UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Matej Bukovinski

Naravni uporabniški vmesniki za večdotične naprave

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: prof. dr. Matija Marolt

Ljubljana, 2010

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: 01694/2010
Datum: 01.09.2010

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

Kandidat: **MATEJ BUKOVINSKI**

Naslov: **NARAVNI UPORABNIŠKI VMESNIKI ZA VEČDOTIČNE NAPRAVE**
NATURAL USER INTERFACES FOR MULTITOUCH DEVICES

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

V diplomski nalogi preučite značilnosti naravnih uporabniških vmesnikov in se osredotočite na naravne uporabniške vmesnike za večdotične naprave. Preučite trenutno stanje na področju večdotičnih tehnologij in izvedite študijo uporabniških vmesnikov za tovrstne naprave. Na podlagi ugotovitev kot primer dobre prakse razvijte večdotično podatkovno aplikacijo.

Mentor:

doc. dr. Matija Marolt

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Besedilo je 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 Matej Bukovinski,

z vpisno številko 63050016,

sem avtor diplomskega dela z naslovom:

Naravni uporabniški vmesniki za večdotične naprave

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom
  prof. dr. Matije Marolta
- 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 10.12.2010 Podpis avtorja:

Zahvala

Zahvaljujem se mentorju prof. dr. Matiji Maroltu za vodenje in pomoč pri izdelavi diplomske naloge. As. Cirilu Bohaku se zahvaljujem za pomoč pri iskanju literature in predlagane korekture. Nenazadnje bi se rad zahvalil staršem, ki so mi omogočili študij.

Kazalo

Seznam uporabljenih kratic in simbolov

HCI - ang. *human–computer interaction*, interakcija človeka in računalnika.

NUI - ang. *natural user interface*, naravni uporabniški vmesnik.

CLI - ang. *command line interface*, vmesniki z ukazno vrstico.

GUI - ang. *graphic user interface*, grafični uporabniški vmesnik.

WIMP - ang. *window, icon, menu, pointing device*, okna, ikone, meniji in sledilne naprave.

CRT - ang. *cathode ray tube*, katodni zaslon.

FTIR - ang. *frustrated total internal reflection*, metoda na osnovi onemogočanja popolnega notranjega odboja.

DI - ang. *diffused illumination*, metoda na osnovi difuzne osvetlitve.

LLP - ang. *laser light plane illumination*, metoda na osnovi laserske svetlobne površine.

DSI - ang. *diffused surface illumination*, metoda na osnovi difuzne površinske osvetlitve.

LCD - ang. *liquid crystal display*, zaslon s tekočimi kristali

API - ang. *application programming interface*, programski vmesnik

GPS - ang. *global positioning system*, globalni pozicijski sistem

ARM - ang. *advanced RISC machine*, vrsta procesorske arhitekture

UMTS - ang. *universal mobile telecommunications system* - univerzalni sistem za mobilno komunikacijo

HDPA - ang. *high-speed downlink packet access* - visoko hitrostni paketni prenos podatkov

REST - ang. *representational state transfer* - programska arhitektura za distribuirane hipermedijske sisteme

HTTP - ang. *hypertext transfer protocol* - protokol za prenos hiperteksta

MVC - ang. *vodel–view–controller* - programska arhitektura modela, pogleda in kontrolerja

URL - ang. *uniform resource rocator* - enotni lokator virov

Izrazi so prevedeni v slovenščino s pomočjo iSlovarja (http://www.islovar.org), računalniškega slovarčka (http://dis-slovarcek.ijs.si), slovarčka krajšav (http://bos.zrc-sazu.si/kratice.html) ali druge slovenske strokovne literature.

Povzetek

Diplomska naloga predstavlja novo vrsto uporabniških vmesnikov, ki so v stroki dobili poimenovanje *naravni uporabniški vmesniki*. Podane so njihove poglavitne značilnosti, zgodovinski razvoj, ter prednosti pred trenutno najbolj razširjenimi grafičnimi uporabniškimi vmesniki. Osrednja pozornost je namenjena podskupini naravnih uporabniških vmesnikov za večdotične naprave.

V okviru diplomske naloge je podrobneje predstavljena večdotična tehnologija, najprej iz tehnološkega vidika, nato pa še praktično, v obliki primerjave šestih popularnih večdotičnih platform. Najpodrobneje je predstavljena platforma iOS, ki omogoča razvoj aplikacij za naprave iPhone, iPod Touch ter iPad.

Osrednji del naloge obravnava značilnosti večdotičnih uporabniških vmesnikov. Prikazane so njihove specifike ter povzete poglavitne ugotovitve in napotki iz strokovne literature ter znanstvenih raziskav. Teoretične ugotovitve so dodatno ilustrirane s študijo primerov petih značilnih večdotičnih vmesnikov.

V zadnjem delu je najprej predstavljeno omrežje Twitter, nato pa še iPad aplikacija *Corkboard*, večdotični Twitter odjemalec, ki je bil razvit v okviru te naloge. Aplikacija Corkboard služi kot primer vključitve konceptov večdotičnih vmesnikov v podatkovno usmerjene aplikacije ter podrobneje predstavlja načrtovanje in razvoj večdotičnih aplikacij.

V samem zaključku so še enkrat povzete poglavitne ugotovitve ter navedene možne smernice za nadaljnje delo.

Ključne besede:

uporabniški vmesnik, naravni uporabniški vmesnik, večdotična tehnologija, Twitter, Corkboard

Abstract

This thesis presents a new class of user interfaces, which a commonly referred to as *natural user interfaces*. It discusses their main characteristics, evolution and advantages over currently dominant graphical user interfaces. Special attention is devoted to the subgroup of natural user interfaces for multitouch devices.

Multitouch technology is firstly presented from a technical point of view and afterwards also in practice in form of a comparative study of six popular multitouch platforms. Most attention is devoted to the iOS platform, which facilitates development for the iPhone, iPod Touch and iPad range of devices.

The central part of this work discusses the characteristics of multitouch user interfaces. This section states the specific concepts of multitouch interfaces and summarizes key findings and directives from specialist literature and scientific research. Theoretical findings are additionally illustrated with a case study of five characteristic multitouch applications.

The closing sections firstly present the social network Twitter. Subsequently we introduce an iPad application called *Corkboard*, a multitouch Twitter client that was developed in the scope of this thesis. By utilizing the Corkboard application, we present a possible approach for integrating multitouch user interface concepts into a data driven application and demonstrate the design and development process of multitouch applications.

We conclude with a summary of principal findings and an overview of directions for possible future work.

Key words:

user interface, natural user interface, multitouch, Twitter, Corkboard

Poglavje 1

Uvod

Študija uporabniških vmesnikov dobiva, z vse večjo vključitvijo tehnologije v človekov vsakdan, vedno večji pomen. Medtem, ko je zgodnje računalniške sisteme lahko uporabljala le peščica usposobljenih strokovnjakov, morajo današnji uporabniški vmesniki zagotoviti enostavno uporabo neprimerno širši množici ljudi. Težnja po izdelavi vse bolj uporabniku prijaznih uporabniških vmesnikov je nedavno privedla do definicije nove oblike uporabniških vmesnikov, t. i. naravnih uporabniških vmesnikov. Ti naj bi pri interakciji izkoriščali človekove naravne - prirojene in naučene sposobnosti, ter tako zmanjšali miselni razkol med napravo in človekom.

Naravno interakcijo lahko dosežemo na mnogo različnih načinov. V zadnjih letih doživlja svoj razcvet predvsem uporaba naprav, ki omogočajo neposredno interakcijo z uporabo več prstov. Naprave, za katere pravimo, da izkoriščajo večdotično tehnologijo, prikazujejo svoj uporabniški vmesnik na na dotik občutljivem zaslonu, ter tako omogočajo neposredno manipulacijo s prikazanimi virtualnimi elementi. Najznamenitejše naprave te vrste so gotovo mobilni telefoni iPhone podjetja Apple, vse bolj pa se uveljavljajo tudi izdelki drugih proizvajalcev.

Vmesniki za večdotične naprave zahtevajo bistveno drugačno obravnavo, kot klasični grafični uporabniški vmesniki. Pri načrtovanju grafičnih vmesnikov se je skozi leta ustalila množica principov in napotkov, ki pa za izdelavo kvalitetnega večdotičnega vmesnika ne zadostuje. Obstoječi principi namreč ne upoštevajo pomen specifičnih značilnosti večdotičnih naprav, ter fiziološke značilnosti pri interakciji, ki bazira na uporabi prstov. Področje večdotičnih uporabniških vmesnikov je zaradi tega zanimivo tako iz znanstvenega, kot komercialnega vidika.

Največji vir informacij na področju večdotične tehnologije in naravnih upo-

rabniških vmesnikov predstavlja skupnost NUI Group [1], ki poleg lastnih publikacij [2] ponuja tudi obsežno zbirko referenc do povezane literature. Velik doprinos pri študiji večdotičnih vmesnikov imajo tudi raziskovalni laboratoriji podjetij, kot so Microsoft [3, 4, 5, 6], Mitsubishi [7, 8] in Apple [9, 10].

Diplomska naloga je sestavljena iz osmih poglavij. V naslednjem poglavju (2) bomo najprej natančneje spoznali pojem naravnih uporabniških vmesnikov, ter si ogledali njihove poglavitne značilnosti. Da bi bolje razumeli, kako je prišlo do njihovega nastanka, si bomo na kratko ogledali tudi zgodovino računalniških uporabniških vmesnikov.

Sledi pregled večdotične tehnologije, ki je povzet v okviru poglavja 3. Poleg opisa zgodovinskega razvoja bomo spoznali tudi strojne in programske tehnologije, ki so potrebne za prepoznavanje več hkratnih dotikov. Na kratko si bomo ogledali tudi temeljne principe, ki omogočajo več uporabnikom sočasno uporabo večdotičnih naprav. Razvijalcem je pri razvoju večdotičnih aplikacij na voljo množica različnih platform, ki pa se med sabo precej razlikujejo. Da bi nekoliko olajšali izbiro najprimernejše, je v zaključnem delu poglavja 3 pripravljena primerjava nekaterih popularnih platform. Platformi iOS, ki je služila kot osnova za izdelavo lastne aplikacije, je posvečeno nekoliko več pozornosti v okviru samostojnega poglavja 4.

Specifične zakonitosti uporabniških vmesnikov večdotičnih aplikacij si bomo ogledali v poglavju 5. Le-te bodo najprej obravnavane iz teoretičnega vidika, nato pa še v obliki študije primerov petih aplikacij.

Poglavji 6 in 7 sta posvečeni večdotični aplikaciji Corkboard, ki je bila razvita v okviru diplomske naloge. V poglavju 6 si bomo najprej ogledali omrežje Twitter, ki služi kot vir podatkov za aplikacijo, v poglavju 7 pa bomo nadaljevali s predstavitvijo same aplikacije, njene zgradbe, ter funkcionalnosti.

Zaključek diplomske naloge predstavlja poglavje 8, v katerem bodo povzete poglavitne ugotovitve ter navedene možnosti za nadaljnje delo.

Poglavje 2

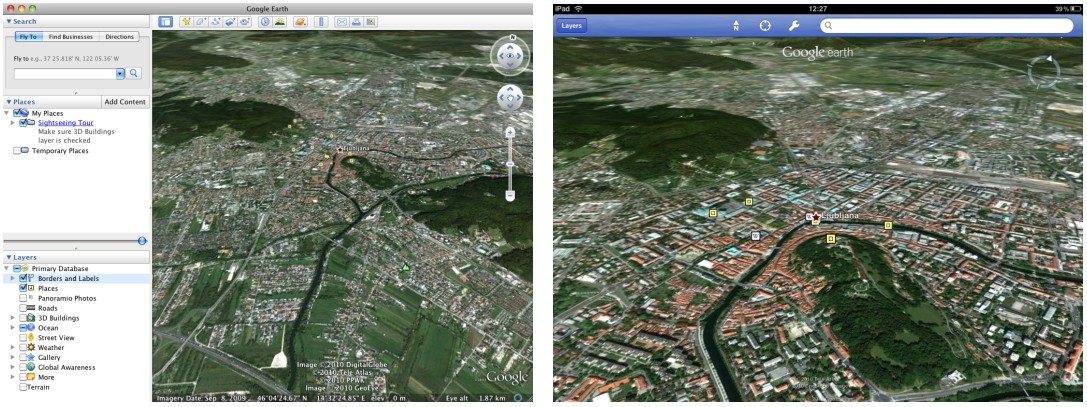
Naravni uporabniški vmesniki

Tehnološkemu napredku na področju računalniških sistemov pogosto sledi napredek pri temeljnih konceptih uporabniških vmesnikov [11]. Zmogljivejši računalniški sistemi vzpodbujajo kreacijo vizualno bogatejših uporabniških vmesnikov, novi tipi vhodno izhodnih naprav pa omogočajo nove pristope za interakcijo človeka in računalnika (*ang. human computer interaction, HCI*). Ravno sedaj smo na pragu takšnega miselnega preskoka, saj klasične grafične uporabniške vmesnike pričenjajo nadomeščati naravni uporabniški vmesniki.

2.1 Definicija in karakteristike

Pojem naravni uporabniški vmesnik (*ang. natural user interface, NUI*) označuje uporabniški vmesnik, ki je za uporabnika neviden, ali postane neviden skozi postopoma naučene interakcije [1, 12]. Pri tem se beseda "naravni" nanaša predvsem na način interakcije med človekom in napravo. Medtem ko klasični uporabniški vmesniki zahtevajo uporabo dodatnih pripomočkov, naravni uporabniški vmesniki predvidevajo bolj neposredno interakcijo z uporabnikom. Naravni vmesniki bazirajo na uporabi človekovih prirojenih in naučenih sposobnostih, kot so dotik, vid, govor, kretnje, pisanje in zaznavanje. Uporabnik ugotovi način uporabe postopoma, tako da izvaja za njega relativno naravne geste in opazuje odzive naprave.

Kar se za uporabnika smatra naravno, pa ni odvisno samo od interakcijske tehnologije, ampak tudi od konteksta, kjer se tehnologija uporablja. Za primer vzamemo mobilni telefon in njegovo funkcijo klicanja. Kadar vozimo je primerni način za izbiro telefonske številke govor, medtem ko je pri vožnji na avtobusu bolje (zaradi prisotnosti drugih in hrupa) številko vtipkati s prsti.



(a) Mac OS X (b) iOS (iPad)

Slika 2.1: Aplikacija *Google Earth* za Mac OS X in iPad.

Naravni vmesnik mora tako izrabljati človekove naravne in naučene sposobnosti, kot tudi biti primeren v konteksu uporabe [6].

Naravni uporabniški vmesniki pogosto iščejo inspiracijo v fizičnem svetu. Kljub temu, posnemanje naravnega okolja ni cilj naravnih uporabniških vmesnikov, temveč metoda s katero lahko pogosto dosežemo boljšo uporabniško izkušnjo. S tem ko vzpostavimo korespondenco med elementi uporabniškega vmesnika in fizičnimi predmeti, uporabniku namigujemo možne interakcije, ki jih že pozna iz izkušenj pri fizičnih predmetih. Interakcija med ljudmi in fizičnimi predmeti, ter ljudmi med sabo, tako služi kot model za naravne uporabniške vmesnike.

Končni cilj naravnih uporabniških vmesnikov je skriti tehnološko kompleksnost interakcije človeka in naprave (tipično računalnika) in jo s tem približati nivoju interakcije med ljudmi. Uporaba tehnologije bi s tem postala enostavnejša (tudi za ljudi z minimalnim tehničnim znanjem), bolj intuitivna in zabavna.

Za ilustracijo si lahko ogledamo aplikacijo Google Earth[1] (slika 2.1). Google Earth je na voljo za več različnih platform, med drugim tudi za Mac OS X in iPad. Verzija za Mac OS X (slika 2.1a) je primer običajnega grafičnega uporabniškega vmesnika. Velik del prostora je namenjen prikazu menijev in drugih navigacijskih kontrol. Po 3D prikazu Zemlje navigiramo s kontrolami v levem zgornjem kotu ali s pomočjo miške. Ker je število gest, ki jih lahko opravljamo

[1]Google earth je aplikacija, ki prikazuje 3D virtualni globus Zemlje skupaj z geografskimi meta-podatki. http://www.google.com/earth/index.html

z miško omejeno, je potrebno za določene akcije (npr. sprememba vertikalnega pogleda) hkrati držati določene tipke na tipkovnici. Za razliko ima iPad verzija (slika 2.1b) že mnoge karakteristike naravnega uporabniškega vmesnika. Po 3D svetu navigiramo s pomočjo enega ali več prstov. Vse običajne operacije (premikanje, sprememba pogleda in sprememba povečave) lahko izvajamo neposredno s premiki prstov. Aplikacija pokriva celotno površino zaslona in njen prevladujoči del je namenjen vsebini. Kljub temu, pa na vrhu opazimo elemente klasičnih grafičnih vmesnikov - gumbe in vnosno polje. To je dejansko pogosta praksa pri aplikacijah z naravnimi uporabniškimi vmesniki. Velja namreč, da so za izvajanje določenih akcij (npr. izbira načina delovanja, nastavitve, iskanje, ipd.) takšne komponente primernejša izbira [8, 4].

2.2 Zgodovina uporabniških vmesnikov

Da bi bolje razumeli doprinos naravnih uporabniških vmesnikov, si velja na kratko ogledati zgodovino razvoja uporabniških vmesnikov.

Uporabniški vmesnik predstavlja stično mesto uporabnika (človeka) in naprave. Preko uporabniškega vmesnika uporabnik napravi podaja vhodne podatke in ukaze, naprava pa mu vrača povratne (izhodne) podatke. Izraz "uporabniški vmesnik" se danes najpogosteje uporablja za označevanje stične točke človeka in računalnika (programska in stojna oprema). To je tudi pomen, na katerega se bo nanašala nadaljnja diskusija.

Za zgodnje računalniške sisteme pravimo, da so delovali v t. i. paketnem načinu procesiranja (ang. *batch processing*). Uporabnik je računalniku podal program s preklapljanjem kablov, s stikali ali luknjastimi karticami. Računalnik je nalogo ob določenem času izvršil in vrnil rezultat, ki je bil tipično v tiskani obliki. Takšnim preprostim vmesnikom pravimo paketni vmesniki (ang. *batch interfaces*) [13].

Ravnanje s paketnimi vmesniki je bilo zelo zamudno opravilo. Pogosto je od vnosa programa do sistemskega odgovora minilo več ur ali celo več dni. Z večanjem zmogljivosti računalniških sistemov, se je pojavila potreba po učinkovitejšem načinu interakcije uporabnika in stroja. Ta zahteva skupaj s pojavitvijo novih vhodno-izhodnih naprav, je privedla do modernih uporabniških vmesnikov. Le-te lahko razdelimo v tri kategorije: vmesniki z ukazno vrstico, grafični uporabniški vmesniki in naravni uporabniški vmesniki [14]. Na sliki 2.2 so prikazani štirje uporabniški vmesniki, po eden za vsako od omenjenih obdobij.

(a) (b) (c) (d)

Slika 2.2: Uporabniški vmesniki skozi čas. Luknjač kartic IBM 029 Key Punch (a), terminal VT100 (b), osebni računalnik Xerox Star (c), večdotična miza Microsoft Surface (d). (Slike povzete po `http://www.catb.org/~esr/writings/taouu/html/` in `http://www.microsoft.com/presspass/presskits/surfacecomputing/gallery.mspx`)

2.2.1 Vmesniki z ukazno vrstico

Prvi računalniški vmesniki z ukazno vrstico (ang. *command line interface*) so bili teleprinterji (ang. *teletype*), priklopljeni na osrednje računalnike (ang. *mainframe*). Teleprinterji, ki so bili takrat (petdeseta leta prejšnjega stoletja) že uveljavljena naprava za pošiljanje in sprejemanje telegrafov, so omogočali vnos programa s tipkovnico in bistveno krajši odzivni čas (nekaj sekund). Kasneje so tiskalnike, ki so tiskali izhodne podatke zamenjali s CRT monitorji. To so bili prvi računalniški terminali, ki jih še danes povezujemo z vmesniki z ukazno vrstico.

Interakcija pri vmesnikih z ukazno vrstico bazira na tekstu. Uporabnik prične transakcijo, tako da računalniškemu sistemu poda ukaz v striktni sintaksi, ta pa mu takoj odgovori. Za vmesnike z ukazno vrstico pravimo, da so funkcijsko osredotočeni, saj tipično najprej podamo ukaz (akcijo) in nato objekt, nad katerim se akcija izvrši (npr. datoteka).

Čeprav je vmesnik z ukazno vrstico pomenil velik napredek pri interakciji človeka in računalnika, zahteva uporaba takšnega vmesnika od uporabnika podrobno tehnično znanje (predvsem poznavanje ukazov in sintakse). Kljub kompleksnosti uporabe, so ukazne vrstice še danes pogosto prisotne, saj imajo kar nekaj prednosti pred grafičnimi vmesniki (npr. večja hitrost uporabe za redne uporabnike, manjša poraba sistemskih virov, večja stopnja nadzora, itd.) [15, 16].

2.2.2 Grafični uporabniški vmesniki

Grafični uporabniški vmesniki (*ang. graphic user interface, GUI*), kot jih poznamo danes, so nastali v sedemdesetih letih prejšnjega stoletja, v širšo uporabo pa so prišli v osemdesetih. Zasluge za nastanek in razvoj grafičnih uporabniških vmesnikov ima mnogo različnih posameznikov, podjetij [17, 18] in akademskih ustanov [19]. Predvsem je tu pomembno izpostaviti Douglasa Engelbarta [20], ki velja za pionirja na tem področju in podjetje Xerox PARC, ki je leta 1973 razvilo grafični uporabniški vmesnik za računalnik *Alto* in kasneje *Star* [21]. Za poznejšo komercialno uveljavitev grafičnih vmesnikov sta zaslužni predvsem podjetji Apple Computer (računalnika *Lisa* in *Macintosh*) ter Microsoft (operacijski sistem *Windows*).

