Prirodoslovno – matematički fakultet
Matematički odsjek
Horvatovac 102a
Zagreb

Završni projektni rad iz Multimedijskih sustava:

Dvosmjerna komunikacija između mobilnog uređaja i računala

Autori:
Jure Milašinović
Email:jure.pmf.hr@gmail.com
Mario Pavlović
Email:pavlek1817@gmail.com
Voditelj kolegija:
Dr. sc. Goran Igaly, v. pred.
Zagreb, 15.2.2016

Sadržaj

1. Uvod:

Na početku bismo opisali što nam je zadatak, tj. što je cilj našeg projekta. Nakon što iskažemo cilj pokušat ćemo korištenjem raznih hardvera i softvera postići naš cilj. Kao glavno uporište izdvojili bi Processing na njemu ćemo graditi naše rješenje.

Cilj nam je realizirati dvije stvari:

- Predavač pokrene prezentaciju na računalu i s njome upravlja preko mobilnog uređaja.
- Predavač pokrene prezentaciju na svom mobitelu koja se šalje računalu i prikazuje na projektoru. Tada korisnik može s njome upravljati preko mobitela ili preko računara.

Hoćemo li to uspjeti, ne znamo. Naš projekt je zamišljen da ga razradimo do neke točke. Nakon toga, ako ne uspijemo doći do cilja, ostavljamo nekim drugim kolegama/icama da nastave s našim radom. Projekt ćemo bazirati na razvoju aplikacije od početka do kraja. Riješit ćemo elementarne stvari, kao što prikaz ekrana mobitela na ekranu računala, uspostave veze između računala i mobilnog uređaja i slično. Zatim ćemo pokušati realizirati neke osnovne naredbe na računalu preko mobitela. Moramo napomenuti da ne znamo koliki nas posao očekuje, tako da je moguće da ne napravimo projekt do kraja ili da ga jako brzo napravimo. Još jedna napomena: Naš dokument prati razvoj aplikacije. Počet ćemo sa elementarnim stvarima. Za naš projekt koristit ćemo biblioteku *oscP5* i program *Control* za mobilni uređaj.

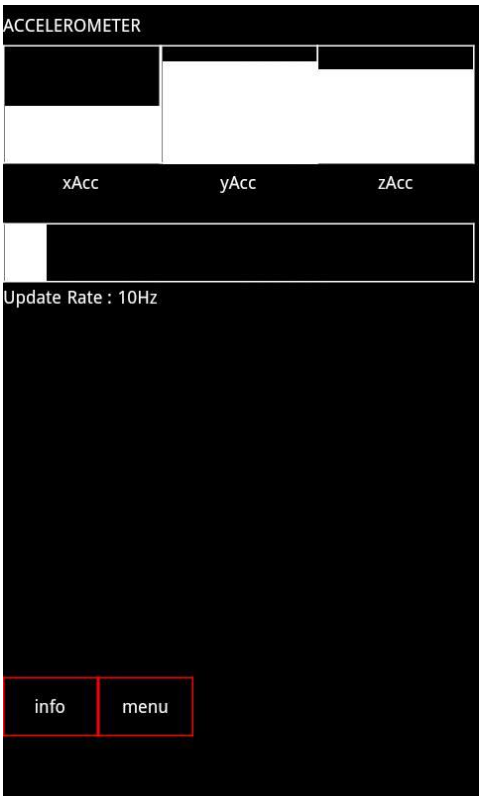
2. Elementarna komunikacija između računala i mobilnog uređaja

Potreban nam je mobilni uređaj na kojem je instaliran *Control* program, računala na kojem je instaliran *Processing* i zajednički wireless Internet na koji su spojeni mobilni uređaj i računalo.

2.1. Akcelerometar

U ovoj točki ćemo pokušati prikazati u *Processing*-u bilo kakvu promjenu lokacije mobilnog uređaja. Za to će nam trebati opcija akcelerometar u *Control* programu. Sučelje programa izgleda kao na slici Slika 2.1.1. Prilikom okretanja mobilnog uređaja šaljemo koordinate mobilnog uređaja u prostoru *Processing*-u i na osnovu njih ispisujemo a.typetag(). Zapravo je riječ o tipu podataka „fff“, „iff“, „fif“, „ffi“, „iif“, „fii“ i „iii“. Pri čemu je f = float, a i = integer.

