko izkušnjo na telefonu (prav tako na naprave iPad in iPod Touch). Tehnologije so bile razvite z namenom pomagati razvijalcem pri razvoju grafično zahtevnih aplikacij.

Za specifične potrebe razvijalcev je bilo razvitih kar nekaj grafičnih tehnologij, ki jih bomo na kratko opisali.

- Quartz 2D je napreden grafični pogon, ki je specializiran za risanje v dveh dimenzijah. Popolnoma enak pogon uporablja operacijski sistem Mac OS X. Podpira delo s slikami, barvami, 2D transformacijami, kreiranje datotek PDF itd. Ta knjižnica je bila uporabljena v igri *Space Pong*, ki je bila razvita.

- Tehnologija animiranja (angl. core animation) se uporablja pri kompleksnih animacijah in grafičnih efektih.

- OpenGL ES je ogrodje, ki se uporablja pri risanju v dveh ali treh dimenzijah. Napisano je v programskem jeziku C in intenzivno komunicira s strojno opremo telefona. Ogrodje se uporablja v povezavi z vmesniki EAGL, ki omogočajo povezavo med napisano kodo in objekti v aplikaciji.

- Zvočne tehnologije (angl. Audio Technologies) so namenjene predvajanju in snemanju zvoka na telefonih . Obstaja široka paleta uporabljenih tehnologij [1].

- Video tehnologije (angl. Video Technologies) so namenjene predvajanju videa na celotnem zaslonu. Predvajajo lahko filme različnih formatov : mov, mp4, m4v in 3gp. Podprta kompresijska standarda sta H.264/AVC in MPEG-4.

Plast osnovnih storitev (angl. core services)

Osnovne storitve, ki jih uporabljajo prav vse aplikacije, so zajete v tej plasti. Sem spadajo tako znana ogrodja (npr. ogrodje kontaktnih podatkov) kot popolnoma nova ogrodja, ki dodatno s pomočjo razvojnega okolja xCode (xCode ločimo kot razvojni paket in razvojno okolje) omogoča risanje shem podatkovnih modelov. Za najboljšo zmogljivost podjetje Apple Inc. priporoča uporabo podatkovne baze SQLite.

Dodatno najdemo tukaj ogrodja za uporavljanje s podatkovnimi bazami, ogrodja za delo s teksti, datumi, nitmi, ogrodja za identificiranje geografske

pozicije naprave. Primer aplikacije, ki tehnologijo izkorišča, je iskanje restavracij, muzejev, knjižnic v okolici 100 metrov glede na pozicijo telefona.

Podpira tudi XML[12].

Plast jedra operacijskega sistem (angl. core OS layer)

Omogoča ogrodja za delo z internetnimi protokoli, npr. SSL, HTTP, FTP, ogrodja za dostop do zunanjih naprav, ogrodja za varnost podatkov, ki jih ustvari aplikacija (uporaba certifikatov, javnih in privatnih ključev).

Zaradi varnostnih razlogov je dostop do sistemskega ogrodja (jedro sistema) omejen. Kljub temu imajo razvijalci možnost uporabe številnih vmesnikov, s katerimi dostopajo do niti, datotečnega sistema in alociranja pomnilnika.

2.2.3 Razvojni paket xCode

V prejšnjem poglavju smo omenili, da so ostale pomembne komponente programskega paketa za razvoj aplikacij (SDK): razvojni paket xCode, iPhone Simulator in razvojna dokumentacija (angl. iPhone Reference Library). Razvojni paket xCode predstavljajo osnovna orodja za uspešen razvoj aplikacij na napravi. Sestavljajo ga trije osnovni programi: razvojno okolje xCode, okolje za razvoj uporabniškega vmesnika (angl. Interface Builder) in dodatno orodje za analiziranje in razhroščevanje razvitih aplikacij, znano pod angleškim imenom Instruments.

Razvojno okolje xCode

Integrirano razvojno okolje (angl. Integrated Development Environment) predstavlja glavno komponento razvojnega paketa xCode. Omogoča spreminjanje, prevajanje, razhroščevanje in izvajanje kode, zapisane v določenem programskem jeziku.

Uporablja spremenjeno prevajalniško zbirko GNU (angl. GNU Compiler Collection ali kar GCC)[13], ki podpira programske jezike C, C++, Objekten-C (angl. Objective-C) [14], Java (za iPhone še ni podpore, obstajajo programi, ki prevedejo kodo v ustrezen, podprt jezik), Apple skripte (angl. AppleScript) in Ruby [1]. Za razvoj aplikacij na operacijskem sistemu Mac OS X je bila dodatno implementirana (s strani razvijalcev) podpora jezikom C#, Ada, Free Pascal, Perl in D.

[12]Označevalni jezik, ki je podoben jeziku HTML.

[13]Množica prosto dostopnih programskih prevajalnikov, plod projekta GNU.

[14]Različica programskega jezika C s poudarkom na objektih.

Okolje omogoča razvoj tako aplikacij za operacijski sistem iOS, kot za operacijski sistem Mac OS X.

Okolje za razvoj uporabniškega vmesnika (angl. Interface Builder)

Predstavlja del razvojnega paketa xCode. S pomočjo vmesnika GUI[15] lahko razvijalci oblikujejo grafično podobo svojih aplikacij.

Razvoj aplikacij torej poteka v dveh različnih okoljih. V tem razvojnem okolju ustvarimo grafično podobo aplikacije, jo shranimo v datoteko s končnico nib in nadaljujemo delo v razvojnem okolju xCode. Prej ustvarjenim objektom dodamo v novem okolju funkcionalnosti s pomočjo programske kode [4].

Razvijalec ima na voljo veliko različnih objektov [1, 4], ki jih lahko uporabi v svoji aplikaciji. Število objektov ni omejeno, kar omogoča dodajo novih.

Dodajanje objekta na podlago aplikacije, ki ji pravimo okno (angl. window), deluje po principu primi in spusti (angl. drag and drop). Na enak način lahko vsak dodan objekt povežemo z objektom, ustvarjenim v kodi. S tem ustvarimo referenco, ki nam omogoča spreminjanje grafične podobe objektov in dodajanje novih funkcionalnosti. Na ta način se objekti inicializirajo pred dejanskim zagonom aplikacije.

2.2.4 Omejitve pri razvoju aplikacij

Kot smo omenili v enem izmed prejšnjih poglavij, je zmogljivost telefona po tehničnih lastnostih v primerjavi z modernimi osebnimi računalniki (PC) omejena. Kot razvijalec aplikacij je potrebno upoštevati določena dejstva [10], ki so specifična za telefon iPhone in se razlikujejo od drugih telefonov:

Samo ena trenutno delujoča aplikacija. Ena izmed tipičnih lastnosti in hkrati kritik operacijskega sistema iOS je, da ne omogoča delovanje več aplikacij hkrati. V času pisanja diplome je podjetje Apple Inc. predstavilo operacijski sistem iOS verzije 4.0, ki bo to funkcionalnost omogočal.

Samo eno okno. Za razliko od drugih operacijskih sistemov, npr. Microsoft Windows, lahko razvijalec pri svojih aplikacijah uporablja le eno okno. Njegova velikost je enaka velikosti zaslona na dotik. Tukaj bi omenil, da je omgočen sistem pogledov (angl. views), ki simulira več oken. Omejitev pri tem pristopu je, da okna ne morejo biti odprta hkrati.

[15] Grafični uporabniški vmesnik.

Omejen dostop. Pri programih osebnih računalnikov, kjer ima razvijalec možnost dostopa do velikega števila sistemskih klicev in datotek, smo pri telefonu iPhone omejeni. Beremo in spreminjamo lahko samo datoteke, ki se nahajajo v posebej za aplikacijo namenjenemu delu datotečnega sistema z imenom peskovnik (angl. sandbox).

Omejen odzivni čas. Pravilo pravi, da se naj prikaz glavnega okna izvede v najkrajšem možnem času. Uporabnik ne sme dobiti občutka, da je z aplikacijo nekaj narobe. Gumb domov (angl. home button) je zasnovana tako, da lahko prekine izvajanje aplikacije kadarkoli. Razvijalec (odvisno od konteksta aplikacije) mora poskrbeti za ustrezno shranjevanje podatkov.

Omejena velikost zaslona. 320×480 točk predstavlja majhno delovno površino v primerjavi s sodobnimi zasloni osebnih računalnikov, ki zmorejo 1680×1050 točk in več.

Omejena strojna oprema. Razvoj aplikacij na telefonu iPhone, s 128 MB ali 256 MB pomnilnika (4. generacija tudi 512 MB pomnilnika), ne moremo primerjati z razvojem zahtevnih aplikacij na napravi z več GB pomnilnika. Grafično kompleksne aplikacije, npr. igre, zasedejo količino razpoložljivega pomnilnika zelo hitro. Razvijalci morajo v procesu razvoja igre to upoštevati in uvesti dodatne postopke optimizacije. Na tem mestu je potrebno omeniti, da aplikacije za delovanje nimajo razpoložljivega celotnega pomnilnika, ampak le del tega. Nerazpoložljiv del zasedajo procesi operacijskega sistema. Dodatno bi opozoril na pomanjkljivost, da operacijski sistem iOS ne pozna niti koncepta strani, niti blokov [14].

Manjkajoč pobiralec smeti (angl. garbage collector). Pobiralec smeti je oblika upravljanja pomnilnika. Aplikaciji sprošča pomnilnik tako, da išče neuporabljene objekte, ki zasedajo prostor v pomnilniku. Tehnologija operacijskega sistema iOS ne podpira pobiralca smeti in dodatno delo prepušča razvijalcu. Vsak objekt, ki ga naredi, mora na ustreznem mestu sprostiti.

2.2.5 Zaključek

Pogledali smo si strojno in programsko opremo telefona iPhone (prav tako se navezuje na naprave iPod Touch in iPad, ki je po strojni opremi primerljiv z iPhonom 4. generacije). Spoznali osnove delovanja, tehnične lastnosti telefona skozi čas (4 generacije), delovanje in arhitekturo operacijskega sistema

ter uporabniški vmesnik. V drugem delu poglavja smo predstavili orodja za razvoj aplikacij, tehnologije, ki sestavljajo operacijski sistem in programe, ki tvorijo razvojno okolje. Na koncu smo omenili nekaj omejitev, s katerimi se srečujejo razvijalci aplikacij.

Poglavje 3

Proces razvoja iger

3.1 Uvod

V prejšnjem poglavju smo spoznali osnovne elemente, potrebne za uspešen razvoj aplikacij, npr iger za iPhone OS. Pogledali smo si tako strojno in programsko opremo naprav kot razvojno okolje in omejitve pri razvoju.

V tem poglavju bomo analizirali proces razvoja iger, ga opisali in razdelali na posamezne stopnje, ter povezali s pojmom cevovod razvoja iger (angl. game development pipeline).

Na začetku se moramo vprašati, kako definiramo razvoj igre. Razvoj igre definiramo kot proces razvoja posebne vrste programske opreme, ki jo s skupnim imenom imenujemo igra. Stopnja kompleksnosti igre določa stopnjo kompleksnoti procesa razvoja. Tako lahko preprosto zastavljena igra zahteva le enega do dva razvijalca, na drugi strani pa srečujemo podjetja, ki imajo 500 ali več zaposlenih, tako programerjev kot grafičnih oblikovalcev. Vsak je strokovnjak na svojem področju in doda le majhen kos končnemu izdelku.