Grafični uporabniški vmesniki temeljijo na konceptu oken, ikon, menijev in sledilne naprave (ang. *window, icon, menu and pointing device, WIMP*). Za uporabo zahtevajo grafični zaslon in sledilno napravo (tipično miška), s katero se izvajajo akcije. Grafični uporabniški vmesniki prikazujejo elemente datotečnega sistema v obliki grafičnih simbolov (ikon), aktivne aplikacije pa kot okna. Meniji se uporabljajo za izbiro akcij. Upravljanje bazira na sledilni napravi, ki je na zaslonu prikazana v obliki kazalca (ang. *pointer*).

Če smo vmesnik z ukazno vrstico označili kot funkcijsko osredotočen, je grafični uporabniški vmesnik objektno osredotočen. Pri grafičnemu vmesniku namreč najprej izberemo objekt (npr. datoteko, prikazano z ikono), nad katerim želimo izvajati akcije in nato ukaz iz menija.

O uspešnosti grafičnih uporabniških vmesnikov ni dvoma. Velika večina današnjih uporabniških vmesnikov temelji na tem principu. V primerjavi z vmesniki z ukazno vrstico je glavna prednost grafičnih vmesnikov bistveno enostavnejša uporaba za nepoznavalce [15]. Grafični uporabniški vmesniki temeljijo na prepoznavanju funkcij, namesto na spominu naučenih ukazov [16]. Ta lastnost jih naredi primernejše za občasne uporabnike in pripomore k položnejši učni krivulji pri učenju uporabe vmesnika.

2.2.3 Naravni uporabniški vmesniki

Grafični uporabniški vmesniki so v uporabi že več desetletij. Kljub stalnemu napredku, pa lahko ugotovimo, da se osnovni koncepti grafičnih vmesnikov od osemdesetih let niso bistveno spremenili. Principi naravnih uporabniških vmesnikov, ki smo si jih ogledali v 2.1, se od klasičnih grafičnih uporabniških vmesnikov dovolj razlikujejo, da jih je vredno obravnavati kot novo obliko uporabniških vmesnikov. Kljub temu pa naravni uporabniški vmesniki ne bodo

nadomestili grafičnih uporabniških vmesnikov, podobno kot slednji niso povsem nadomestili vmesnikov z ukazno vrstico. Za določena opravila (npr. pisanje članka) je grafični uporabniški vmesnik tisti, ki ga večina smatra kot naravni oziroma najučinkovitejši. Naravni uporabniški vmesniki so dopolnilo, ki pokriva področja, kjer so se obstoječi uporabniški vmesniki izkazali za manj primerne.

Podobno kot je veljalo za grafične uporabniške vmesnike, sta tudi za komercialno uveljavitev naravnih uporabniških vmesnikov zaslužni predvsem podjetji Microsoft in Apple s svojimi večdotičnimi napravami. Večdotično tehnologijo bomo natančneje pregledali v naslednjem poglavju (3), izdelke podjetja Apple pa v poglavju 4.

Poglavje 3

Večdotična tehnologija

V prejšnjem poglavju smo ugotovili, da lahko izboljšamo interakcijo človeka in naprave tako, da izkoristimo človekove prirojene sposobnosti. V tem pogledu je eden izmed osnovnih in najbolj uporabljenih principov vpeljava dotika.

S pojmom večdotična tehnologija (ang. *multitouch*) označujemo množico interakcijskih pristopov in naprav, ki omogočajo uporabniku, da upravlja z grafičnimi aplikacijami s pomočjo več prstov [2]. Naprave, ki omogočajo večdotično interakcijo so v splošnem sestavljena iz dveh delov - strojnega in programskega. Strojna oprema zaznava dotike prstov na svoji površini, ter jih v obliki zaznanih *blobov* (regije na sliki, ki so bodisi svetlejše bodisi temnejše od okolice) posreduje programski opremi. Programski del nato na osnovi surovih podatkov iz strojne opreme vrši identifikacijo in sledenje posameznih prstov, ter zaznavanje višje-nivojskih gest.

V primerjavi s tehnologijo za zaznavanje enega dotika, je bistvena prednost večdotične tehnologije večja izrazna moč. Z več prsti lahko izvajamo geste, ki so z enim samim prstom nemogoče. Pri uporabi ene stične točke smo obsojeni na enake interakcije, kot pri uporabi miške (zaradi pomanjkanja gumbov je nabor celo manjši). Večdotična tehnologija v tem pogledu ponuja več. Z njo lahko tvorimo naravne uporabniške vmesnike, ki uporabniku ponujajo boljšo uporabniško izkušnjo. Tehnologija pa se ne omejuje samo na uporabo prstov ene roke. Današnje večdotične naprave lahko zaznavajo tudi do več deset prstov hkrati, kar na eni napravi omogoča hkratne interakcije več uporabnikov.

3.1 Zgodovina

Naprave, ki omogočajo zaznavanje več sočasnih dotikov, so se v širši javnosti pričele pojavljati v zadnjih nekaj letih, prej pa smo jih bili vajeni predvsem iz

znanstveno-fantastičnih filmov. Mnogi zato sklepajo, da je tehnologija povsem nova, čeprav ima dejansko že skoraj 30 letno zgodovino. Njena predhodnica (tehnologija za zaznavanje enega dotika) sega celo pred čas računalnikov [3].

Začetki večdotične tehnologije segajo v leto 1982, ko so na Univerzi v Torontu izdelali prvo večdotično tablico, namenjeno interakciji z računalnikom [22]. Naprava je delovala na optičnem principu, tako da je kamera pod površino zaznavala dotike prstov na brušenem steklu. Prsti so se na sicer beli površini odražali kot črna področja.

Že takrat so znanstveniki pričeli razmišljati o novih uporabniških vmesnikih, ki bi jih (več)dotična tehnologija omogočala. V teoriji uporabniških vmesnikov je znan članek avtorjev Nakatani in Rohrlich [23], ki opisuje "mehke"" uporabniške vmesnike (ang. *soft machines*). Avtorja v članku opisujeta možnost prikazovanja grafičnih predstavitev fizičnih kontrol na zaslonu, ter njihovega upravljanja s pomočjo dotika. To je bila idejna zasnova, iz katere so se kasneje razvili današnji naravni večdotični vmesniki.

V desetletjih, ki so sledila, se je mnogo institucij in podjetij vključilo v razvoj večdotične tehnologije. Izdelanih je bilo več sistemov, ki pa so bili dragi in pretežno omejeni na uporabo znotraj laboratorijev, kjer so bili razviti. Od zgodnjih dosežkov velja omeniti prvi večdotični CRT zaslon, ki so ga razvili v laboratorijih podjetja Bell leta 1984, in prvo kapacitivno večdotično tablico, razvito naslednje leto na Univerzi v Torontu [3].

Širša komercialna uveljavitev večdotične tehnologije se je pričela po letu 2000. Prvi komercialni produkt, ki je uporabljal zgolj večdotično tehnologijo za implementacijo svojega vmesnika, je bila leta 2004 predstavljena glasbena mešalna miza *Lemur* [3]. Leta 2006 je na konferenci TED svojo optično večdotično mizo predstavil Jeff Han [24]. Han je kasneje ustanovil podjetje Perceptive Pixel, ki to tehnologijo danes tudi komercialno trži [25]. Največja prelomnica v komercialnem smislu je bilo leto 2007, ko sta podjetji Apple in Microsoft predstavili svoji paradni večdotični napravi - *iPhone* in *Surface*. Podjetji sta do sedaj objavili še vrsto drugih podobnih naprav (*iPod touch*, *iPad*, *Microsoft Touch Wall*,...). Njunemu zgledu je pozneje sledilo še mnogo drugih proizvajalcev računalniške in mobilne opreme.

3.2 Strojna oprema

Osnovna zahteva za izvajanje večdotične interakcije je naprava, ki je sposobna zaznavati dotike in jih pretvoriti v računalniku primerno obliko. Najpogosteje se takšne naprave pojavljajo v obliki na dotik občutljivih zaslonov (ang. *touch*

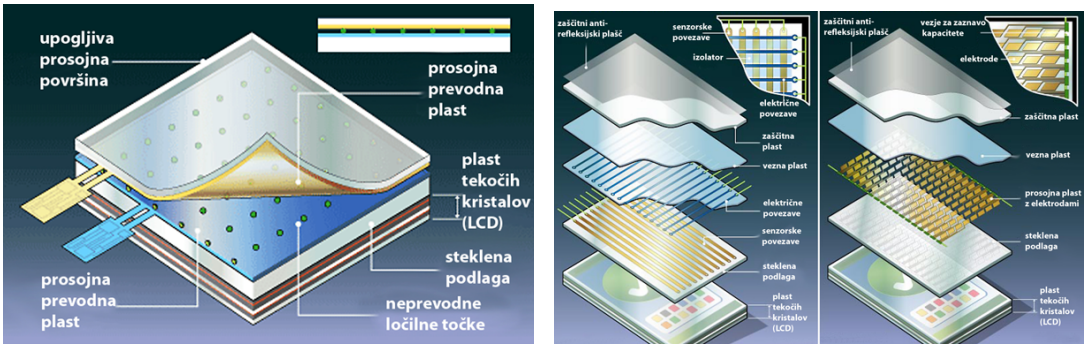
screen) ali sledilnih ploščic (ang. *touch pad*) [3]. Pri prvih gre za vhodno izhodne naprave, ki tako zaznavajo dotike kot prikazujejo sliko, pri slednjih pa zgolj za vhodno napravo. Z vidika naravnih uporabniških vmesnikov so zanimivi predvsem na dotik občutljivi zasloni, saj omogočajo neposredno manipulacijo z virtualnimi elementi, prikazanimi na zaslonu.

Pristopov za zaznavanje dotikov je mnogo. Med drugim lahko temeljijo na zaznavanju pritiska, toplote, elektromagnetne indukcije, infrardeče svetlobe, senc ali nadzvočnega valovanja. Danes se najpogosteje uporabljajo rezistivna, kapacitivna in različne optične tehnologije. Na kratko si oglejmo osnovne značilnosti ter glavne prednosti in pomanjkljivosti omenjenih tehnologij (povzeto po [2, 26, 27, 28]).

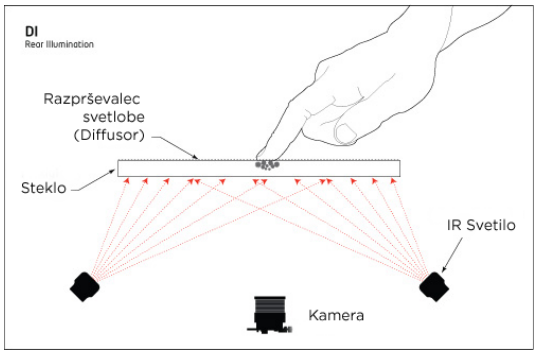
Rezistivni na dotik občutljivi zasloni (ang. *resistive touchscreen*) so sestavljeni iz več plasti, med katerimi sta za delovanje najpomembnejši prevodna plast in rezistivna plast (slika 3.1a). Plasti sta med sabo ločeni z majhnim neprevodnimi točkami. Pri pritisku na zaslon povzročimo stik obeh plasti in s tem spremembo toka, ki teče po prevodni plasti. Na osnovi spremembe toka lahko spremljajoča logika natančno določi mesto dotika. Rezistivna tehnologija je poceni in zaradi tega zelo razširjena. Zaznava lahko tako dotike s prsti kot s poljubnimi drugimi predmeti, poleg tega pa nudi tudi zelo visoko resolucijo. Od slabosti velja izpostaviti relativno slabo prepustnost svetlobe in občutljivost na poškodbe pri uporabi.

Pri kapacitivnih na dotik občutljivih zaslonih (ang. *capacitive touchscreen*) je steklena površina prepletena s prosojno mrežo, katere elementi lahko hranijo električni naboj. Možnih je več variacij te tehnologije (vzajemna kapacitivnost, lastna kapacitivnost [29], slika 3.1b), osnovni princip pa je pri vseh enak. Ko se površine dotaknemo z drugim prevodnikom (npr. s prstom), se del naboja iz površine sprosti. To spremembo zazna elektronika na robovih zaslona, ki z merjenjem količine in distribucije motenj ugotovi lokacijo dotika. Kapacitivni zasloni so zelo odzivni, robustni in prepuščajo več svetlobe kot rezistivni. Tehnologija je sicer dražja od rezistivne, vendar se cene zaradi masovne proizvodnje hitro spuščajo. Glavna slabost je omejitev na zaznavanje zgolj prevodnih snovi.

Pri optičnih tehnologijah poznamo celo vrsto možnih implementacij (*FTIR*, *DI*, *LLP* in *DSI* [2]). Vsem je skupna uporaba infrardeče svetlobe in kamer, ki lahko takšno svetlobo zaznavajo. Za prikazovanje se ponavadi uporabljajo projektorji, pri nekaterih implementacijah pa je možna tudi uporaba LCD matrik. Pogosto uporabljena metoda je metoda na osnovi difuzne osvetlitve (ang. *Diffused Illumination, DI*). Pri tej se na površino iz akrilnega stekla prevlečeno s projekcijskim platnom od zadaj projecirata slika ter infrardeča svetloba (slika



(a) (b)



(c)

Slika 3.1: Shematski prikaz nekaterih tehnologij za zaznavanje dotikov. (Slike povzete po `http://www.mojmikro.si/v_srediscu/tehnologije/tehnologija_zaslonov_multi-touch` in `http://www.touchscreenmagazine.nl/multitouch-techniques/direct-illumination`)

3.1c). Ko se površine dotaknemo s prsti, se del infrardeče svetlobe odbije. Ta odboj zazna infrardeča kamera, ki sliko posreduje programski opremi. Optične metode uporabljajo že obstoječo tehnologijo (infra-rdeča svetila, kamere, projektorji), zato so relativno poceni in popularne za domačo izdelavo. Primerne so predvsem za večje površine (mize, stene), saj optična konstrukcija zahteva veliko prostora.

3.3 Programska oprema

Programski del večdotičnega sistema se konceptualno in praviloma tudi implementacijsko deli na dva dela. Prvi del je proces, ki izvaja sledenje (ang.

tracker), drugi pa aplikacija, ki podatke o položaju prstov uporablja za prepoznavanje gest in izvajanje akcij.

3.3.1 Sledenje

Da bi lahko dotike, zajete na stojni opremi uporabili, jih je najprej potrebno pretvoriti v aplikacijam primerno obliko. Podatke iz večdotične strojne opreme, si lahko pogosto prestavljamo kot zaporedje sivinskih slik, na katerih so dotiki prestavljeni kot področja visoke ali nizke intenzitete (odvisno od implementacije). To velja še posebej za optične metode, ki si jih bomo v tem delu najpodrobneje ogledali.

Naloga procesa za sledenje je, da na posamezni sliki razpozna lokacije prstov, ter jih poveže z znanimi položaji iz prejšnjih slik. Pri nekaterih izvedbah lahko položaje dotikov identificira že sama strojna oprema. V tem primeru se proces za sledenje ustrezno poenostavi. Prepoznane dotike proces za sledenje nato posreduje ciljni aplikaciji v obliki sporočil. Ta sporočila tipično obsegajo vsaj sporočilo o zaznanem novem dotiku (ang. *touch down*), o premiku dotika (ang. *touch moved*) in o prenehanju dotika (ang. *touch ended*).

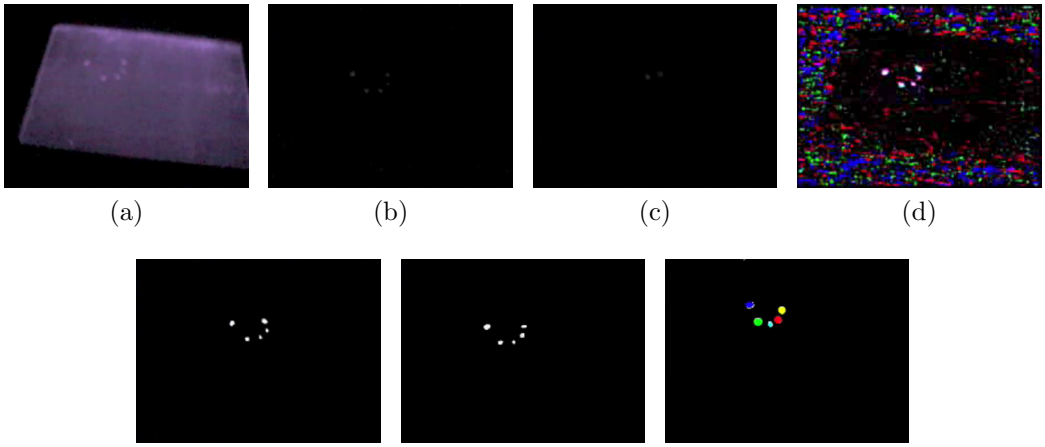
Proces za sledenje je lahko že vgrajen v platformo ali pa je implementiran kot dodatna aplikacija, ki vseskozi teče v ozadju. Na voljo je kar nekaj odprto-kodnih sistemov za sledenje (*Community Core Vision*[1], *Touchlib*[2], *Touché*[3], ipd.). Sledenje poteka tako, da se na vsaki sliki iz sekvence zaporedoma vršijo slikovne operacije, ki v končni fazi pripeljejo do podatkov o dotiku. Za ilustracijo si poglejmo faze pri sledenju s pomočjo sistema Touché [30].

1. Iz vhodne slike (3.2a) najprej odstrani ozadje. To se izvede tako, da se od slike odšteje vnaprej zajeta slika, ki vsebuje samo ozadje.

2. V idealnem primeru sedaj na sliki ostanejo le področja z dotiki in nekaj šuma (3.2b). Šum se zmanjša z glajenjem slike (Gaussov filter).

3. Področja z dotiki so po aplikaciji predhodnih operacij le slabo vidna (3.2c). Ta problem se odpravi s povečavo kontrasta. Kot stranski učinek se pri postopku, ki ga uporablja Touché zelo poveča količina šuma v barvnem spektru (3.2d).

[1]Na voljo na http://ccv.nuigroup.com/
[2]Na voljo na http://nuigroup.com/touchlib/
[3]Na voljo na http://gkaindl.com/software/touche



(a) (b) (c) (d)

(e) (f) (g)

Slika 3.2: Prikaz stopenj pri zaznavanju dotikov s pomočjo sistema Touché. (Povzeto po videu, dostopnem na http://gkaindl.com/software/touche/videos)

4. Šum, ki je bil vpeljan v prejšnjem koraku se odstrani s posebno transformacijo, ki preferira bele točke - dotike, ostale pa zatira. Po tem so dotiki lepo razpoznavni (3.2e).

5. Z aplikacijo praga (ang. *threshold*) se slika spremeni v binarno obliko, na kateri so zaznani dotiki označeni kot področja z drugo vrednostjo kot ozadje (3.2f).

6. Na zaznanih področjih se poiščejo središča, ki predstavljajo lokacijo dotika. Podatki o lokacijah se primerjajo s podatki iz prejšnjih slik, s čimer se zagotovi, da so isti dotiki na različnih slikah enako označeni (3.2g).

Sledenje predmetom je eden izmed osnovnih problemov iz področja računalniškega vida. V splošnem je zanesljiva identifikacija predmetov v zaporedju slik (videu) zelo težek problem. To je še posebej res, če se predmet nahaja pred dinamičnim ozadjem ali je delno zakrit. Za namene sledenja je bilo razvitih mnogo kompleksnih metod, ki uporabljajo adaptivne modele za odstranjevanje motenj iz ozadja, ter matematične metode za boljšo lokalizacijo [2]. Na srečo, se lahko pri sledenju dotikov izognemo mnogim problemom, če uporabljamo dovolj kvalitetno večdotično strojno opremo. Dobra strojna oprema lahko že sama izloči večino motenj iz ozadja (pri tem imajo prednost metode, ki ne temeljijo na optiki). Dodatna poenostavitev izhaja iz dinamičnih lastnosti premikanja prstov. V primeru, da je večdotična oprema sposobna zajemati

podatke z dovolj veliko frekvenco, se prst med dvema slikama le malo premakne. Asociacijo dotikov pri procesu sledenja lahko v tem primeru preprosto vršimo s primerjavo evklidske razdalje (metoda najbližjih sosedov).

Aplikacije za delo z dotiki potrebujejo podatek o njihovem natančnem položaju, vendar prsti takšne natančnosti ne morejo ponuditi. Dotik prsta in večdotične površine se zazna v obliki eliptične površine, katere oblika je med drugim odvisna od tega, kateri prst je bil uporabljen, kako močen je bil pritisk in pod kakšnim kotom je bil dotik izvršen. Enostavna rešitev tega problema je iskanje središč elips, vendar ta ni vedno najboljša. Velja namreč, da se površine praviloma zaznajo pod mestom, kjer uporabnik misli, da je izvršil dotik [31]. Dobri procesi za sledenje upoštevajo takšne lastnosti in uporabljajo hevristične metode za ugotavljanje najprimernejših lokacij.

