Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Operacijski sistemi 2

Končno poročilo seminarske naloge MAIRM

Avtorja:
Matej Jakop, 63070025
Gregor Kališnik, 63070041

Ljubljana, 25. maj 2010

Kazalo

1 Opis izbrane teme in problemskega področja

Tema seminarske naloge je izdelava virtualne naprave vhodno/izhodnega sistema za popolni nadzor računalnika preko mobilnega telefona pri čemer je povezava med njima vspostavljena preko bluetooth vmesnika. Sama tema seminarske naloge obsega naslednja področja:

- komunikacij med napravami,
- poznavanja različnih platform (mobilne, računalniške),
- strojnega učenja (osnove),
- poznavanje operacijskih sistemov

2 Izbrani cilji, zahteve ter omejitve

Cilj seminarske naloge je izdelati delujoč sistem in se pri tem naučiti nove stvari iz področij, ki jih seminarska naloga obsega. Bolj konkretni cilji sistema so:

- omogočiti enostavno uporabo sistema,
- s pomočjo premikov telefona upravljamo z računalniško miško (premiki levo, desno, gor, dol),
- omogočiti uporabniku drsenje (scrolling) preko strani s preprostim nagibanjem telefona naprej oziroma nazaj,
- omogočiti uporabniku vnos krajših besedil(za daljša besedila telefonska tipkovnica ni preveč primerna),
- omogočiti uporabniku, da računalnik "nauči"kako naj bi neka kretnja izgledala,
- tako "naučen"računalnik naj bi potem ob uporabnikovi izvedbi kretnje naredil določeno akcijo (npr. zagnal določen program) in
- vse skupaj narediti z odprtokodnimi orodji/knjižnjicami ipd.

Ob vseh zgoraj podanih ciljih se pojavi vprašanja kakšnim zahtevam je potrebno zadostiti, da bo uporaba končne rešitve sploh možna. Zahteve za delovanje sistema maj bi bile sledeče:

- spodoben računalnik z urejeno Java podporo,
- bluetooth vmesnik,
- mobilni telefon z merilnikom pospeška: v najinem primeru sva se omejila na telefona znamke Nokia s Symbian operacijskim sistemom, kljub temu je prenosljivost na ostale mobilne platforme trivialna.

Kot je opaziti iz zahtev rešitev deluje na vseh operacijskih sistemih, kjer je podpora za programski jezik Java. Samo delovanje sistema je bilo testirano na Windows in Linux platformi, kjer je deloval brez težav.

Po preučitvi podobnih rešitev na spletu sva prišla do ugotovitve, da rešitve, ki bi vključevala vse najine cilje še ni na voljo. Obstoječe rešitve omogočajo več ali manj samo katero izmed funkcionalnosti/ciljev, ki sva si jih midva zadala. Ena izmed takšnih rešitev je WiiGee. WiiGee omogoča učenje in razpoznavo kretenj. Na žalost/srečo je ta rešitev osnovana za lastnike igralne konzole Wii.

3 Arhitektura rešitve

Rešitev je zasnovana s pomočjo principa strežnik-odjemalec. Strežnik in odjemalec med sabo komunicirata preko MAIM protokola, ki je osnovan na JSON formatu ter je predstavljen v nadaljevanju poročila.

Na strežnikovi (računalnikovi) strani teče MAIRM strežniška aplikacija. Njene naloge so:

- ustvari bluetooth service (z imenom MAIRM) preko katerega se MAIRM odjemalec poveže do same aplikacije,
- poskrbi za pravilno prepoznavo odjemalčevih akcij:
  - premikanje miške po zaslonu,
  - pomikanje po straneh (scrolling),
  - uporaba mobilne tipkovnice,
  - prepoznavanje 3D kretenj
- skrbi za pravilen odziv glede na odjemalčevo akcijo:
  - če zazna zahtevo po premiku miške, premakne miško
  - če zazna zahtevo po pomikanje preko strani, simulira potreben dogodek
  - če zazna zahtevo po pritisku tipke, simulira pritisk
  - če zazna 3D kretnjo naredi akcijo (akcija je določena v XML datoteki), ki je določena za njo:

