```
/* 12 dicembre 2007.  Martina, Marina, Ivan.
Haction assemblaggio
*/

////////////////////////////////////////////////////////////////////////////////
// Names Of Graphics & How Load Them
////////////////////////////////////////////////////////////////////////////////

PImage backgroundP;
PImage ok;
PImage back;
PImage send;
PImage write;

PImage hacktion;
PImage hcktion;
PImage a2;
PImage a3;
PImage a4;
PImage a7;
PImage a9;
PImage a12;
PImage a14;

PImage startLogin;
PImage startSignup;
PImage stariconFocus;

PImage loginUser;
PImage loginPsw;
PImage loginForgot;
PImage loginText;

PImage userName;
PImage userPoint;
PImage userAvatar;
PImage userOnline;
PImage userOffline;
PImage userHack;
PImage userCanovaccio;
PImage userBot;

PImage newI;
PImage new_othI;
PImage othI;
PImage oth_oldI;
PImage oldI;
PImage old_newI;

PImage wall;
PImage stick;
PImage shadow;
PImage avA;
PImage avB;
PImage avC;

PImage newTitle;
PImage newWhen;
PImage newWhere;
PImage newBox;

PImage newBdesc;
PImage newBbox;
PImage newBimage;
PImage newImage;

PImage txt;
PImage txt2;
PImage txtW;
PImage msgBox;

PImage mose;
PImage ins;

PImage partecipate;

PFont font;// Declare font
```

```
String nickname = " ";
String password = " ";
String title = " ";
String where = " ";
String when = " ";
String description = " ";
String wmsg = " ";

String textToSend = " ";        // variable to store the text (string) to be sent

void loadImages()
{
  backgroundP = loadImage ("bg3.png");
  ok = loadImage ("ok.png");
  back = loadImage ("back.png");
  send = loadImage ("send.png");
  write = loadImage ("writeMsg.png");
  hacktion = loadImage ("hacktion.png");
  hcktion = loadImage ("hction.png");
  a2 = loadImage ("a2.png");
  a3 = loadImage ("a3.png");
  a4 = loadImage ("a4.png");
  a7 = loadImage ("a7.png");
  a9 = loadImage ("a9.png");
  a12 = loadImage ("a12.png");
  a14 = loadImage ("a14.png");
  startLogin = loadImage ("login.png");
  startSignup = loadImage ("signup.png");
  stariconFocus = loadImage ("star.png");
  loginUser = loadImage ("iconaUser.png");
  loginPsw = loadImage ("iconaKey.png");
  loginForgot = loadImage ("forgot.png");
  loginText = loadImage ("boxLoginSignup.png");
  userName = loadImage ("name.png");
  userPoint = loadImage ("points.png");
  userAvatar = loadImage ("boxAvatar.png");
  userOnline = loadImage ("online.png");
  userOffline = loadImage ("offline.png");
  userHack = loadImage ("hacktions.png");
  userCanovaccio = loadImage ("canovacio.png");
  userBot = loadImage ("button.png");
  newI = loadImage("new.png");
  new_othI = loadImage("beforeNew.png");
  othI = loadImage("current.png");
  oth_oldI = loadImage("beforeOthers.png");
  oldI = loadImage("old.png");
  old_newI = loadImage("beforeOld.png");
  wall =loadImage ("bg4.png");
  stick = loadImage ("sticker.png");
  shadow = loadImage ("ombra.png");
  newTitle = loadImage ("title.png");
  newWhen = loadImage ("when.png");
  newWhere = loadImage ("where.png");
  newBox = loadImage ("boxDefoult.png");
  newBdesc = loadImage ("description.png");
  newBbox = loadImage ("boxDescription.png");
  newImage = loadImage ("image.png");
  msgBox = loadImage ("boxMsg.png");
  avA = loadImage ("avA.png");
  avB = loadImage ("avB.png");
  avC = loadImage ("avC.jpg");
  txt = loadImage ("txt_a.png");
  txt2 = loadImage ("txt_b.png");
  txtW = loadImage ("txt_c.png");
  mose = loadImage ("mo.png");
  ins = loadImage("ins.png");
  partecipate = loadImage("participate.png");

}



////////////////////////////////////////////////////////////////////////////////////////////
// Screen Drawing Functions
////////////////////////////////////////////////////////////////////////////////////////////
```

```