3.3.2 Aplikacije in geste

Aplikacije pridobijo podatke o dotikih tako, da vključijo knjižnice in implementirajo programske vmesnike (ang. *application programming interface, API*), ki jih sistemi za sledenje narekujejo. Podrobnosti o tem, kako se vrši komunikacija med procesom za sledenje in aplikacijami se med platformami razlikujejo in so pri vgrajenih rešitvah pogosto pred uporabniki skrite. Pri odprto-kodnih rešitvah poteka komunikacija praviloma po principu strežnik - odjemalec. Sistem za sledenje igra vlogo strežnika, ki dogodke po vnaprej določenem protokolu posreduje aplikacijam (odjemalcem). Protokol, ki je za to vrsto komunikacije že postal defacto standard se imenuje TUIO [32]. Po implementaciji metod oz. funkcij, ki jih določajo programski vmesniki, lahko aplikacija izvaja akcije na osnovi dogodkov, ki jih tvori proces za sledenje.

Pogosto aplikacije ne zanimajo zgolj goli podatki o dotikih, temveč znanje o gestah, ki jih uporabniki vršijo na večdotični strojni opremi. Gesta (ang. *gesture*) je formalno opredeljena kot "gib, navadno z rokami, s katerim se kaj izraža ali poudarja" [33]. V splošnem gre torej za obliko neverbalne komunikacije z deli telesa. V kontekstu večdotične tehnologije ima pojem nekoliko ožji pomen. Odraža kretnjo (oz. zaporedje kretenj) prstov, ki ima znotraj aplikacije določen pomen oz. povzroči izvedbo akcije. Primer pogosto uporabljene geste je kretnja "povečaj - pomanjšaj" (ang. *pinch*), kjer dva prsta premikamo narazen ali skupaj, da bi dosegli povečavo ali pomanjšavo.

Geste so povezane z akcijami znotraj aplikacij. Akcije pa niso samo odvisne od vrste geste, temveč tudi od konteksta, v katerem uporabnik gesto izvrši. Tako ima lahko gesta "povečaj - pomanjšaj" znotraj pregledovalnika fotografij povsem drugi pomen, če jo izvršimo na fotografiji (sprememba veli-

kosti fotografije) ali na ozadju (sprememba velikosti površine, na kateri ležijo fotografije). Z izvedbo geste tako ne le določimo akcijo, ki se naj izvrši, temveč tudi objekt, na katerem se naj akcija izvede.

Nekatere večdotične platforme (npr. iOS in Adobe Flash), ponujajo že vgrajene rešitve za prepoznavanje določenih osnovnih gest. Tako se poenostavi relativno kompleksno ravnanje z detekcijo gest in zagotovi, da veljajo enaki kriteriji za detekcijo pri vseh aplikacijah. Kadar takšnih rešitev na platformi ni ali kadar aplikacija definira lastne geste, je potrebno logiko za detekcijo implementirati individualno znotraj aplikacije.

Na mnogih večdotičnih platformah so se sčasoma izoblikovali standardni nabori gest. Za te geste velja, da se znotraj platforme pogosto uporabljajo in da so v vseh aplikacijah povezane z zelo podobnimi akcijami. Uporaba standardnih gest je smiselna, saj povečuje njihovo prepoznavnost in možnost za intuitivno odkrivanje funkcionalnosti s poskušanjem. Velja namreč, da bo uporabnik za izvedbo določene akcije najpogosteje poskusil gesto, ki jo že pozna iz drugih aplikacij. Pogosto uporabljene geste si uporabnik lažje zapomni [8] in smatra kot bolj primerne za izvedbo z njimi povezanih akcij [4]. Standardni nabor gest na platformi iOS si bomo ogledali v poglavju 4.

Geste kljub veliki izrazni moči niso vedno najprimernejši pristop. Sicer si lahko zamislimo aplikacijo, ki bi povsem temeljila na uporabi gest, vendar bi takšna aplikacija gotovo pomenila slabo uporabniško izkušnjo. Za izvajanje vseh funkcij, ki jih aplikacija ponuja, bi potrebovali kompleksen nabor gest, ki pa bi se ga uporabniki le stežka naučili in zapomnili [8]. Ravno to je razlog, da današnji večdotični vmesniki poleg uporabe gest, še vedno posegajo po klasičnih menijih za izvedbo naprednejših operacij.

Geste izhajajo iz človekovega socialnega vedenja, zato absolutnih zakonov za njihovo pravilno uporabo ni. Ne obstajajo geste, ki bi bile "naravne" ali univerzalne za vse uporabnike. Tudi za izvedbo enostavnih operacij (kot je recimo translacija predmeta) je strinjanje uporabnikov glede primerne geste le približno 50% [4]. Tudi to je eden izmed razlogov, zakaj je koristno ponovno uporabljati geste, ki so na platformi že uveljavljene. Število prstov na človeških rokah omogoča zelo velik nabor možnih gest. Na žalost pa so raziskave pokazale, da nevešči uporabniki številu prstov med izvajanjem gest posvečajo le malo pozornosti [4]. Uporabnikom najprijaznejše geste so zato tiste, ki uporabljajo le eden ali dva prsta, ter so robustne na morebitne dodatne dotike. Če vse skupaj povzamemo, lahko rečemo, da so dobre geste tiste, ki so enostavne, ponovno uporabljive, robustne glede števila prstov in imajo standardiziran pomen znotraj platforme.

3.4 Večuporabniška uporaba

Tradicionalni namizni računalniški sistemi, ki bazirajo na miški in tipkovnici, so bili od nekdaj namenjeni uporabi enemu samemu uporabniku. S tem, ko računalniki dobivajo vse večjo vlogo v družbenem življenju, pa se je pojavila tudi potreba po sodelovanju več uporabnikov na isti napravi. Posplošitev tradicionalnih sistemov v obliki več mišk, se je za te namene izkazala kot zgrešeno in nenaravno, saj uporabnikom povzroča zmedo in zahteva preveč medsebojne koordinacije [7].

Večdotične naprave so na drugi strani same po sebi večuporabniško naravnane. Razlikujejo lahko med več dotiki na svoji površini, pri tem pa ni pomembno ali so vir dotikov prsti iste roke ali celo iste osebe. Ta osnovna lastnost omogoča, da lahko isto večdotično napravo hkrati uporablja več ljudi. Večuporabniška uporaba je najprimernejša pri večjih večdotičnih napravah v obliki miz ali sten. Uporaba takšnih naprav se zdi ljudem naravna, saj izkoriščajo naravni občutek dotika ter obstoječe socialne navade ljudi, kot je delo za skupno mizo ali tablo.

Ne-razlikovanje uporabnikov, ki tako enostavno omogoča večuporabniško uporabo, pa je na drugi strani tudi največja hiba večdotičnih naprav, kadar jih uporabljamo v večuporabniških okoljih. Skoraj vse današnje večdotične tehnologije (med drugim tudi tiste, opisane v podpoglavju 3.2) aplikacijam ne ponujajo podatkov, ki bi dotike enolično dodelil enemu izmed uporabnikov. Aplikacije tako na primer niso sposobne razlikovati, ali dva zaznana dotika pomenita dvoprstni dotik enega uporabnika ali enoprstna dotika dveh različnih uporabnikov. To dejstvo ima pomembne implikacije na nabor večuporabniških aplikacij, ki jih na takih napravah lahko poganjamo. Na srečo za nekatere aplikacije podatek o uporabniku ni pomemben. Primer take aplikacije je pregledovalnik slik, kjer je načeloma vseeno ali eden uporabnik premika dve sliki ali dva uporabnika vsak svojo. Na drugi strani pa obstaja tudi cela množica aplikacij, ki brez podatka o identiteti uporabnika ne morajo delovati. Trivialen primer je že program za risanje, kjer bi vsak od uporabnikov rad povlekel črto s svojo barvo. To je nemogoče, saj aplikacija ob zaznanem dotiku ne mora vedeti katero barvo naj uporabi.

Za problem identifikacije uporabnika obstajajo tudi rešitve. Ena izmed njih je sistem *DiamondTouch*, ki so ga razvili pri Mitsubishi Electric Research Laboratories [7]. Miza DiamondTouch ustvarja v svoji notranjosti lokacijsko odvisna električna polja, ki se ob dotiku prenesejo na uporabnika. V okolici uporabnika (tipično na stolih) so nameščeni sprejemniki, ki prenesen signal zaznajo in ustrezno obvestijo sledilni proces. Ta pristop omogoča relativno

robustno identifikacijo do štirih hkratnih uporabnikov. Poleg sistema DiamondTouch obstajajo tudi drugi pristopi, ki skušajo omogočiti identifikacijo na obstoječi opremi. Ti najpogosteje predvidevajo uporabo dodatne kamere, ki prepoznava lokacijo rok, ali programske rešitve, ki temeljijo na lokacijsko naravnanih hevristikah ali ločenih uporabniških področjih [34].

S problemom identifikacije je tudi povezan problem avtentikacije. Pri klasičnih namiznih sistemih pridobi uporabnik po uspešni prijavi dostop do celotnega zaslona. Pri večuporabniških sistemih, pa to ni vedno zaželeno, saj imajo hkratni uporabniki lahko povsem drugačne dostopne pravice. Reševanje avtentikacijskih problemov je brez robustne identifikacije zelo težavno [35].

Pri skupni uporabi večdotičnih naprav so neizogibni konflikti. Primer je lahko skupna uporaba virtualne karte, kjer bi v nekem trenutku vsak uporabnik rad mapo premaknil v drugo smer. Takšna interakcija bo sistem, ki ne ločuje med uporabniki, verjetno zmedla ali povzročila registracijo povsem druge geste (npr. povečaj - pomanjšaj). Pristopov za reševanje konfliktov je lahko več. Njihov vpliv lahko poskušamo zmanjšati z vpeljavo dodatnih virtualnih kontrol, ki jih lahko uporablja le eden uporabnik hkrati [36]. Alternativni pristop je, da konfliktov striktno ne preprečujemo, temveč predvidimo njihovo reševanje s socialnim dialogom med uporabniki [37].

Dodaten izziv pri večuporabniških sistemih, še posebej pri tistih v obliki mize, je orientacija posameznega uporabnika. Večdotične mize, kot je Microsoft Surface, nimajo privzete orientacije, zato uporabniki na njih vršijo interakcije iz različnih strani. Reševanje tega problema je praviloma prepuščeno aplikacijskem nivoju in je odvisno od vrste posamezne aplikacije. Tipično aplikacije za večdotične mize ponujajo opcije (geste) za rotacijo predmetov, s katerimi je možno stopiti v interakcijo. Možen je tudi pristop, kjer se objekti vedno samodejno orientirajo tako, da so usmerjeni stran od središča površine. Primer slednjega principa je Microsoftov koncept *Sphere* [5].

Kot je razvidno iz zgornje diskusije, nudi področje večuporabniških večdotičnih naprav mnoge zanimive izzive. Zaradi tega ni presenetljivo, da predstavlja njihovo reševanje eno izmed najaktivnejših raziskovalnih področij pri večdotični tehnologiji.

3.5 Platforme

Mnoge računalniške platforme imajo danes že vgrajeno podporo za večdotično tehnologijo. Te platforme ob prisotnosti primerne strojne opreme že same podpirajo ravnanje z večdotičnimi dogodki, brez potrebe po dodatnih sistemih za

sledenje. Večdotična podpora je na voljo na nekaterih namiznih operacijskih sistemih, mobilnih platformah in splošnih aplikacijskih okoljih. Pri namiznih operacijskih sistemih prednjači *Windows 7*, večdotično podporo pa imajo tudi določene distribucije *Linux* in *Mac OS X*. Pri mobilnih platformah ima odlično večdotično podporo predvsem Applov operacijski sistem *iOS*, prisotna pa je tudi pri nekaterih drugih, kot so *Andoid*, *webOS* in *Sybian*. Večdotična podpora se pričenja pojavljati tudi na svetovnem spletu v obliki razširitev jezika *Java script* ter z *Adobe Flash* predvajalnikom.

Nivo podpore večdotični interakciji se zelo razlikuje med posameznimi platformami. Na kratko si oglejmo, kakšno podporo nudijo nekatere od omenjenih platform, ter kakšne so poglavitne prednosti in pomanjkljivosti posameznih implementacij. Platformo iOS bomo podrobneje obravnavali v poglavju 4.

3.5.1 Windows 7

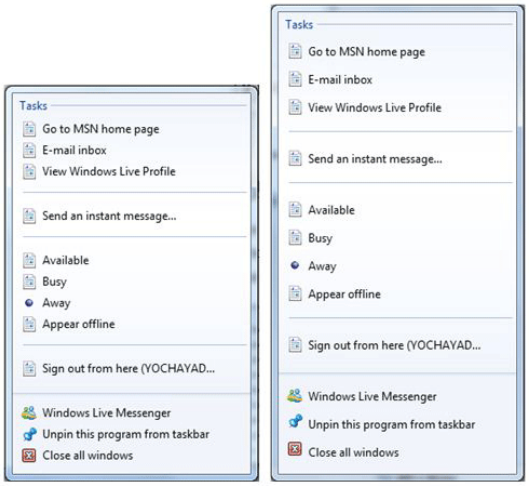
Windows 7 je bil prvi namizni operacijski sistem, ki je vseboval celovito podporo za večdotično tehnologijo. Sistem preko podsistema *Windows Touch Platform* ponuja tako vmesnike za osnovne večdotične dogodke, kot tudi vgrajene mehanizme za prepoznavanje nekaterih gest [38].

Windows 7 predvideva tri nivoje podpore za večdotično interakcijo [39]. Prvi nivo predstavlja osnovno podporo, ki so jo deležne vse obstoječe aplikacije. Pri tem se nekatere večdotične geste prevedejo v običajne dogodke, ki so bili do sedaj vezani na tipkovnico in miško. Tako lahko izvajamo določene geste tudi v aplikacijah, ki sicer niso prilagojene večdotični interakciji. Primer takšnega delovanja je že prej omenjena gesta povečaj - pomanjšaj. Le-ta se interno prevede v dogodek pritiska tipke *control* in premika miškinega kolesa. Če aplikacija to kombinacijo obravnava, bo za njo implementacija geste povečaj - pomanjšaj povsem transparentna.

Drugi nivo predstavljajo aplikacije, ki sistemske geste eksplicitno obravnavajo. Win32 aplikacije v ta namen prejemajo sporočila *WM_GESTURE*, ki jih sistem prevzeto pošilja vsem aplikacijam. Aplikacije lahko pridobivajo podatke o naslednjih sistemskih gestah:

- povečaj - pomanjšaj (ang. *zoom*),
- premik z enim ali dvema prstoma (ang. *single finger or two finger pan*),
- rotacijo (ang. *rotate*),
- dotik z dvema prstoma (ang. *two fingers tap*),
- ter pritisk in dotik (ang. *press and tap*).

Z eksplicitno obravnavo lahko aplikacije dobijo več podatkov o izvedenih gestah (npr. lokacijo središča pri gesti povečaj - pomanjšaj) in tako nudijo boljšo



Slika 3.3: Prilagoditev menijev na Windows 7. Sistem zazna kadar aktiviramo t. i. *Jump List* z dotikom in v tem primeru poveča razmik med menijskimi izbirami. Izbiranje s prsti tako postane bolj zanesljivo. (Slika povzeta po `http://msdn.microsoft.com/en-us/magazine/ee336016.aspx`)

uporabniško izkušnjo. Sistem na tem nivoju ne omogoča hkratnega zaznavanja več gest, zato morajo aplikacije, ki to potrebujejo posegati po lastnih rešitvah.

Aplikacije, ki zahtevajo naprednejšo večdotično interakcijo sodijo v tretji nivo. Takšne aplikacije prejemajo sporočila *WM_TOUCH*, ki vsebujejo podatke o vseh zaznanih dotikih. Sporočila lahko aplikacije neposredno uporabljajo ali na osnovi njih vršijo detekcijo lastnih gest. Za lažjo implementacijo večdotične logike nudi Windows 7 nekatere dodatne programske knjižnice. Koristni sta predvsem knjižnica *Manipulation*, ki lahko na osnovi dotikov tvori dvodimenzionalne transformacijske matrike, in knjižnica *Inertia*, ki nudi animacije na osnovi preproste fizike [40].

Windows Touch Platform lahko s pomočjo ustreznih gonilnikov deluje na celi vrsti večdotičnih naprav. Naprave imajo lahko zelo različne lastnosti, kar se tudi odraža pri sporočilih, ki jih aplikacije prejemajo. Nekatere naprave tako omogočajo zaznavanje največ dveh dotikov, medtem ko jih lahko druge zaznajo po več deset. To so pomembne omejitve, ki jih aplikacije morajo upoštevati v primeru, da želijo ohraniti polno funkcionalnost ne glede na stojno opremo, ki se uporablja v ozadju.

Uporabniški vmesnik sistema Windows 7 se ne razlikuje bistveno, če ga uporabljamo v povezavi z miško ali z večdotično napravo. Posamezni aspekti grafičnega vmesnika so sicer prilagojeni uporabi s prstom (npr. večji razmiki

v nekaterih menijih - slika 3.3, večje aktivne cone pri določenih kontrolah, izboljšana zaslonska tipkovnica, ipd. [40]), vendar bistvenih sprememb ni. To je na eni strani prednost, saj se ohrani poznani videz okolja, na drugi strani pa slabost, saj se nekatere obstoječe kontrole le težko uporabljajo s prsti [41]. Sistemi, ki so glede strojne opreme bolj omejeni (npr. iOS) imajo tu prednost, saj lahko celotni uporabniški vmesnik prilagodijo samo enemu tipu uporabe.

3.5.2 Mac OS X

Za razliko od operacijskega sistema Windows 7, Mac OS X ni namenjen uporabi na večdotičnih zaslonih. Grafični uporabniški vmesnik je zato povsem klasičen in prilagojen uporabi s sledilno napravo (miško oz. drsno tablico). Večdotična podpora je omejena na večdotične tablice, ki jih podjetje Apple ponuja za svoje prenosne in namizne računalnike. Na teh tablicah je predvsem zanimiva izvedba gest, zato so tudi ponujeni programski vmesniki bolj prilagojen detekciji gest, kot obravnavi posameznih dotikov.

Večdotična podpora se je na Mac-u pojavila v prvi posodobitvi sistema *Leopard* (Mac OS X 10.5.1) in je obsegala sistemske vmesnike za omejeni nabor gest. Ti vmesniki so bili v sitemu Leopard nedokumentirani in s tem nedostopni ne-sistemskim aplikacijam .

Vmesniki za sistemske geste so postali dostopni razvijalcem z izdajo sistema *Snow Leopard* (Mac OS X 10.6) [42]. Skupaj z njimi se je pojavila tudi možnost za opcijsko prejemanje podrobnih podatkov o posameznih dotikih. Podpora večdotični tehnologiji je implementirana kot razširitev obstoječih razredov *NSResponder* in *NSEvent*. NSResponder predstavlja razred od katerega dedujejo vse kontrole, s katerimi je možna interakcija. Razred definira množico metod, ki jih podrazredi lahko implementirajo in tako pridobijo podatke o dogodkih. Za izvedbo večdotične podpore je bilo k NSResponder dodano nekaj novih metod, ki jih sistem kliče ob nastopu geste ali dotikov. Vsak dogodek ima pripadajoč objekt tipa NSEvent, ki vsebuje vse za kontrolo relevantne podatke. S prihodom večdotične podpore so bili definirani novi tipi objektov NSEvent, ki vsebujejo podatke o zaznanih gestah in dotikih (*NSTouch*).

Podobno kot pri pri Windows 7, lahko tudi pri Mac OS X aplikacije koristijo določene večdotične geste brez posebnega obravnavanja novih dogodkov. Tako se premik z dvema prstoma (ang. *two finger pan*) v vseh aplikacijah, ki uporabljajo standardne drsne trakove (ang. *scroll bar*) odraža podobno kot premik miškinega kolesa.

Kljub dobrim aplikacijskim vmesnikom, se večdotična interakcija na Mac OS X še ni bistveno uveljavila. Poglavitna razloga za to sta neprilagojen

uporabniški vmesnik ter pomanjkanje strojne opreme (večdotični zasloni), ki bi uporabnikom omogočala neposrednejšo interakcijo.

3.5.3 Android

Mobilni operacijski sistem Google Android v svojih prvih verzijah ni podpiral večdotične interakcije, čeprav so nekatere naprave, na katerih je bil nameščen, to omogočale. Odprto-kodna narava sistema pa je kljub temu omogočila, da so nekateri proizvajalci takšno funkcionalnost dodali kar sami (npr. HTC pri svojem telefonu HERO [43]). Uradna podpora se je v sistemu pojavila z izdajo verzije Android 2.0 [44].

Podpora za večdotične dogodke je na nivoju aplikacijskega vmesnika implementirana kot razširitev obstoječega razreda *MotionEvent* [27]. Aplikacije prejemajo objekte tega tipa v svojih dogodkovnih metodah (ang. *event listener*) kot podane parametre. MotionEvent pri večdotičnih dogodkih vsebuje podatke, kot so število dotikov in njihovo položaj, na osnovi katerih lahko aplikacije gradijo svojo večdotično logiko.

Sistemska podpora za prepoznavanje večdotičnih gest je zelo omejena. Pojavila se je šele v trenutno zadnji verziji sistema (2.2) in obsega le en razred (ScaleGestureDetector), ki ga aplikacije lahko uporabljajo za detekcijo geste povečaj - pomanjšaj [45]. To je tudi edina večdotična gesta, ki se je na sistemu Android širše uveljavila.