    * požene nek program
    * pritisne neko kombinacijo tipk

Na mobitelovi strani oziroma odjemalčevi strani teče MAIRM odjemalčeva aplikacija. Njene naloge so:

- zajemanje podatkov iz merilnika pospeška vgrajenega v sam mobitel,
- zajemanje pritiskov tipk,
- posredovanje zajetih podatkov po MAIRM protokolu vse do MAIRM strežniške aplikacije,
- skrbi za preklapljanje med naslednjimi načini delovanja:
  - Mouse mode,
  - Scrolling mode,
  - Gesture mode
- izris uporabniškega vmesnika za touch screen telefone.

Obe komponenti sistema skupaj tvorita delujočo osnova s katero se (lahko) doseže zastavljene začetne cilje.

3.1 MAIRM protokol

Sam protol je v JSON formatu in teče preko Bluetooth Serial Port ter je zelo enostaven. V trenutni verziji je podprta samo enosmerna komunikacija, saj potrebe po dvosmerni komunikaciji (zaenkrat) ni bilo.

3.1.1 Uporabljeni podatkovni tipi

boolean = true ali false

double = decimalno število

buttonState = up ali down

buttonStateMiddle = up ali down ali scrolling
scrolling pomeni, da telefon preide iz načina nadziranja miške na način nadziranja scrollerja preko nagibanja

keys = ASCII ali ENTER ali BACKSPACE ali ...
Dodatne tipke so stvar dogovora in potrebe

modifiers = SHIFT in/ali ALT in/ali CONTROL

3.1.2 Format za GESTURE mode

Oblika:

```
1 {"gesture":{"x":"double","y":"double","z":" double ","start":"boolean","end
    ":"boolean"}}
```

Listing 1: Oblika formata za GESTURE mode

Razlaga:
x = vrednost akselometra po x-osi.
Privzeta vrednost: 0.0

y = vrednost akselometra po y-osi.
Privzeta vrednost: 0.0

z = vrednost akselometra po z-osi.
Privzeta vrednost: 0.0

start = določa ali prejeti podatki pomenijo začetek gesture ali ne. Vrednost true ima lahko samo pri prvem sporočilu nekega gesture.
Privzeta vrednost: false

end = določa ali prejeti podatki pomenijo konec gesture ali ne. Vrednost true ima lahko le pri zadnjem sporočilu za nek gesture.
Privzeta vrednost: false

3.1.3 Format za MOUSE mode

Oblika:

```
1 {"mouse":{"x":"double","y":"double","z":"double","leftbutton":"buttonState"
    ,"middlebutton":"buttonStateMiddle","rightbutton":"buttonState"}}
```

Listing 2: Oblika formata za MOUSE mode

Razlaga:
x = vrednost akselometra po x-osi.
Privzeta vrednost: 0.0

y = vrednost akselometra po y-osi.
Privzeta vrednost: 0.0

z = vrednost akselometra po z-osi.
Privzeta vrednost: 0.0

leftbutton = levi miškin gumb
Privzeta vrednost: up

middlebutton = srednji miškin gumb

Privzeta vrednost: up

rightbutton = desni miškin gumb
Privzeta vrednost: up

3.1.4 Format za KEYBOARD mode

Oblika:

```
1 {"keyboard": {"key":"keys","modifiers":[modifiers]}}
```

Listing 3: Oblika formata za KEYBOARD mode

Razlaga:
key = tipka, ki naj bi bila pritisnjena na računalniški tipkovnici. Posebni znaki morajo biti zapisani s polnim imenom (npr. "). modifiers = v array se navede katere tipke so dodatno še pritisnjene. Npr. za veliki A moramo dodati v modifiers element ŠHIFT".