Začetki razvoja iger segajo v leto 1952, ko je Alexander S. Douglas z univerze Cambridge predstavil prvo delujočo igro na digitalnem zaslonu OXO, bolje poznano pod imenom križci in krogci (angl. Noughts and Crosses). Sledile so druge, vendar v večjih časovnih razmikih. V obdobju od leta 1970 do 1980 [13], ob začetku prihoda osebnih računalnikov, so razvijalci prvih znanih iger (*Zork*, *Air Warrior* in *Adventure*) začeli tržiti svoje produkte. Tako se je začelo obdobje industrije iger. Vsak, ki je imel osnovno znanje iz programiranja, idejo in preveč časa, je razvijal igre. Posledica takšnega načina razvoja, kazanje prednosti količini izdanih iger in ne kvaliteti (angl. quantity over qual-

ity), je pripeljala do padca industrije v Združenih državah Amerike in uspeha na Japonskem, zlasti pri podjetju Nintendo[1]. V 90. letih 20. stoletja so industrijo iger izboljšale nove tehnologije operacijskih sistemov, še posebej vmesnik GUI.

Razvoj grafičnih kartic, kot samostojnih enot osebnih računalnikov, v povezavi z novimi tehnologijami predstavlja revolucijo izgleda in igralnosti iger. Danes predstavljajo ločeni grafični procesorji standard skoraj vseh naprav z grafičnim uporabniškim vmesnikom. Telefon iPhone ni nobena izjema.

3.2 Razvoj iger danes

Časi, ko sta eden ali dva razvijalca dokončala projekt izdelave računalniške igre, so mimo. Zaradi možnosti izvajanja iger na različnih napravah, npr. igralnih konzolah, osebnih računalnikih in telefonih, obstajajo izjeme, kjer lahko posamezen uporabnik z dobro zasnovano idejo prevzame pobudo in razvije manj kompleksno igro za telefon, npr. iPhone.

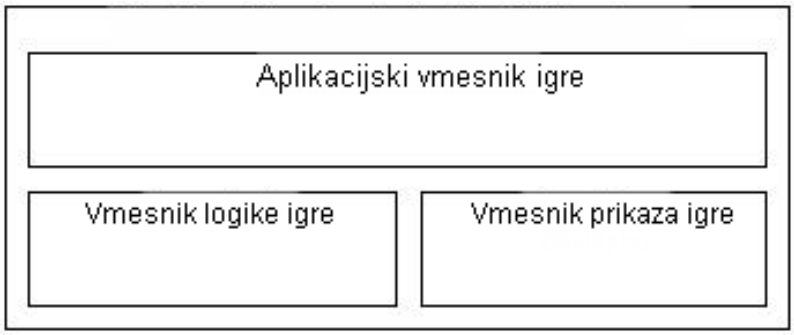
Razvoj iger 21. stoletja zahteva ekipe programerjev in grafičnih oblikovalcev, ki poskušajo razviti grafično bogato in za uporabnika zanimivo igro [13]. Projekti se lahko zavlečejo na obdobje več let. Za razvoj tržno uspešne igre so potrebni nadarjeni razvijalci, strokovnjaki na svojem področju, grafični oblikovalci, kreativnost, dober način, kako napraviti igro prepoznavno (trženje) in sreča.

Trenutno se nahajamo v obdobju, kjer grafična podoba igre ne predstavlja ene izmed najpomebnejših točk razvoja. Še vedno je pomembno, da so igre na visokem grafičnem nivoju, vendar tisto, kar dejansko spremeni način razvoja je uporabniški vmesnik. Kako uporabnika dodatno vključiti, integrirati v igro? Govorimo o poskusu zavračanja standardnih pripomočkov pri igranju iger, npr. tipkovnica in miška ali igralni plošček (angl. gamepad). Pionir na tem področju je bila leta 2006 razvita igralna konzola Nintendo Wii [2], ki uporabnika prisili v fizično sodelovanje ob igranju igre. Tako npr. ob igranju tenisa uporabnik posnema gibe iz tenisa, namesto loparja drži posebno obliko igralnega ploščka, ki reagira na gibanje v treh dimenzijah. Prav tako lahko omenim telefon iPhone, ki s svojim inovativnim zaslonom na dotik, pospeškometrom in možnostjo uporabe telefona v različnih načinih delovanja (ležeč in pokončen), uporabniku ponuja nov način interakcije z igro.

[1] http://www.nintendo.com
[2] http://www.nintendo.com/wii

3.3 Sestavni deli igre

Preden začnemo analizirati proces razvoja iger, je dobro poznati zgradbo arhitekture igre. Tukaj ni pravila, vsak razvijalec ali skupina razvijalcev ima svoj model realizacije. Kljub različnim modelom pa se pri vsaki igri srečujemo s tremi primarnimi kategorijami (slika 3.1 [12]): aplikacijski vmesnik igre (angl. game application layer), vmesnik logike igre (angl. game logic layer) in vmesnik prikaza igre (angl. game view layer). Aplikacijski vmesnik skrbi za komunikacijo z operacijskim sistemom in strojno opremo (v našem primeru operacijskim sistemom iOS in telefonom iPhone), vmesnik logike igre upravlja stanja in se odziva na spremembe stanj v igri, vmesnik prikaza igre pa predstavlja grafiko in zvok igre.



Slika 3.1: Tri primarne kategorije igre

3.3.1 Aplikacijski vmesnik igre

Aplikacijski vmesnik igre delimo naprej na naprave, operacijski sistem in življenjsko dobo iger.

Naprave predstavljajo povezavo med uporabnikom in igro. Poznamo tipkovnice, miške, igralne ploščke in že prej omenjen zaslon na dotik telefona iPhone. V skupino naprav uvrščamo dodatno še datotečni sistem, medpomnjenje grafičnih elementov (pri grafično kompleksnih igrah mora razvijalec razviti sistem ustreznega prikazovanja oddaljenih in bližnjih objektov) in upravljanje pomnilnika.

Operacijski sistem predstavlja okolje, kjer bo igra delovala. Za razvijalca je pomembno, da pozna glavne značilnosti arhitekture, npr. podpora izmenjavi strani ali blokov (v kontekstu navideznega pomnilnika), podpora nitim, eno ali več-jedrnim procesorjem (angl. multi-core CPU), podpora omrežni povezavi

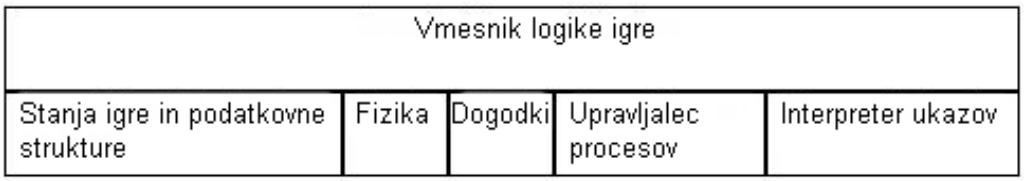
itd. Bistvene značilnosti operacijskega sistema iOS, ki predstavlja okolje v okviru diplomskega dela razvite igre *Space Pong*, so opisane v prejšnjem poglavju.

Življenjska doba igre predstavlja njeno inicializacijo, glavno zanko (vsaka igra deluje v neskončni zanki in čaka na uporabnikov ukaz) in možnost zaključitve. Za razliko od različnih programov, ki brez uporabnikove interakcije sprostijo večji del pomnilnika in zasedejo le tisti del, ki je potreben za delovanje, je pri igrah malo drugače. Igre so simulacije, ki imajo svoj življenjski cikel. Če uporabnik ne naredi ničesar, ga določeni dogodki v igri prisilijo k interakciji.

Na najvišjem nivoju aplikacijski vmesnik igre ustvari in naloži vmesnik logike igre, ki ga poveže z vmesnikom prikaza igre (slika 3.1 [12]).

3.3.2 Vmesnik logike igre

Vmesnik logike igre predstavlja dušo in srce igre, jo definira, predstavi vse njene objekte in njihovo povezanost. Ob uporabnikovi interakciji ali sprožitvi procesa umetne inteligence vmesnik opiše spremembe njenih stanj. Delimo ga na stanja igre in podatkovne strukture, uporabljeno fiziko (angl. physics)[3], dogodke, upravljalec procesov in interpreter ukazov (angl. command interpreter) (slika 3.2 [12]).



Slika 3.2: Delitev vmesnika logike igre

Ena izmed prvih odločitev razvijalcev je izbira ustrezne podatkovne strukture za shranjevanje objektov igre. Pri enostavnih igrah zadostuje struktura v obliki seznama, pri kompleksnih so potrebni drugi pristopi. Obstajati mora možnost hitrega iskanja in spremembe stanja objektov. Zelo uspešna igra *Ultima* je za shranjevanje objektov uporabljala 2D tabelo. Za lažje razumevanje lahko podam kot primer sistem točk, ki sestavljajo neko površino v 3D prostoru. Vsaka točka je objekt s svojimi značilnostmi, npr. koordinate (x, y, z), opis itd. Lastnosti objektov, npr. moč motorja avtomobila, število prestav motorja avtomobila, točke poškodb na avtomobilu (avtomobil predstavlja objekt) se ponavadi shranjujejo na posebej za te namene razvitih podatkovnih strukturah. *Ultima Online* je shranjevala lastnosti objektov v obliki teksta,

igra *Thief: Deadly Shadows* pa je uporabljala zapleten sistem, ki je poleg spreminjanja obstoječih lastnosti objektov omogočal dodajanje novih lastnosti [12].

Fizika spada pod kategorijo pravila igre. S pomočjo fizikalnih formul lahko simuliramo trke, padce in druge interakcije med objekti. Več o tem v naslednjem podpoglavju.

Razlogi za dinamičnost neke igre se skrivajo v spremembi njenih stanj. Samo kreiranje novih objektov (nek dogodek) sproži aktivacijo določenih sistemov igre. Tako na primer prikaz avtomobila v igri aktivira grafični sistem, ki naloži 3D objekt, kreiran v kakšnem profesionalnem okolju in prilepi nanj teksture. Ob vožnji zvočni sistem igre poskrbi za zvok avtomobila. Če dodamo procese umetne inteligence, lahko simuliramo druge voznike, ki se odzovejo na naše vozilo. Zavedati se moramo, da te sisteme igre obveščajo dogodki, ki predstavljajo sestavni del vmesnika logike igre.

Upravljalca procesov uporabljajo bolj kompleksne igre. V bistvu govorimo o večprocesnem sistemu, kjer ima vsak proces namenjen nek določen čas izvedbe. Primer bi lahko bila ročna granata, uporabljena v vojnih igrah. Določen proces lahko upravlja katerokoli gibanje objektov pred trkom, ob istem času drug proces čaka na izvedbo simulacije eksplozije.

Interpreter ukazov omogoča odziv igre na igralčeve ukaze. V primeru prvoosebnih iger (angl. first person shooter) lahko določena tipka na tipkovnici predstavlja premik naprej, druga tipka premik nazaj itd.

3.3.3 Vmesnik prikaza igre

Vmesnik prikaza igre komunicira z vmesnikom logike igre (slika 3.1 [12]), s pomočjo katerega omogoča prikaz igre uporabniku. Njegove glavne naloge so odzivi na dogodke v igri, prikaz na zaslonu, predvajanje zvočnih datotek in ustrezno predstavitev uporabniškega vmesnika igre.

Posebej bi izpostavil prikaz igre na zaslonu, ki omogoča prikaz objektov v igri (izrisovanje odvisno od zvrsti igre, v 3D okoljih se osredotočimo na izrisovanje objektov, ki so v videnm polju igralca) in ustrezno predstavitev uporabniškega vmesnika igre, ki predstavlja način kako igralec uporablja igro (npr. izgled menija igre, navodilo za igranje, izhod iz igre).