Android je močno prilagojen uporabi na mobilnih napravah z na dotik občutljivimi zasloni. Temu primeren je tudi uporabniški vmesnik, ki vsebuje kontrole, ki so povsem prilagojene uporabi s prsti. Na drugi strani pa je podpora za večdotično interakcijo relativno šibka in slabo uveljavljena. Delni razlog za to lahko iščemo v pozni in slabo implementirani uveljavitvi večdotične aplikacijske podpore, ter pomanjkanju naprednejših sistemskih rešitev za prepoznavanje gest [27]. Drugi razlog je povezan z dejstvom, da lahko podobno kot Windows 7, tudi operacijski sistem Android poganjamo na velikem naboru različnih naprav, ki imajo zelo različne sposobnosti glede prepoznavanja več dotikov. Aplikacijski razvijalci se ne morejo zanesti na to, da bo večdotična interakcija povsod možna, zato ji pogosto posvečajo le malo pozornosti.

3.5.4 HTML in Java script

Podpora večdotični tehnologiji v spletnih okoljih je še v povojih. Trenutno je implementirana le znotraj nekaterih spletnih brskalnikov (mobilni Safari na iOS [46], Firefox 4 beta na Windows 7 [47]). Podpora, ki je spletnim aplikaci-

jam na voljo, kot razširitve jezika Java script, je trenutno še povsem nestandardizirana. To pomeni, da mora spletna aplikacija v vsakem brskalniku uporabljati nekoliko drugačne programske vmesnike za dostop do večdotičnih dogodkov. Zaradi omejenega števila ustreznih brskalnikov in razlik med njimi, se je do sedaj pojavilo le izredno malo spletih aplikacij, ki bi izkoriščale večdotično interakcijo. To stanje se bo lahko izboljšalo šele po uveljavitvi univerzalnih programskih vmesnikov in njihovi vključitvi v najbolj razširjene spletne brskalnike.

3.5.5 Flash player

V okolju Adobe Flash player so se programski vmesniki za večdotične dogodke pojavili s trenutno zadnjo verzijo 10.1 [48]. Flash predvajalnik nudi aplikacijam tako podatke o dotikih, kot tudi vgrajeno prepoznavanje določenih gest. Podpora pa je, podobno kot pri Java scriptu, precej odvisna od tega, kje se virtualni stroj izvaja, saj v ozadju uporablja večdotične podatke, ki jih nudi gostujoči operacijski sistem. Popolni nabor funkcionalnosti je na vojo le na Windows 7, medtem ko je na Mac OS X in Windows Mobile podprto le obveščanje o prepoznanih gestah [49]. Predvidena je tudi podpora za iOS, vendar mobilni brskalnik Safari Flash predvajalnika trenutno ne podpira.

Vsa potrebna orodja za aplikacijsko večdotično podporo so zbrana v razredu *Multitouch* [49]. Aplikacije uporabljajo ta razred za preverjanje, ali je večdotična podpora pri trenutni izvedbi aplikacije na voljo. Če preverjanje uspe, lahko z uporabo Multitouch razreda tudi ugotovijo, ali so na voljo podatki o dotikih in katere geste sistem samodejno prepoznava. Podobno kot pri drugih platformah, se nato aplikacija registrira na sprejemanje dogodkov o gestah in dotikih. Vse potrebne večdotične podatke, aplikacija pridobi iz objektov tipa *TouchEvent* ali *GestureEvent*, ki so podani kot parametri pri klicu dogodkovnih metod.

Podobno kot pri večini prej omenjenih platform, je glavna hiba implementacije v Flashu negotovost glede nabora večdotične funkcionalnosti, ki bo aplikaciji na voljo v nekem trenutku. Razvijalci se zaradi tega pri splošno namenskih aplikacijah le redko poslužujejo novih interakcijskih možnostih. Kljub temu, pa je Flash platforma zelo popularna v krogih, ki razvijajo svoje lastne večdotične naprave. Flash predvajalnik lahko namreč relativno enostavno povežemo tudi z odprto-kodnim procesi za sledenje. To, skupaj z možnostjo hitrega razvoja bogatih grafičnih vmesnikov naredi platformo zelo privlačno.

Poglavje 4

iOS

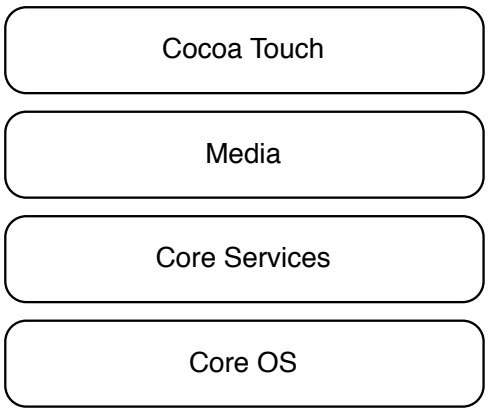
iOS je operacijski sistem, ki ga podjetje Apple uporablja pri svojih mobilnih napravah. Ker predstavlja tudi večdotično platformo, na kateri je bil razvit praktični del te naloge, si ga bomo v tem poglavju ogledali nekoliko podrobneje.

4.1 Operacijski sistem

iOS ima relativno kratko zgodovino. Izšel je leta 2007 skupaj s prvim mobilnim telefonom iPhone. V začetku sistem ni imel določenega imena in ni omogočal aplikacij, ki bi jih lahko poganjali neposredno na platformi. To se je spremenilo leta 2008, z izdajo druge verzije operacijskega sistema, ki je dobil poimenovanje *iPhone OS*. Končno ime iOS se je izoblikovalo z izdajo četrte revizije leta 2010 [50].

iOS temelji na operacijskem sistemu Mac OS X in si z njim deli veliko skupnih značilnosti ter tehnologij. Skupno je tudi razvojno okolje ter večina programskih vmesnikov. Sistem ima v ozadju štiri-nivojsko zgradbo [51] (slika 4.1). Vsaka plast predstavlja množico tehnologij in knjižnic, ki na eni strani nudijo izvajalno okolje aplikacijam, na drugi pa skrbijo za upravljanje s strojno opremo. Višje plasti so grajene na funkcionalnosti, ki jo nudijo nižje plasti, in predstavljajo objektne abstrakcije nižjenivojskih konceptov. Aplikacije najpogosteje dostopajo do najvišjih dveh, po nižjih pa posegajo le, ko potrebujejo funkcionalnosti, ki so na višjih nivojih abstrahirane.

Vrhnja plast (*Cocoa Touch*) nudi podporo za implementacijo uporabniškega vmesnika in dostop do mnogih visoko-nivojskih sistemskih storitev. V plasti *Media* je zbrana podpora za grafiko, zvok in video. Tretja plast (*Core services*) vsebuje nižjenivojske storitve, kot so neposreden dostop do omrežnih povezav, podatkovnih baz in podatkov o napravi. V najnižji plasti (*Core OS*) se nahaja



Slika 4.1: Štiri plasti operacijskega sistema iOS.

jedro operacijskega sistema ter določene nizkonivojske knjižnice za kriptiranje in matematične operacije.

Osnovo za razvoj iOS aplikacij predstavljajo programski jezik Objective-C (možna je tudi uporaba C in C++), ter sistemski knjižnici *Foundation* in *UIKit* [51]. V knjižnici Foundation so definirani osnovni razredi in podatkovne strukture, medtem ko UIKit ponuja ključne aplikacijske vmesnike za izgradnjo grafičnih aplikacij. Poleg teh dveh obstaja še množica drugih specializiranih knjižnic, ki med drugim ponujajo dostop do podatkovnih baz, omrežno komunikacijo, dostop do sistemskega adresarja in koledarja, podatke o lokaciji in orientaciji naprave, animacijske sposobnosti, 3D risanje, ipd. Možen je tudi razvoj spletnih aplikacij, ki se izvajajo znotraj mobilnega brskalnika Safari in temeljijo na uporabi jezika *JavaScript* (3.5.4).

Sistemski uporabniški vmesnik je, podobno kot pri prej omenjenem sistemu Android, popolnoma prilagojen uporabi s prsti. Programske kontrole (gumbi, drsniki in stikala) so velike in zaznavajo dotike tudi izven svojih meja. Pogosta je uporaba animacij in drugih vizualnih pristopov, ki skušajo nadomestiti pomanjkanje fizičnega odziva pri uporabi kontrol. V nekaterih primerih se uporablja tudi zvok. Dober primer za to je grafična tipkovnica, pri kateri se ob pritisku na tipko predvaja zvok in izriše povečana tipka. Z zvokom uporabnik dobi povratni odziv o zaznanem dotiku, povečana tipka pa mu prikaže katero od tipk je dejansko zadel. Za razliko od sistema Android je pri iOS velik del interakcije zasnovan na večdotičnih gestah, brez ustreznih ekvivalent v obliki klasičnih kontrol. Osnovni sistemski uporabniški vmesnik je t. i. *SpringBoard*, ki prikazuje vmesnik za izbiranje aplikacij ter upravlja z njihovim poganjanjem

(a) (b) (c)

Slika 4.2: Naprave, ki podpirajo operacijski sistem iOS: iPhone (a), iPod Touch (b), iPad (3). (Slike povzete po `http://www.apple.com/pr/products/`)

in zapiranjem. Po zagonu aplikacije zasedejo vso površino prikazovalnika. iOS vključuje nekaj vgrajenih aplikacij, katerih natančen nabor je odvisen od posamezne naprave. Poleg teh, je uporabniku na voljo že preko 250.000 dodatnih aplikacij v trgovini *Apple AppStore* [52].

4.2 Naprave

Operacijski sistem iOS trenutno podpira tri mobilne naprave: iPhone, iPod Touch in iPad (slika 4.2). Posebna verzija iOS teče tudi na zadnji verziji multimedijske naprave Apple TV. To okolje programerjem zaenkrat ni dostopno, zato Apple TV tukaj ne posebej obravnavan.

4.2.1 iPhone

Apple iPhone je pametni mobilni telefon, ki uporabnikom ponuja mobilne storitve, dostop do interneta in multimedijskih vsebin. Prva verzija je bila izdana leta 2007, trenutno pa je na voljo že četrta generacija, ki nosi ime iPhone 4.

Poglavitna zunanja značilnost strojne opreme je 3.5’ (9 cm) večdotični zaslon, ki temelji na kapacitivni tehnologiji (poglavje 3.2). Visoko-resolucijski zaslon (640 x 960 točk) lahko zaznava dotike do enajstih prstov hkrati. Zaslon predstavlja glavni vmesnik za interakcijo z napravo, medtem ko je ostalih fizičnih kontrol le 5. Te so: gumb za zapustitev aplikacije (*Home button*), gumb

za prižig in izklop, stikalo za tihi način delovanja, ter dva gumba za spreminjanje glasnosti. Od teh ima le gumb za zapustitev aplikacije neposredno vlogo pri menijski navigaciji. Naprava vsebuje tudi dve kameri (ena spredaj in ena zadaj), ki omogočata zajem slik in videa. [51, 53]

Poleg večdotične površine in kamer ima iPhone tudi množico drugih senzorjev. Ti obsegajo stereo mikrofon, merilnik pospeška (ang. *accelerometer*), žiroskop, merilnik intenzitete svetlobe, merilnik bližine (ang. *proximity sensor*), digitalni kompas in GPS [51]. Z uporabo podatkov iz senzorjev lahko implementiramo množico novih interakcij, ki dopolnjujejo večdotični zaslon. Pogosta je uporaba pospeškomerja, s pomočjo katerega lahko aplikacije prilagodijo svoj grafični vmesnik trenutni orientaciji naprave.

Poglavitna komponenta v notranjosti naprave je integrirano vezje Apple A4, ki vsebuje 1 Ghz ARM centralno procesno enoto in grafični procesor [54]. iPhone 4 vključuje 512 MB hitrega pomnilnika ter 16 ali 32 GB flash pomnilnika [51]. iOS omejuje količino navideznega pomnilnika z količino fizičnega pomnilnika, zato morajo aplikacije biti pozorne, da ne prekoračijo prostora, ki jim je na voljo na hitrem pomnilniku. Omrežna povezljivost je zagotovljena s podporo raznim mobilnim standardom (tudi *UMTS/HDPA*), ter lokalnim brezžičnim omrežjem (*Wi-Fi*) [51].

4.2.2 iPod Touch

iPod Touch je v mnogih pogledih zelo podoben iPhonu. Lahko bi ga celo opisali kot okrnjeno verzijo prej opisane naprave. Poglavitna razlika med obema pa je, da iPod Touch ne vsebuje funkcij povezanih z mobilno telefonijo. Namenjen je predvsem predvajanju multimedijskih vsebin, uporabi interneta, ter igranju iger. Podobno kot iPhone, je tudi iPod Touch do sedaj doživel štiri prenovitve.

iPod Touch si s telefonom iPhone deli enak večdotični zaslon in večino strojnih senzorjev. To omogoča, da lahko iz vidika uporabniških vmesnikov obe napravi pogosto obravnavamo kot eno. Pri obeh napravah je enako tudi osrednje integrirano vezje, ki ga predstavlja Apple A4. iPod Touch vsebuje polovico manj hitrega pomnilnika kot iPhone (256 MB) in ima bodisi 32 bodisi 64 GB flash pomnilnika. [55, 56]

4.2.3 iPad

iPad predstavlja najnovejšo skupino Applovih mobilnih izdelkov. V osnovi gre za podobno napravo, kot je iPod Touch, le da vsebuje bistveno večji zaslon. Tudi v funkcionalnem smislu pokriva naprava podobna področja kot iPod To-

uch. Večji zaslon in zmogljivejša baterija ga naredita atraktivnega tudi za branje elektronskih publikacij in urejanje preprostejših dokumentov.

Kapacitivni večdotični zaslon, ki ga uporablja iPad meri 9.7' (25 cm) po diagonali in ima resolucijo 1024 x 768 slikovnih točk [57]. Aplikacije imajo pri tem zaslonu na voljo precej več slikovnega prostora, zato za iPad pogosto vsebujejo posebej prilagojen grafični vmesnik. Večji zaslon je tudi bistveno bolj primeren za večdotično interakcijo in oblikovanje naravnih uporabniških vmesnikov. To je tudi razlog, da je bila večdotična aplikacija, ki predstavlja praktični del te naloge, razvita prav za to napravo.

Ostala strojna oprema je zelo podobna tisti, ki jo najdemo na iPod Touchu. Tudi pri iPadu se uporablja Apple A4 in enaka količina hitrega pomnilnika (256 MB). iPad nima kamere, lahko pa ima opcijsko podporo za mobilni internet (UMTS) in GPS.

4.3 Večdotična podpora

Operacijski sistem iOS je bil od samega začetka predviden uporabi na napravah, za katere predstavlja večdotični zaslon poglaviten interakcijski vmesnik. To je omogočilo načrtovalcem, da so se povsem posvetili tej vrsti interakcije in za njo razvili dobre programske rešitve.

4.3.1 Ravnanje z dotiki

Obravnavanje dotikov na iOS temelji na konceptu dogodkov. Dogodki so objekti tipa UIEvent, ki jih sistem na osnovi uporabniških akcij pošilja aplikacijam. iOS pozna več vrst dogodkov. Tukaj bomo obravnavali večdotične dogodke, ki se prožijo med tem, ko se dotikamo površine zaslona s prsti. Objekti UIEvent imajo predpisan tip, ter množico drugih atributov, ki vsebujejo podrobnejše podatke o dogodku. [31]

Drugi pomemben koncept, ki se uporablja v povezavi z dogodki so objekti poslušalci (ang. *responder objects*). Poslušalci lahko dogodke prejemajo in obravnavajo. Programski vmesniki za ravnanje z dogodki so definirani v razredu UIResponder, od katerega dedujejo vsi objekti poslušalci. Posebno pomembna skupina poslušalcev so pogledi (ang. *views*), ki predstavljajo temeljne objekte za gradnjo uporabniškega vmesnika. Objekti, ki bi želeli prejemati večdotične dogodke morajo redefinirati nekatere metode, ki so definirane znotraj razreda UIResponder. Za zaznavanje dogodkov so pomembne naslednje štiri [58]:

- `touchesBegan:withEvent:` - novi dotik enega ali več prstov,
- `touchesMoved:withEvent:` - eden ali več obstoječih dotikov je spremenilo položaj,
- `touchesEnded:withEvent:` - eden ali več dotikov se je končalo,
- `touchesCancelled:withEvent:` - sistem je preklical dotik.

Vsak dotik je predstavljen z enim samim UITouch objektom, ki skozi potek večdotične sekvence spreminja svoje atribute (npr. koordinate dotika). Zgoraj omenjene metode prejmejo množico novih oz. spremenjenih UITouch objektov in objekt UIEvent, iz katerega lahko med drugim dobimo tudi podatke o ostalih dotikih.

Podatke o dotikih lahko hkrati prejme samo en objekt, ki ga sistem izbere na podlagi zaporedja geometrijskih preverjan (ang. *hit-testing*). Praviloma se izbere pogled, ki leži neposredno pod dotikom ali eden od njegovih nadpogledov. Objekt, na katerega se nanaša prvi dotik, nato prejema podatke o celotni sekvenci, četudi se dotik kasneje premakne izven njegove površine.

4.3.2 Geste

Tudi za ravnanje z gestami je pri iOS dobro poskrbljeno. Razvijalcem so na voljo t. i. prepoznavalniki gest (ang. *gesture recognizer*), ki poenostavljajo logiko za detekcijo in uporabo gest. Možna je uporaba že vgrajenih prepoznavalnikov, kot tudi definiranje povsem novih.

iOS definira šest standardnih tipov gest, ki jih lahko obravnavamo tudi z vgrajenimi prepoznavalniki [31]:

- dotik (ang. *tapp*) - s poljubnim številom prstov se enkrat ali večkrat dotaknemo,
- povečaj - pomanjšaj (ang. *pinch*) - dva prsta premikamo narazen za povečavo in skupaj za pomanjšavo,
- premik (ang. *pan*) - enega ali več prstov položimo na površino in poljubno premikamo,
- hitri pomik (ang. *swipe*) - enega ali več prstov na hitro premaknemo levo, desno, gor ali dol,
- rotacija (ang. *rotate*) - dva prsta krožno premikamo v nasprotnih smereh,
- dolgi dotik (ang. *long press*) - s poljubnim številom prstov se nič ali večkrat dotaknemo in nato enkrat za daljši čas.

Omenjeni tipi imajo več različnih podvrst, ki se med sabo razlikujejo po številu uporabljenih prstov (ang. *touches*) in številu zaporednih dotikov (ang. *taps*).

Prepoznavalniki so objekti, ki dedujejo od abstraktnega razreda UIGestureRecognizer. Konkretne podtipe uporabljamo tako, da jih priredimo posameznim pogledom. Sistem nato poskrbi, da prepoznavalniki dobivajo enaka sporočila, kot bi jih sicer pogledi. Vsak pogled lahko vsebuje več prepoznavalnikov, ki lahko delujejo tudi povezano.

Prepoznavalniki interno delujejo na principu končnega avtomata. Medtem, ko opazujejo sekvenco dotikov, spreminjajo svoje stanje, sistem pa poskrbi, da se na osnovi trenutnega stanja izvršijo ustrezne akcije.

Poglavje 5

Večdotični vmesniki

V zadnjih dveh poglavjih smo si natančneje ogledali strojne in programske komponente, ki omogočajo večdotično interakcijo z računalniškimi sistemi. Kljub percepciji večine ljudi o naprednosti večdotične tehnologije, pa je njen dejanski doprinos v tehnološkem smislu precej skromen. Tehnologija gradi močno na obstoječih rešitvah in je glede na siceršnjo kompleksnost računalniških sistemov relativno preprosta in z znanstvenega vidika nezanimiva. Osrednji fokus raziskovanja večdotične tehnologije pa ni v tehnologiji sami, temveč v študiji z njo povezanih interakcij in uporabniških vmesnikov.

5.1 Značilnosti

Za načrtovanje računalniških uporabniških vmesnikov lahko pravimo, da poteka na osnovi principov in smernic [59]. Principi so temelj vsakega vmesnika. Določajo koncepte kot so poglavitna vrsta interakcije, osnovne gradnike in akcije. Za današnje grafične uporabniške vmesnike (2.2.2) so bili principi definirani že pred desetletji. Temeljijo na uporabi miške, konceptu WIMP in akcijah, kot so povleci in spusti (ang. *drag and drop*). Smernice se izoblikujejo postopoma, saj izvirajo iz raziskav in izkušenj. Govorijo o tem, kako v določenem kontekstu izoblikovati najprimernejši vmesnik in s tem nuditi uporabniku najboljšo možno izkušnjo.

Večdotična tehnologija lahko predstavlja prelomno točko tako pri principih, kot pri smernicah uporabniških vmesnikov. Omogoča namreč uporabo principov naravnih uporabniških vmesnikov, o katerih smo že govorili v poglavju 2. Novi principi zahtevajo seveda tudi nove smernice.

Vse večdotične vmesnike ne moremo označiti kot naravne. Velja celo nasprotno. Vmesniki, ki se danes izvajajo na večdotičnih napravah večinoma

niso naravni (npr. vmesnik sistema Windows 7), temveč vsebujejo večino značilnosti grafičnih vmesnikov. Tudi grafičnih vmesnikov pa ne moremo učinkovito uporabljati na večdotičnih napravah, če niso zgrajeni na podlagi smernic, ki veljajo za večdotično tehnologijo.

Cilj načrtovanja uporabniškega vmesnika je narediti uporabnikovo interakcijo kolikor se da enostavno in učinkovito [60]. Dober uporabniški vmesnik je zato tisti, ki upošteva njegovega uporabnika na vseh nivojih - njegove zahteve, pričakovanja, kognitivne ter fizične sposobnosti. Tej filozofiji, ki jo je smiselno uporabljati tudi pri večdotičnih vmesnikih, pravimo uporabniško osredotočeno oblikovanje (ang. *user-centered design*) [61].