4 Izvedba rešitve in problemi pri tem

Sama izvedba rešitve na prvi pogled izgleda čisto enostavna. Ko pa se zatopimo v delo in poskušamo rešitev dejansko izvesti, pa pridemo do različnih problemov, ki jih je potrebno rešiti. Midva sva prišla do naslednjih problemov:

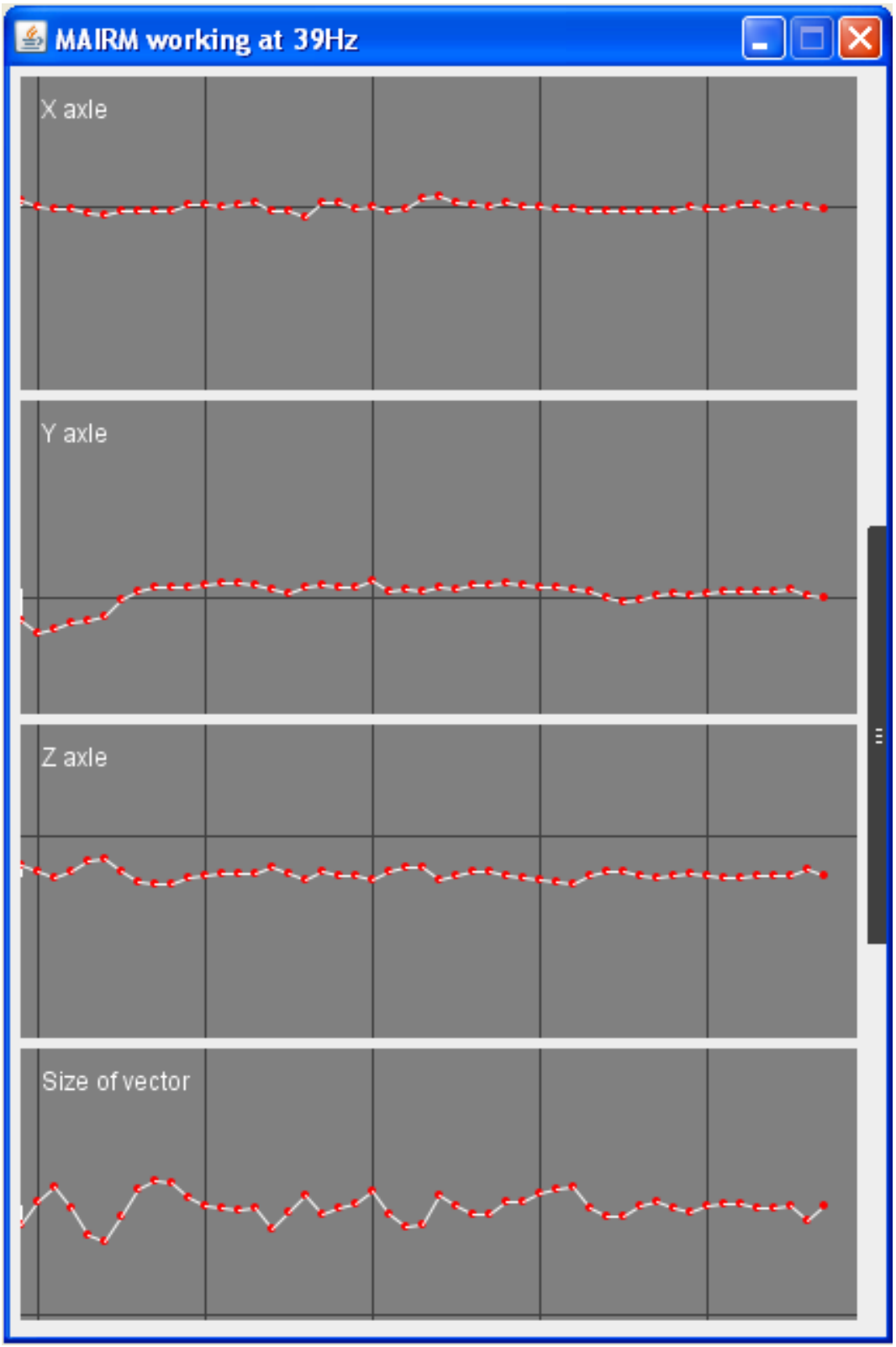
1. Vzpostavitev povezave med različnimi napravami
2. Pridobitev podatkov iz pospeškometra v mobilnem telefonu
3. Simulacija delovanja miške in tipkovnice
4. Učenje in razpoznava kretenj

Vse naštete probleme, ki so bolj podrobno opisani v nadaljevanju, sva uspešno rešila.