[3]Igre lahko zajemajo fizikalne formule za simulacije dogodkov iz realnega sveta.

3.4 Posamezne stopnje procesa razvoja igre

3.4.1 Cevovodni razvoj igre

V prejšnjih podpoglavjih smo spoznali, da se igre razlikujejo po kompleksnosti. Stopnje procesa razvoja igre narekuje prav kompleksnost igre.

Pri razvoju zahtevnih iger, ki se odvijajo v 3D okolju in uporabljajo svoj grafični fizikalni pogon ter poseben sistem za umetno inteligenco, je število stopenj procesa razvoja veliko. Znanje ljudi, ki sodelujejo na teh projektih, je zelo različno. Tako imamo na eni strani programerje, ki so tehnično zelo podkovani, na drugi strani grafične oblikovalce, katerih primarna naloga je razvoj grafičnih modelov. Programerji razvijejo okolje, namenjeno grafičnim oblikovalcem, ki razumejo delovanje sistemov le na najvišjem nivoju. Razvita programska okolja morajo biti oblikovalcu prijazna in tukaj se velikokrat zatakne. Potrebno je torej skrbno planiranje procesa in posvetovanje različnih ljudi na različnih področjih.

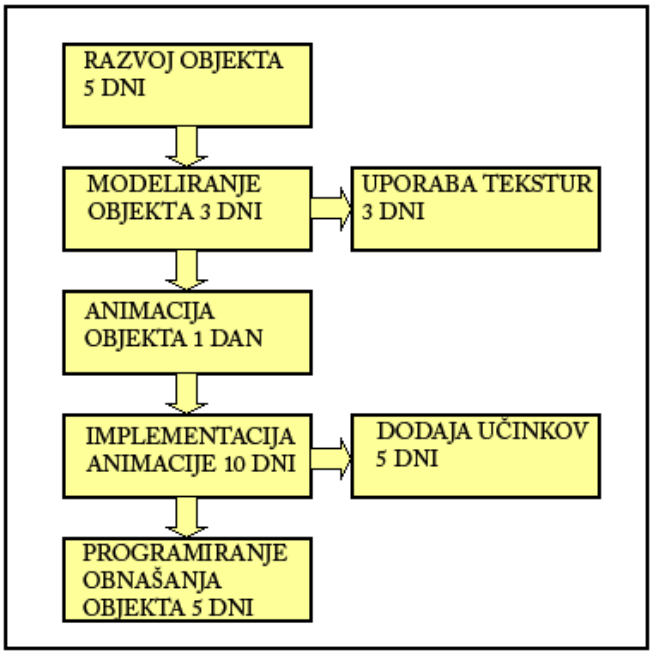
Tako kot večina industrij (npr. avtomobilska) je industrija iger pod časovnim pritiskom. Večina podjetij, ki razvija igre za igralne konzole ali osebne računalnike (visoka stopnja kompleksnosti), je prisiljena proces razvoja iger optimizirati do te stopnje, da določene aktivnosti izvaja sočasno. Na tem mestu se prvič srečamo z besedo cevovodni razvoj (angl. pipeline development) [6].

Beseda cevovod se v računalništvu uporablja kot realizacija CPE (angl. CPU), ki omogoča sočasno izvrševanje ukazov tako, da se posamezni koraki ukazov prekrivajo. Več si lahko bralec prebere v [5].

Sodoben razvoj iger predstavlja potrebo po cevovodnem razvoju, ki razvijalcem omogoča krajši čas izvajanja projekta. Obstaja veliko načinov, kako gledamo na cevovodni razvoj. Eden je na primer seznam točk, ki jih razvijalska ekipa poskuša uresničiti. Žal takšen pristop ne prikazuje povezave med aktivnostmi procesa razvoja. Aktivnosti so tiste, ki povzročajo časovne zamike.

Odvisnosti med posameznimi aktivnostmi so sestavni del procesa razvoja iger. Tako lahko opazimo, da je v danem primeru na sliki 3.3 [6], modeliranje objektov (3 dni) odvisno od razvoja objektov (5 dni), uporaba tekstur (3 dni) pa od modeliranja objektov (3 dni), pri čemer lahko proces ob končani aktivnosti modeliranja objektov (3 dni) nadaljujemo, ker animacija objektov (1 dan) ni odvisna od uporabe tekstur (3 dni). Na ta način si lahko sliko 3.3 [6] samodejno razlagamo naprej.

Lahko rečemo, da je čas trajanja podprocesa 24 dni (opazujemo glavni tok: 5 dni + 3 dni + 1 dan + 10 dni + 5 dni). Aktivnosti uporaba tekstur (3



Slika 3.3: Graf podprocesa cevovodnega razvoja igre

dni) in dodajanje učinkov (5 dni) lahko izpustimo (izvedemo sproti), ker sta izven glavnega toka in njuna izdelava ne presega obdobja 24 dni. Izvedemo ju lahko v obdobju glavnega toka podprocesa. Če predpostavimo, da podproces, predstavljen na sliki 3.3, omogoča razvoj 3D objekta, na primer človeka, ki bo nastopil v igri, lahko čas trajanja podprocesa za razvoj posamičnega objekta dodatno izboljšamo. Na tem mestu si pomagamo s cevovodnim razvojem.

Cevovodni razvoj je torej namenjen sočasnemu izvajanju, zato je logično, da smo v primeru razvoja posamičnega objekta prisiljeni na 24 dni trajajoče obdobje. V vseh sodobnih igrah nastopa zagotovo več kot le en 3D objekt in tukaj pride cevovodni razvoj do izraza. Ko strokovnjaki končajo delo na razvoju objekta (5 dni) in realizirane rešitve predajo naprej v fazo modeliranja objekta (3 dni), lahko prevzamejo razvoj novega 3D objekta. Ob istem času, ko poteka modeliranje prvega objekta, se razvija že nov objekt. S takšnim tempom razvoj poteka skozi vse aktivnosti podprocesa. Čas razvoja prvega objekta bo vedno 24 dni, pridobili bomo pri vseh ostalih objektih, ki sledijo. Če ne bi uporabljali cevovodnega razvoja, bi razvoj dveh objektov trajal 48 dni, treh 72 dni itn. Pri cevovdnem razvoju predstavlja končanje prvega objekta (torej 24. dan razvoja), 5. dan aktivnosti implementacije animacije na drugem objektu. Torej je čas potreben za dokončanje obeh objektov 34 dni (24 dni +

10 dni) in ne 48 dni. Tako lahko predpostavimo, da je dodaten čas potreben za razvoj vsakega nadaljnega objekta, ob končanju trenutnega, enak 10 dni, kar je čas najdlje trajajoče aktivnosti.

V cevovodni CPE se vsak korak izvaja enako dolgo. Pri razvoju igre, kjer sodeluje več ljudi in vsaka aktivnost predstavlja različno težavnostno stopnjo, je velik izziv predvideti čas izvajanja posamezne aktivnosti. Za popolnoma izkoriščen cevovod, je pogoj izvajanje vseh aktivnosti istočasno, kar je v primeru iger nemogoče. Različna podjetja imajo različne pristope, kako izboljšati in bolje izkoristiti cevovod. Ena izmed rešitev je izboljšati razvojna orodja, ki smo jih omenili v prejšnjem poglavju.

Bralec si lahko sam razlaga, da je takšen način razvoja igre možen le v večjih podjetjih, kjer na projektu sodeluje veliko ljudi. Razvoj igre postane avtomatiziran proces. Pri manjših razvojnih skupinah lahko razvijalec upošteva posamezne korake pri procesu razvoja igre, ne more pa pospešiti razvoja. Izvajanje aktivnosti poteka zaporedno in ne vzporedno kot pri cevovodnem razvoju. Manjše razvojne skupine se posledično pogosteje odločajo za manj kompleksne projekte, ki so namenjeni bolj preprostim napravam, npr. telefonom.

Besedo cevovodni razvoj iger ne srečujemo samo v povezavi s cevovodno CPE, ampak tudi pri povezanosti med posameznimi fazami procesa razvoja igre [3]. Tvorimo odvisnostne verige posameznih aktivnosti procesa, katere skupaj predstavljajo cevovod razvoja. Ob dostopu do posamezne faze, lahko način implementacije, ki nastopa v določeni aktivnosti uporabimo na različnih objektih, npr. implementacijo animacij gibov. Torej cevovod v tem primeru poimenovanja predstavlja vse osnovne faze in njihovo ustrezno povezanost v procesu razvoja igre, posamezni koraki cevovoda pa predstavljajo posamezne faze, ki nastopijo v eni izmed osnovnih faz.

Pri razvoju igre *Space Pong* sem poskušal upoštevati vse osnovne faze procesa razvoja igre: predprodukcijska, produkcijska faza in faza izdaje igre. V naslednjih podpoglavjih bom posamezne faze razdelal na podfaze in opisal, kako je v okviru diplomske naloge nastala igra *Space Pong*.

3.4.2 Predprodukcijska faza

Sestavna dela predprodukcijske faze sta:

- faza analiziranja (angl. analysis phase)
- faza ustvarjanja (angl. creative phase)

Faza analiziranja

Začetna faza razvijalcem omogoča izbiro ustrezne naprave (npr. osebni računalnik ali telefon), na kateri bo igra delovala, in izbiro tematike igre. Če pogledamo področje manj kompleksnih iger, kjer zadostujeta le eden ali dva razvijalca, se motivi za razvoj igre razlikujejo. Nekateri želijo uporabiti igro za samopromocijo in jo ponujajo zastonj, spet drugi želijo razviti najbolj prodajano igro. Motivi večjih podjetij (več sto razvijalcev) so bolj povezani s prodajo iger kot pa s samopromocijo.

Po ustrezni izbiri ciljne naprave in tematike igre je potrebno analizirati trg, kjer bo igra ponujena. Kaj je tisto, kar uporabniki želijo igrati? Kakšni so najnovejši trendi? Ta faza je za trženje igre najpomembnejša. Velikokrat v zgodovini industrije iger so znana podjetja ob napačni presoji trga propadla.

Pri razvoju igre *Space Pong* je bila osnovna motivacija spoznavanje razvojnega okolja xCode in telefona iPhone. Spoznati je bilo potrebno programski jezik Objektni-C ter korake razvoja igre.

Tukaj bi omenil, da ideja igre *Pong* sega v začetek razvoja iger nasploh. Leta 1972 je Allan Alcom iz podjetja Atari Inc.[4] razvil prvo različico igre. Ideja je relativno preprosta, govorimo o igranju tenisa v 2D, pri čemer se lahko ploščici (oz. loparja) premikata le levo in desno. Zmaga tisti, ki doseže večje število točk. Igra je bila znotraj podjetja tako priljubljena, da so se odločili za produkcijo. Zaradi velikega uspeha lahko trdimo, da je bila igra *Pong* odločilna za uspešen razvoj industrije iger. Danes obstaja veliko različic na vseh mogočih napravah (osebni računalniki, igralne konzole, telefoni itd.). Po 38 letih od uradnega izida še vedno velja za zelo priljubljeno igro.

V fazi analiziranja sem pregledal različice igre *Pong* na spletni trgovini aplikacij (angl. App Store), torej konkurenco. Nad večino sem bil razočaran, predvsem zaradi dejstva, da je bilo vloženega malo truda v sam razvoj. Fizikalni sistem in umetna inteligenca sta bila preprosto implementirana, kar je bil razlog, da sem se odločil za svojo različico igre. Poimenoval sem jo *Space Pong*, ker je tematika igre vesolje.