Človek uporablja za krmiljenje večdotičnega sistema svoje prste. Prsti so pritrjeni na dlani, ki so del roke in posledično celotnega telesa. Roke se kot vhodni mehanizem bistveno razlikujejo od klasičnih sledilnih naprav. Miškin kazalec predstavlja breztelesen prst, ki ga krmilimo posredno s premiki sledilne naprave. Medtem, ko je natančnost miškinega kazalca praktično na nivoju velikosti zaslonske pike, je lahko prst le toliko natančen, kot je premer njegovega vrha oz. blazinice (8 - 14 mm) [62]. Prste ljudje tudi le težko držimo povsem pri miru in z njimi ne moremo oblikovati povsem pravilnih kretenj. Ker so prsti pritrjeni na dlani jih tudi ne moremo poljubno premikati in sukati. To vse ima zelo pomembne implikacije za oblikovanje večdotičnih uporabniških vmesnikov. Kinetične in fiziološke lastnosti ljudi, ki so bile do nedavno predvsem skrb industrijskih oblikovalcev, so naenkrat postale pomembne tudi za oblikovalce računalniških uporabniških vmesnikov [62]. Vse to nosi za sabo mnoge praktične implikacije. Tako morajo na primer aktivna področja večdotičnih kontrol pokrivati primerno površino za uporabo s prsti. To velja še posebej, kadar lahko vršimo akcije z več prsti. Pri definiciji gest moramo biti pozorni predvsem na omejitve pri motoriki prstov. Uporabnik tako ne bo mogel držati prstov povsem pri miru, kadar izvaja gesto dolgega dotika, ali rotirati dveh prstov za polni kot pri gesti rotacije. Tudi pri izvršitvi gest, kot je hitri pomik v določeni smeri (ang. *swipe*), bodo težave, če ne bomo hevristično upoštevali odstopanja od povsem pravilne poti.

Nenatančnost pa ni edina slabost, ki jo imajo prsti in roke v primerjavi z miškinim kazalcem. Roke zaradi svoje velikosti zakrivajo bistveno več prostora, kot običajen miškin kazalec. V praksi to pomeni, da bo precejšen del zaslona pri večdotičnih mizah in prenosnih napravah pogosto zakrit [63]. Najbolj viden del predstavlja zgornji del zaslona, zato je le-ta najprimernejši za prikaz pomembnih podatkov [9].

Druga pomembna sprememba v primerjavi s klasičnimi vmesniki, je pomanjkanje fizičnega odziva na izvršene akcije (ang. *haptic feedback*). Lju-

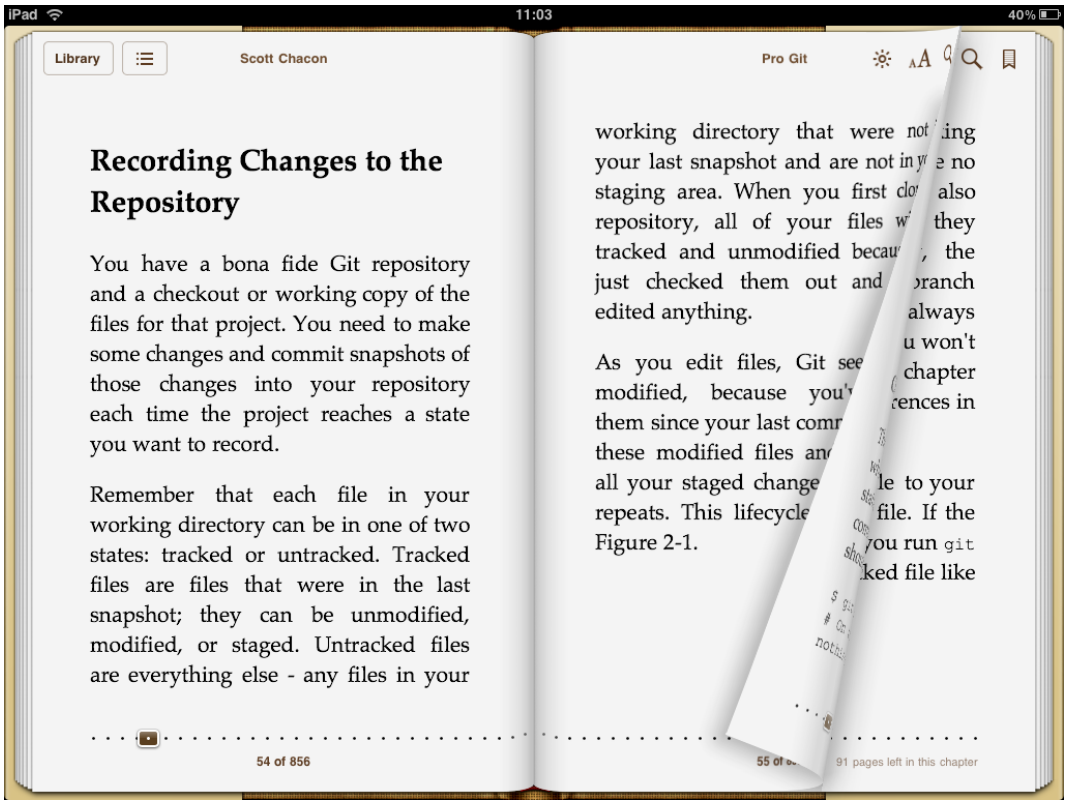
dje smo vajeni, da pri uporabi vmesnikov dobimo takojšnji fizični povratni učinek. Gumbi se ob pritisku pogreznejo in izvedejo specifičen zvok, stikala preskočijo, drsniki pri premikanju nudijo fizični odpor, miška pa pri pritisku "klikne". Fizični odziv nam omogoča, da dobimo takojšnjo povratno informacijo o uspešno zaznani akciji. Brez tega interakcijske metode, kot so slepo pisanje, ne bi bile možne. Večdotični zasloni fizičnega odziva ne nudijo. Kjerkoli se dotaknemo površine je občutek povsem enak. Da bi uporabnik vseeno bil deležen povratne informacije, je potrebno posegati po drugih metodah. Fizični odziv lahko tako nadomestimo z vizualnim ali zvočnim. Virtualni gumbi lahko ob dotiku spremenijo barvo ali obliko, stikala se animirajo v drugi položaj, drsniki pa takoj sledijo premiku prstov. Dodamo lahko tudi realističen zvok, ki dodatno izboljša percepcijo. Za odzive je pomembno, da so takojšnji (manj kot 100ms) in povezani s pričakovanimi odzivi iz realnega sveta [64].

Dobri večdotični uporabniški vmesniki naj zahtevajo le minimalno količino dodatne razlage in dokumentacije [9]. Uporabniki se uporabe takšnih aplikacij naučijo intuitivno s poskušanjem. Zaradi tega je pomembno, da se vključijo tudi mehanizmi za razveljavitev akcij ali se omogoči povrnitev na predhodno stanje z izvedbo ekvivalentne inverzne akcije [64].

Kot smo že omenili pri obravnavi naravnih uporabniških vmesnikov, imajo pomembno vlogo za doseganje intuitivnosti uporabniškega vmesnika metafore iz fizičnega sveta. Z njihovo uporabo povežemo virtualne objekte z njihovimi realnimi inačicami in tako sugeriramo možno uporabo in sprejete akcije. Metafora za večdotični pregledovalnik slik je lahko miza, na kateri so razvrščene slike. Ljudje intuitivno pričakujemo, da bomo lahko slike po mizi premikali in sukali. Pri tem pa se nikakor ne smemo omejiti zgolj na nabor funkcionalnosti iz realnega sveta, temveč moramo izkoristiti dodatne možnosti, ki nam jih ponuja digitalna tehnologija. Fotografije lahko tako v aplikaciji s posebno gesto tudi povečamo, kar je v realnem svetu nemogoče. Dober primer je tudi aplikacija iBooks, ki izkorišča metaforo knjige (slika 5.1) in s tem takoj sugerira na možnost listanja. Metafore naj bodo izbrane tako tako, da so za večino ljudi lahko prepoznavne, sicer se jim je bolje ogniti [9].

Večdotična tehnologija, z odstranitvijo vmesne naprave (miške), izboljšuje občutek neposredne manipulacije. Občutek je koristen, saj uporabniki lažje razumejo rezultate svojih akcij, če jih lahko izvajajo neposredno na dotičnih virtualnih objektih [9]. S primernim uporabniškim vmesnikom lahko ta občutek še povečamo. Koristna je predvsem uporaba gest, s katerimi lahko istočasno določimo ciljni objekt kot tudi izvedemo akcijo.

Pomembno implikacijo na obliko večdotičnega vmesnika ima tudi ciljna naprava, njen namen in lastnosti (predvsem velikost površine). Naprave z



Slika 5.1: Aplikacija iBooks izkorišča obliko knjige, ki takoj sugerira možnost listanja. Funkcije pa niso zgolj omejene na sposobnosti realnih knjig, saj iBooks med drugim nudi tudi iskanje, izbiro pisave in barv, ter vgrajen slovar.

(a) (b)

Slika 5.2: Različne dimenzije večdotičnih naprav. MP3 predvajalnik Apple iPod nano (a) in večdotični zid RingWall v nemškem mestu Nürburgring (b). (Slike povzete po `http://www.apple.com/pr/products/` in `http://infosthetics.com/archives/2009/11/ringwall_world_largest_multi-touch_multi-user_wall.html`)

večdotično podporo segajo danes vse od miniaturnih glasbenih predvajalnikov (*iPod nano* - $15cm^2$, slika 5.2a), pa vse do ogromnih večdotičnih zidov (*Ring° Wall* - $425m^2$, slika 5.2b). Velike naprave so tipično namenjene skupni uporabi več ljudi (3.4). Naprave uporabljamo od daleč in krmilimo s premiki celotne roke. Uporabniški vmesnik naj bo temu primerno velik in pregleden. Na drugi strani je pri majhnih napravah večdotična interakcija omejena na preproste kretnje z enim ali dvema prstoma iste roke. Uporabnik napravo uporablja od blizu, zato lahko predvidimo nekoliko manjše elemente in s tem bolje izkoristimo prostor, ki je na voljo.

Kot pri klasičnih grafičnih uporabniških vmesnikih tudi pri večdotičnih vmesnikih ne smemo pozabiti na estetiko. Uporabniški vmesnik mora biti urejen, logičen in pregleden. Oblika vmesnika naj sledi funkcionalnosti aplikacije in jo smiselno dopolnjuje. Ne-vsebinski deli vmesnika naj bodo subtilni in prilagojeni celostnem izgledu. Oblikovna dovršenost je sicer sekundarnega pomena, vendar zaželena, saj izboljšuje percepcijo o kvaliteti [10] in uporabnosti [37] celotne aplikacije. Poleg vizualne estetike je pri večdotičnih uporabniških vmesnikih pomembna tudi t. i. interakcijska estetika (ang. *interaction aesthetics*), ki obravnava uporabnikove občutke pri interakciji z vmesnikom. V idealnem primeru naj bo interakcija zasnovana tako, da se uporabniki niti ne zavedajo orodja, ki ga uporabljajo, temveč ga smatrajo kot naravni podaljšek svoje roke [37].

Slika 5.3: Pregledovalnik slik Microsoft Surface Photos. (Slika povzeta po `http://technet.microsoft.com/en-us/library/ee692031(Surface.10).aspx`)

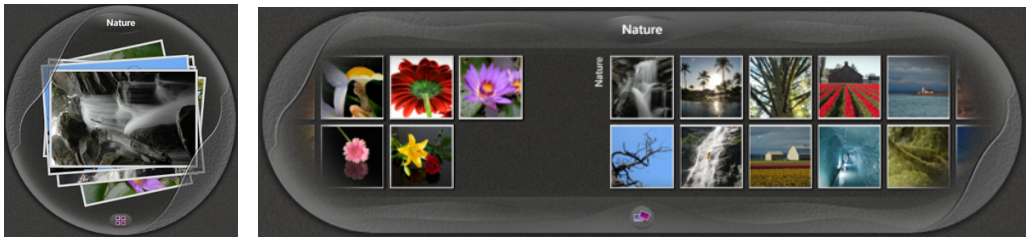
5.2 Študija primerov

Da bi bolje razumeli opisane koncepte, si velja ogledati nekaj konkretnih primerov. Zaradi možnosti lastnega preskušanja se bomo osredotočili predvsem na iOS platformo.

5.2.1 Microsoft Surface Photos

Verjetno najbolj prepoznavna vrsta večdotičnih aplikacij so pregledovalniki fotografij. Takšne aplikacije so odlične za prikazovanje naravne interakcije in večdotičnih gest, zato so skoraj obvezni del vsake večdotične demonstracije. Reprezentativni predstavnik te vrste aplikacij je *Microsoft Surface Photos*, ki predstavlja vgrajen pregledovalnik fotografij in video posnetkov na platformi Microsoft Surface [65].

Uporabniški vmesnik je oblikovan v obliki mize, na kateri so razpršene slike (slika 5.3). Vsaka izmed slik predstavlja fotografijo ali video posnetek (videoposnetki imajo na sredini gumb za predvajanje). Uporaba aplikacije temelji na večdotičnih gestah. S pomočjo standardnih gest lahko slike premikamo, rotiramo in skaliramo. Aplikacijo lahko brez težav uporablja tudi več uporab-



(a) (b)

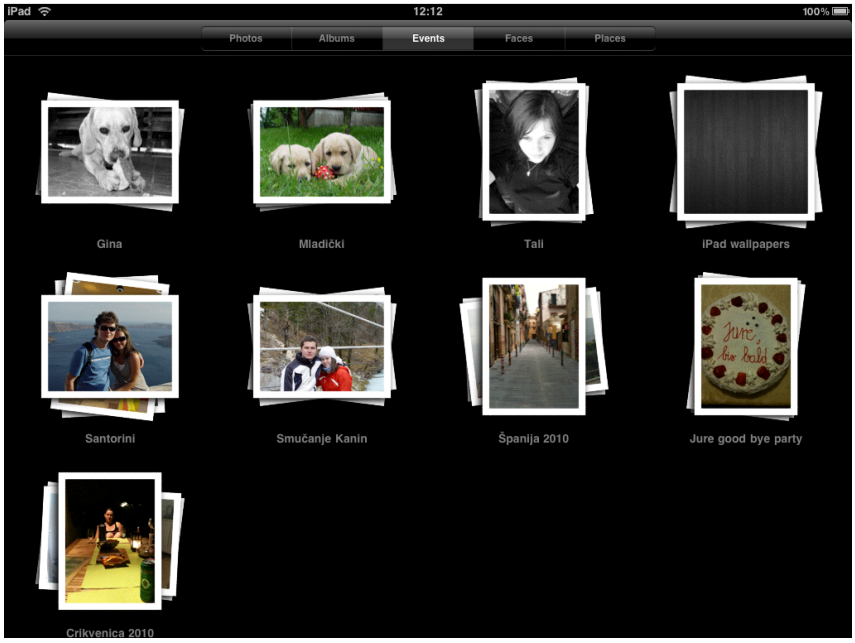
Slika 5.4: Surface Photos *stack* (a) in *scroller* (b). (Sliki povzeti po `http://technet.microsoft.com/en-us/library/ee692031(Surface.10).aspx`)

nikov hkrati, tako da vsak manipulira s svojo sliko [66]. Klasičnih menijev je le malo. Vsi so kontekstne narave in se prikažejo le po potrebi.

Razpršeni pogled vsebine, ki ga pri Microsoftu imenujejo *scatter view*, je za uporabnike naraven in intuitiven, vendar ima tudi svoje slabosti. Kadar je potrebno ravnati z večjo količino fotografij, hitro postane nepregleden in s tem težaven za uporabo. V tem primeru so veliko bolj pregledni pogledi, ki vsebino uredijo v pravilno strukturo (npr. v mrežo). Pri Microsoftu pa so izbrali nekoliko drugačno rešitev. Pri trenutno zadnji verziji aplikacije (1.0 SP1) so vpeljali dve novi komponenti - *stack* (sklad) in *scroller* (drsnik), ki sta namenjeni organizaciji večje količine fotografij. Komponenti sta dejansko en objekt, ki lahko ob pritisku na gumb spremeni svojo obliko. Kot pove že samo ime, organizira *stack* fotografije v obliki sklada slik (slika 5.4a). Z gestami lahko fotografije na skladu pregledujemo, tako da jih postavljamo iz vrha sklada na dno. Fotografije lahko tudi potegnemo iz sklada ven ali iz mize nazaj na sklad. *Scroller* opravlja podobno funkcijo kot sklad, le da organizira fotografije v obliki mreže (slika 5.4b). Obe komponenti prikazujeta fotografije ločeno po kategorijah. Trenutna kategorija je vedno navedena na zgornjem delu, med njimi pa izbiramo s pomočjo kontekstnega menija. Tudi pri fotografijah lahko z daljšim pritiskom prikličemo kontekstni meni. Ta nam omogoča, da izvajamo določene kompleksnejše akcije, kot so pošiljanje fotografij po elektronski pošti, objava na socialnih omrežjih ali arhiviranje [67].

5.2.2 iPad Photos

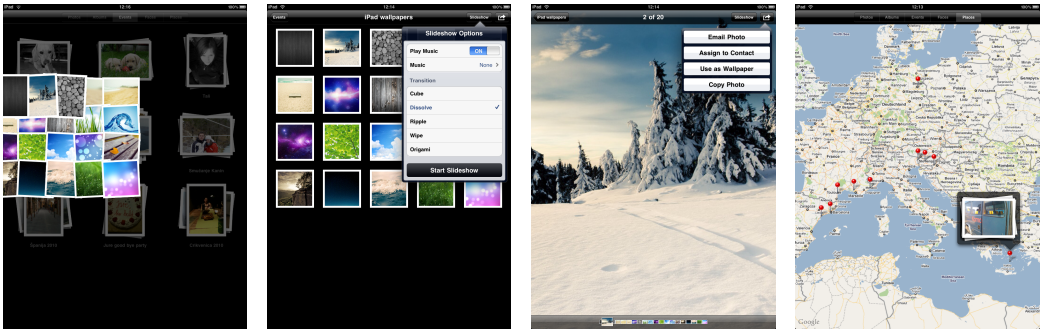
Pri podjetju Apple se načrtovalci niso odločili za uporabo metafore mize, temveč so izbrali nekoliko bolj striktno organizacijo fotografij. Kljub temu aplikacija iPad Photos močno izkorišča večdotično interakcijo, zato je prav, da



Slika 5.5: Pregledovalnik slik iPad Photos. Slike so grupirane v sklade, ki jih lahko z gesto povečaj - pomanjšaj razširimo.

si jo podrobneje ogledamo.

Aplikacija nudi več pogledov, med katerimi se premikamo z gumbi v orodni vrstici na vrhu aplikacije. Najpomembnejša sta pogled albumov in dogodkov, ki organizirata slike iz istega albuma (dogodka) v sklade (slika 5.5). Skladi prikazujejo reprezentativno fotografijo in ime skupine, s čimer omogočajo hiter pregled nad vsemi albumi. Posamezni sklad je možno razširiti in s tem pridobiti pogled nad vsemi slikami iz albuma. To lahko storimo bodisi z enim samim dotikom ali z gesto povečaj. V slednjem primeru animacija razširitve sklada postopoma sledi razširitvi prstov (slika 5.6a), s čimer uporabnik dobi dober občutek neposredne manipulacije. Fotografije iz določene skupine so v podrobnem pogledu razvrščene v obliki mreže (slika 5.6b). Mreža nudi dober pregled nad vsemi fotografijami tudi kadar je le-teh zelo veliko. Z uporabo enakih gest, kot na prvem nivoju, lahko povečamo posamezno sliko v celozaslonski pogled (slika 5.6c). Tu lahko navigiramo med slikami z gesto hitrega pomika (*swipe*) ali s pomočjo traka na spodnjem delu. V celozaslonskem pogledu lahko sliko tudi približamo ali pomanjšamo. To lahko storimo z običajno gesto povečaj - pomanjšaj ali z dvakratnim zaporednim dotikom (enoprstnim za povečavo, dvoprstnim za pomanjšavo).



(a) (b) (c) (d)

Slika 5.6: Ravnanje s slikami v iPad photos

Kot je razvidno iz slik 5.6b in 5.6c, je orodna vrstica prisotna na vseh nivojih, njena vsebina pa se razlikuje glede na to, kje se nahajamo. Nabor kontrol v vrstici je majhen, kadar je potrebnih več opcij pa ponujajo dodatne kontekstne menije, ki se prikažejo na zahtevo. Kot smo že omenili je zgornji del zaslona pri napravah kot je iPad najbolj viden in zato najprimernejši za takšne komponente.

V nasprotno smer se po slikovni hierarhiji premikamo z uporabo inverzne geste. Če fotografijo v celozaslonskem načinu dovolj pomanjšamo, se vrnemo na mrežni pogled. Z aplikacijo geste pomanjšaj na mreži, pa se vrnemo na osnovni nivo. Prehodi med nivoji so animirani, s čimer se uporabniku nazorno prikaže, kje se je fotografija ali album nahajal v višjih nivojih hierarhije. Ta način navigacije morebiti ni takoj očiten vsem uporabnikom. Med tem, ko se lahko po hierarhiji navzdol premikamo tudi z uporabo vsem očitnega enoprstnega dotika, je za navigacijo navzgor potrebna dvoprstna gesta pomanjšaj. Da uporabniki ne bi imeli težav pri navigaciji, je za vrnitev na višji nivo na voljo tudi tipka v orodni vrstici.

iPad Photos nudi v primeru sinhronizacije slik z namiznim programom iPhoto, še dva zanimiva pogleda. Prvi se imenuje obrazi (ang. *faces*) in prikazuje fotografije grupirane v sklade glede na osebe, ki se na njih nahajajo (prepoznavanje obrazov). Drugi pogled nosi ime kraji (ang. *places*). Pogled prikazuje albume, ki vsebujejo geografske podatke, na interaktivni mapi (slika 5.6d). Tudi navigacija po mapi povsem bazi na uporabi večdotičnih gestah. Albumi so prikazani v obliki označba, ki ob dotiku prikažejo gručo slik. Tudi tu lahko gručo razširimo in tako preidemo v mrežni pogled.