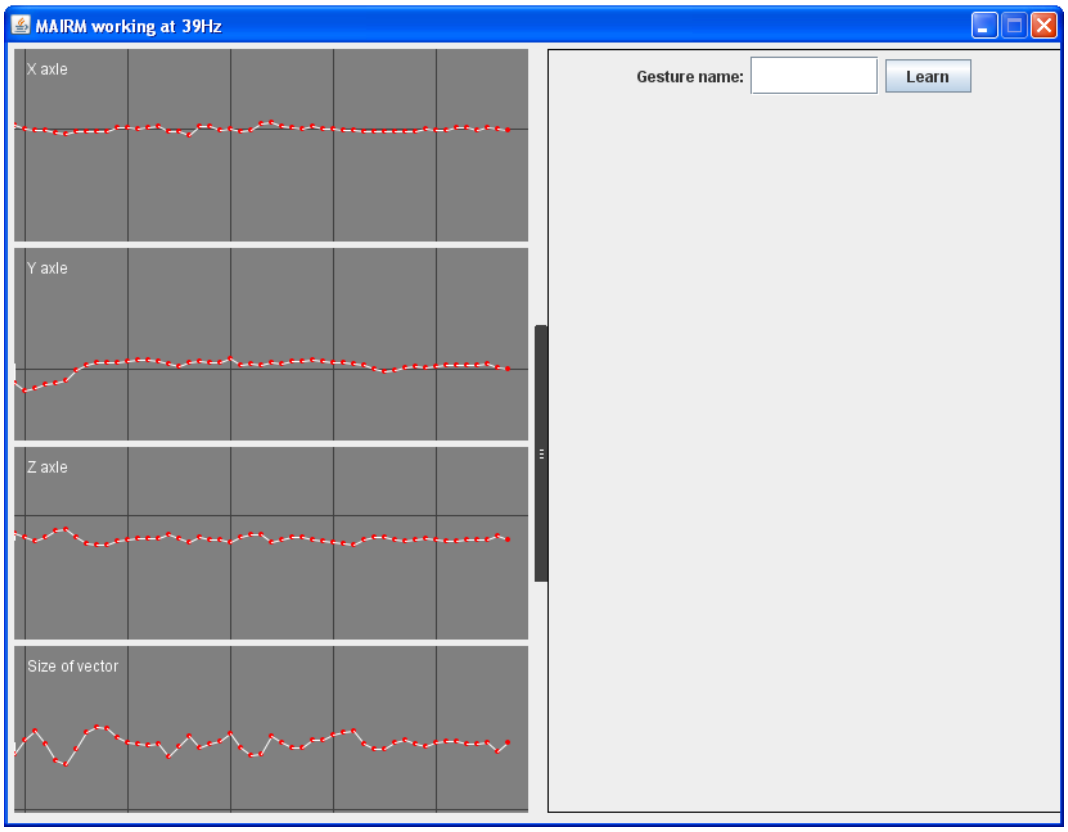
4.1 Pregled uporabniškega vmesnika

4.1.1 Osnovno okno

Osnovno okno, ki ga odbimo ob zagonu:



Razširjeno okno za vnos/učenje kretnje: Raširjeno okno dobimo po kliku na temno sivi zavihek.



4.1.2 Delovanje v ozadju

Po minimizaciji glavnega okno gre sistem v delovanje v ozadju. Ko je sistem v tem načinu delovanja se na mestu poleg ure pojavi ikonca, ki predstavlja program.

Ko se v sistem poveže telefon ikonca spremeni svojo barvo in s tem je ponazorjena uspešna povezava.