Slijedi kod samog programa:

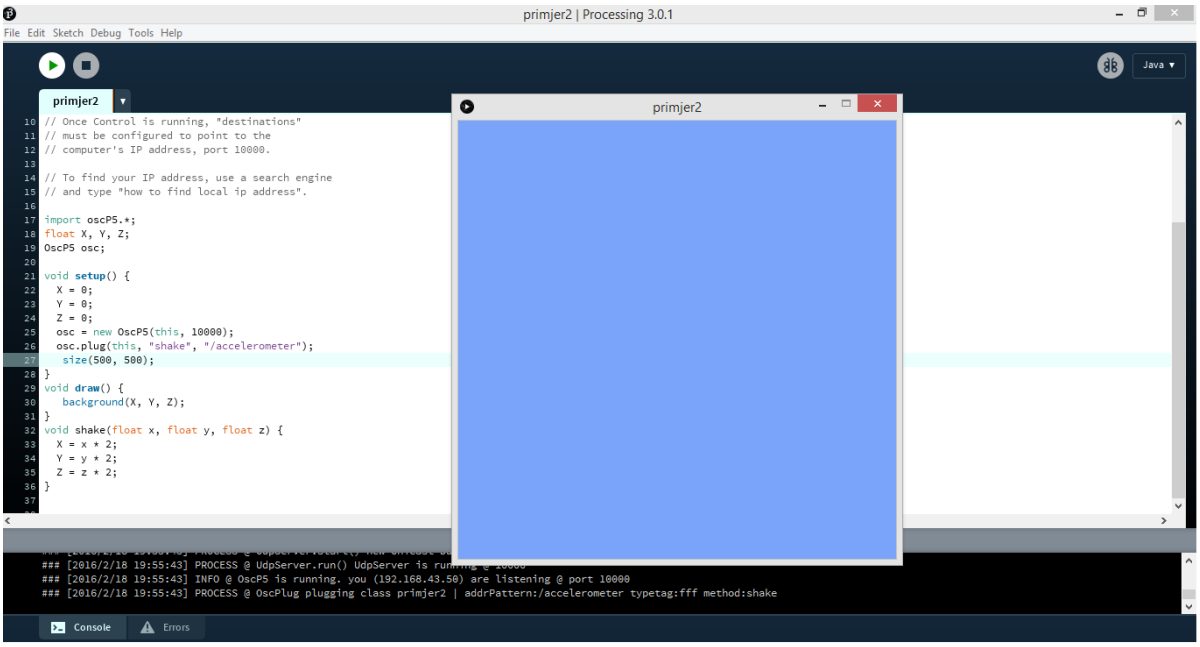


Slika 2.1.1

```
import oscP5.*;

OscP5 osc;

void setup() {
osc = new OscP5(this, 10000);
}
void draw() {
}
void oscEvent(OscMessage a){
println(a.addrPattern() + " , " + a.typetag());
}
```

Bitno je naglasiti da konstruktor OscP5 prima port preko kojeg će komunicirati sa mobilnim uređajima, te je potrebno isti taj port napisati u programu *Control* zajedno sa odgovarajućom IP adresom. U našem slučaju 10000 je port preko kojeg komuniciraju mobilni uređaj i računalo. Program se nalazi u mapi „Akcelerometar“.

Drugi primjer koji smo napravili sa akcelerometrom je promjena boje boju pozadine našeg prozora. Kako to izgleda možete vidjeti na slici Slika 2.2.2.



Slika 2.2.2.

Način na koji smo to napravili je sljedeće. Stvorili smo funkciju shake() koju smo proslijedili funkciji osc.plug(). Funkcija osc.plug() prosljeđuje osc poruku funkciju shake() koja prima tri argumenta koja određuju koordinate mobilnog uređaja u prostoru. Na osnovu koordinata obojimo površinu pozadine našeg prozora. Slijedi kod programa u *Processing*-u*:*

```
import oscP5.*;
float X, Y, Z;
OscP5 osc;

void setup() {
  X = 0;
  Y = 0;
  Z = 0;
  osc = new OscP5(this, 10000);
  osc.plug(this, "shake", "/accelerometer");
   size(500, 500);
}
void draw() {
   background(X, Y, Z);
}
void shake(float x, float y, float z) {
  X = x * 2;
  Y = y * 2;
  Z = z * 2;
}
```

Program se nalazi u mapi „AkcelerometarPozadina“.