5.2.3 Google Earth

Aplikakacijo Google Earth smo na kratko že spoznali v poglavju 2. Ker aplikacija dobro uporablja določena načela večdotičnih vmesnikov, si jo bomo na tem mestu ogledali še nekoliko podrobneje.

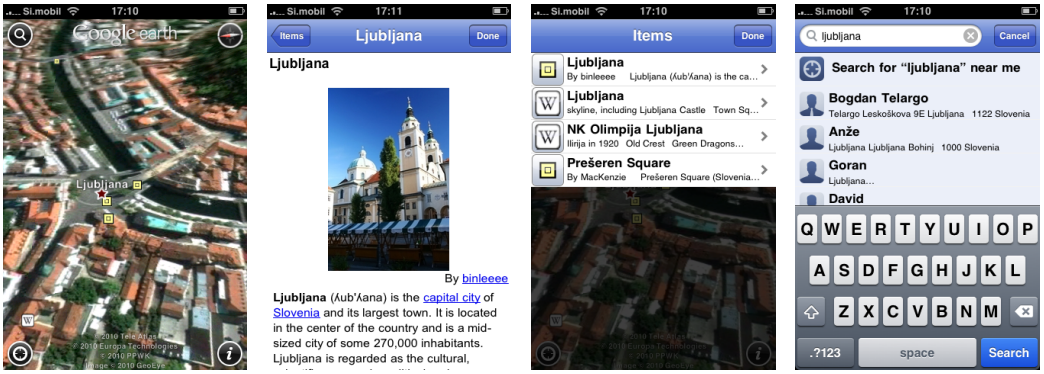
Kot že omenjeno, prikazuje Google Earh 3D virtualni globus Zemlje skupaj z geografskimi meta-podatki. Uporabniki se lahko po prikazani pokrajini premikajo, spreminjajo pogled in povečavo. Verzija aplikacije za iOS platformo (sliki 5.7a in 2.1b) je popolnoma prilagojena uporabi na večdotičnih zaslonih, zato za navigacijo po mapi sploh ne ponuja klasičnih kontrol. Osnovno premikanje po pokrajini je vezano na premike enega prsta. Možno je uporabiti tudi hitro zaporedje dveh dotikov, s čimer se takoj premaknemo na izbrano mesto. Za povečavo in rotacija se predvideva uporaba standardnih gest povečaj - pomanjšaj ter rotacije dveh prstov. Nekoliko naprednejša operacija spreminjanja pogleda pa se uporablja s hkratnim premikanjem dveh prstov. Izbrana razporeditev gest ni naključna in je dober primer povezave kompleksnosti akcije in geste. Operacije, ki so za uporabo aplikacije kritične imajo dodeljene preproste geste, medtem ko kompleksne akcije lahko posegajo po zahtevnejših nestandardnih gestah.

Pokrajina v Google Earth je dodatno opremljena z označbami, ki jih lahko izberemo in tako dobimo več informacij o določenem kraju (slika 5.7b). Za posamezni kraj je lahko prikazana precejšnja množica označb in pogosto je težko s prstom zadeti točno tisto, ki nas zanima. Načrtovalci so za ta problem poiskali dobro in poučno rešitev. Kadar ni mogoče enolično ugotoviti izbrano označbo, se prikaže dodatni meni, ki prikaže označbe pod prstom v preglednejši obliki (slika 5.7c).

Aplikacija vsebuje tudi funkcijo iskanja. Kadar jo poganjamo na manjših napravah (iPhone, iPhod Touch) je vrstica za iskanje skrita, s čimer je uporabnikom na voljo več zaslonskega prostora za ogled vsebine. Zaradi pomanjkanja fizičnega odziva, je vnos teksta na večdotičnih napravah otežen. Google Earth to omejitev upošteva tako, da nam na osnovi podatkov, ki so shranjeni v napravi sam priporoča kraje, ki bi nas morebiti zanimali (slika 5.7d). Na ta način je pogosto dovolj, da določen naslov vnesemo le delno, ter nato izberemo ustrezno izbiro iz seznama predlogov.

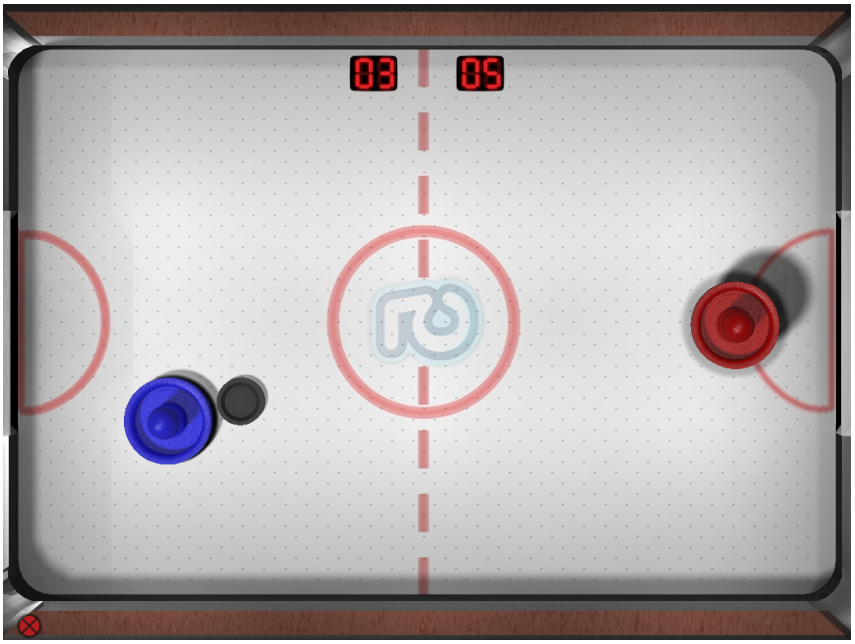
5.2.4 Touch hockey

Večdotična tehnologija postaja vse bolj atraktivna tudi za področje računalniških iger. Z njo se odpirajo inovativne interakcijske možnosti, ki ponujajo



(a) (b) (c) (d)

Slika 5.7: Google Earth na mobilnem telefonu iPhone (iOS).



Slika 5.8: Touch hockey za iPad.

razvijalcem priložnost za ustvarjanje povsem novih vrst uporabniških vmesnikov. Takšni vmesniki lahko bolje simulirajo interakcijo, ki so jo uporabniki vajeni iz naravnega okolja, ter tako pripomorejo k zmanjšanju razkola med fizičnimi in virtualnimi igrami. Neposredna interakcija omogoča, da se uporabnik bolje poveže z virtualnim svetom, ki ga uporablja na njemu naraven in zabaven način.

Kot preprosti primer večdotične računalniške igre si bomo ogledali večuporabniško iOS igro *Touch hockey* (slika 5.8). Igra predstavlja virtualno simulacijo znane fizične igre zračni hokej (ang. *air hockey*).

Zračni hokej je igra za dva igralca. Za igranje potrebujemo posebno prirejeno mizo, dva plastična pripomočka za odbijanje (*mallet*) ter plošček (*puck*). Miza na svoji površini ustvarja zračno blazino, po kateri drsi plošček skoraj brez trenja. Igralca vsak na svoji strani odbijata plošček in skušata zadeti režo, ki predstavlja gol na nasprotnikovi strani mize. Na koncu zmaga tisti, ki je uspel zadeti največ golov.

Touch Hockey izkorišča način interakcije pri realni igri, tako da ustvarja okolje kjer igralci krmilijo virtualne mallete s premiki svojih prstov. Privzeto malleti lebdijo, ob pritisku pa se spustijo na površino in pričnejo slediti prstu. Pri tem se postavijo rahlo nad površino dotika, tako da jih prsti ne zakrivajo. Igralno področje je razdeljeno na dva dela, ki sta ločena s prekinjeno sredinsko črto. Vsak igralec ima dodeljeno svoje aktivno področje znotraj katerega se vsi dotiki upoštevajo kot akcije istega uporabnika. Igre, ki temeljijo na takšnem principu razdelitve igralne površine, ne potrebujejo strojne sposobnosti za identifikacijo uporabnika in lahko zaradi tega tečejo na večina današnje večdotične opreme. Sam potek igre temelji na preprostem fizičnem pogonu, ki simulira realne odboje ploščka od sten ter malletov.

Grafično je igra povsem hiper-realistična predstavitev realne igre. Asociacija z realno igro je očitna, vendar je grafika vseeno nekoliko idealizirana. Povezava z realno igro pripomore k prepoznavnosti in takoj namiguje na način uporabe vsakemu, ki je igro že kdaj srečal v fizični obliki. Občutek igranja fizične igre še dodatno poglobi realistični zvok mize in odbojev.

Igre, kot so Touch Hockey navkljub vsemu še vedno ne morajo povsem pričarati občutka igranja fizične igre, vendar jim večdotična tehnologija omogoča, da so temu cilju sedaj bližje kot kadarkoli doslej.

5.2.5 SurfaceTwitter

Seveda niso vsi večdotični uporabniški vmesniki primeri dobrih naravnih uporabniških vmesnikov. Za konec si zato poglejmo nekoliko slabši vmesnik apli-

Slika 5.9: Večdotični Twitter odjemalec SurfaceTwitter. (Slika povzeta po `http://bit.ly/bsXoaw`)

kacije *SurfaceTwitter* [68].

SurfaceTiwtter je večdotični odjemalec za socialno mrežo Twitter, ki deluje na platformi Micfrosoft Surface. Ob zagonu aplikacija za trenutno prijavljenega uporabnika naloži zadnjih 25 sporočil in jih prikaže raztresene po površini virtualne mize (slika 5.9). Nabor sporočil se nato samodejno posodablja, pri čemur nova sporočila prepoznamo po dodani zvezdici. S priklicem skritega vnosnega polja ob strani je možno tudi kreiranje novih sporočil. Za vnos se pri tem uporablja sistemska virtualna tipkovnica.

SurfaceTiwtter za prikaz uporablja vgrajeno komponento Scatter View, podobno kot že obravnavana aplikacija Surface Photos (5.2.1). Ta komponenta omogoča enostavno naključno razporejanje objektov, ter vsebuje vgrajeno podporo za translacijo, rotacijo in povečavo prikazanih elementov. Kot smo že ugotovili, je funkcionalnost, ki jo ponuja Scatter View, odlična za pregledovanje manjše količine fotografij, saj namiguje na ravnanje s fotografijami v realnem svetu. Koncepti, ki dobro delujejo v eni vrsti aplikaciji, pa seveda niso univerzalno prenosljivi. Tako naključno razporejanje sporočil iz socialne mreže nima neposredne koristi za uporabnika. Takšna razporeditev zelo otežuje pregled nad časovnim zaporedjem sporočil in s tem onemogoča sledenje že prebranih

sporočil. Metafora mize, ki pri fotografijah deluje dobro, pri množici tekstovnih sporočil deluje neurejeno in nepregledno. Tekst za razliko od fotografij ne ponuja dovolj velike prepoznavnosti pri delni zakritosti, zato je prepoznavanje in izbira sporočil težavna.

Tudi iz grafičnega vidika je uporabniški vmesnik šibek. Uporablja generične grafične komponente, ki ne nudijo povezave z realnostjo in s svojo obliko ne namigujejo na način uporabe.

Dober večdotični vmesnik minimizira prostor, ki ga zasedajo neaktivne komponente. SurfaceTvitter to načelo dobro upošteva pri vnosnem polju za nova sporočila. V neaktivnem stanju je skrito izven prikazovalne površine, ob pritisku na gumb pa se prikaže. Na drugi strani, pa aplikacija presenetljivo vedno prikazuje formo za vnos uporabniškega imena. To je nenavadno, saj bo tipični uporabnik konfiguracijo izvedel le enkrat (na začetku uporabe) in kasneje tega polja več ne bo potreboval.

Aplikacija vsebuje tudi nekaj dobrih aspektov. Tako je sposobna iz sporočil razbrati povezave do slik in jih prikazati kot dodane objekte, pritrjene na sporočila. S tem dobi uporabnik pregled nad celotno vsebino sporočila, brez potrebe po odpiranju povezave s spletnem brskalniku. Žal je ta funkcionalnost omejena zgolj na slike iz spletne strani *TwitPic*.

V poglavju 7 bomo na osnovi pomanjkljivosti tega vmesnika skušali implementirati naravnejšo večdotično predstavitev za storitev Twitter.

Poglavje 6

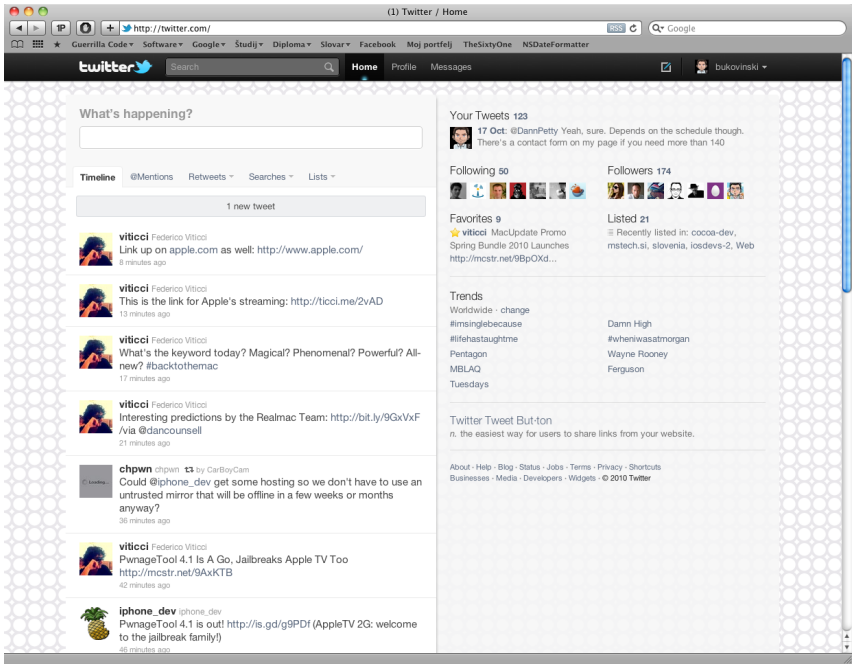
Twitter

Twitter je popularna spletna stran, ki svojim uporabnikom ponuja brezplačno socialno omrežje, zasnovano na storitvi spletnega mikro dnevnika[1] [69]. Ker bo Twitter v nadaljevanju služil tudi kot vir podatkov za našo aplikacijo, si ga na tem mestu na kratko oglejmo.

6.1 Funkcionalnost

Twitter temelji na dveh osnovnih funkcijah. Prva je tvorjenje in objava kratkih sporočil, imenovanih *tweets*. Tweeti so kratka tekstovna sporočila, ki ne smejo presegati 140 znakov. Vsebina sporočil je povsem poljubna, avtorji pa pogosto odgovarjajo kar na vprašanje: "Kaj se dogaja?" (ang. *"What's happening?"*), ki predstavlja slogan socialne mreže. Del tekstovnega sporočila so lahko tudi hiperpovezave, ki vsebino sporočila povežejo z drugimi relevantnimi vsebinami (npr. spletne strani, fotografije ali videi). Druga osnovna funkcija, ki jo ponuja Twitter, je agregacija sporočil, ki jih objavljajo drugi uporabniki. Uporabniki socialne mreže lahko v ta namen sledijo (ang. *follow*) objavam določenega uporabnika. Twitti vseh sledenih uporabnikov se nato prikažejo v obliki kronološko urejenega seznama objav (ang. *timeline*) na uporabnikovi Twitter domači strani (slika 6.1).

[1] Spletni mikro dnevnik (*ang. micro blog*) je oblika spletnega dnevnika (ang. *blog*), pri katerem so posamezni vnosi bistveno krajši, kot pri klasičnih dnevnikih (tipično le nekaj stavkov).



Slika 6.1: Spletna stran `twitter.com`.

6.1.1 Sporočila

Sporočila so privzeto javno objavljena in tako dostopna vsakomur. Opcijsko lahko uporabnik vidnost svojih sporočil tudi omeji na ožji krog prijateljev.

Določene kombinacije znakov in besed imajo pri sporočilih poseben pomen. Besede, ki imajo dodani prefix # se smatrajo kot označbe (ang. *hashtag*), ki določajo ključne besede v sporočilu. Označbe se lahko kasneje uporabljajo za iskanje in grupiranje sporočil, ter za ugotavljanje trenutno zanimivih tem (ang. *trending topics*). Posebno funkcijo ima tudi črka **d**. Kadar jo uporabimo na začetku sporočila, le-to ne bo objavljeno javno, temveč poslano kot privatno sporočilo (ang. *direct message*) uporabniku, ki ga poimenujemo z besedo, ki neposredno sledi črki. Kadar želimo v sporočilu omeniti določenega twitter uporabnika, to storimo tako, da njegovemu uporabniškemu imenu dodamo prefix **@**. Če omembo navedemo na začetku sporočila, se sporočilo smatra kot odgovor na eno izmed sporočil navedenega uporabnika. S tem povezana je tudi funkcija *retweet*, ki obstoječe sporočilo ponovno objavi. Takšno sporočilo ima na začetku črki **RT**, ki jima sledijo prej omenjena oblika poimenovanja originalnega avtorja, dvopičje, ter preostanek izvirnega sporočila. Ponovno objavljanje sporočil prestavlja glavni način širjenja zanimivih vsebin po omrežju.

6.1.2 Agregacija

Twitter ponuja veliko različnih načinov za pregledovanje objavljenih sporočil. Privzeti pogled prikazuje agregirana sporočila vseh sledenih uporabnikov, obstajajo pa tudi mnogi drugi pogledi, pri katerih si lahko med drugim ogledamo zgolj sporočila enega avtorja, samo privatna sporočila ali izključno sporočila iz določenega pogovora.

Omrežje je pred kratkim uvedlo tudi novi koncept twitter seznamov (*twitter lists*). Seznami predstavljajo grupiranja avtorjev, ter prikazujejo javna sporočila vseh avtorjev iz seznama. Podobno kot lahko sledimo sporočilom določenega uporabnika, lahko sledimo tudi sporočilom seznama, ter tako prejemamo sporočila od vseh vsebovanih uporabnikov.

6.2 Programski vmesnik

Omrežje Twitter ponuja drugim aplikacijam dostop do svojih storitev preko obsežnega programskega vmesnika. Twitter v tem primeru deluje kot strežnik, druga aplikacija pa kot odjemalec. Komunikacija med strežnikom in odjemalcem temelji na programski arhitekturi *REST*[2], ki omogoča pridobivanje podatkov o oddaljenih virih s pomočjo preprostih HTTP poizvedb.

Twitter dejansko ponuja tri različne programske vmesnike. Prvi je že omenjeni REST vmesnik, preko katerega lahko odstopamo do večine podatkov. Poleg tega je razvijalcem na voljo tudi vmesnik za iskanje (*Search API*) in pretočni vmesnik (*Streaming API*) [71]. Kot pove že samo ime, je vmesnik za iskanje namenjen poizvedbam, povezanim z iskanjem sporočil in ugotavljanjem aktualnih tem. Zaradi zgodovinskih razlogov, vsebuje vmesnik za iskanje nekoliko drugačno predstavitev podatkov kot običajni REST vmesnik. Pretočni vmesnik ponuja stalno povezavo z Twitter strežniki ter tako omogoča realno-časovno spremljanje sporočil. Primeren je predvsem za pomožne spletne storitve.

Za dostop do nekaterih oddaljenih virov je potrebna predhodna avtentikacija uporabnika. Twitter za te namene predvideva uporabo protokola *OAuth* ali njegove variante *xAuth* [71]. S pomočjo protokola OAuth (ali xAuth) aplikacija pridobi dostopni žeton (ang. *tolken*), ki ga nato skupaj s svojim privatnim ključem uporablja za podpisovanje vseh poslanih zahtev.

[2]REST (ang. *representational state transfer*) je vrsta programske arhitekture za distribuirane hipermedijske sisteme [70]. Arhitektura REST definira interakcije med strežnikom in odjemalcem na principu ponovne uporabe vokabularja obstoječih protokolov aplikacijske plasti (npr. HTTP).

Poglavje 7

Izvedba večdotične aplikacije

Praktični del pričujoče diplomske naloge predstavlja iPad aplikacija *Corkboard*[1]. Z njo bomo v tem poglavju predstavili razvoj večdotične aplikacije, ter ilustrirali koncepte, ki so bili podrobneje predstavljeni v poglavju 5.

7.1 Aplikacija Corkboard

Aplikacija Corkboard je preprosti odjemalec za socialno omrežje Twitter. Uporabnikom omogoča prebiranje sporočil, ogled v njih vsebovanih povezav, ter tvorjenje novih sporočil. Povod za nastanek aplikacije predstavlja večdotična aplikacija Surface Twitter, ki smo jo opisali v podpoglavju 5.2.5. Surface Twitter smo označili kot primer slabega večdotičnega vmesnika. Interakcijske možnosti, ki jih ponuja so nenaravne in uporabnikom ne omogočajo kvalitetnejše uporabe socialnega omrežja. Glavni cilj aplikacije Corkboard je prikazati, kako lahko večdotično interakcijo vpeljemo v takšno vrsto aplikacij na boljši način.