Faza ustvarjanja

Po fazi analiziranja imamo jasen pregled nad tem, kaj želimo. Na tej točki je vsak razvijalec osredotočen na sestavne dele igre, torej vsebino, pravila, model, igralnost in zgodbo.

Pri komplesnih projektih lahko sodeluje v tej fazi tudi več 100 ljudi na

[4] http://www.atari.com/

različnih področjih, od grafičnih oblikovalcev do programerjev. Manjši projekti zahtevajo manj dela in sestavne dele igre lahko razvijalec razvije sam. Tako je bilo na začetku industrije iger v 70. in 80. letih 20. stoletja. Glavni programer je predstavljal hkrati grafičnega oblikovalca. Danes je zahtevnost uporabnikov na zelo visokem nivoju in proces razvoja iger se je temu moral prilagoditi. Tako imajo velika podjetja, npr. Electronic Arts[5] posebej zaposlene ljudi, ki razvijajo koncept igre.

Kot že prej omenjeno je osrednja naloga te faze izdelava seznama opisov sestavnih delov igre. Skupna pravila zaradi različnih zvrsti iger ne obstajajo. Na naslednji strani v dokumentu tabela 3.1 [11] so predstavljeni opisi igre *Space Pong*.

[5]http://www.ea.com/

Naslov igre	Space Pong
Kratek opis	Igra predstavlja simulacijo tenisa v 2D. Premik loparja ali ploščka je možen le levo in desno, kar je bistvena omejitev v primerjavi s simulacijo tenisa. Zvok in tema igre je vesolje. Zmaga tisti, ki prvi zbere 5 točk. Ob doseženi točki se plošček igralcu zmanjša, nasprotniku pa poveča. Igralec lahko izbira med tremi stopnjami težavnosti. Spreminjamo lahko tudi barvo svojega ploščka.
Zvrst igre	Arkadna igra, ki je podobna igram na igralnih avtomatih. Zaradi hitrosti zahteva hitro refleksno odzivnost igralca.
Osnovni elementi igre	Igra ima implementiran fizikalni sistem, ki je sestavljen iz dveh delov glede na plošček. Kadar imamo situacijo, kjer trčita dve gibajoči se telesi (žoga in plošček), uporabimo formulo za neprožni trk dveh gibajočih se teles. Ob trku statičnega ploščka in žoge se pogleda točka trka na ploščku in glede na njeno pozicijo izračunamo kot, ki vpliva na odbojno hitrost. Umetna inteligenca deluje na osnovi predikcije. Ob trku igralčevega ploščka in žoge se preračuna pozicija žoge na nasprotnikovi strani glede na predviden odboj od stranskih sten. **Osnovni objekti:** žoga; plošček (igralec, nasprotnik); točkovna sistema (igralec, nasprotnik); stena (ovira); gumb za vrnitev v meni. **Dogodki:** preverjanje trkov žoge in ploščka, žoge in stene; preverjanje, kdo je dosegel točko; proženje umetne inteligence; naključno proženje ovir.
Način igranja	Plošček se odziva na dotik. Igralec začne igro tako, da se dotakne ploščka in ga s prstom premika do željene pozicije (levo ali desno).
Struktura navigacije	Aplikacijo sestavljajo štirje pogledi: meni, nastavitve, informacije (podatki o avtorju igre), izvajanje igre.
Implementiran zvok	Zvok se pojavi ob trku žoge in ob doseženi točki.

Tabela 3.1: Opis sestavnih delov igre *Space Pong*

3.4.3 Produkcijska faza

Ko imamo v rokah dokument opisa sestavnih delov igre (tabela 3.1 [11]), lahko pričnemo z realizacijo projekta. Pri večjih projektih govorimo o fazi začetka sodelovanja vseh članov ekipe.

Produkcijska faza se dodatno deli na:

- faza razvoja (angl. development phase)
- faza testiranja (angl. testing phase)

Faza razvoja se naprej deli na posamezne faze. Odvisna je od kompleksnosti, zvrsti igre ter politike podjetja. Lahko rečemo, da je specifična za vsak razvoj posebej. Pri igri *Space Pong* je izvedba faze razvoja potekala po naslednjem zaporedju:

- faza postavitve ogrodja igre
- razvoj uporabniškega vmesnika igre
- detekcija trkov, implementacija fizike
- razvoj umetne inteligence
- razvoj navigacijske strukture (meni, podmeni itd.)
- razvoj grafičnih elementov
- dodaja zvočnih datotek

Nato je sledila še faza testiranja.

Faza postavitve ogrodja igre

Na začetku zahteva razvoj vsake programske opreme postavitev ogrodja. Ogrodje predstavlja kreiranje ustreznega projekta z razredi, kreiranje določenih objektov ter inicializacijo spremenljivk in metod.

Ker faza razvoja grafičnih elementov pri nas nastopa v fazi celotnega razvoja šele na predzadnjem mestu, je bilo potrebno pri postavitvi ogrodja določene objekte (npr. žogo in ploščka) povezati s preprostimi slikami s spleta. Ob ustrezni povezavi objektov in inicializaciji ostalih spremenljivk smo nadaljevali z definiranjem ustreznih metod. Na tem mestu bi omenil, da ima vsak razred namenjen razvoju aplikacij na telefonu iPhone (iPad in iPod Touch) vnaprej definirane metode, ki predstavljajo osnovno strukturo oziroma ogrodje aplikacije. Te metode so:

- *(void)didReceiveMemoryWarning*
- *(void)viewDidUnload*
- *(void)dealloc*
- *(id)initWithNibName*
- *(void)loadView*
- *(void)viewDidLoad*
- *(BOOL)shouldAutorotateToInterfaceOrientation*

Prve tri so obvezne, ostale so opcijske in jih lahko ignoriramo.

(void)didReceiveMemoryWarning je metoda, ki jo uporabljamo pri aplikacijah, ki vsebujejo več pogledov (angl. multiple views), npr. pogled na meni igre, nastavitve in igro samo. Kot omenjeno v poglavju 2 je upravljanje s pomnilnikom v operacijskem sistemau iOS prepuščeno programerju. Ker nimamo navideznega pomnilnika, je za določeno aplikacijo pomembno, da ima na razpolago čim več pomnilniškega prostora. Pri več pogledih nam ta metoda omogoča, da lahko neaktivni pogled odstranimo s pomnilnika in ga s tem sprostimo za trenutno aktivni pogled.

(void)viewDidUnload je metoda, ki jo uporabljamo pri aplikacijah, ki vsebujejo več pogledov. Razlika od prej definirane metode je, da nam ta omogoča kontrolo nad objekti, katerim lahko dodelimo določene lastnosti, npr. obstoj (angl. retain). Ta lastnost nam omogoča, da objekt ostane v pomnilniku tako dolgo, dokler ne zaključimo obstoječe aplikacije (koristno za objekte, ki se jim med delovanjem aplikacije spreminjajo vrednosti). Problem nastane, ko ima aplikacija več pogledov. Telefon iPhone (tudi naprave iPad in iPod Touch) uporabljajo razred *ViewController* za prikaz različnih pogledov. Ta nam omogoča, da pogled ustvarjen s pomočjo okolja za razvoj uporabniškega vmesnika (angl. Interface Builder), naložimo v pomnilnik in ob končanju sprostimo iz pomnilnika. Ob sprostitvi pogleda (angl. release view) njegovi objekti, ki imajo definirano lastnost obstoj, ostanejo v pomnilniku, česar v določenih situacijah ne želimo. Uporaba metode je na tem mestu koristna, ker nam omogoča spreminjanje vrednosti teh objektov. Če želimo biti dober upravljalec pomnilnika, je potrebno tak objekt odstraniti. Ob ponovni naložitvi pogleda v pomnilnik, plast *Cocoa Touch* poskrbi za ustrezno inicializacijo vseh njegovih objektov.

(void)dealloc se uporablja za sprostitev objektov iz pomnilnika ob zaključitvi aplikacije.

(id)initWithNibName je konstruktor razreda, ki skrbi za njegovo udejanjanje (instantizacijo). Ob inicializaciji razreda mu lahko dodamo dodatne vrednosti. Aplikacije, ki imajo več pogledov, lahko uporabljajo konstruktorje za prenos vrednosti spremenljivk, npr. stanje nastavitev igre lahko na tak način shranimo.

(void)loadView se uporablja, kadar želimo sami kreirati poglede, torej s pisanjem programske kode. Razredi aplikacij telefona iPhone imajo poleg vnaprej definiranih metod povezave s pogledi (angl. views). Poglede lahko ustvarimo s pomočjo razvojnega okolja uporabniškega vmesnika (angl. Interface Builder), katerega smo omenili v poglavju 2. Drug način kreiranja pogledov je s pomočjo te metode.

(void)viewDidLoad uporabljamo, kadar želimo v razvojnem okolju uporabniškega vmesnika (angl. Interface Builder) kreiran pogled dodatno razširiti z določenimi lastnostmi, npr. ob inicializaciji igre prikazati sporočilo.

(BOOL)shouldAutorotateToInterfaceOrientation se sproži ob spremembi orientacije telefona. V drugem poglavju sem omenil, da telefon lahko uporabljamo pokončno ali ležeče. Aplikacije, še posebej igre, velikokrat uporabljajo možnost ležečega položaja, ker uporabnikom ta način olajša igranje iger.

Ob spoznanju osnovnih metod in implementiranih povezavah med objekti in slikami s spleta, pri čemer si pomagamo z razvojnim okoljem uporabniškega vmesnika (angl. Interface Builder), smo lahko začeli z implementacijo metode *viewDidLoad*. Zakaj *viewDidLoad* in ne *loadView*? Ker razvijamo že kreiran pogled. Na tem mestu bi opozoril, da je potrebno v razvojnem okolju ustvariti dodatno povezavo med pogledom in razredom.

Vse bistvene operacije igre *Space Pong* se izvajajo v neskončni zanki, imenovani *gameLoop*. V resnici to ni zanka, ampak metoda, ki jo prožimo v neskončnosti, dokler uporabnik ne zapre ali prekine igro. Ob inicializaciji igre v metodi *viewDidLoad* prvič prikličemo metodo *gameLoop* in jo s pomočjo razreda merilnik časa (angl. timer) prožimo na n[6] sekund. Objektu žoga lahko v metodi *viewDidLoad* nastavimo hitrost in jo kasneje v metodi *gameLoop* spreminjamo. S tem omogočimo gibanje objekta, ki predstavlja sliko žoge.

[6] $n = 0.008$ sekund, uporabljen čas je omogočal tekoče delovanje igre.

Rezultati faze:

- ustvarjen pogled s preprostimi slikami objektov (žoga, ploščka)
- implementacija metode *viewDidLoad*
- implementacija metode *gameLoop*
- spreminjanje pozicije objekta žoge s pomočjo proženja metode *gameLoop*

Razvoj uporabniškega vmesnika igre

Ob uspešni postavitvi ogrodja igre je potrebno realizirati uporabniški vmesnik. V dokumentu opisa sestavnih delov igre (tabela 3.1[11]) je definiran način igranja. Zaslon na dotik nam omogoča uporabo treh metod, in sicer *(void)touchesBegan*, *(void)touchesMoved* in *(void)touchesEnded*. Prožijo se ob začetku dotika na zaslonu, ob premiku po zaslonu in ob umiku prsta ali prstov z zaslona.