int moove = 0;

// function to draw the menu with specified focus
int starIconFocusX = 20 ; // begin with startIconFocus X at Login X
int starIconFocusY = 50 ; // begin with menuIconFocus Y at Login Y

int slideX=-415; // Image initial position_X

void drawAnima ()
{
  image(backgroundP, 0, 0);

  moove = moove + 15;

  if (moove < 10)
  {
    image(hacktion,0,60);
  }
  if (moove >= 10 && moove < 20)
  {
    image(hcktion,0,60);
    image(a2,33,90);
  }
  if (moove >= 20 && moove < 30)
  {
    image(hcktion,3,63);
    image(a3,33,93);
  }
  if (moove >= 30 && moove < 40)
  {
    image(hcktion,-1,59);
    image(a4,28,79);
  }
  if (moove >= 40 && moove < 50)
  {
    image(hcktion,2,58);
    image(a7,22,78);
  }
  if (moove >= 50 && moove < 60)
  {
    image(hcktion,0,60);
    image(a9,20,77);
  }
  if (moove >= 60 && moove < 70)
  {
    image(hcktion,0,60);
    image(a12,15,70);
  }
  if (moove >= 70 && moove < 80)
  {
    image(hcktion,-2,62);
    image(a14,12,62);
  }
  if (moove >= 80 && moove < 90)
  {
    image(hcktion,0,60);
    image(a14,10,60);
  }
  if (moove >= 90 && moove < 100)
  {
    image(hcktion,2,63);
    image(a12,17,73);
  }
  if (moove >= 100 && moove < 110)
  {
    image(hcktion,-1,60);
    image(a9,19,77);
  }
  if (moove >= 110 && moove < 120)
  {
    image(hcktion,1,58);
    image(a7,21,78);
  }
  if (moove >= 120 && moove < 130)
  {
    image(hcktion,-3,60);
    image(a4,25,80);
  }
```

```
  if (moove >= 130 && moove < 140)
  {
    image(hcktion,0,62);
    image(a3,30,92);
  }
  if (moove >= 140 && moove < 200)
  {
    image(hcktion,0,60);
    image(a2,33,90);
  }

}


void drawStart(int focus)
{
  image (backgroundP, 0, 0);   // draw background image for the menu area
  image (ok,145,175);
  image (back,8,184);
  image (startLogin, 50, 60);      // draw login
  image (startSignup, 50, 90);    // draw signup

  // calculate position of focus highlight for the icon in focus
  if(focus == STARTFOCUS_LOGIN){      // if we are focused on login
    starIconFocusX = 25;
    starIconFocusY = 60;
  }
  else if(focus == STARTFOCUS_SIGNUP){     // if we are focused on signup
    starIconFocusX = 25;
    starIconFocusY = 90;
  }
  image (stariconFocus,starIconFocusX,starIconFocusY);

}

void drawLogin(int focus)
{
  image (backgroundP,0,0);
  image (ok,145,175);
  image (back,8,184);
  image (loginUser,27,67);
  image (loginPsw,29,115);
  image (loginForgot,29,140);
  if(focus == LOGINFOCUS_USER){
    starIconFocusX = 8;
    starIconFocusY = 70;
  }
  else if(focus == LOGINFOCUS_PSW){     // if we are focused on psw
    starIconFocusX = 8;
    starIconFocusY = 110;
  }
  else if(focus == LOGINFOCUS_FORGOT){     // if we are focused on forgot
    starIconFocusX = 8;
    starIconFocusY = 140;
  }
  image (stariconFocus,starIconFocusX,starIconFocusY);
  image (loginText,63,70);
  image (loginText,63,110);
}

void drawNickname()
{
  nickname = textInput("INSERT YOUR NICKNAME", "", 20);  // Get the string from the device input method
}

void drawPassword()
{
  password = textInput("INSERT YOUR PASSWORD", "", 20);  // Get the string from the device input method
}

void drawTre()
{
  image (backgroundP,0,0);
}

void drawUser(int focus, int stat)
{
  image (backgroundP,0,0);
```

```
image (ok,145,175);
image (back,8,184);
image (userName,20,20);
image (userPoint,20,40);
image (userAvatar,90,20);
image (userBot,40,100);
image (userBot,40,130);
image (userBot,40,160);
image (userHack,50,133);
image (userCanovaccio,45,163);
if (stat == STATUS_ON)
{
  image (userOnline,65,103);
}
else if (stat == STATUS_OFF)
{
  image (userOffline,65,103);
}
if(focus == USERFOCUS_STATUS){
  starIconFocusX = 15;
  starIconFocusY = 103;
}