Sada kada smo uspostavili komunikaciju računalo – mobilni uređaj želimo kontrolirati slanje podataka.

2.2. Broadcast klijent

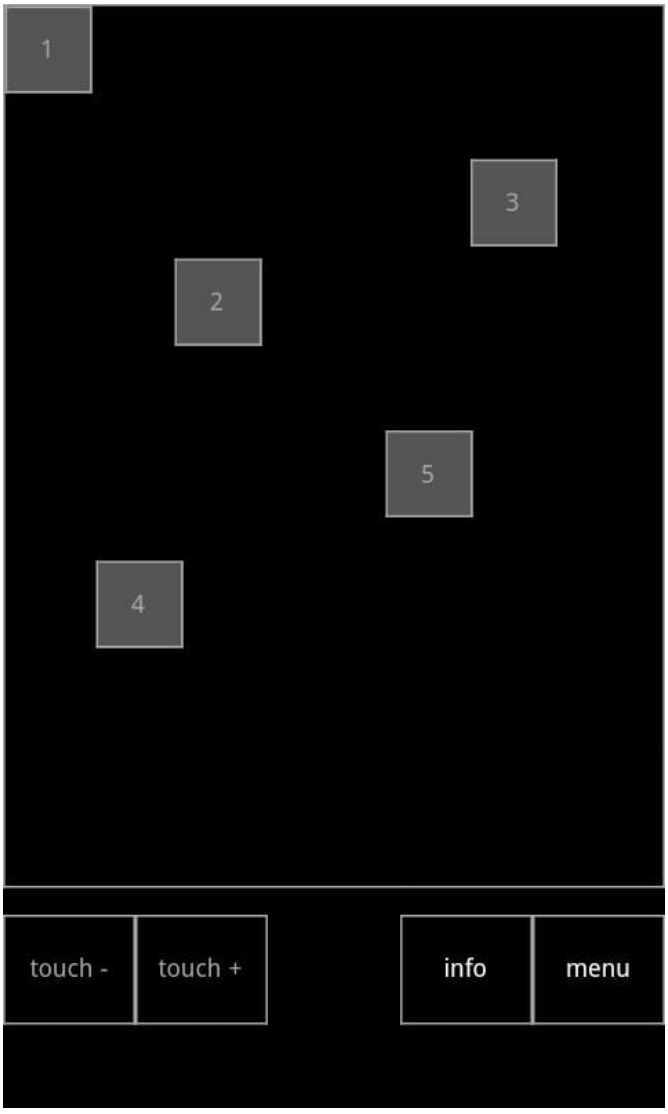
U ovoj točki ćemo pokušati simulirati *Broadcast* klijenta. Naš klijent će biti naš program napisan u *Processing*-u. On će komunicirati sa mobilnim uređajem na portu 10000. I slati sam sebi poruku prilikom klika na miš. Otvaranje konekcije se ostvaruje pritiskom na tipku 'c', a zatvaranje 'd'. To ćemo simulirati ispisom u *Processing*-u. Slijedi kod programa. Napominjem da je kod komentiran, pa nema potrebe dodatno komentirati pojedine dijelove.

```
/**
 * oscP5broadcastClient by andreas schlegel
 * an osc broadcast client.
 * an example for broadcast server is located in the oscP5broadcaster exmaple.
 * oscP5 website at http://www.sojamo.de/oscP5
 */

import oscP5.*;
import netP5.*;
```

```
OscP5 oscP5;

/* a NetAddress contains the ip address and port number of a remote location in the network.
*/
NetAddress myBroadcastLocation;

void setup() {
  size(400,400);
  frameRate(25);

  /* create a new instance of oscP5.
   * 10000 is the port number you are listening for incoming osc messages.
   */
  oscP5 = new OscP5(this, 10000);

  /* create a new NetAddress. a NetAddress is used when sending osc messages
   * with the oscP5.send method.
   */

  /* the address of the osc broadcast server */
  myBroadcastLocation = new NetAddress("192.168.43.50", 10000);
}


void draw() {
  background(0);
}


void mousePressed() {
  /* create a new OscMessage with an address pattern, in this case /test. */
  OscMessage myOscMessage = new OscMessage("/test");
  /* add a value (an integer) to the OscMessage */
  myOscMessage.add(100);
  /* send the OscMessage to a remote location specified in myNetAddress */
  oscP5.send(myOscMessage, myBroadcastLocation);
}


void keyPressed() {
  OscMessage m;
  switch(key) {
    case('c'):
      /* connect to the broadcaster */
      m = new OscMessage("/server/connect",new Object[0]);

    // fulsh() a static method to send an OscMessage straight out of the box without having to
instantiate oscP5.
```

```
      oscP5.flush(m, myBroadcastLocation);
      break;
    case('d'):
      /* disconnect from the broadcaster */
      m = new OscMessage("/server/disconnect",new Object[0]);
      oscP5.flush(m, myBroadcastLocation);
      break;

  }
}


/* incoming osc message are forwarded to the oscEvent method. */
void oscEvent(OscMessage theOscMessage) {
  /* get and print the address pattern and the typetag of the received OscMessage */
  println("### received an osc message with addrpattern "+theOscMessage.addrPattern()+"
and typetag "+theOscMessage.typetag());
  theOscMessage.print();
}
```