Igra *Space Pong* spreminja pozicijo objekta ploščka (slika ploščka) samo ob uporabnikovem dotiku. Implementacija metod *(void)touchesBegan* in *(void) touchesEnded* ni bila potrebna. Ko uporabnik prične z dotikom na plošček, se spremeni stanje igre iz zaustavljenega v delujoče, z drugimi besedami izvajati začnemo metodo *gameLoop*. Premik prsta po zaslonu levo ali desno omogoča premik ploščka (sledi uporabnikovemu prstu).

Rezultata faze:

- implementiran uporabniški vmesnik igre (objekt plošček sledi uporabnikovemu prstu)
- nadgradnja logike, ki je odvisna od uporabniškega vmesnika

Detekcija trkov, implementacija fizike

Ogrodje je postavljeno, uporabniški vmesnik deluje, sledi implementacija trkov med objekti (žoga, plošček, ovire in igralna površina).

Igra *Space Pong* detekcijo trkov določi s pomočjo integrirane metode *CGRect-IntersectsRect*, ki vrne ustrezno vrednost ob preseku dveh kvadratov. Vsak objekt s sliko je v resnici kvadrat (kvadrat, ki predstavlja žogo je pomanjšan do te mere, da lahko del med ostrimi in oglatimi robovi zanemarimo. Na izvajanje trkov v končni igri nima vpliva), zato je izbira metode ob detekciji trkov ustrezna. Trk med ploščkom in žogo je razdeljen na dva dela:

- trk dveh gibajočih se teles
- trk statičnega in gibajočega se telesa

Preden začnemo z implementacijo odbojev, bi opozoril, da se nahajamo v 2D prostoru in posledično računamo vrednosti v odvisnosti od x in y osi.

V primeru **trka dveh gibajočih se teles** se hitrost žoge po y osi izračuna z zamenjavo predznaka hitrosti, [8]:

$$v(y) = -v(y) \tag{3.1}$$

Kadar govorimo o pozitivni vrednosti hitrosti, potuje žoga navzdol, kadar govorimo o negativni vrednosti hitrosti, potuje žoga navzgor. Če primerjamo s koordinatnim sistemom, opazimo, da se vrednosti spreminjata na popolnoma obraten način. Razlog se skriva v začetni poziciji točk x in y, ki nista na sredini zaslona ekrana kot na koordniatnem sistemu, ampak na zgornjem levem robu. Dodatno bi želel omeniti, da je poenostavitev enačbe (3.1) po y osi posledica gibanja ploščka. V dokumentu opisa sestavnih delov igre (tabela 3.1 [11]) je bilo predstavljeno, da lahko plošček premikamo levo ali desno. Z drugimi besedami, ob trku dveh gibajočih se teles se y os ploščka ne spreminja, spreminja se samo x os ploščka, kar je razlog za uporabo enačbe (3.1).

Podoben pristop smo uporabili pri omejevanju vidnega polja igre in naključni pojavitvi ovir. Govorimo o odboju žoge od stene (rob zaslona in ovire). Ob poskusu prečkanja žoge čez širino (leva ali desna stran) zaslona ali poskusu prečkanja čez oviro se po enakem principu, zamenjava predznaka po enačbi (3.1), kjer pa uporabimo x, odbije žoga nazaj na igralno površino. Ob poskusu prečkanja žoge čez višino zaslona (zgornja ali spodnja stran) je implementacija odboja izpuščena. Predstavlja namreč indikator doseganja točk enega izmed igralcev.

Hitrost odboja žoge po x osi od gibajočega se ploščka se izračuna po fizikalni enačbi za neprožni trk dveh gibajočih se teles [8]. Razlog za uporabo formule neprožnega trka je bila predpostavka, da sta žoga in plošček neprožni telesi, čeprav v realnosti to ne velja. Poglejmo na primer odbijanje košarkarske žoge, vsak odboj predstavlja prožni trk, pri katerem se kinetična energija ne ohranja. Posledica je padec hitrosti in nižji doseg višine ob odboju (upor zraka zanemarimo). Kljub temu je ob številnih poskusih uporaba formule neprožnega trka prikazala najboljše rezultate igralnosti.

Pri neprožnem trku moramo upoštevati tako ohranitev kinetične energije, kot ohranitev gibalne količine. Obe enačbi uporabimo pri izpeljavi hitrosti po

trku. Enačba ohranitve gibalne količine:

$$m_1 v_1(x) + m_2 v_2(x) = m_1 v_1^{'}(x) + m_2 v_2^{'}(x) \tag{3.2}$$

in enačba ohranitve kinetične energije:

$$\frac{m_1 v_1^2(x)}{2} + \frac{m_2 v_2^2(x)}{2} = \frac{m_1 v_1^{'2}(x)}{2} + \frac{m_2 v_2^{'2}(x)}{2} \tag{3.3}$$

pri čemer $m_1 v_1(x)$ predstavlja produkt mase in hitrosti ploščka pred trkom in $m_2 v_2(x)$ produkt mase in hitrosti žoge pred trkom, $m_1 v_1^{'}(x)$ pa produkt mase in hitrosti ploščka po trku in $m_2 v_2^{'}(x)$ produkt mase in hitrosti žoge po trku. Ker nas zanima hitrost žoge, torej $v_2^{'}(x)$, jo s pomočjo pravil za reševanje linearnih enačb izpeljemo kot

$$v_2^{'}(x) = \frac{v_2(x)(m_2 - m_1) + 2m_1 v_1(x)}{m_1 + m_2} \tag{3.4}$$

Naslednji korak je ustrezna določitev mas objektov. Ob določenem številu poskusov (večji poudarek na upoštevanju dinamike igre, kot realnosti) je končna predpostavka $m_1 = 2m_2$. Enačba (3.4) se ustrezno spremeni

$$v_2^{'}(x) = \frac{-v_2(x) + 4v_1(x)}{3} \tag{3.5}$$

Zaradi ustrezne hitrosti izvajanja igre je bilo potrebno končno izračunano hitrost omejiti na vrednost 6,2 (po testiranju zadnja meja, ki omogoča ustrezno igralnost). Doseganje te vrednosti je odvisno od hitrosti ploščka in žoge ob trku, enačba (3.5).

V primeru **trka statičnega in gibajočega se telesa** nismo toliko upoštevali realnosti, kot smo nepredvidljivost in dinamičnost igre.

Koncept je bil zastavljen tako, da ob trku mirujočega ploščka in žoge izračunamo razdaljo med središčema objektov. Večja kot je oddaljenost središča žoge od središča ploščka, manjši je izračunan kot ob trku. Manjši kot je kot, večja je odbojna hitrost $v_2^{'}(x)$.

V praksi to pomeni, da ob trku med mirujočim ploščkom in žogo upoštevamo pozicijo odboja žoge na ploščku. Če pride do odboja na robovih ploščka, bo posledično večja odbojna hitrost. Bolj ko igralec odbija žogo s središčem mirujočega ploščka, manjša bo odbojna hitrost.

Najprej izračunajmo razdaljo med središčema objektov

$$dx = T_{sredisceZoge}(x) - T_{srediscePloscka}(x) \tag{3.6}$$

in

$$dy = T_{srediscZoge}(y) - T_{srediscePloscka}(y) \tag{3.7}$$

Po izračunanih razdaljah sledi uporaba metode *atan2(y, x)*, ki omogoča izračun kota med x osjo (plošček) in oddaljenostjo točke (x, y), v našem primeru (dx, dy). Definicija metode *atan2(y, x)*:

$$atan2(y,x) = \begin{cases} \arctan(\frac{y}{x}) & x > 0 \\ \pi + \arctan(\frac{y}{x}) & x < 0, y \geq 0 \\ -\pi + \arctan(\frac{y}{x}) & x < 0, y < 0 \\ \frac{\pi}{2} & x = 0, y > 0 \\ -\frac{\pi}{2} & x = 0, y < 0 \\ 0 & x = 0, y = 0 \end{cases} \tag{3.8}$$

Na tem mestu bi omenil, da v igri *Space Pong* zadnje vrednosti 0 funkcija $atan2(y,x)$ ne more doseči. Sicer se lahko zgodi, da je ob trku vrednost $dx = 0$, vendar je vrednost dy minimalna in velja $dy \neq 0$. Sledi izračun kota v radianih:

$$\varphi = atan2(dy, dx) \tag{3.9}$$

sedaj lahko izračunamo še hitrost žoge po trku:

$$v_2^{'}(x) = v_2^{''}(x)\cos(\varphi) \tag{3.10}$$

pri čemer predstavlja $v_2^{''}(x)$ novo naključno generirano hitrost, kar daje igri prej omenjeno dinamiko in nepredvidljivost.

Na popolnoma enak način je s pomočja algoritma umetne inteligence (opis v naslednji fazi) implementirana logika trkov pri nasprotniku, ki ga upravlja telefon iPhone.

Rezultata faze:

- implementiran fizikalni sistem ob trku dveh gibajočih se teles, ločeno za uporabnika in nasprotnika
- implementiran fizikalni sistem ob trku statičnega in gibajočega se telesa, ločeno za uporabnika in nasprotnika

Razvoj umetne inteligence

Za delujočo igro potrebujemo zadnji korak in sicer razvoj umetne inteligence. Najpreprostejša oblika implementacije le-te je sledenje ploščka gibanju žoge. Na ta način je bila implementirana umetna inteligenca pri večini iger ob pregledovanju spletne trgovine aplikacij (angl. App Store). Obstaja primer, kako lahko s pomočjo te oblike umetne inteligence ustvarimo nepremagljivega nasprotnika. Potrebno je nastaviti večjo hitrost gibanja ploščka od hitrosti gibanja žoge.

Z vidika igralnosti je bolj učinkovita metoda predikcija (bližje načinu igranja ljudi) . Ob trku igralčevega ploščka z žogo poskuša algoritem predvideti pozicijo žoge na nasprotnikovi strani. Pri tem upošteva možnost trkov z levo ali desno stranjo igralne površine. Na tak način se izognemo nepotrebnemu sledenju žoge, kjer plošček pošiljamo v popolnoma napačno smer.

V vsaki iteraciji glavne zanke *gameLoop* se izračunava enačba predikcije za optimalno pozicioniranje nasprotnikovega ploščka. Smer žoge začne analizirati šele, ko ta prečka zgornjo polovico igralne površine. Lahko bi rekli območje nasprotnika.

Osnovna ideja je pretvorba poti žoge v premico. Na tak način ugotovimo, v katero smer se žoga premika. Prvi korak je določitev smernega koeficienta poti žoge, ki ga določimo s pomočjo naslednje enačbe:

$$k = \frac{y - y_1}{x - x_1} \tag{3.11}$$

$y - y_1$ in $x - x_1$ predstavljata razliki točk, ki se nahajata na premici gibanja žoge (kjerkoli na premici). Ti razliki lahko imenujemo *dy* in *dx*. Če vsako izmed razlik delimo s časom t dobimo enačbo, ki ohranja razmerje:

$$\frac{y - y_1}{x - x_1} = \frac{\frac{d_y}{t}}{\frac{d_x}{t}} = \frac{v_y}{v_x} \tag{3.12}$$

pri čemer v_y in v_x predstavljata hitrosti žoge po y in x osi. Sledi izpeljava predikcijske enačbe:

$$x = x_1 + \frac{(y - y_1)v_x}{v_y} \tag{3.13}$$

x predstavlja končno pozicijo žoge in algoritmu omogoča ustrezno pozicioniranje ploščka. Vendar bi želel opozoriti na pomanjkljivost. Z izpeljano formulo za predikcijo pozicije žoge ni nič narobe v situaciji, kadar žoga ne želi zapustiti igralne površine (leva ali desna stran). V primeru zapustitve igralne površine

in posledičnega odboja žoge nazaj na igralno površino je potreben dodaten razmislek, ki nas pripelje do manjšega popravka rezultata predikcije.