else if(focus == USERFOCUS_HACK){     // if we are focused on psw
  starIconFocusX = 15;
  starIconFocusY = 133;
  }
  else if(focus == USERFOCUS_CAN){     // if we are focused on forgot
    starIconFocusX = 15;
    starIconFocusY = 163;
  }
  image (stariconFocus,starIconFocusX,starIconFocusY);
}

void drawHack(int focus)
{
  image (backgroundP,0,0);
  image (ok,145,175);
  image (back,8,184);
  if (focus == IDEAFOCUS_NEW)
  {
    image (newI,20,20);
  }
  if (focus == IDEAFOCUS_NEOT)
  {
    image (new_othI,20,50);
  }
  if (focus == IDEAFOCUS_OTHER)
  {
    image (othI,20,20);
  }
  if (focus == IDEAFOCUS_OTOL)
  {
    image (oth_oldI,20,50);
  }
  if (focus == IDEAFOCUS_OLD)
  {
    image (oldI,20,20);
  }
  if (focus == IDEAFOCUS_OLNE)
  {
    image (old_newI,20,50);
  }
}

void drawWall (int focus)
{
  if(focus == WALLFOCUS_C)
  {
    slideX=slideX;
  }
  else if (focus == WALLFOCUS_P)
  {
    slideX=slideX-15;
    slideX = constrain(slideX,(width-wall.width),0);   // Prevent image from panning beyond width
    //selectX=selectX+10;
  }
  else if (focus == WALLFOCUS_M)
  {
```

```
  slideX=slideX+15;
  slideX = constrain(slideX,(width-wall.width),0);
}

image(wall,slideX,0);

int stXa = slideX+500;
int stYa = 120;
int stXb = slideX+550;
int stYb = 40;
int stXc = slideX+400;
int stYc = 70;
int stXd = slideX+470;
int stYd = 20;
int stXe = slideX+330;
int stYe = 110;
int stXf = slideX+300;
int stYf = 15;
int stXg = slideX+250;
int stYg = 90;
int stXh = slideX+600;
int stYh = 115;
int stXi = slideX+630;
int stYi = 10;
int stXl = slideX+680;
int stYl = 90;
int stXm = slideX+715;
int stYm = 20;
int stXn = slideX+760;
int stYn = 120;

image(stick,stXa,stYa);
image(avC,stXa+3,stYa+5);
image(stick,stXb,stYb);
image(avC,stXb+6,stYb+4);
image(stick,stXc,stYc);
image(avB,stXc+6,stYc+3);
image(stick,stXd,stYd);
image(avA,stXd+3,stYd+5);
image(stick,stXe,stYe);
image(avA,stXe+3,stYe+3);
image(stick,stXf,stYf);
image(avB,stXf+6,stYf+4);
image(stick,stXg,stYg);
image(avC,stXg+6,stYg+3);
image(stick,stXh,stYh);
image(avA,stXh+3,stYh+5);
image(stick,stXi,stYi);
image(avB,stXi+6,stYi+4);
image(stick,stXl,stYl);
image(avC,stXl+6,stYl+3);
image(stick,stXm,stYm);
image(avA,stXm+3,stYm+5);
image(stick,stXn,stYn);
image(avB,stXn+6,stYn+4);

if(stXa >= ((width / 2)-40)){
  if(stXa <= ((width / 2)-10)){
    image(shadow,stXa,stYa);
  }
}

if(stXb >= ((width / 2)-40)){
  if(stXb <= ((width / 2)-10)){
    image(shadow,stXb,stYb);
  }
}

if(stXc >= ((width / 2)-40)){
  if(stXc <= ((width / 2)-10)){
    image(shadow,stXc,stYc);
  }
}

if(stXd >= ((width / 2)-40)){
  if(stXd <= ((width / 2)-10)){
    image(shadow,stXd,stYd);
  }
```

```
}

if(stXe >= ((width / 2)-40)){
  if(stXe <= ((width / 2)-10)){
    image(shadow,stXe,stYe);
  }
}

if(stXf >= ((width / 2)-40)){
  if(stXf <= ((width / 2)-10)){
    image(shadow,stXf,stYf);
  }
}

if(stXg >= ((width / 2)-40)){
  if(stXg <= ((width / 2)-10)){
    image(shadow,stXg,stYg);
  }
}

if(stXh >= ((width / 2)-40)){
  if(stXh <= ((width / 2)-10)){
    image(shadow,stXh,stYh);
  }
}
```

```
  if(stXi >= ((width / 2)-40)){
    if(stXi <= ((width / 2)-10)){
      image(shadow,stXi,stYi);
    }
  }

  if(stXl >= ((width / 2)-40)){
    if(stXl <= ((width / 2)-10)){
      image(shadow,stXl,stYl);
    }
  }

  if(stXm >= ((width / 2)-40)){
    if(stXm <= ((width / 2)-10)){
      image(shadow,stXm,stYm);
    }
  }

  if(stXn >= ((width / 2)-40)){
    if(stXn <= ((width / 2)-10)){
      image(shadow,stXn,stYn);
    }
  }
  image (ok,145,175);
  image (back,8,184);
}
```

```
void drawNew(int focus)
{
  image (backgroundP,0,0);
  image (ok,145,175);
  image (back,8,184);
  image (newBox,40,58);
  image (newBox,40,98);
  image (newBox,40,138);
  image (newTitle,40,40);