Pri aplikaciji Surface Twitter je razvijalec za prikaz sporočil izbral metaforo virtualne mize, na kateri so sporočila naključno razporejena. Takšen pogled je privlačen za demonstracijske namene in enostaven za implementacijo, vendar ne ustreza običajnemu načinu prebiranja sporočila iz socialnega omrežja. Sporočila, ki so dejansko kratke novice, ljudje najraje prebiramo tako kot elektronsko pošto ali internetne novice - v kronološko urejeni obliki. Prvi izziv, ki ga je bilo potrebno rešiti pri razvoju aplikacije Corkboard, je iskanje ustreznejše metafore za prikaz vsebine omrežja Twitter. Izbrana je bila obliko

[1]Corkboard je angleško poimenovanje za oglasno desko zgrajeno iz plute (ang. *cork*), na kateri ljudje objavljajo javna sporočila.

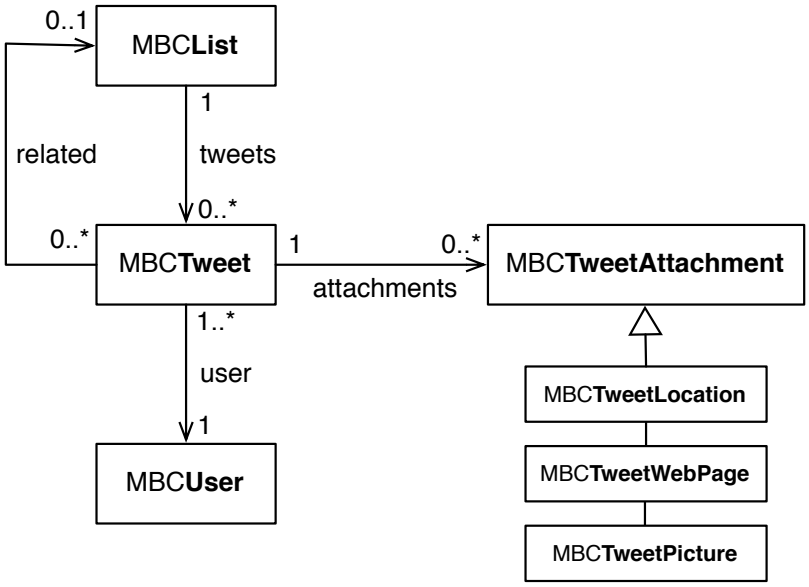
Slika 7.1: Oglasna deska je služila kot inspiracija za izdelavo aplikacije Corkboard. (Vir slike http://www.hickokcenter.org/tour.html)

oglasne deske (slika 7.1), saj jih ljudje že od nekdaj uporabljamo za javno objavo sporočil, podobno kot to počnemo v elektronski obliki na omrežju Twitter. Sporočila so na virtualni oglasni deski predstavljena v obliki listkov, ki imajo lahko pripete še druge vsebine (slike, mape in spletne povezave).

Drugi izziv, ki ga je bilo potrebno rešiti že na začetku, je izbira načina vključitve večdotične interakcije. Surface Twitter omogoča uporabnikom klasično večdotično manipulacijo s sporočili. Z gestami jih lahko premikamo, obračamo ter povečujemo. Premikanje in rotacija sta zaradi izbranega načina ureditve sporočil nujni operaciji, med tem ko povečava uporabnikom ne ponuja bistvene dodane vrednosti. Pri aplikaciji Corkboard takšnih manipulacij ne potrebujemo, saj je vsebina fiksno urejena v obliki mreže. Pri naši aplikaciji lahko večdotično interakcijo bolj smiselno vključimo tako, da uporabnikom z gestami ponudimo hitrejši in enostavnejši dostop do pogosto uporabljenih funkcij.

7.2 Načrtovanje

Aplikacija Corkboard je, tako kot večina drugih iOS aplikacij, zgrajena na osnovi programske arhitekture modela, pogleda in kontrolerja (ang. *Model-View-Controller, MVC*). Arhitektura MVC predvideva razdelitev aplikacijske



Slika 7.2: Diagram modelnih razredov aplikacije Corkboard.

logike na tri plasti. Plast modela vsebuje podatkovne strukture in funkcije, ki so potrebne za upravljanje s podatki. Na drugi strani, plast pogleda poskrbi, da so podatki iz modelne plasti ustrezno prikazani v obliki uporabniškega vmesnika. Plast kontrolerja je vmesna plast, ki povezuje uporabniški vmesnik z ustreznimi modelnimi strukturami. Na tej plasti se obravnavajo uporabniški vnosi, ter na osnovi njih kličejo ustrezne funkcije za posodobitev modelne in pogledne plasti. Na tem mestu si bomo na kratko ogledali načrt prvih dveh plasti aplikacije Corkboard.

7.2.1 Podatkovni model

Podatkovni model aplikacije Corkboard (slika 7.2) temelji na podatkovnih strukturah, ki jih predvideva omrežje Twitter. Korenski objekt predstavlja seznam (*MBCList*[2]), ki združuje kronološko urejeno množico sporočil. Vsako sporočilo (*MBCTweet*) nosi podatke o svoji vsebini in času nastanka. Vsebino je možno pridobiti tudi v obliki obogatenga besedila (*NSAttributedString*), ki omembe drugih uporabnikov, označbe ter povezave, prikazuje z drugačnim sti-

[2]Objektni-C ne pozna imenskih prostorov. Da bi se preprečile morebitne imenske kolizije, se razredom tipično dodajajo prefiksi, ki označujejo avtorja in/ali projekt. V našem primeru je izbrani prefiks MBC (Matej Bukovinski Corkboard).

lom pisave. Vsako sporočilo ima tudi referenco na svojega avtorja (*MBCUser*). Objekti MBCUser predstavljajo avtorjevo ime, uporabniško ime (ang. *screen name*) in URL naslov njegove profilne slike. Poleg tega imajo sporočila še dve lastnosti, ki ne izvirajo neposredno iz omrežja Twitter. Prava je seznam sorodnih (ang. *related*) sporočil, ki lahko vsebuje poljuben nabor pomensko povezanih sporočil. Pri trenutni implementaciji so tukaj vsebovana ostala sporočila istega avtorja. Zadnja pomembna lastnost sporočil je seznam pripetih vsebin (*MBCTweetAttachment*). MBCTweetAttachment je abstraktni razred, ki ima tri konkretne podrazrede. Le-ti lahko predstavljajo vsebine, ki jih avtorji prilagajo sporočilom v obliki spletnih povezav ali drugih meta-podatkov. Pri trenutni implementaciji je možno prepoznavati spletne strani (*MBCTweetWebPage*), slike (*MBCTweetPicture*) ali lokacije na zemljevidu (*MBCTweetLocation*).

7.2.2 Uporabniški vmesnik

Načrtovanje uporabniškega vmesnika je iterativni proces. Prične se z izdelavo grobih skic in v zaključni fazi pripelje do končnega videza aplikacije in produkcijskih grafičnih komponent.

Na sliki 7.3 je prikazana prva skica aplikacije Corkboard. Kot je razvidno, je že ta pripravljena z računalniškim programom. Komponente na tej skici so površno izrisane in slabe kvalitete. Po večini so predstavljene s pripravljenimi grafični elementi iz različnih internetnih baz. Kljub temu, že prva skica nazorno ilustrira poglavitne koncepte uporabniškega vmesnika. Oblikovan je v obliki oglasne deske, ki vsebuje pripeta sporočila. Sporočila, poleg besedila, vsebujejo še nekaj podatkov o avtorju ter relativni čas nastanka. Tista, ki imajo sorodne vsebine, so označena z več spetimi listki. Nekatera sporočila imajo lahko dodane pripete vsebine, ki dodatno ilustrirajo povezave in meta podatke iz sporočila. Na skici je to ponazorjeno v obliki slike in zemljevida, ki pripadata sredinskemu sporočilu. Podobno, kot druge aplikacije z naravnim uporabniškim vmesnikom, tudi Corkboard ne more povsem brez klasičnih gumbov za kontekstno neodvisne akcije. Le-ti so zbrani znotraj orodne vrstice na vrhu aplikacije.

Končni rezultat načrtovanja uporabniškega vmesnika je prikazan na sliki 7.4. Grafični elementi so tukaj že dokončno definirani in se v enaki obliki pojavljajo tudi v sami aplikaciji. Za razliko od začetne skice so listki sedaj povsem poravnani. Takšna razporeditev olajša branje, saj oko lažje sledi vsebini na enakih višinah. Dodelana je tudi orodna vrstica, ki se sedaj bolje vključuje v celoto in deluje kot del okvira oglasne deske.

Slika 7.3: Prva skica uporabniškega vmesnika.

Slika 7.4: Končna skica uporabniškega vmesnika.

7.3 Izvedba

Osnovo za razvoj iOS (iPad) aplikacij predstavljajo orodja in knjižnice, ki so vključene v t. i. iOS SKD. Pri izdelavi aplikacije Corkboard je bila uporabljena trenutno zadnja verzija 4.2.

V aplikaciji so bile uporabljene tudi nekatere odprto-kodne komponente, med katerimi velja izpostaviti:

- **MGTwitterEngine** - knjižnica za dostop do programskega vmesnika omrežja Twitter (izboljšana in poenostavljena lastna verzija - http://github.com/matej/MGTwitterEngine)
- **AQGridView** - komponenta, ki omogoča razporeditev v mrežo (lastna verzija z dodano opcijo horizontalnega načina razporejanja in drugimi izboljšavami - http://github.com/matej/AQGridView)
- **MBProgressHUD** - lastna komponenta, ki omogoča enostaven prikaz poteka operacij (http://github.com/matej/MBProgressHUD)
- **MBURLExpander** - lastna komponenta, ki omogoča razširitev kratkih spletnih naslovov (http://github.com/matej/MBURLExpander)
- **three20** - splošno namenska knjižnica, uporabljena za izris nekaterih gumbov (http://github.com/facebook/three20/)

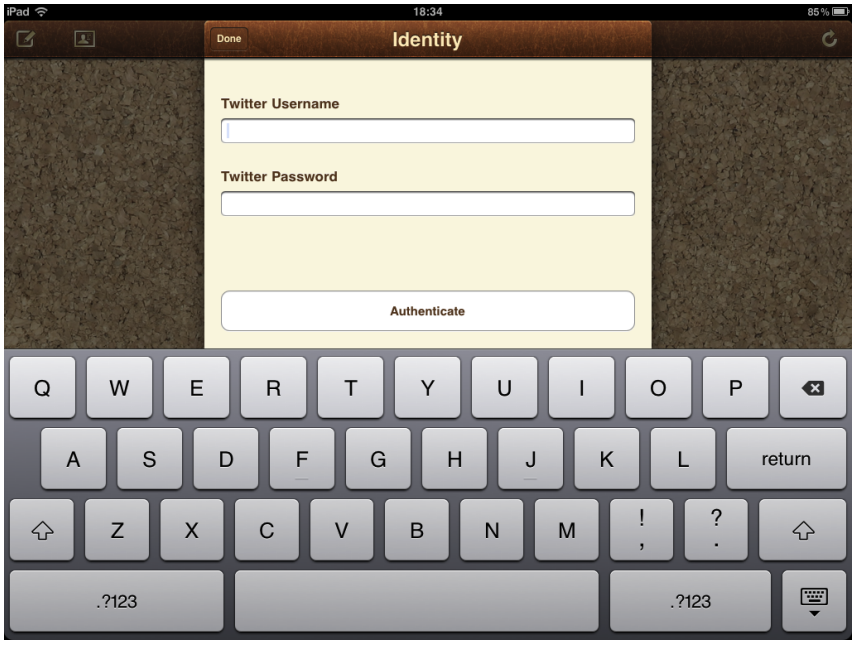
7.3.1 Dostop do podatkov

Pred začetkom uporabe se mora uporabnik prijaviti. V ta namen se ob prvem zagonu prikaže prijavno okno (slika 7.5), v katerega uporabnik vpiše svoje uporabniško ime in geslo. Ti podatki se nato izmenjajo za dostopni žeton, ki se shrani, in uporablja kot avtentifikacijsko sredstvo pri vseh nadaljnjih dostopih. Uporabnik lahko prijavno okno kadarkoli ponovno prikliče z enim od gumbov v orodni vrstici.

Ob vsakem zagonu se za trenutno prijavljenega uporabnika samodejno naloži zadnjih sto sporočil. Seznam je možno tudi kadarkoli osvežiti z gumbom v orodni vrstici. Za dostop do podatkov se v ozadju uporablja knjižnica MGTwitterEngine, ki poskrbi za klicanje ustreznih REST poizvedb in pretvorbo povratnih podatkov v domorodne objekte. Za ta namen imajo modelni objekti predvidene konstruktorje, ki sprejemajo podatke iz knjižnice.

7.3.2 Prikaz sporočil

iOS SDK ne ponuja vgrajene komponente, ki bi omogočila prikaz množice elementov v obliki mreže, zato je bilo potrebno poseči po lastni rešitvi. Osnovo



Slika 7.5: Prijavno okno.

za implementacijo mrežnega pogleda predstavlja odprto-kodna komponenta AQGridView, ki pa jo je bilo potrebno dopolniti z možnostjo horizontalnega razvrščanja.

Posamezni element mreže je sestavljen iz pogleda, ki predstavlja sporočilo in do dveh pripetih vsebin. Slike, ki izvirajo iz spleta (predvsem profilne slike in pripete vsebine) se nalagajo asinhrono in nato lokalno predpomnijo, kar zagotavlja dobro odzivnost pri ponovnih dostopih.

Pripete vsebine ponujajo uporabniku grafični predogled vsebinsko povezanih lokacij, slik in spletnih strani. Za izvedbo teh komponent je bilo potrebno poseči po več različnih spletnih storitvah:

- **lokacije** - Google Static Maps API (prikaz lokacij na zemljevidu),
- **slike** - Twitpic API, YFrog API in img.ly API (prikaz vzorcev slik),
- **spletne strani** - Pageglimpse API (posnetki spletnih strani) in Longurl API (razširitev skrajšanjih URL naslovov).

iPad nima fiksno določene orientacije uporabe, zato je priporočljivo, da se uporabniški vmesnik prilagodi trenutni orientaciji naprave [10]. Corkboard sledi temu načelu in avtomatično prilagodi razporeditev elementov ob zasuku naprave (slika 7.6).

(a) (b)

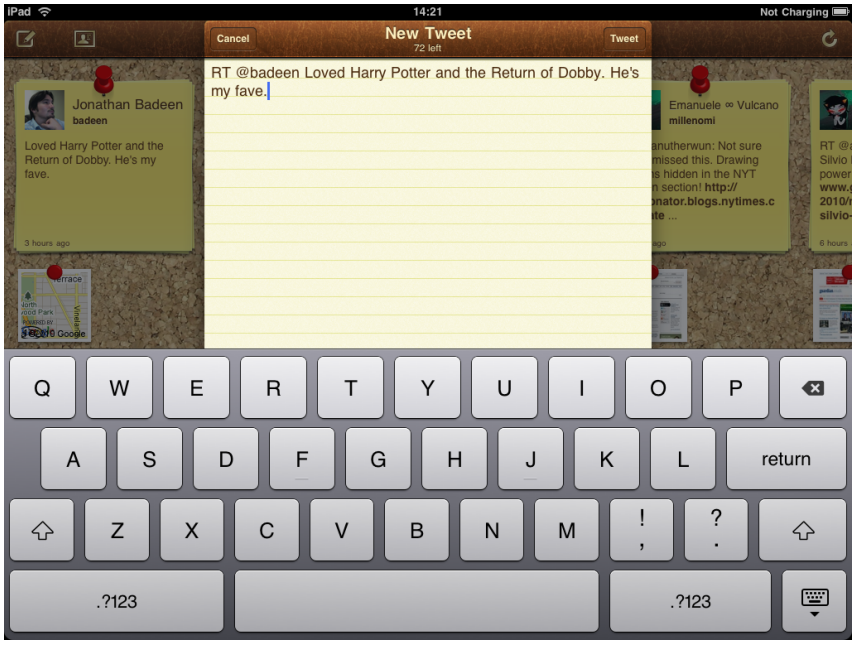
Slika 7.6: Vmesnik se prilagodi trenutni orientaciji naprave.

7.3.3 Tvorba sporočil

Aplikacija omogoča tudi tvorbo novih sporočil. Za ta namen je predviden gumb v orodni vrstici, ki prikliče pogled za sestavljanje sporočil (slika 7.7). Pogled je oblikovan v obliki beležnice, s čimer sledi celostni podobi aplikacije. Besedilo vnašamo s pomočjo sistemske tipkovnice, pri čemur se na vrhu okna izpisuje število preostalih znakov. Sporočilo lahko pošljemo s pritiskom na gumb “Tweet” ali zavržemo z gumbom “Cancel”. Alternativno lahko pogled zapremo tudi z uporabo geste pomanjšaj.

7.3.4 Interakcija s sporočili

Wobbrock, Morris in Wilson so s svojim raziskovalnim delom [4] pokazali, da večina neveščih uporabnikov pri interakciji z večdotičnim napravami intuitivno posega po podobnih gestah, kot jih lahko izvajajo z miško. Uporabniški vmesnik aplikacije Corkboard je zaradi tega zasnovan tako, da omogoča dostop do vseh poglavitnih funkcij aplikacije že z uporabo zaporedja enoprstnih dotikov. Ob dotiku kateregakoli od sporočil, se le-to obrne (slika 7.8) in s tem prikaže gumbe, ki omogočijo izvedbo raznih kontekstno odvisnih akcij. Vsak gumb



Slika 7.7: Pogled za sestavljanje sporočil.

je predstavljen s simbolom[3], ki prikazuje večdotično gesto, s katero je možno izvesti ekvivalentno akcijo.

Na sporočilih se prepoznavanje gest vrši s pomočjo sistema prepoznavalikov gest, ki smo ga opisali v podpoglavju 4.3.2. Pri sporočilih poleg dotika (UITapGestureRecognizer) prepoznavamo še naslednje tri geste:

- **enoprstni hitri pomik navzdol** (UISwipeGestureRecognizer) - prikliče pogled za tvorjenje sporočil s pripravljeno vsebino za odgovor na izbrano sporočilo,
- **dvoprstni hitri pomik navzdol** (UISwipeGestureRecognizer) - prikliče pogled za tvorjenje sporočil s pripravljeno vsebino za ponovno objavo izbranega sporočila,
- **gesta povečaj** (UIPinchGestureRecognizer) - prikaže sorodna sporočila, če so na voljo.

Tehnično najzanimivejša je izvedba prikaza sorodnih sporočil z gesto povečaj. Prikličemo jih tako, da v osnovnem pogledu postavimo dva prsta na sporočilo in ju postopoma razširjamo. Pri premikanju prstov narazen, pred-

[3]Simboli bazirajo na notaciji Gesturecons (http://gesturecons.com).

Slika 7.8: Kontekstne akcije na zadnji strani sporočila.

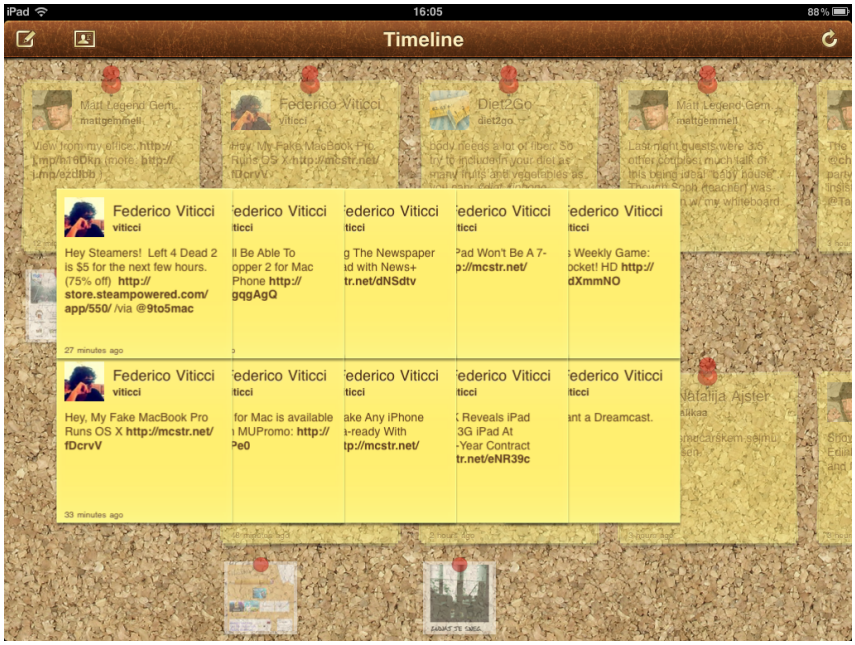
hodni nabor sporočil izginja, nova sporočila pa se skladno s premiki prstov razporejajo narazen (slika 7.9a).

Ko odmaknemo vsaj enega od dveh potrebnih prstov, aplikacijska logika preveri, koliko so bili prsti premaknjeni narazen. Na osnovi tega se nova sporočila animirajo na svoja končna mesta ali grupirajo nazaj. V prvem primeru preidemo v pogled sorodnih sporočil (slika 7.9b), sicer pa ostanemo v osnovnem pogledu. Povratek iz pogleda sorodnih sporočil lahko izvedemo z gumbom v orodni vrstici ali z izvedbo geste pomanjšaj, ki izvede podoben inverzni postopek.