Kot prvo si poglejmo možnost gibanja žoge izven leve strani igralne površine. Ob tem dogodku se uporabi sledeča enačba:

$$x = -(x \bmod sirinaZaslona) \tag{3.14}$$

Uporabimo torej prej izračunano predikcijo x po enačbi (3.13) in s pomočjo operacije modul kljub odboju z robom igralne površine predvidimo pozicijo žoge na nasprotnikovi strani. Širina zaslona je v enačbi (3.15) predstavljena kot *sirinaZaslona*, pri telefonu iPhone je 320 točk (enako velja za iPod Touch). Relevantna predikcija lahko torej zaseda vrednosti na območju od 0 do 320. Če žoga želi zapustiti levo stran igralne površine, potem je izračunana predikcija x manjša od 0. Operacija modul med negativno predikcijo in širino igralne površine omeji izračunano predikcijo na območju med točko 0 in 320. Minus pred enačbo je potreben za zagotovitev pozitivne vrednosti.

Iz zgoraj povedanega je v drugem primeru ob potovanju žoge izven desne strani igralne površine rešitev trivialna (vrednost predikcije je višja od 320)

$$x = sirinaZaslona - (x \bmod sirinaZaslona) \tag{3.15}$$

Algoritem umetne inteligence nam omogoča takojšnje pozicioniranje nasprotnikovega ploščka. Cilj vsake igre je dobra igralnost, kar posledično dovoljuje zmotljivost nasprotnika. Nepremagljiv nasprotnik je v vseh igrah nezaželjen. Enačbi za določitev predikcije je bilo potrebno dodati določeno napako (angl. error), s pomočjo katere nasprotnik v določenih primerih nepravilno pozicionira plošček in dobi točko. Igralec na ta način lahko pride do zmage.

Igra pozna tri težavnostne stopnje, ki se ločujejo po velikost napak. Tako ima najlažja težavnostna stopnja največjo napako in najtežja stopnja najmanjšo napako. Napako izračunamo po naslednji enačbi:

$$napaka = konstanta + dodatek \tag{3.16}$$

pri čemer predstavlja *konstanta* neko določeno vrednost pridobljeno na osnovi testiranja (najboljša igralnost), *dodatek* pa delež dolžine ploščka. V tabeli 3.1 smo omenili, da ob doseženi točki ustrezno prilagodimo dolžine ploščkov. Če ima nasprotnik (v našem primeru telefon iPhone) zelo majhen plošček, mu napako dinamično zmanjšamo (ustrezno zmanjšamo *dodatek*) in s tem omogočimo boljšo igralnost. Tabela 3.2 prikazuje vrednosti napak (konstante in delitelje) treh težavnostnih stopenj igre *Space Pong*. Delitelje uporabimo pri izračunu *dodatka*.

	lahka tež. stopnja	srednja tež. stopnja	težka tež. stopnja
konstanta	20	16	12
delitelj	1,0	1,5	2,0

Tabela 3.2: Vrednosti napak treh težavnostnih stopenj igre *Space Pong*

V primeru vrednosti delitelja 1,0 sestavlja napako vsota konstante in dolžine ploščka, v primeru srednje težavnostne stopnje pa se uporabi vsota konstante in delež dolžine ploščka, ki ga dobimo pri deljenju z deliteljem 1,5 (delimo dolžino ploščka). Enak postopek, le druga števila se uporabijo pri izračunu tretje (težje) težavnostne stopnje.

Torej lahko povzamemo, da v okviru določene težavnostne stopnje izboljšamo igralnost tako, da minimalno prilagajamo napako.

Rezultat faze:

- implementirana umetna inteligenca na osnovi predikcije

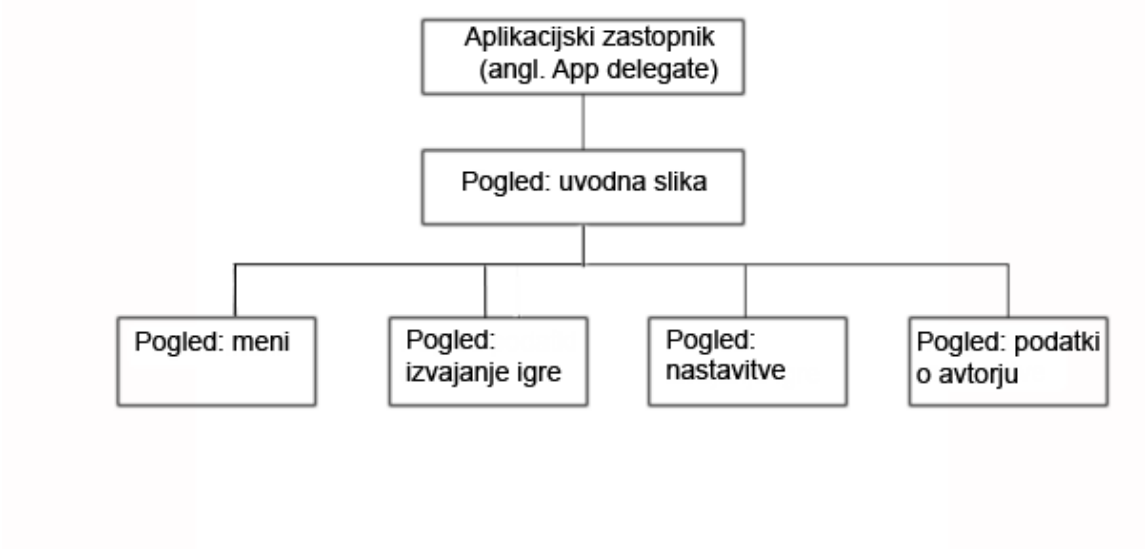
Razvoj navigacijske strukture

Osnovna logika Igre je z implementacijo umetne inteligence končana. Sledijo testiranja določenih situacij in manjši popravki. V tej fazi se skoncentriramo na razvoj menija in nastavitev (angl. settings) ter povezavo na pogled, kjer se igra izvaja. Dodan je bil še pogled, ki vsebuje podatke o avtorju.

Navigacijska struktura (slika 3.4) predstavlja medsebojno povezanost posameznih pogledov. Besedo pogled (angl. view) omenjamo zato, ker imajo aplikacije možnost spreminjanja pogledov in ne oken, kot je to poznano v drugih operacijskih sistemih.

Najprej nekaj besed o aplikacijskem zastopniku (angl. App delegate). Sestavni del vsake aplikacije je primerek razreda *UIApplication*, ki poskrbi za njeno pravilno delovanje. Razvijalec ima dostop do pesebnega razreda z imenom aplikacijski zastopnik. Ta omogoča definicijo metod, ki se lahko izvedejo ob določenem času izvajanja aplikacije. Razred *UIApplication* ima dostop do teh metod. Tako lahko na primer definiramo metodo, ki se izvede ob inicializaciji aplikacije, ali metodo, katero sprožimo tik ob zaprtju aplikacije (sprostimo pomnilnik).

Igra *Space Pong* razred aplikacijskega zastopnika izkorišča za prikaz uvodne slike. Ta pogled aplikacija uporablja za preklapljanje med različnimi pogledi (ostane inicializiran). Potek delovanja je sledeč: prikaz uvodne slike, nato sledi



Slika 3.4: Prikaz navigacijske strukture igre *Space Pong*

prikaz meni, kjer uporabnik izbira med izvajanjem igre, nastavitvami in podatki o avtorju (slika 3.4); ob ustrezni izbiri se v ozadju pokliče razred pogleda z uvodno sliko, ki ustrezno sprosti trenutno aktiven pogled in inicializira zahtevan pogled (dodatek A).

V nastavitvah ima uporabnik možnost izbire med tremi težavnostnimi stopnjami. Težavnostna stopnja je določena na osnovi napake (angl. error), ki je bila omenjena v koraku implementacije umetne inteligence. Nastavitve dodatno omogočajo igralcu izbiro barve ploščka.

Rezultati faze:

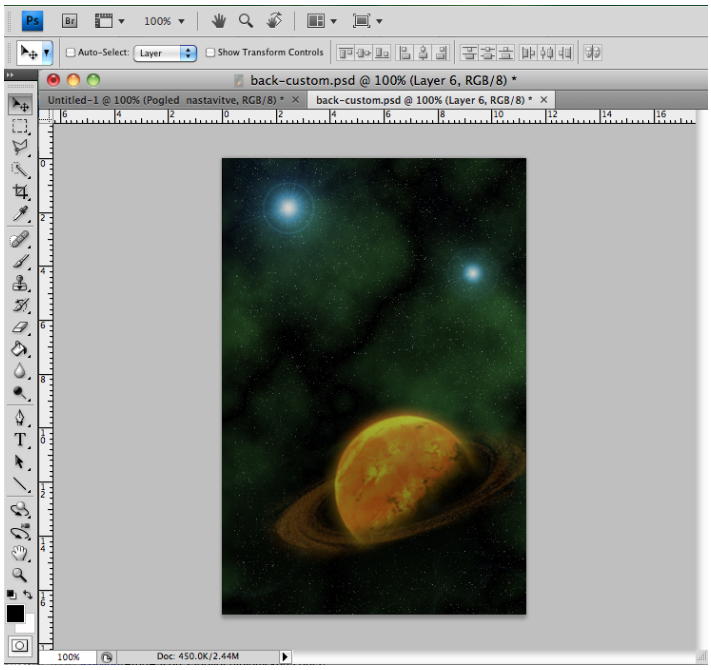
- razvit in v navigacijsko strukturo dodan pogled z uvodno sliko
- razvit in v navigacijsko strukturo dodan pogled z menijem
- razvit in v navigacijsko strukturo dodan pogled s podatki o avtorju
- v navigacijsko strukturo dodan pogled za izvajanje igre (razvit v prejšnjih fazah)
- razvit in v navigacijsko strukturo dodan pogled z nastavitvami

Razvoj grafičnih elementov

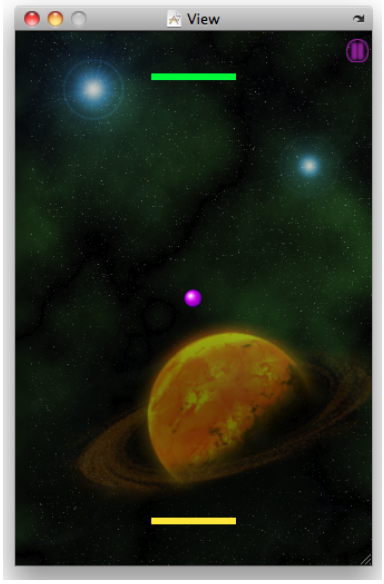
Pri razvoju kompleksnih iger, je ta faza ena najpomembnejših. Grafični učinki in elementi imajo velik vpliv na končen izgled igre, še posebej v okolju 3D.