  image (newWhen,40,80);
  image (newWhere,40,120);
  if(focus == USERFOCUS_STATUS){
    starIconFocusX = 15;
    starIconFocusY = 40;
  }
  else if(focus == USERFOCUS_HACK){     // if we are focused on psw
    starIconFocusX = 15;
    starIconFocusY = 80;
  }
  else if(focus == USERFOCUS_CAN){     // if we are focused on forgot
    starIconFocusX = 15;
    starIconFocusY = 120;
  }
  image (stariconFocus,starIconFocusX,starIconFocusY);
}
```

```
void drawTitle()
{
  title = textInput("INSERT TITLE", "", 20);  // Get the string from the device input method
}

void drawWhere()
{
  where = textInput("INSERT A PLACE", "", 20);  // Get the string from the device input method
}

void drawWhen()
{
  when = textInput("INSERT DATE & H", "", 20);  // Get the string from the device input method
}

void drawNewB (int focus)
{
  image (backgroundP,0,0);
  image (send,130,184);
  image (back,8,184);
  image (newBdesc,30,5);
  image (newBbox,7,23);
  image (newImage,30,120);
  image (userAvatar,83,125);
  if(focus == USERFOCUS_STATUS){
    starIconFocusX = 10;
    starIconFocusY = 5;
  }
  else if(focus == USERFOCUS_HACK){
    starIconFocusX = 10;
    starIconFocusY = 120;
  }
  image (stariconFocus,starIconFocusX,starIconFocusY);
}

void drawDescription()
{
  description = textInput("INSERT THE DESCRIPTION", "", 80);  // Get the string from the device input method
}

void drawSticker()
{
  image (backgroundP,0,0);
  image (write,70,184);
  image (back,8,184);
  image (userName,20,17);
  image (userPoint,20,37);
  image (userAvatar,90,17);
  image (avA,93,20);
  image (newBbox,7,87);
  image (txt,12,90);
}

void drawStickerB()
{
  image (backgroundP,0,0);
  image (write,70,184);
  image (back,8,184);
  image (userName,20,17);
  image (userPoint,20,37);
  image (userAvatar,90,17);
  image (avC,95,21);
  image (newBbox,7,87);
  image (txt2,12,90);
}

void drawMsg()
{
  image (backgroundP,0,0);
  image (send,130,184);
  image (write,15,10);
  image (msgBox,7,30);
}

void drawWmsg()
{
  wmsg = textInput("WRITE A MSG", "", 160);  // Get the string from the device input method
```

```
}

void drawSms()
{
  image (backgroundP,0,0);
  image (send,130,184);
  image (write,15,10);
  image (msgBox,7,30);
  if(textToSend != " ")          // if text has been entered...
  {
    text(textToSend, 15, 50);   // draw the text the user entered (on top of the field)
  }
}

void drawWallO (int focus)
{
  if(focus == WALLFOCUS_C)
  {
    slideX=slideX;
  }
  else if (focus == WALLFOCUS_P)
  {
    slideX=slideX-15;
    slideX = constrain(slideX,(width-wall.width),0);   // Prevent image from panning beyond width
    //selectX=selectX+10;
  }
else if (focus == WALLFOCUS_M)
{
  slideX=slideX+15;
  slideX = constrain(slideX,(width-wall.width),0);
}

image(wall,slideX,0);

int stXa = slideX+503;
int stYa = 120;
int stXb = slideX+550;
int stYb = 40;
int stXc = slideX+400;
int stYc = 70;
int stXd = slideX+470;
int stYd = 20;
int stXe = slideX+330;
int stYe = 110;
int stXf = slideX+300;
int stYf = 15;
int stXg = slideX+250;
int stYg = 90;
int stXh = slideX+600;
int stYh = 115;
int stXi = slideX+630;
int stYi = 10;
int stXl = slideX+680;
int stYl = 90;
int stXm = slideX+715;
int stYm = 20;
int stXn = slideX+760;
int stYn = 120;

image(stick,stXa,stYa);
image(ins,stXa+3,stYa+5);
image(stick,stXb,stYb);
image(mose,stXb+6,stYb+4);
image(stick,stXc,stYc);
image(stick,stXd,stYd);
image(mose,stXd+6,stYd+5);
image(stick,stXe,stYe);
image(stick,stXf,stYf);
image(stick,stXg,stYg);
image(stick,stXh,stYh);
image(stick,stXi,stYi);
image(stick,stXl,stYl);
image(stick,stXm,stYm);
image(stick,stXn,stYn);

if(stXa >= ((width / 2)-40)){
  if(stXa <= ((width / 2)-10)){
    image(shadow,stXa,stYa);