U programu *Control* smo koristili *Accelerometer* i *MultiTouchXY*. Nismo vidjeli kako izgleda sučelje od *MultiTouchXY*. Slijedi njegov prikaz:



Kao što vidimo u sučelju se nalaze gumbovi. Možemo dodavati i oduzimati gumb. Svaki gumb ima svoj svoje ime. Ono je određeno brojem koji je upisan u gumbu. Gumb ima opciju *onClick()* koji šalje odgovarajuću poruku računalu. Naš program će ispisivati super stvar. Prvo ćemo prikazati slikovno, pa onda opisati.

```
--------------------
### received an osc message with addrpattern /accelerometer and typetag fff
-OscMessage----------
received from       /192.168.43.101:35822
addrpattern         /accelerometer
typetag   fff
[0] 76.39844
[1] 98.22656
[2] 115.09375

--------------------
### received an osc message with addrpattern /multi/5 and typetag ff
-OscMessage----------
received from       /192.168.43.101:35822
addrpattern         /multi/5
typetag   ff
[0] 0.26964286
[1] 0.21317829

--------------------
```

Vidimo odličnu stvar prilikom ispisa. Točno znamo odakle nam je došao signal, tko je poslao(Accelerometer ili MultiTouchXY). Ukoliko nam je signal poslan od *Accelerometer*-a dobili smo koordinate mobilnog uređaja u prostoru. Ukoliko smo signal dobili od MultiTouchXY dobili smo koji je gumb pritisnut i koordinate našeg gumba u svom okviru. Okvir za gumb-ove *MultiTouchXY* je dimenzije 1x1. Program se nalazi u mapi „oscP5broadcastClient“.

Dosta dobrih ideja se može ostvariti s ovom informacijom. Npr. Mogli bi smo s mobilnim uređajem upravljati nekom igricom napisanom u *Processing*-u. Na neki način naš mobilni uređaj se ponaša kao *joystic* za *Processing.*

Mislim da je na ovom mjestu dobro stati što se tiče osnovnih primjera. Iz ovog primjera vidimo da bi mogli ostvariti ozbiljniju komunikaciju. Na osnovu gumba i njegovog imena možemo ostvariti veću kontrolu našeg koda u *Processing*-u. Sada ćemo napisati kod koji će ispisivati koji smo gumb pritisnuli na mobitelu.

```
import oscP5.*;
import netP5.*;

OscP5 oscP5;

void setup() {
  size(400,400);
  frameRate(25);
  oscP5 = new OscP5(this, 10000);
```

```
}


void draw() {
  background(0);
}
void oscEvent(OscMessage theOscMessage) {

if(theOscMessage.addrPattern().indexOf("multi")
!=-1 )
  {
    String tmpStr =
theOscMessage.addrPattern();
    int br =  Integer.parseInt(tmpStr.substring(7));
    println(br);
  }
}
```

Izolirali smo ime gumba koje poslao signal. Program se nalazi u mapi „gumblme“. Na ovom mjestu smo ostvarili veću kontrolu komunikacije između mobilnog uređaja i računala.