4.1.3 Uporabniški vmesnik na mobilnem telefonu

Uporabniški vmesnik na mobilnem telefonu ni nič posebnega, saj po tem ni potrebe. Potrebno bistvo se skriva v ozadju.

4.2 Vzpostavitev povezave med različnimi napravami

4.2.1 Opis problema:

Imamo dve napravi, ki sta obe sposobni bluetooth komunikacije. Pri tem se pojavijo naslednja vprašanja:

1. Katera izmed naprav naj bo gospodar, ki prejema zahteve po povezavi od druge naprave?

2. Kako programsko, hitro in čimbolj enostavno vzpostaviti povezavo s pogojem, da to deluje na vseh sistemih?

4.2.2 Opis rešitve:

1. Ta problem sva rešila zelo enostavno. Ker je potrebno za delovanje sistema oziroma samo sporočanje premikov telefona zagnati aplikacijo na mobilniku (predpostavljava, da je na računalniku že pognana), sva se odločila, da naj bo mobilnik tisti, ki poda zahtevo po vzpostavitvi povezave. Tako je tudi rešeno, da ni potrebno aplikaciji na računalniku skenirati na okoli za napravami, ki imajo MAIRM sposobnost. S tem se prihrani nekaj tudi na energijski porabi (ki je sploh na prenosnih napravah ključnega pomena), ker se povezava vzpostavlja in ostaja aktivna samo takrat, ko je dejansko potrebna.

2. Naslednji problem je že bil malce težji. Težji iz vidika, da želiva to početi preko programskega jezika Java in to na vseh platformah. Po brskanju na spletu sva prišla do knjižnjice BlueCove, ki omogoča točno to kar potrebujeva. Potem je bilo potrebno samo še spoznati uporabo te knjižnjice in standard, ki je osnova implementacije. Končna implemetacija povezave je bila rešene na sledeči način:

   (a) Strežniška aplikacija na računalniku ob zagonu kreira storitev z imenom MAIRM. Storitev teče na vrhu protokola Bluetooth Serial Port Profile,

   (b) Uporabnik po zagonu MAIRM mobilne aplikacije dobi na voljo izbiro bluetooth naprave ter nato še izbiro storitve (v primeru, da bi bilo storitev več ali pod različnimi imeni)

   (c) Če je vse pravilno storjeno, se komunikacija vzpostavi.

4.3 Pridobitev podatkov iz pospeškometra v mobilnem telefonu

4.3.1 Opis problema:

Že od samega začetka sva vedela katere mobilne telefone bova uporabila pri najini seminarski nalogi. Preostala je le še izbira programskega jezika v katerem se je mogoče dokopati do teh podatkov in kako to storiti v izbranem programskem jeziku.

4.3.2 Opis rešitve:

Za programiranje na Symbian platformi imamo na voljo naslednje programske jezike:

- Java
- C/C++
- Python

Od vseh možnosti najprej odpade programski jezik Java, ker nima (še) dodanega aplikacijskega vmesnika do senzorjev, ki so v telefonu. Zaradi enostavnosti implementacije sva izbrala programski jezik Python, čeprav se v samem interpreterju jezika in dokumentaciji pojavljajo napake. Osnovna implementacija v Pythonu je naslednja:

1. uporabi razred sensor in mu podaj funkcijo/metodo za katero želiš, da se kliče ob dogodkih vzorčenja podatkov iz senzorja
2. v klicani metodi pošlji podatke na računalnik

Ob tem velja omeniti, da je potrebno rešiti izgubljanje podatkov v primeru napake pri pošiljanju podatkov preko bluetooth vmesnika.

4.4 Simulacija delovanja miške in tipkovnice

4.4.1 Opis problema:

Pri seminarski nalogi sva prišla do točke, ko je bilo potrebno začeti simulirati delovanje miške in tipkovnice. Torej, želela sva programsko operacijskemu sistemu sporočiti, da se je miška premaknila desno, da se je zgodil klik,....