```

```
  }
}

if(stXb >= ((width / 2)-40)){
  if(stXb <= ((width / 2)-10)){
    image(shadow,stXb,stYb);
  }
}

if(stXc >= ((width / 2)-40)){
  if(stXc <= ((width / 2)-10)){
    image(shadow,stXc,stYc);
  }
}

if(stXd >= ((width / 2)-40)){
  if(stXd <= ((width / 2)-10)){
    image(shadow,stXd,stYd);
  }
}

if(stXe >= ((width / 2)-40)){
  if(stXe <= ((width / 2)-10)){
    image(shadow,stXe,stYe);
  }
}

if(stXf >= ((width / 2)-40)){
  if(stXf <= ((width / 2)-10)){
    image(shadow,stXf,stYf);
  }
}

if(stXg >= ((width / 2)-40)){
  if(stXg <= ((width / 2)-10)){
    image(shadow,stXg,stYg);
  }
}

if(stXh >= ((width / 2)-40)){
  if(stXh <= ((width / 2)-10)){
    image(shadow,stXh,stYh);
  }
}

if(stXi >= ((width / 2)-40)){
  if(stXi <= ((width / 2)-10)){
    image(shadow,stXi,stYi);
  }
}

if(stXl >= ((width / 2)-40)){
  if(stXl <= ((width / 2)-10)){
    image(shadow,stXl,stYl);
  }
}

  if(stXm >= ((width / 2)-40)){
    if(stXm <= ((width / 2)-10)){
      image(shadow,stXm,stYm);
    }
  }

  if(stXn >= ((width / 2)-40)){
    if(stXn <= ((width / 2)-10)){
      image(shadow,stXn,stYn);
    }
  }
  image (ok,145,175);
  image (back,8,184);
}

void drawStickerWall()
{
  image (backgroundP,0,0);
  image (partecipate,70,184);
  image (back,8,184);
  image (newTitle,20,17);
```

```
  image (newWhere,20,37);
  image (newWhen,20,57);
  image (userAvatar,90,17);
  image (mose,94,24);
  image (newBbox,7,87);
  image (txtW,12,90);
}



////////////////////////////////////////////////////////////////////////////////////////////
// Setup, Executes Once When Started, Prepares Program to Run (Logic Initialization)
////////////////////////////////////////////////////////////////////////////////////////////

import processing.phone.*; // import phone library to go fullscreen
import processing.messaging.*;  // import messaging library to send a message

Phone myPhone;             // named reference to phone instance
Messenger m;                  // named reference to messenger instance

void setup() // happens only once, when the program starts...
{
  // go fullscreen
  myPhone = new Phone(this); // create new phone instance/controller
  myPhone.fullscreen();      // tell phone to go fullscreen
  m = new Messenger(this);

  loadImages(); // load images
  font = loadFont("Futura-Medium-14.mvlw", color(0,0,0));   // Load the font file from the data folder
  textFont(font);                        // Use the font for rendering text

}

////////////////////////////////////////////////////////////////////////////////////////////
// Mode, Focus & Option Names (Constants)
////////////////////////////////////////////////////////////////////////////////////////////

int MODE_ANIMA = 0;
int MODE_START = 1;
int MODE_LOGIN = 2;
int MODE_NICKNAME = 3;
int MODE_PASSWORD = 4;
int MODE_USER = 5;
int MODE_TRE = 6;
int MODE_HACK = 7;
int MODE_WALL = 8;
int MODE_NEW = 9;
int MODE_TITLE = 10;
int MODE_WHERE = 11;
int MODE_WHEN = 12;
int MODE_NEWB = 13;
int MODE_DESCRIPTION = 14;
int MODE_STICKER = 15;
int MODE_STICKERB = 16;
int MODE_MSG = 17;
int MODE_WMSG = 18;
int MODE_SMS = 19;
int MODE_WALLO = 20;
int MODE_STICKWALL = 21;


int STARTFOCUS_LOGIN = 0;
int STARTFOCUS_SIGNUP = 1;
int LOGINFOCUS_USER = 0;
int LOGINFOCUS_PSW = 1;
int LOGINFOCUS_FORGOT = 2;
int USERFOCUS_STATUS = 0;
int USERFOCUS_HACK = 1;
int USERFOCUS_CAN = 2;
int STATUS_ON = 0;
int STATUS_OFF = 1;
int IDEAFOCUS_NEW = 0; //riferiti a MODE_HACK
int IDEAFOCUS_NEOT = 1;
int IDEAFOCUS_OTHER = 2;
int IDEAFOCUS_OTOL = 3;
int IDEAFOCUS_OLD = 4;
int IDEAFOCUS_OLNE = 5;
```

```
int WALLFOCUS_C=0; // C=centro della pagina...per ora non mi serve...credo però serva per selezionare gli stikers...lasciamolo scritto per ora.