Potrebni grafični elementi:

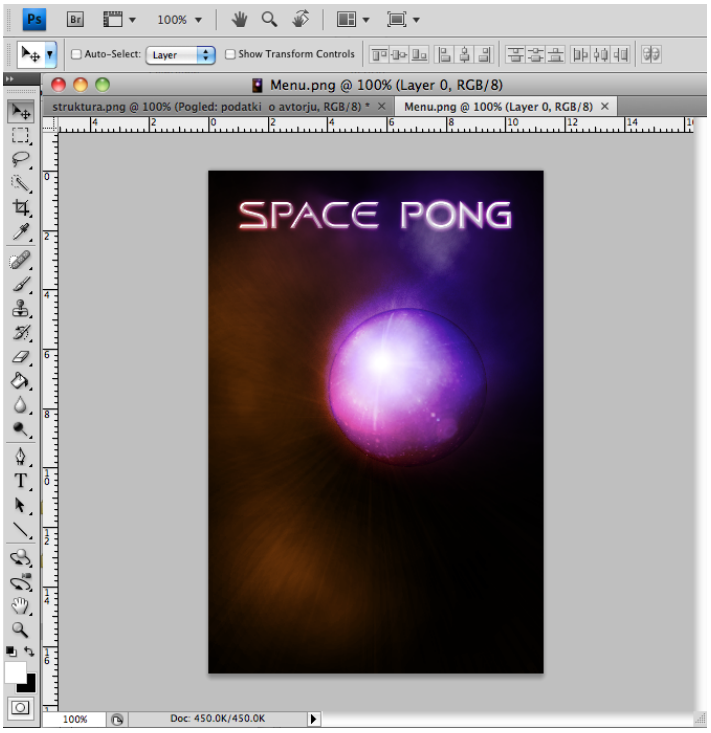
- slika ozadja (slika 3.5)
- slika žoge (slika 3.6)
- slika gumba za prekinitev igre (slika 3.6)
- oblikovanje ploščkov (slika 3.6)
- uvodna slika (enaka v meniju, nastavitvah in pogledu podatki o avtorju, slika 3.7)



Slika 3.5: Primer razvoja slike ozadja s pomočjo orodja *Adobe Photoshop CS4*



Slika 3.6: Pogled: izvajanje igre v razvojnem okolju uporabniškega vmesnika



Slika 3.7: Primer razvoja uvodne slike s pomočjo orodja *Adobe Photoshop CS4*

Okoli objektov žoge in ploščka smo v okviru faze razvoja grafičnih elementov samostojno izdelali učinek sija (angl. glow effect). Deluje na osnovi vidnosti izrisanih objektov. Okoli ploščkov in žoge se izrišejo kvadrati bodisi krogi, ki s pomočjo funkcije vidnosti simulirajo učinek sija.

Dodaja zvočnih datotek

Končno vzdušje igre ustvarimo z dodajo zvoka. Operacijski sistem iOS zahteva posebno vrsto zvočnih datotek s končnico caf (digitalen zvočni zapis na osnovi Applove tehnologije Core Audio). Razvijalec ima možnost implementacije kompleksnih knjižnic *OpenAL* in *Audio Queue* [1].

Igra *Space Pong* ne zahteva kompleksnega zvoka, dodani so le kratki posnetki. Ob dosegu točke lahko zaslišimo ploskanje občinstva in ob trku žoge s ploščkom ter trku žoge z ovirama zaslišimo ustrezen zvok odboja. Edina omejitev pri implementaciji kratkih posnetkov s pomočjo knjižnice *Audio Services*, ki je del vgrajenega ogrodja *AudioToolbox*, je, da lahko ob istem času predvajamo samo eno zvočno datoteko. Preprosto dodajanje zvoka omogočajo vnaprej definirani ukazi knjižnice.

Rezultat faze:

- dodane zvočne datoteke

Faza testiranja

Razvoj logike igre, implementiran navigacijski sistem, ustrezna dodaja grafičnih elementov in zvočnih datotek so bistvene faze, ki smo jih uspešno zaključili. V tej fazi je potrebno aplikacijo namestiti na dejansko napravo in začeti preverjati delovanje. Ob napaki ali želji po izboljšanju aplikacije so potrebni določeni ukrepi. Pri kompleksnih projektih ima ta faza velik pomen, ker predstavlja zadnji korak pred izidom igre. Najhuje, kar se lahko podjetjem zgodi je, da kljub dolgoletnemu razvoju v zadnji fazi ugotovijo določene napake, ki jih popravijo šele po izidu igre z izdajo popravkov (angl. patches). Zato je testiranje potrebno izvajati v vsaki fazi. Osrednja naloga faze testiranja je tako v našem primeru namenjena le manjšim popravkom in izboljšavam.

3.4.4 Faza izdaje igre

Ta faza predstavlja pri razvoju aplikacij za telefon iPhone (enako velja za naprave iPad in iPod Touch) distribucijo preko spletne trgovine aplikacij (angl.

App Store). V drugem poglavju, natančneje v razdelku programska oprema in podrazdelku operacijski sistem je opisan postopek izvajanja aplikacij na telefonu ali napravah. Do distribucije je potrebno upoštevati vse opisane korake. Dodatno moramo preveriti le, ali smo aplikacijo ustrezno prevedli za distribucijo.

Pri večjih projektih govorimo o kompleksni fazi, kjer se podjetja, razvijalci iger povežejo z drugimi podjetji, s katerimi razvijejo določene strategije prodaje. Možnost je recimo izdelava spletne strani, ki predstavlja končno igro.

3.5 Zaključek

V tem poglavju smo se seznanili s pojmom razvoj, pogledali začetek industrije iger in ga primerjali z razvojem danes. Analizirali smo sestavne dele ter posamezne stopnje procesa razvoja igre. Razjasnili smo pojem cevovodni razvoj in dodatno razgradili stopnje procesa razvoja igre *Space Pong* ter vsako stopnjo podrobno opisali.

Poglavje 4

Napredne grafične tehnologije

4.1 Uvod

V prejšnjem poglavju smo podrobno spoznali posamezne faze razvoja igre. Ker je igra *Space Pong* 2D igra in posledično ni tako grafično kompleksna, je zahtevala faza implementacije grafičnih elementov le dodatno znanje v okolju *Adobe Photoshop CS4*. Vsi grafični elementi v igri so statične slike, razen učinek sija (angl. glow effect).

Dinamični učinki vplivajo na boljšo igralnost igre. V poglavju 2 omenjena multimedijska plast predstavlja zlitje video, grafičnih in zvočnih tehnologij s ciljem ustvariti najboljšo možno multimedijsko izkušnjo. Obstajajo tehnologije, ki razvijalcem omogočajo lažjo implementacijo grafičnih učinkov.

V tem poglavju si bomo na kratko ogledali tehnologije Quartz 2D, OpenGL ES (predvsem za 3D okolje) ter tehnologijo Cocos2D, ki omogoča dodajanje 3D učinkov v 2D okolje.

4.2 Tehnologija Quartz 2D

Quartz 2D je napreden grafičen pogon (del SDK), katerega osnovna lastnost je risanje objektov v dveh dimenzijah. Robustnost mu omogoča lastnost neodvisnosti glede končne naprave (telefon iPhone, naprave iPad in iPod Touch in računalniki podjetja Apple Inc.) in integracijo z drugimi tehnologijami (npr. OpenGL ES).

Igra *Space Pong* uporablja to tehnologijo za izrisovanje ploščkov. Na tem mestu bi želel omeniti eno pomanjkljivost. Sicer tehnologija Quartz 2D omogoča delo s pisavo(npr. dodaja barve ozadja), ne omogoča pa dodajanje

novih. Razvojno okolje dovoljuje uporabo le šestih osnovnih vrst pisave, kar je za igre premalo. Obstaja rešitev in sicer redefinicija razreda za prikazovanje teksta in ročno dodajanje datotek s končnico ttf [1].

Quartz 2D uporabljamo predvsem pri:

- podpori delu s slikami
- risanju tako primitivnih kot kompleksnih objektov
- 2D transformacijah
- podpori delu z barvami
- kreiranju PDF datotek

Delo s slikami

Za delo s slikami se uporablja slikarski model (angl. painter model), ki omogoča iz več primitivnih slik sestaviti bolj kompleksne slike. Ko je slika enkrat izrisana, se je ne da več spreminjati. Razvijalec lahko z dodatnimi primitivnimi objekti izdela kompleksno sliko.

Dodatni sta možnosti kreiranja ponavljajoče se primitivne slike (vzorec, npr. šahovnica) in uvajanje postopka senčenja na primitvnih objektih [1].

Risanje objektov

Tehnologija omogoča risanje od preprostih objektov, npr. črt, krogov, kvadratov, do bolj kompleksnih, abstraktnih oblik. Kot že prej omenjeno imamo pri telefonu iPhone poglede in ne okna. Pogled predstavlja tudi področje, kjer lahko izrisujemo grafične objekte. Ob klicu metode *drawRect* (nariši kvadrat) se pogled ustrezno prilagodi risanju. Metoda se lahko izvede takoj.

2D transformacije

Vgrajene funkcije omogočajo vse vrste transformacij (povečave; vodoravni, navpični premiki; rotacije itd.) na objektih. Razvijalec ima dostop do transformacijske matrike, ki jo lahko ustrezno spreminja in s tem vpliva na transforme objektov.

[1] Datoteke, ki vsebujejo matematično izračunano velikost znaka (črka, številka itd.) izbrane pisave.

Delo z barvami

Vsaka naprava ima svoj specifičen način, kako interpretira različne barve. Quartz 2D predstavlja barve kot množico vrednosti, ki so odvisne od barvnega prostora (neodvisen od naprave, možnost kreiranja tudi drugih barvnih prostorov). Dodatno ima vsaka barva definirano *alpha* vrednost, ki omogoča nastavitev transparentnosti barve.

Kreiranje PDF datotek

PDF dokumenti shranjujejo slike, tekst in druge grafične elemente neodvisno od resolucije naprave. Quartz 2D omgoča kreiranje PDF dokumentov, dodatno jih lahko optimizira za določeno uporabo (npr. za tiskanje ali za spletno stran). Ena izmed funkcij je možnost uporabe transformacij na PDF dokumentih.

4.3 Tehnologija OpenGL ES

OpenGL ES je kratica za odprto grafično knjižnico (angl. Open Graphics Library Embedded Systems). Napisana je v programskem jeziku C in omogoča kreiranje tako 2D, kot 3D objektov [1, 11], prostorov, simulacij in razvoj iger. Razvijalci lahko modele definirajo kot črte, točke ali poligone, ki jih s pomočjo določenih tehnik senčenja pretvorijo v ustrezno obliko. OpenGL funkcje pošiljajo grafične ukaze strojni opremi, kar omgoča hiter način izrisovanja.

OpenGL ES je poenostavljena različica knjižnice. Namenjena je predvsem mobilnim napravam, računalniškim sistemom v avtomobilih in igralnim konzolam [1].

4.4 Tehnologija Cocos2D

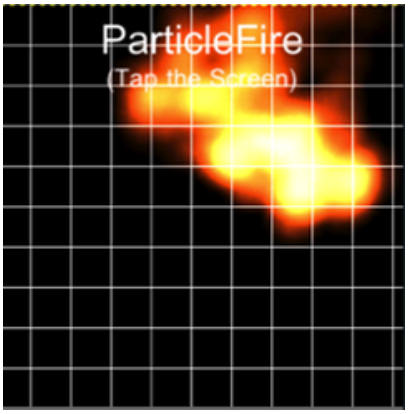
Cocos2D[2] za telefon iPhone (prav tako naprave iPad in iPod Touch) je ogrodje namenjeno razvijalcem 2D iger in drugih grafično interaktivnih aplikacij. Za prikaz 3D učinkov v 2D okolju se uporablja knjižnica OpenGL ES. Ogrodje je napisano v programskem jeziku Objekten-C (angl. Objective-C).