4.4.2 Rešitev problema:

Sprva sva mislila, da bo v Javi to problem, ki ga bo potrebno reševati preko JNI (java native interface). Kar bi celotno implementacijo malce otežilo. Na srečo sva prišla do spoznanja, da obstaja razred Robot, ki omogoča vse kar sva potrebovala. Z razredom Robot je mogoče:

- Zajemati zaslonsko sliko,
- Premikati miško,
- Pritiskati tipke,
- Izvajati drsenje po straneh(scrollanje)
- ...

Dobra stvar tega razreda je tudi ta, da je že vsebovan v standardni Javi.

4.5 Učenje in razpoznava kretenj

4.5.1 Opis problema:

Iz prejetih podatkov, ki jih dobiva iz mobilnega telefona je bilo potrebno nekako zagotoviti, da bo sistem znal prepoznati, kaj je uporabnik izvedel. Npr. uporabnik v zraku nariše krog in sistem ga uspešno prepozna. Sprva se nama je problem zdel težek in ob enem zanimiv, kar je samo dal dodatno motivacijo za njegovo razrešitev.

4.5.2 Rešitev problema:

Problem sva želela rešiti na več bolj ali manj uspešnih postopkov:

1. **Poskus odkrivanje v katero izmed smeri se gibje telefon (gor, dol, levo, desno, postrani v levo...):** Delovanje sistema na tak način bi bilo bolj slabo, čeprav je v teoriji vsaj v enem delu izgledal obetavno (ne bi prišlo do napačnih detekcij, detekcija bi bila ali pa ne). Problem bi bil v tem, da bi težko prepoznaval krivulje ter, da bi bilo pri izvajanju potrebno zagotoviti ostre prehode med posameznimi smermi (uporabnik bi moral delat zelo nazorne gibe).

2. **Uporaba algoritma strojnega učenja k-nearest neighbours:** Idejo za ta algoritem sva dobila po pregledu diplomske naloge študenta iz naše fakultete. Dodatno motivacijo za uporabo sledečega algoritma sva dobila še zaradi zelo dobrih rezultatov prepoznavanja, ki jih je dosegel pri svojem delu.

Torej za rešitev problema učenja in razpoznave kretenj sva izbrala algoritem strojnega učenja k-nearest neighbours. Edina slabost te izbire je, da algoritem ni med najhitrejšimi pri sami razpoznavi. Vzrok tega je, da se sistem ne uči ampak šele pri razpoznavi poskuša klasificirati kateremu razredu (kretnji) pripada nek seznam vrednosti. Kljub temu, se nama je zdel dobra naložba. Sam algoritem pri svojem delovanju uporablja računanje razdalje med dvema zaporedjema. Ker se vrednosti pri izvajanju kretnje lahko časovno razlikujejo (nek gib ne izvedemo vedno časovno čisto enako) sva za merjenje razdalje uporabila algoritem DTW (dynamic time warp). Osnovna implementacija obeh algoritmov je dokaj enostavna, a počasna. Če bi želela hitro implementacijo bi bilo potrebno precej več dela za dosego spodnje meje implementacije. Zato sva raje uporabila že obstoječo rešitev s tega področja, in sicer implementacijo v java programski knjižnjici JML (java machine learning). Po preučitvi uporabe knjižnjice je bil problem rešen.

5 Morebitno nadaljnje delo

Projekt je objavljen na google code (http://code.google.con/p/mairm/) in kot tak je na voljo vsem, ki bi radi kaj dodali/spremenili. Torej odpte so vse opcije za sodelovanje drugih ljudi in tem uresničevanje njihovih idej. Kot uporabnika in razvijalca sistema sva prišla do ugotovitve, da bi bilo dobro za boljšo uporabniško izkušnjo narediti še sledeče stvari:

1. V XML datoteko uvesti pogoje glede na trenutno aktivno aplikacijo. Tako bi se lahko zagotovilo, da bi ista kretnja imela različen pomen v različnih programih. Npr. kretnja v desno bi v predvajalniku glasbe prestavila na drugo pesem, medtem ko v pregledovalniku slik na naslednjo sliko.