int WALLFOCUS_P= 1;
int WALLFOCUS_M=-1;
int NEWFOCUS_TITLE = 0;
int NEWFOCUS_WHEN = 1;
int NEWFOCUS_WHERE = 2;
int NEWBFOCUS_D = 0;
int NEWBFOCUS_I = 1;


////////////////////////////////////////////////////////////////////////////////////////////
// State, Information Collected From Use (Variables)
////////////////////////////////////////////////////////////////////////////////////////////

int mode = MODE_ANIMA;
// main menu information
int startFocus = STARTFOCUS_LOGIN; // initially, the main menu focus is on the first item login
int loginFocus = LOGINFOCUS_USER;
int userFocus = USERFOCUS_STATUS;
int userStatus = STATUS_ON;
int hackFocus = IDEAFOCUS_NEW;
int wallFocus = WALLFOCUS_C;
int newFocus = NEWFOCUS_TITLE;
int newBFocus = NEWBFOCUS_D;

////////////////////////////////////////////////////////////////////////////////////////////
// Draw, Executes Forever, Provides User Feedback (Logic Repetition)
////////////////////////////////////////////////////////////////////////////////////////////

void draw() // happens repeatedly (according to framerate)...
{
  if (mode == MODE_ANIMA){
    drawAnima();
    if (moove >= 200){
      mode = MODE_START;
    }
  }
  if (mode == MODE_START){
    drawStart(startFocus); // draw image for menu with a variable focus
  }
  else if (mode == MODE_LOGIN){
    drawLogin(loginFocus);

    text(nickname, 68, 87);
    text(password, 68, 127);
  }
  else if (mode== MODE_NICKNAME){
    drawNickname();
    mode = MODE_LOGIN;
  }
  else if (mode== MODE_PASSWORD){
    drawPassword();
    mode = MODE_LOGIN;
  }
  /*else if (mode== MODE_IDEAS){
    drawIdeas();
  }*/
  else if (mode== MODE_TRE){
    drawTre();
  }
  else if (mode == MODE_USER){
    drawUser(userFocus,userStatus);
  }
  else if (mode == MODE_HACK){
    drawHack(hackFocus);
  }
  if (mode == MODE_WALL){
    drawWall(wallFocus); // draw image for menu with a variable focus...i moove the wall!
  }
  else if (mode == MODE_NEW){
    drawNew(newFocus);
    text(title, 45, 71);
    text(where, 45, 111);
    text(when, 45, 151);
  }
  else if (mode== MODE_TITLE){
    drawTitle();
    mode = MODE_NEW;
```

```
}
else if (mode== MODE_WHERE){
  drawWhere();
  mode = MODE_NEW;
}
else if (mode== MODE_WHEN){
  drawWhen();
  mode = MODE_NEW;
}
else if (mode == MODE_NEWB){
  drawNewB(newBFocus);
  text(description,15,40);
}
else if (mode == MODE_DESCRIPTION){
  drawDescription();
  mode = MODE_NEWB;
}
else if (mode == MODE_STICKER){
  drawSticker();
}
else if (mode == MODE_STICKERB){
  drawStickerB();
}
else if (mode == MODE_MSG){
  drawMsg();
  text(wmsg,15,50);
  }
  else if (mode == MODE_WMSG){
    drawWmsg();
    mode = MODE_MSG;
  }
  else if (mode== MODE_SMS){
    drawSms();
  }
  else if (mode == MODE_WALLO){
    drawWallO(wallFocus);
  }
  else if (mode == MODE_STICKWALL){
    drawStickerWall();
  }


}

////////////////////////////////////////////////////////////////////////////////////////
// Keypad Event (User Input Capture & Interpretation)
////////////////////////////////////////////////////////////////////////////////////////

void keyPressed ()
{
  if (mode == MODE_HACK)
  {
    if(hackFocus == IDEAFOCUS_NEW){
      if(keyCode == RIGHT){
        hackFocus = IDEAFOCUS_NEOT;
      }
      else if(keyCode == LEFT){
        hackFocus = IDEAFOCUS_OLNE;
      }
    }
    else if(hackFocus == IDEAFOCUS_OTHER){
      if(keyCode == RIGHT){
        hackFocus = IDEAFOCUS_OTOL;