2.3. Konkretnan primjer

Prikazat ćemo sada jedan zanimljiviji primjer.

```
import oscP5.*;
import netP5.*;

OscP5 oscP5;
int br;
void setup() {
  size(400,400);
  frameRate(25);
  oscP5 = new OscP5(this, 10000);
  br = 0;
}

void draw() {
    background(50*br);
}
void oscEvent(OscMessage theOscMessage) {
  if(theOscMessage.addrPattern().indexOf("multi") !=-1 )
```

```
  {
    String tmpStr = theOscMessage.addrPattern();
    br =  Integer.parseInt(tmpStr.substring(7));
    println(br);
  }
}
```

Primjer radi slijedeće: Na osnovu pritisnutog gumba se mijenja boja pozadine. U sučelju *MultiTouchXY* možemo dodavati gumbe. Prilikom klika na gumb, uzimamo njegovu vrijednost i na osnovu toga mijenjamo boju pozadine. Program se nalazi u mapi „gumbPozadina“.

Slijedi nam jedan primjer koji smo gore naveli kao mogućnost *Processing*-a.

2.4. Primjena komunikacije mobilni uređaj – računalo na igrici Zmija u Processing-u

Slijedi upravljanje zmijom pomoću mobilnog uređaja. Radi se o jednostavnoj verziji zmije gdje je pomičemo gumbovima. Zmija uvijek stoji, kada kliknemo na gumb ona se pomiče. Napominjemo da nije naglasak na Zmiji kao igrici, već primjenu naše ostvarene komunikacije.

Upravljanje je sljedeće:

- Gumb 1 za desno
- Gumb 2 za lijevo
- Gumb 3 za gore
- Gumb 4 za dolje

```
import oscP5.*;
import netP5.*;

OscP5 oscP5;
int br;

class Location {
  int row;
  int column;
  Location(int row, int column) {
    this.row = row;
    this.column = column;
```

```
  }
  boolean equals(Location l) {
    return l.row == row && l.column == column;
  }
}

Location food;
ArrayList<Location> snake;
int score;

final int COLS = 20;
final int ROWS = 20;

void setup() {
  size(500, 500);
   frameRate(25);
  oscP5 = new OscP5(this, 10000);
  br = 0;
  reset();
}

void reset() {
  snake = new ArrayList<Location>();
  growSnake(1, 3);
  growSnake(1, 2);
  growSnake(1, 1);
  addFood();
}

void addFood() {
  // add food:
  food = new Location( (int) random(COLS), (int) random(ROWS));
}

void growSnake(int x, int y) {
  snake.add(new Location(x, y));
}

void drawFood() {
 drawRect(food);
}
void drawSnake() {
  for(Location l : snake) {
    drawRect(l);
  }
}
```

```
void drawRect(Location value) {
  int w = width / COLS;
  int h = height / ROWS;
  stroke(255);
  rect(value.column * w, value.row * h, w, h);
}

void draw() {
  background(100);
  fill(#005500);
  drawSnake();
  fill(#550000);
  drawFood();
}

void oscEvent(OscMessage theOscMessage) {
  if(theOscMessage.addrPattern().indexOf("multi") !=-1 )
  {
    String tmpStr = theOscMessage.addrPattern();
    br =  Integer.parseInt(tmpStr.substring(7));

    int direction = RIGHT;
    if (key == CODED) {
      direction = keyCode;
    }
    Location head = snake.get(0); // current head
    Location newHead = null;

    switch (br) {
      case 1:
        newHead = new Location(head.row, head.column + 1);
        break;
      case 2:
        newHead = new Location(head.row, head.column - 1);
        break;
      case 3:
        newHead = new Location(head.row - 1, head.column);
        break;
      case 4:
        newHead = new Location(head.row + 1, head.column);
        break;
      default:
        return;
    }
    if (newHead.equals(food)) {
```

```
      score++;
      addFood();
    } else {
      snake.remove(snake.size() - 1); // remove old tail
    }
    snake.add(0, newHead); // add new head
    }
}
```

Program se nalazi u mapi „Zmija“.