Uporabniki lahko namesto sporočila izberejo tudi eno od pripetih vsebin. V tem primeru se, glede na vrsto pripete vsebine, odpre ali vgrajeni spletni brskalnik (slika 7.10a), ali vgrajen zemljevid *Google maps* (slika 7.10b). Obe komponenti podpirata tudi večdotično interakcijo (premik, povečava in pomanjšava vsebine).

7.4 Problemi

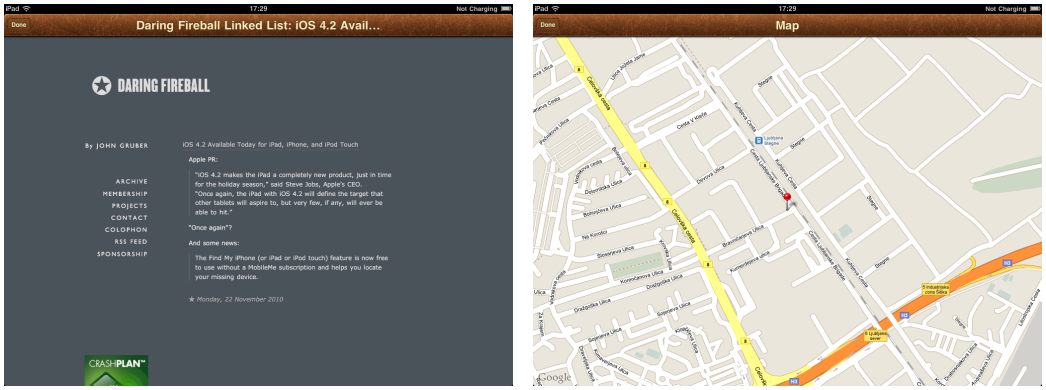
Pri delu z napravo kot je iPad, lahko hitro pozabimo, da se dejansko ukvarjamo z mobilno napravo, ki je kljub napredni tehnologiji, v performančnem smislu



(a)

(b)

Slika 7.9: Prikaz sorodnih sporočil.



(a) (b)

Slika 7.10: Prikaz pripetih vsebin.

bistveno šibkejša od današnjih osebnih računalnikov. iPad ima v primerjavi z modernimi osebnimi računalniki le kakšno petino njihove procesorske moči ter desetino kapacitete hitrega pomnilnika. To dejstvo zahteva, da so razvijalci pri izvedbi iOS aplikacij izredno pozorni na porabo sistemskih virov.

Med razvojem aplikacije Corkboard, so se težave s porabo sistemskih virov pojavile večkrat. Aplikacija je zaradi kompleksnih tekstur, senčenja in obilne uporabe prosojnosti, grafično precej zahtevna. To privede do nezanemarljive porabe pomnilnika (med-pomnjenje grafičnih elementov) in procesorske moči (izvajanje kompozicije in izrisa). Da bi vseeno omogočili dobro odzivnost pri pomikih (ang. *scrolling*), se večina pogledov ponovno uporablja. Pogledi sporočil, ki se zaradi pomika skrijejo, se posodobijo z novimi podatki, ter postavijo na ustrezno novo mesto na drugi strani mreži. To omogoča, da imamo lahko v pomnilniku vselej le peščico pogledov, četudi prikazujemo seznam, ki lahko obsega več sto sporočil. Podoben postopek se uporablja tudi pri ozadju, ki je dejansko sestavljeno le z dveh manjših slik, ki sledita premikom celotne mreže. Tudi sicer skuša aplikacija čim bolj varčevati s pomnilnikom. Komponente, ki ob zagonu aplikacije niso vidne in se morebiti med izvedbo sploh ne prikažejo, se inicializirajo le na zahtevo, ter se takoj po uporabi sprostijo (leno nalaganje).

Drugi večji problem, ki se je pojavil med razvojem aplikacije, je povezan z omrežjem Twitter. Twitter je v sklopu prenove spletne strani, preko spletnega vmesnika pričel ponujati podatke o sorodnih sporočilih (odgovori, ponovne objave, druga sporočila istega avtorja, sporočila s podobno vsebino ipd.). Ta funkcija je bila tudi povod za predvidene ekvivalentne operacije znotraj apli-

kacije Corkboard. Žal pa se je med razvojem izkazalo, da Twitter še ne ponuja programskih vmesnikov, ki bi omogočili implementacijo takšne funkcionalnosti pri odjemalcih. To dejstvo je privedlo do nekoliko skromnejšega pogleda sorodnih sporočil, ki pri trenutni verziji aplikacije vsebuje samo druga sporočila istega avtorja.

Poglavje 8

Zaključek

Zaradi stalnega tehnološkega razvoja, lahko v prihodnosti pričakujemo bistvene spremembe pri načinu interakcije človeka in računalnika. Klasične vhodno-izhodne naprave, kot so miška, tipkovnica in sledilna ploščica, bodo sčasoma izgubljale na svojem pomenu. Na drugi strani bomo lahko pri vse več napravah uporabljali naravnejše interakcijske možnosti, ki bazirajo na človekovih naravnih sposobnostih, kot so govor, vid in dotik. Novi uporabniški vmesniki bodo interakcijo z računalnikom naredili enostavnejšo in zabavnejšo, ter jo približali nivoju interakcije med ljudmi. Kljub temu je za pričakovati, da bodo klasični grafični uporabniški vmesniki še dolgo ostali pomemben in koristen faktor pri interakciji z računalniki, podobno kot danes velja za nekdaj prevladujoče vmesnike z ukazno vrstico.

Zgodnje oblike naravnih vmesnikov že srečujemo v našem vsakdanu. Pojavljajo se predvsem v obliki na dotik občutljivih naprav, ki so zmožne zaznavati več dotikov hkrati. Ta tehnologija, katere začetki segajo že slabih 30 let v zgodovino, se je šele pred kratkim začela komercialno širše uveljavljati, s čimer predstavlja odličen primer pojava, ki mu Bill Buxton pravi *"the long-nose of innovation."* [3]. Večdotične naprave se danes že pojavljajo v najrazličnejših oblikah, vse od miniaturnih mobilnih naprav, pa do obsežnih virtualnih sten. V bližnji prihodnosti lahko pričakujemo, da se bo trend večdotičnih naprav še nadaljeval, s čimer se bo večala potreba po izdelavi inovativnih naravnih vmesnikov za tovrstne naprave.

Velika raznolikost večdotičnih naprav pa ima tudi svoje pomanjkljivosti. Pri platformah, ki omogočajo izvajanje aplikacij na različnih tipih naprav, morajo razvijalci pogosto zagotavljati brezhibno delovanje tudi na tistih napravah, ki imajo le omejeno večdotično podporo ali pa je sploh nimajo. Razvoj univerzalnih večdotičnih aplikacij še dodatno otežujejo velike razlike med samimi

platformami, ki onemogočajo enostavno prenosljivost aplikacij. Dandanes bi tako zaman iskali mehanizem, ki bi omogočal učinkovito in cenovno ugodno med-platformno (ang. *cross platform*) razvijanje večdotičnih aplikacij.

Ena izmed trenutno najzanimivejših večdotičnih platform je gotovo platforma iOS, ki omogoča razvoj aplikacij za iPhone, iPod Touch in iPad. Omenjene mobilne naprave nimajo le odlične podpore za večdotično interakcijo, temveč so tudi cenovno ugodne ter enostavno dostopne. Za razvoj večdotičnih aplikacij je najzanimivejša naprava iPad, ki s svojim relativno velikim zaslonom omogoča udobno izvajanje gest, ter implementacijo bogatih uporabniških vmesnikov.

Uporabniške vmesnike za večdotične naprave moramo zasnovati nekoliko drugače, kot običajne grafične vmesnike. Nekatere probleme lahko delno rešimo z uporabo ustaljenih principov, pri mnogih drugih pa ne moremo mimo upoštevanja specifik, ki so povezane z uporabo večdotične opreme. Med drugim smo prišli do naslednjih ugotovitev:

- velikost komponent naj bo primerna za uporabo s prsti,
- roke pri uporabi zakrivajo spodnji del površine, zato naj bodo pomembni elementi zgoraj,
- videz komponent naj namiguje na način njihove uporabe,
- pri interakciji je pomemben takojšnji vizualni (zvočni) odziv,
- animacije dodatno izboljšajo občutek neposredne manipulacije,
- kjer je smiselno, uporabimo vedenje na osnovi naivne fizike,
- vsebina naj bo v ospredju, navigacijski elementi pa naj bodo subtilni in integrirani v celostno podobo,
- vmesnik naj bo vizualno bogat, ter predstavlja idealizirano realnost.

Kot smo videli pri študiji primerov, nekatere aplikacije odločno izkoriščajo večdotično interakcijo, ter s tem ponudijo uporabniško izkušnjo, ki jo pri vmesnikih, ki bazirajo na uporabi miške, ne bi mogli doseči. Pri vključitvi večdotične interakcije moramo biti pozorni, da le-ta smiselno dopolnjuje funkcionalnost aplikacije in ne služi le kot sredstvo za doseganje večje atraktivnosti uporabniškega vmesnika.

Rezultat praktičnega dela naloge je iPad aplikacija Corkboard, ki ponuja inovativen pristop k prikazovanju vsebine omrežja Twitter. Poglavitni značilnosti aplikacije sta vizualno bogat uporabniški vmesnik ter možnost uporabe več-prstnih gest. Interakcija, ki bazira na uporabi gest, je vključena v vse aspekte aplikacije. Geste omogočajo hitrejši dostop do kontekstnih funkcij, izvajanje operacij na modalnih oknih, ter manipulacijo pri prikazu pripetih vsebin. Aplikacijo je možno uporabljati tudi povsem brez uporabe naprednih

gest, kar jo naredi prijaznejšo do uporabnikov, ki so vajeni interakcije z miško. V tem primeru ponujajo kontekstni meniji implicitno pomoč pri odkrivanju večdotičnih "bližnjic" za ponujene akcije.

Aplikacija ima veliko potenciala za nadaljnje delo. Modularna objektno usmerjena zgradba omogoča enostavno vključitev novih funkcij, s čimer bi lahko v končni fazi izdelali aplikacijo, ki bi pokrivala vse aspekte omrežja Twitter. Zanimiva bi bila predvsem razširitev funkcije sorodnih sporočil s prikazom drugih relevantnih vsebin. Razširitve so možne tudi pri prikazu pripetih vsebin. Poleg dopolnitve obstoječih vrst z dodatno podporo za več spletnih storitev, bi bila zanimiva tudi vključitev predogledov za druge vrste vsebin (video posnetki, glasba, aplikacije, ipd.).

Slike

Literatura

[1] (2010, jul) Natural user interface. Dostopno na: http://wiki.nuigroup.com/Natural_User_Interface

[2] N. G. Authors, *Multi-Touch Technologies*. NUI Group, 2009.

[3] B. Buxton, ,,Multi-touch systems that i have known and loved,“ *Microsoft Research*, 2007.

[4] J. Wobbrock, M. Morris, in A. Wilson, ,,User-defined gestures for surface computing,“ v *Proceedings of the 27th international conference on Human factors in computing systems*. ACM, 2009, str. 1083–1092.

[5] Sphere: A multi-touch interactive spherical display. Dostopno na: http://research.microsoft.com/en-us/um/people/benko/projects/sphere/

[6] B. Buxton. Ces 2010: Nui with bill buxton (video interview). Dostopno na: http://channel9.msdn.com/posts/LarryLarsen/CES-2010-NUI-with-Bill-Buxton/

[7] P. Dietz in D. Leigh, ,,Diamondtouch: a multi-user touch technology,“ v *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*. New York, NY, USA: ACM, 2001, str. 219–226. Dostopno na: http://portal.acm.org/ft_gateway.cfm?id=502389&type=pdf&coll=Portal&dl=GUIDE&CFID=108523546&CFTOKEN=54401249

[8] M. Wu, C. Shen, K. Ryall, C. Forlines *in sod.*, ,,Gesture registration, relaxation, and reuse for multi-point direct-touch surfaces,“ *IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TableTop)*, str. 185–192, jan 2006.

[9] (2010, sep) iphone human interface guidelines. Dostopno na: http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html

[10] (2010, sep) ipad human interface guidelines. Dostopno na: http://developer.apple.com/library/ios/#documentation/General/Conceptual/iPadHIG/Introduction/Introduction.html

[11] J. Nielsen, ,,Noncommand user interfaces,“ *Commun. ACM*, zv. 36, št. 4, str. 83–99, 1993.

[12] (2010, sep) Natural user interface. Dostopno na: http://en.wikipedia.org/wiki/Natural_User_Interface

[13] (2010, sep) User interface. Dostopno na: http://en.wikipedia.org/wiki/User_interface

[14] A. de los Reyes, ,,Predicting the past,“ v *Web Directions South 2008*. Sydney Convention Centre: Web Directions, 2008.

[15] A. Unwin in H. Hofmann, ,,GUI and Command-line-Conflict or Synergy?“ *Computing Science and Statistics*, str. 246–253, 1999.

[16] (2010, sep) Command line vs. gui. Dostopno na: http://www.computerhope.com/issues/ch000619.htm

[17] J. Reimer. (2005, may) A history of the gui. Dostopno na: http://arstechnica.com/old/content/2005/05/gui.ars

[18] M. Tuck, ,,The real history of the gui,“ *SitePoint*, zv. 13, 2001. Dostopno na: http://articles.sitepoint.com/article/real-history-gui

[19] B. A. Myers, ,,A brief history of human-computer interaction technology,“ *interactions*, zv. 5, št. 2, str. 44–54, 1998.

[20] D. C. Engelbart, ,,The demo,“ dec 1968. Dostopno na: http://sloan.stanford.edu/MouseSite/1968Demo.html

[21] J. Johnson, T. Roberts, W. Verplank, D. Smith *in sod.*, ,,The xerox star: A retrospective,“ *IEEE Computer*, zv. 22, št. 9, str. 11–26, 1989.

[22] N. Mehta, ,,A flexible human machine interface,“ *MA Sc. Thesis, Department of Electrical Engineering, University of Toronto*, 1982.

[23] L. Nakatani in J. Rohrlich, ,,Soft machines: A philosophy of user-computer interface design,“ v *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, 1983, str. 19–23.

[24] Speakers jeff han: Human-computer interface designer. Dostopno na: http://www.ted.com/speakers/jeff_han.html

[25] J. Han. (2006) Multi-touch interaction research. Dostopno na: http://cs.nyu.edu/~jhan/ftirtouch/

[26] M. Buchanan. (2008, aug) Giz explains: The magic behind touchscreens. Dostopno na: http://gizmodo.com/5036516/giz-explains-the-magic-behind-touchscreens

[27] Comparing touchscreen technologies. Dostopno na: http://www.touchscreens.com/intro-touchtypes.html

[28] (2010, sep) Touchscreen. Dostopno na: http://en.wikipedia.org/wiki/Touchscreen

[29] M. Kodelja, ,,Tehnologija zaslonov »multi-touch«,“ *Moj mikro*, feb 2008. Dostopno na: http://www.mojmikro.si/v_srediscu/tehnologije/tehnologija_zaslonov_multi-touch

[30] G. Kaindl. (2008, aug) Touché multitouch framework - introduction. Dostopno na: http://gkaindl.com/software/touche/videos

[31] (2010, aug) Event handling guide for ios. Dostopno na: http://developer.apple.com/library/ios/#documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/Introduction/Introduction.html

[32] M. Kaltenbrunner, T. Bovermann, R. Bencina, in E. Costanza, ,,Tuio - a protocol for table based tangible user interfaces,“ v *Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation (GW 2005), Vannes, France*, 2005.

[33] I. za slovenski jezik Frana Ramovša ZRC SAZU, *Slovar slovenskega knjižnega jezika*, V. Likar, ured. Slovenska akademija znanosti in umetnosti, 2000.

[34] K. Kim, T. Kulkarni, in N. Elmqvist, ,,Interaction workspaces: Identity tracking for multi-user collaboration on camera-based multi-touch tabletops,“ *Collaborative Visualization on Interactive Surfaces-CoVIS'09*, zv. 1, št. P2, str. 1.

[35] J. Sch
"oning, M. Rohs, in A. Kruger, ,,Spatial authentication on large interactive multi-touch surfaces," *IEEE Tabetop*, 2008.

[36] J. Spadaccini. (2010, apr) Case study of l.a. zone multitouch, multiuser table. Dostopno na: http://www.exhibitfiles.org/la_zone_multitouch_multiuser_table

[37] J. Eden. (2009, apr) Designing for multi-touch, multi-user and gesture-based systems. Dostopno na: http://www.drdobbs.com/architecture-and-design/216402697

[38] J. R. Townsen in Y. Kiriaty. (2007) Windows 7 multi touch overview. Dostopno na: http://channel9.msdn.com/blogs/yochay/windows-7-mutli-touch-overview

[39] Y. Kiriaty, ,,Multitouch capabilities in windows 7," *MSDN Magazine*, zv. August 2009, 2009. Dostopno na: http://msdn.microsoft.com/en-us/magazine/ee336016.aspx

[40] W. T. Team. (2009, mar) Engineering windows 7. Dostopno na: http://blogs.msdn.com/b/e7/archive/2009/03/25/touching-windows-7.aspx

[41] P. Miller. (2009, feb) Windows 7 multitouch: it's a gimmick (for now). Dostopno na: http://www.engadget.com/2009/02/05/windows-7-multitouch-its-a-gimmick-for-now/

[42] (2009, sep) Mac os x snowleopard release notes. Dostopno na: http://developer.apple.com/library/mac/#releasenotes/Cocoa/AppKit.html

[43] (2010, okt) Htc hero. Dostopno na: http://en.wikipedia.org/wiki/HTC_Hero

[44] (2009, okt) Android 2.0, release 1. Dostopno na: http://developer.android.com/sdk/android-2.0.html

[45] (2010, July) Android 2.2 platform. Dostopno na: http://developer.android.com/sdk/android-2.2.html

[46] (2010, mar) Safari web content guide. Dostopno na: http://developer.apple.com/library/safari/#documentation/appleapplications/reference/safariwebcontent/handlingevents/handlingevents.html

[47] (2010, aug) Mozilla firefox 4 beta release notes. Dostopno na: http://www.mozilla.com/en-US/firefox/4.0b3/releasenotes/

[48] (2010, sep) Flash player 10.1 release notes. Dostopno na: http://kb2.adobe.com/cps/838/cpsid_83808.html

[49] C. Cantrell. (2009, nov) Multitouch and gesture support on the flash platform. Dostopno na: http://www.adobe.com/devnet/flash/articles/multitouch_gestures.html

[50] (2010, okt) ios (apple). Dostopno na: http://en.wikipedia.org/wiki/IOS_(Apple)

[51] (2010, aug) ios technology overview. Dostopno na: http://developer.apple.com/library/ios/#documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html

[52] (2010, sep) Apple introduces new ipod touch. Dostopno na: http://www.apple.com/pr/library/2010/09/01ipodtouch.html

[53] (2010, okto) iphone. Dostopno na: http://en.wikipedia.org/wiki/IPhone

[54] (2010, sep) Apple a4. Dostopno na: http://en.wikipedia.org/wiki/Apple_A4

[55] (2010) Technical specifications for ipod touch. Dostopno na: http://www.apple.com/ipodtouch/specs.html

[56] (2010, okt) ipod touch. Dostopno na: http://en.wikipedia.org/wiki/IPod_Touch

[57] (2010) ipad technical specifications. Dostopno na: http://www.apple.com/ipad/specs/

[58] (2010, aug) Uiresponder class reference. Dostopno na: http://devworld.apple.com/library/ios/#documentation/UIKit/Reference/UIResponder_Class/Reference/Reference.html

[59] D. Wixon. (2008) Ux week 2008 - nui principals. Dostopno na: http://vimeo.com/2893051

[60] (2010, sep) User interface design. Dostopno na: http://en.wikipedia.org/wiki/User_interface_design

[61] (2010, okt) User-centered design. Dostopno na: http://en.wikipedia.org/wiki/User-centered_design

[62] D. Saffer. (2008) Ux week 2008: Tap is the new click. Dostopno na: http://vimeo.com/2815881

[63] P. Brandl, J. Leitner, T. Seifried, M. Haller *in sod.*, ,,Occlusion-aware menu design for digital tabletops,“ v *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*. ACM, 2009, str. 3223–3228.

[64] D. Saffer, *Designing Gestural Interfaces: Touchscreens and Interactive Devices*, illustrated edition izd. O’Reilly Media, Inc., 2008.

[65] Photos 1.0 sp1. Dostopno na: http://technet.microsoft.com/en-us/library/ee692031(Surface.10).aspx

[66] (2009, mar) Table microsoft surface - application photos en action. Dostopno na: http://www.youtube.com/watch?v=7VklPI5kliU

[67] (2008, jan) Microsoft surface demo @ ces 2008. Dostopno na: http://www.youtube.com/watch?v=Zxk_WywMTzc

[68] (2009, jul) Welcome to surfacetwitter, a multi-touch twitter client for the microsoft surface! Dostopno na: http://blogs.msdn.com/b/swiss_dpe_team/archive/2009/07/16/welcome-to-surfacetwitter-a-multi-touch-twitter-client-for-the-microsoft-surface.aspx

[69] (2010, okt) Twitter. Dostopno na: http://en.wikipedia.org/wiki/Twitter

[70] R. Fielding, ,,Architectural styles and the design of network-based software architectures,“ Doktorska disertacija, Citeseer, 2000.

[71] (2010, okt) Api documentation. Dostopno na: http://dev.twitter.com/doc