2. Nekako poskrbeti, da bi sistem sam izboljševal svoje znanje o kretnjah (brez sodelovanja uporabnika pri temu).

3. Prestaviti delovanje programa na višji prioritetni nivo (ker se drugače zna zgoditi, da ob veliki zasedenosti računalnika celoten sistem deluje upočasnjeno in ne tako kot prava miška), čeprav je ta pojav redek.

Bistveno za nadaljnje delo je to, da morava poskrbeti, da bo več ljudi spoznalo najin projekt in ga morebiti začelo uporabljati. Le-tako bo smiselno dodajati nove funkcionalnosti in izboljšave. Če neka rešitev nima uporabnikov, vsak (pa naj bo še tako dober) projekt slej ko prej umre. Kar je včasih škoda.

6 Navodila za uporabo

6.1 Zahteve za delovanje sistema

Zahteve za delovanja sistema so skromne. Potrebno je imeti le:

- Operacijski sistem Windows/Linux/Mac
- računalnik z bluetooth vmesnikom
- mobilni telefon znamke Nokia s Symbian S60 3rd Edition ali Nokia s Symbian S60 5rd Edition, nameščeno python podporo (pyS60) in prisotnost merilnika pospeška v samem telefonu

Zadnja točka seznama se bo lahko s časom malce posplošila. V prihodnosti projekta bo verjetno dovolj, da bo uporaben poljuben telefon z merilnikom pospeška.

6.2 Potek namestitve

6.2.1 Mobilni telefon

Možnih je več opcij:

1. V primeru že nameščenega pyS60 na mobilni telefon prenesemo (preko bluetooth ali kako drugače) samo mairm.py datoteko, ki se nahaja v paketu za inštalacijo. Na mobilnem telefonu jo shranimo na pomnilniško kartico v imenik data/python. Tako bo datoteka vidna v python vmesniku do interpreterja. Namestitev končana.
2. V primeru, da je v namestitvenemu paketu prisotna datoteka mairm.sis le-to prenesemo na mobilni telefon in poženemo namestitev ter sledimo korakom na zaslonu.

6.2.2 Računalnik

Na računalniku je namestitev sila preprosta. Potrebno je samo razpakirati namestitveni paket ali pognati namestitveno datoteko.

6.3 Povezava mobilnega telefona in računalnika ter pričetek delovanja

Za vzpostavitev povezave je potrebno slediti naslednjim korakom:

1. Vklopimo bluetooth vmesnik v kolikor je izključen
2. Sparimo telefon in računalnik v primeru, daj eto potrebno storiti

3. Poženemo program MAIRM (če smo namestili z namestitvenim programom) oziroma v direktoriju namestitvenega pakete v konzoli poženemo ukaz

   java -jar mairm.jar

4. Na mobilnem telefonu poženemo aplikacijo MAIRM (če smo namestili z mairm.sis) oziroma Python 2.x.x. Če smo pognali Python 2.x.x moramo v njem še odpreti skripto. To storimo tako, da izberemo öptionsïn nato ŕun script∴ Poiščemo datoteko mairm.py in jo izberemo.

5. Prikaže se nam seznam bluetooth naprav in izberemo naš računalnik na katerem teče mairm. Po potrditvi tega nas bo telefon morebiti še povprašal po storitvi na katero se želimo povezati. Enostavno izberemo MAIRM.

6. Če so bili vsi koraki uspešno izvedeni, bi se sedaj morala miška začeti premikati po zaslonu v skladu s premikanjem telefona.