      }
      else if(keyCode == LEFT){
        hackFocus = IDEAFOCUS_NEOT;
      }
    }
    else if(hackFocus == IDEAFOCUS_OLD){
      if(keyCode == RIGHT){
        hackFocus = IDEAFOCUS_OLNE;
      }
      else if(keyCode == LEFT){
        hackFocus = IDEAFOCUS_OTOL;
      }
    }
  }
  if(mode == MODE_WALL)
```

```
  {
    if(keyCode== RIGHT)
    {
      wallFocus=WALLFOCUS_P;
    }
    else if(keyCode== LEFT)
    {
      wallFocus=WALLFOCUS_M;
    }
  }
  if(mode == MODE_WALLO)
  {
    if(keyCode== RIGHT)
    {
      wallFocus=WALLFOCUS_P;
    }
    else if(keyCode== LEFT)
    {
      wallFocus=WALLFOCUS_M;
    }
  }

}

void keyReleased()
{
  if (mode == MODE_START)
  {
    if(startFocus == STARTFOCUS_LOGIN){
      if(keyCode == DOWN){
        startFocus = STARTFOCUS_SIGNUP;
      }
      else if(keyCode == SOFTKEY2){
        mode = MODE_LOGIN;
      }
      else if(keyCode == FIRE){
        mode = MODE_LOGIN;
      }
     }
     else if(startFocus == STARTFOCUS_SIGNUP)
     {
       if(keyCode == UP){
         startFocus = STARTFOCUS_LOGIN;
       }
     }
  }

  else if (mode == MODE_LOGIN)
  {
    if (keyCode == SOFTKEY1){
      mode = MODE_START;
    }
  else if (keyCode == SOFTKEY2)
  {
    mode = MODE_USER;
  }
  else if (keyCode == GAME_B)
  {
    mode = MODE_USER;
  }
  else if(loginFocus == LOGINFOCUS_USER){
    if(keyCode == DOWN){
      loginFocus = LOGINFOCUS_PSW;
    }
    else if (keyCode == FIRE){
      mode = MODE_NICKNAME;
    }
  }
  else if(loginFocus == LOGINFOCUS_PSW){
    if(keyCode == DOWN){
      loginFocus = LOGINFOCUS_FORGOT;
    }
    else if(keyCode == UP){
      loginFocus = LOGINFOCUS_USER;
    }
    else if (keyCode == FIRE){
      mode = MODE_PASSWORD;
```

```
    }
  }
  else if(loginFocus == LOGINFOCUS_FORGOT)
  {
    if(keyCode == UP){
      loginFocus = LOGINFOCUS_PSW;
    }
    else if (keyCode == FIRE){
      mode = MODE_TRE;//provvisorio
    }
  }
}

else if (mode == MODE_USER)
{
  if (keyCode == SOFTKEY1){
    mode = MODE_LOGIN;
  }
  else if (userFocus == USERFOCUS_STATUS)
  {
    if (keyCode == DOWN)
    {
      userFocus = USERFOCUS_HACK;
    }
    else if (userStatus == STATUS_ON)
  {
    if (keyCode == FIRE)
    {
      userStatus = STATUS_OFF;
    }
  }
  else if (userStatus == STATUS_OFF)
  {
    if (keyCode == FIRE)
    {
      userStatus = STATUS_ON;
    }
  }
}
else if (userFocus == USERFOCUS_HACK)
{
  if (keyCode == DOWN)
  {
    userFocus = USERFOCUS_CAN;
  }
  if (keyCode == UP)
  {
    userFocus = USERFOCUS_STATUS;
  }
  else if (keyCode == SOFTKEY2)
  {
    mode = MODE_HACK;
  }
  else if (userFocus == USERFOCUS_CAN)
  {
    if (keyCode == UP)
    {
      userFocus = USERFOCUS_HACK;
    }
    else if (keyCode == SOFTKEY2)
    {
      mode = MODE_WALL;//provvisorio
    }
  }
}

else if (mode == MODE_HACK)
{
  if (keyCode == SOFTKEY1){
    mode = MODE_USER;
  }
  else if(hackFocus == IDEAFOCUS_NEOT){
    if(keyCode == RIGHT){
      hackFocus = IDEAFOCUS_OTHER;
    }
    else if(keyCode == LEFT){
      hackFocus = IDEAFOCUS_NEW;
```

```
  }
}
else if(hackFocus == IDEAFOCUS_OTOL){
  if(keyCode == RIGHT){
    hackFocus = IDEAFOCUS_OLD;
  }
  else if(keyCode == LEFT){
    hackFocus = IDEAFOCUS_OTHER;
  }
}
else if(hackFocus == IDEAFOCUS_OLNE){
  if(keyCode == RIGHT){
    hackFocus = IDEAFOCUS_NEW;
  }
  else if(keyCode == LEFT){
    hackFocus = IDEAFOCUS_OLD;
  }
}