2.5. Upravljanje prekidima na računalu pomoću mobilnog uređaja

U ovoj točki ćemo pokušati ostvariti ono što smo napisali kao cilj našeg projekta.

```
import java.awt.AWTException;
import java.awt.event.KeyEvent;
import java.awt.Robot;
import oscP5.*;
import netP5.*;

Robot robot;
OscP5 oscP5;
int br;

void setup() {
  size(400, 400);
  frameRate(25);
  oscP5 = new OscP5(this, 10000);
  br = 0;

  try {
  robot = new Robot();
  } catch (AWTException e) {
    e.printStackTrace();
    exit();
  }
}
void oscEvent(OscMessage theOscMessage) {
  if(theOscMessage.addrPattern().indexOf("multi") !=-1 )
  {
    String tmpStr = theOscMessage.addrPattern();
    br =  Integer.parseInt(tmpStr.substring(7));
```

```
    if(br == 1)
      robot.keyPress(KeyEvent.VK_RIGHT);
    if(br == 2)
      robot.keyPress(KeyEvent.VK_LEFT);
    if( br == 3)
      robot.keyPress(KeyEvent.VK_ESCAPE);
    if( br == 4)
      robot.keyPress(KeyEvent.VK_F5);
  }
}
```

Ovaj kod simulira naše izvođenje prezentacije. Kontrole se sljedeće:

- Gumb 1 za sljedeći slajd (Strelica desno)
- Gumb 2 za prethodni slajd (Strelica lijevo)
- Gumb 3 za izlaz iz prezentacije kada je pokrenuta (ESCAPE)
- Gumb 4 za pokretanje prezentacije (F5)

Glavna klasa koja nam to omogućuje je Robot. Robot izaziva prekid na računalu u ovisnosti o pritisnutom gumbu na mobilnom uređaju u programu *Control* sučelju *MultiTouchXY*. To je cilj koji smo htjeli ostvariti i ostvarili smo ga. Nema potrebe komentirati kod jer smo ga postupno gradili. Program se nalazi u mapi „Prezentacija“.

Sada slijedi drugi cilj.

3. Održavanje prezentacije na računalu koja se nalazi na mobilnom uređaju

U ovoj dijelu dokumenta ćemo pokušati upravljati prezntacijom koja se nalazi na mobilnom uređaju. Naime, na mobilnom uređaju se nalazi prezentacija koju planiramo održati. Umjesto da prezentaciju prebacujemo na računalu i da je pokrenemo sa računalu, mi ćemo prezentaciju pokrenuti na mobilnom uređaju te „zrcaljenjem“ ekrana na računalo održati prezentaciju.

3.1. Zrcaljenje ekrana mobilnog uređaja na ekran računala

U ovoj točki ćemo pokušati prikazati ekran mobilnog uređaja na ekran računala. Za to nam je potreban softver. Odličan i besplatan softver za zrcaljenje ekrana je *Mobizen*, no nažalost nemamo mobilni uređaj koji je kompatibilan sa softverom. Tako da ovo moramo odustati od ovog cilja.

Prema tome, najjednostavniji način zrcaljenja ekrana je korištenjem toga softvera.

4. Zaključak

Naš projekt je donio puno novih mogućnosti u pogledu komunikacije mobilni uređaj – računalo. Sada je jednostavno napraviti niz programa koji će omogući upravljanje radom OS preko mobilnog uređaja. Radnje kao što su: ugasi računalo, upravljanje zvukom, upravljanje svjetlinom ekrana, pisanju u nekom editoru i mnoge druge se mogu ostvariti preko mobilnog uređaja. Također, možemo urediti sučelje našeg programa *Control* i tako olakšati komunikaciju svrhu izvođenja nekih radnjih

5. Literatura

[Control]: http://charlie-roberts.com/Control/

[oscP5]: http://www.sojamo.de/libraries/oscP5/

[mobizen]: https://www.mobizen.com/?locale=en

[Processing]: https://processing.org/