6.4 Uporabni napotki pri uporabi

6.4.1 Miško premikamo ekranu v skladu s temi načeli:

- Nagib telefona naprej, premakne miško navzgor,
- Nagib telefona nazaj, premakne miško navzdol,
- Nagib telefona levo, premakne miško levo,
- Nagib telefona desno, premakne miško desno,
- Pritisk leve akcijske tipke na telefonu povzroči levi klik,
- Pritisk desne akcijske tipke na telefonu povzroči desni klik
- Pritisk na tipko za izbiro na telefonu povzroči:
  - Preklop na način drsenja (scrolling mode) ali
  - Začetek zajemanja izvajanja kretnje

  Katera stvar se bo zgodila je odvisno od tega, koliko časa držimo tipko. Če jo držimo dlje kot 100ms potem začne izvajati način kretnje (gesture mode), drugače pa enostavno anredi preklop.
- Pritisk tipke urejanje (edit, svinčnik) povzroči, da se pojavi okno v katerega vnesemo besedilo, ki se naj izpiše na računalniku.

6.4.2 Novo kretnjo posnamemo na sledeči način:

1. Odpremo glavno okno aplikacije (če se aplikacija slučajno skriva v System tray področju)
2. Aplikacijo razširimo s klikom na sivi gumb, da dobimo opcijo vnosa imena kretnje
3. Vnesemo ime kretnje ter kliknemo na Learn
4. S pomočjo telefona (seveda mora biti najprej vzpostavljena povezava med mobitelom in računalnikom) izvedemo kretnjo kot običajno
5. Če še nismo tega storili vsaj 5x se vrnemo 3. korak in ponovimo z istim imenom
6. Če še nimamo shranjeni vsaj 2 kretnji, ponovimo postopek vendar tokrat z drugo kretnjo.
7. Ko vse opravimo kot je bilo opisano v prejšnjih korakih, bi moral sistem začeti prepoznavati uporabnikove definirane kretnje

6.4.3 Nastavitev kaj naj sistem naredi ob izvedbi določene kretnje

1. Odpremo datoteko actions.xml, ki se nahaja v istem direktoriju, kot je bil nameščen sam program.
2. V datoteki bi že moral biti prisoten seznam vseh kretenj, ki jih sistem pozna.
3. Odločiti se je potrebno kaj bi radi, da sistem naredi:
   - izvede nek program
     V tem primeru med značke <exec> in </exec> vnesemo pot in ime programa.
   - pritisne določeno kombinacijo tipk
     V tem primeru med značke <keys> in </keys> dodamo nove tipke in sicer podobno kot v em primeru:

     <key delayRelease="false">F1</key>

     To pomeni, da se bo v primeru izvedbe kretnje simuliral pritisk tipke F1. Atribut delayRelease pa pomeni ali naj sistem drži tipko vse dokler se ne izvedejo še preostali pritiski tipk. Še en primer tega za pošiljanje windows+r (požene okno za zagon programa):

     <key delayRelease="true">WINDOWS</key>

     <key delayRelease="false">r</key>

     Seznam vseh tipk, ki so na voljo za takšen vnos je zapisan v datoteki keyboard_keys_available.txt po prvem zagonu MAIRM programa.
4. Datoteko shranimo
5. Kliknemo z desno miškino tipko na ikonco mairm programa in izberemo opcijo R̈eload gesture action bindings:

6.4.4 Izboljšava zaznavanja kretenj

Če se opazi, da sistem dve kretnji pogosto zamenjuje potem lahko to rešite dokaj enostavno. Potrebno je posneti samo več ponovitev problematičnih kretenj (pri številu 15x ponovitev sistem že zelo dobro loči tudi skoraj čisto enake kretnje, npr. krog v levo in krog v desno).

Literatura

[1] Kononenko, I. (1997). Strojno učenje. Ljubljana: Fakulteta za računalništvo in informatiko.

[2] BlueCove documentation.[Online].[Citirano: 2. april 2010; 18:00]. Dostopno na spletnem naslovu: http://code.google.com/p/bluecove/wiki/Documentation

[3] Java machine learning.[Online].[Citirano: 2. april 2010; 19:00]. Dostopno na spletnem naslovu: http://java-ml.sourceforge.net/content/getting-started

[4] Strle, B. (2008). Prepoznavanje človeških gibov s pospeškometri in strojnim učenjem. Ljubljana: Fakulteta za računalništvo in informatiko.