Nekaj glavnih značilnosti

- lastna struktura pogledov

[2]http://www.cocos2d-iphone.org

- integrirani učinki prehodov med pogledi
- možnost uporabe učinkov nad objekti
- integriran fizikalni sistem
- sistem delcev (angl. particle system), ki omogoča kreiranje učinkov ognja, sledi (npr. žoga pušča za sabo sled), dima (slika 4.2).
- dodatna zvočna podpora
- podprta oba načina uporabe telefona (ležeče in pokončno)



Slika 4.1: Primer učinka ognja s pomočjo sistema delcev

4.5 Zaključek

V tem poglavju smo spoznali osnove značilnosti pogona Quartz 2D, na hitro smo pogledali značilnosti OpenGL ES, ki se večinoma uporablja v 3D okolju, na koncu poglavja so sledile še značilnosti Cocos2D. Obstaja še veliko drugih tehnologij, ki omogočajo izdelavo grafično kompleksnih iger, npr. orodje Unity[3].

[3]http://unity3d.com/unity/

Poglavje 5

Zaključek

Industrija iger predstavlja velik delež svetovne industrije. Leta 2008 je bila njena vrednost 22 milijard ameriških dolarjev. Po nekaterih analizah je njena letna rast 22,9%. Govorimo torej o eni izmed svetovno pomembnejših industrij. Podjetje Apple Inc. je leta 2007 z izdajo prve generacije telefona iPhone in dodatno uvedbo spletne trgovine aplikacij (angl. App Store) postalo eno izmed vplivnejših podjetij na trgu prodaje telefonov. Bili so prvi, druga podjetja so jim sledila. Po zadnjih podatkih je število aplikacij, ki so na voljo za telefon iPhone, preraslo številko 225.000 in še raste. Zabeležili so več kot 5.000.000 prenosov. Vsi ti podatki vzpodbujajo tako posamezne razvijalce, kot velika podjetja, npr. Electronic Arts[1] ali Ubisoft[2] za razvoj aplikacij.

V diplomskem delu so predstavljena dejstva telefona iPhone, tako na področju strojne kot programske opreme. Posebna pozornost je namenjena zaslonu na dotik, ki je revolucioniral uporabniški vmesnik telefonov nasploh. Lahko bi rekli, da je prisilil druga podjetja v uvajanje zaslonov na dotik, ker je predstavljal izboljšan način interakcije elektronske naprave in uporabnika. Tipkovnica postane nepotrebna in se prikaže samo takrat, ko jo uporabnik potrebuje. Ne smemo pozabiti na operacijski sistem iOS in razvojna orodja, ki omogočajo aplikacijam tekoče delovanje in poenostavljen razvoj. Delovanje aplikacij ni odvisno le od operacijskega sistema in razvojnega okolja, ampak dodatno od razvijalcev, ki so primorani upoštevati določene omejitve naprave.

V drugem poglavju smo torej spoznali vse osnovne značilnosti telefona iPhone, ter tako v tretjem začeli razvijati aplikacijo, igro *Space Pong*. Seveda je potrebno poznati način, kako se lotiti dela. Odločilnega pomena na tej točki sta znanje o procesu razvoja igre in poznanje sestavnih delov igre. Do-

[1] http://www.ea.com/

[2] http://www.ubi.com/UK/default.aspx

bro je poznati cevovodni razvoj igre in poskušati optimizirati lasten proces razvoja. V začetni fazi je najtežje spisati dokument opisov sestavnih delov igre: zamisel scenarija, tematike, zvrsti, realizacijo uporabniškega vmesnika in strukturo navigacije igre. Po tej fazi je nekoliko lažje, idejo imamo, potrebna je implementacija le-te. Sicer ne smemo mimo dejstva, da se lahko produkcijska faza, kjer idejo realiziramo, zavleče. Zaradi pomanjkanja znanja ali neizkušenosti jo velikokrat prilagajamo. Zavedati se moramo, da je začetek vsake produkcijske faze programiranje. Faza, ki po končanem programiranju jedra igre lahko ogrozi projekt, je faza razvoja grafičnih elementov. Lahko si pomagamo s številnimi orodji, od manj zahtevnih do bolj zahtevnih, npr. *Adobe Photoshop CS4*[3], *3D Studio MAX* [4] (namenjen predvsem 3D objektom) ali orodje *Maya* (namenjen prav tako 3D objektom)[5]. Na tem mestu bi želel omeniti, da ni dovolj poznavanje orodij in znanje programiranja. Potrebno je tudi *"umetniška žilica"*. Ob zaključku te faze sledi faza testiranje, ki hkrati predstavlja sestavni del vsake faze. Končna faza testiranja nam omogoča le ugotovitve manjših napak in njihoviho odpravljanje. Ko je igra končana, jo lahko izdamo. Večja razvojna podjetja se na tej točki ponavadi povežejo z določenimi podjetji, specializiranimi za marketing in prodajo iger.

Igra, izdelana v okviru diplomske naloge, z imenom *Space Pong*, predstavlja primerek svetovno uspešne igre *Pong*. Posebnost izdelane igre je vesoljska tematika, izboljšan fizikalni sistem odbojev, umetna inteligenca na osnovi predikcije, uporabniški vmesnik, kjer plošček dejansko držimo s prstom, tri težavnostne stopnje in naključen pojav ovir. Izboljšave so možne na vseh področjih, še posebej grafičnem. Tako bi lahko izboljšali grafično podobo igre s pomočjo napredne tehnologije Cocos2D, izboljšali sam koncept igranja, torej določili zgodbo igri in vsebinske stopnje (angl. levels). Lahko bi imeli npr. 10 vsebinskih stopenj in vsaka bi imela drugo ozadje in druge ovire ter višjo težavnostno stopnjo. Ena izmed zanimivih implementacij bi lahko bila možnost igranja dveh igralcev hkrati. Največja sprememba bi verjetno predstavljala pretvorbo celotne igre v okolje 3D.

Ob vseh izboljšavah ne smemo pozabiti osnovnega poslanstva in razloga, zakaj smo se za igro odločili. Namen je bil spoznati nova razvojna okolja, delovanje telefona iPhone in izboljšati igre zvrsti *Pong*.

Naj zaključim s citatom znanega razvijalca iger Sida Meierja: *"Igra je določeno število pomembnih in zanimivih igralčevih odločitev, ki zasledujejo čist in utemeljen cilj"* [13].

[3]http://www.adobe.com/
[4]http://usa.autodesk.com/
[5]http://usa.autodesk.com/

Dodatek A

Igra *Space Pong*

A.1 Opis in značilnosti

V okviru diplomske naloge je bila izdelana igra z imenom *Space Pong*, ki deluje na telefonu iPhone. Igralec s pomočjo ploščka, ki ga vodi s prstom, poskuša odbiti žogo nasprotniku tako hitro, da je ta ne more udariti nazaj. Na ta način pride do točke. Kdor prvi doseže 5 točk, zmaga. Za bolj zanimiv potek igre se naključno prikazujeta oviri v obliki dodatne stene.

Osnovne značilnosti igre:

- implementiran fizikalni sistem odbojev
- umetna inteligenca na osnovi predikcije
- dodani grafični elementi (razvita slika ozadja, slika žoge in ploščka, učinek sija)
- navigacijska struktura (meni, nastavitve, pogled s podatki o avtorju in izvajanje igre)
- implementirana ustrezna pisava glede na vesoljsko tematiko igre

A.2 Navodila za uporabo igre

Ob zagonu igre se prikaže pozdravno okno (predstavitev igre), nato se samodejno spremeni pogled na meni. V meniju lahko uporabnik izbere novo igro, nastavitve iali si pogleda podatke o avtorju (slika A.3). Slednja možnost ponuja

tudi aktivni spletni naslov skupine *GameTeam*, ki deluje pod okriljem Fakultete za računalništvo in informatiko Univerze v Ljubljani[1]. Ob izbiri nastavitev (angl. settings) se trenutni pogled (slika A.1) spremeni in uporabnik dobi možnost izbire med tremi težavnostnimi stopnjami. Dodatno lahko izbere barvo ploščka (slika A.2).

Z dotikom na gumb nova igra (angl. new game) začnemo z izvajanjem igre (slika A.4). Uporabnik prične igrati tako, da s prstom prime plošček in ga premakne v smeri potovanja žoge. Hitreje kot ga zadene z žogo, hitreje se žoga odbije. Ob doseženi točki se plošček uporabnika pomanjša in hkrati poveča nasprotniku. Ob nasprotnikovi točki se zgodi ravno obratno. Zaradi boljše dinamike igre se naključno pojavljata oviri v obliki stene. Uporabnik ima možnost prekinitve igre kadarkoli, tako da pritisne na gumb v zgornjem desnem kotu.

Slika A.1: Meni igre *Space Pong*

[1] http://gameteam.fri.uni-lj.si/

Slika A.2: Nastavitve igre *Space Pong*

Slika A.3: Podatki o avtorju igre *Space Pong*

Slika A.4: Izvajanje igre *Space Pong*

Slike

Tabele

Literatura

[1] Apple Inc., iPhone OS Reference Library. Dostopno na: http://developer.apple.com/iphone/library/navigation/index.html (datum zadnjega obiska 1.7.2010)

[2] Avsec R., Peer P.: OpenCV in iPhone: detekcija obrazov v realnem času. V: Potočnik B. (ur.). ROSUS 2010 : računalniška obdelava slik in njena uporaba v Sloveniji 2010 : zbornik 5. strokovne konference, Maribor, 18. marec 2010. V Mariboru: Fakulteta za elektrotehniko, računalništvo in informatiko, Inštitut za računalništvo, 2010, str. 131-138. Dostopno na: http://lrv.fri.uni-lj.si/~peterp/publications/rosus10_2.pdf (datum zadnjega obiska: 28.6.2010)

[3] Carter B.: The Game Asset Pipeline. Charles River Media, Boston, ZDA, 2004, strani: 241 - 245

[4] Dalrymple M., Knaster S.: Learn Objective-C on the Mac. Apress, New York, ZDA, 2009

[5] Dumas II J.D.: Computer Architecture, Fundamentals and Principles of Computer Design. CRC Press, Miami, ZDA, 2006, strani: 181 - 183

[6] Goodman D.: Game Tools Tune-Up: Optimize Your Pipeline Through Usability. Dostopno na: http://www.gamasutra.com/view/feature/4016/ (datum zadnjega obiska: 25.6.2010)

[7] Honan M.: Apple unveils iPhone. Dostopno na: http://www.macworld.com/article/54769/2007/01/iphone.html (datum zadnjega obiska: 25.6.2010)

[8] Jacobs G, Schulman J.: AP Physics B&C. McGraw-Hill, New York, ZDA, 2010, strani: 136 - 137

[9] Mark D.: Learn C on the Mac. Apress, New York, ZDA, 2009, strani: 34 - 37, 121 - 161, 207 - 249, 283 - 317

[10] Mark D., LaMarche J.: iPhone 3 Development, Exploring the iPhone SDK. Apress, New York, ZDA, 2009

[11] Mark D., Carbera PJ: iPhone Games Projects. Apress, New York, ZDA, 2009, strani: 131 - 155, 191 - 202

[12] McShaffry M.: Game Coding Complete, Third Edition. Charles River Media, Boston, ZDA, 2009 strani: 21 - 44, 50 - 56, 88 - 89, 169 - 172, 623 - 654

[13] Rabin S.: Introduction to Game Development. Charles River Media, Boston, ZDA, 2005, strani: 3 - 51, 69 - 99, 335 - 393, 441 - 443

[14] Stallings W.: Operating Systems: Internals and Design Principles, Sixth Edition.
Upper Saddle River, New Jersey, ZDA, 2009, strani: 345 - 383

[15] Wilson T.V.: How the iPhone Works. Dostopno na: http://electronics.howstuffworks.com/iphone3.htm (datum zadnjega obiska: 20.6.2010)