else if(hackFocus == IDEAFOCUS_NEW){
  if (keyCode == SOFTKEY2){
    mode = MODE_NEW;
  }
}
else if(hackFocus == IDEAFOCUS_OTHER){
  if (keyCode == SOFTKEY2){
    mode = MODE_WALLO;
    }
  }
}

else if(mode ==MODE_WALL)
{
  if(keyCode == SOFTKEY1)
  {
    mode = MODE_USER;
  }
  else if(keyCode == RIGHT)
  {
    wallFocus=WALLFOCUS_C;
  }
  else if(keyCode== LEFT)
  {
    wallFocus=WALLFOCUS_C;
  }
  else if (slideX>=-425)
  {
    if(slideX<=-405){
      if(keyCode == SOFTKEY2)
      {
         mode = MODE_STICKER;
      }
    }
  }
  /*else if (slideX>=-380)
  {
    if(slideX<=-360){
      if(keyCode == SOFTKEY2)
      {
         mode = MODE_STICKER;
      }
    }
  } */
  else if (slideX>=-445)
  {
    if(slideX<=-425){
      if(keyCode == SOFTKEY2)
      {
         mode = MODE_STICKERB;
      }
    }
  }
}

else if (mode == MODE_NEW)
{
  if(keyCode == SOFTKEY1)
  {
    mode = MODE_HACK;
```

```
}
else if(keyCode == SOFTKEY2)
{
  mode = MODE_NEWB;
}
else if(newFocus == NEWFOCUS_TITLE)
{
  if(keyCode == DOWN)
  {
    newFocus = NEWFOCUS_WHEN;
  }
  else if (keyCode == FIRE){
    mode = MODE_TITLE;
  }
}
else if(newFocus == NEWFOCUS_WHEN)
{
  if(keyCode == DOWN)
  {
    newFocus = NEWFOCUS_WHERE;
  }
  else if(keyCode == UP)
  {
    newFocus = NEWFOCUS_TITLE;
  }
    else if (keyCode == FIRE){
      mode = MODE_WHERE;
    }
  }
  else if(newFocus == NEWFOCUS_WHERE)
  {
    if(keyCode == UP)
    {
      newFocus = NEWFOCUS_WHEN;
    }
    else if (keyCode == FIRE){
      mode = MODE_WHEN;
    }
  }
}

else if (mode == MODE_NEWB)
{
  if(keyCode == SOFTKEY1)
  {
    mode = MODE_NEW;
  }
  else if(keyCode == SOFTKEY2)
  {
    mode = MODE_USER;
  }
  else if(newBFocus == NEWBFOCUS_D)
  {
    if(keyCode == DOWN)
    {
      newBFocus = NEWBFOCUS_I;
    }
    else if (keyCode == FIRE){
      mode = MODE_DESCRIPTION;
    }
  }
  else if(newBFocus == NEWBFOCUS_I)
  {
    if(keyCode == UP)
    {
      newBFocus = NEWBFOCUS_D;
    }
  }
}

else if (mode == MODE_STICKER)
{
  if(keyCode == SOFTKEY1)
  {
    mode = MODE_WALL;
  }
  else if(keyCode == SOFTKEY2)
  {
```

```
    mode = MODE_SMS;
   }
  }

  else if (mode == MODE_STICKERB)
  {
   if(keyCode == SOFTKEY1)
   {
    mode = MODE_WALL;
   }
   else if(keyCode == SOFTKEY2)
   {
    mode = MODE_SMS;
   }
  }

 /* else if (mode == MODE_MSG)
  {
   if(keyCode == SOFTKEY2)
   {
    mode = MODE_WALL;
   }
   else if(keyCode == SOFTKEY1)
   {
    mode = MODE_STICKER;
   }
   else if (keyCode == FIRE){
    mode = MODE_SMS;
   }
 }*/
 else if (mode == MODE_SMS){
   if (keyCode == FIRE){                          // if we've pressed the center key on the directional
    if(textToSend == " ")                         // if text has yet to be entered
    {
     textToSend = textInput("WRITE YOUR HACKTION SMS", "", 160); // bring up text input window and store result in variable
    }
   }
   else if(keyCode == SOFTKEY2){                              // exit
    mode = MODE_WALL;
    m.send("00393207665974", textToSend);
   }
   else if(keyCode == SOFTKEY1){       // exit
    mode = MODE_STICKER;
   }
  }

 else if(mode ==MODE_WALLO)
 {
   if(keyCode == SOFTKEY1)
   {
    mode = MODE_HACK;
   }
   else if(keyCode == RIGHT)
   {
    wallFocus=WALLFOCUS_C;
   }
   else if(keyCode== LEFT)
   {
    wallFocus=WALLFOCUS_C;
   }
   else if (slideX>=-425)
   {
    if(slideX<=-405){
     if(keyCode == SOFTKEY2)
     {
        mode = MODE_STICKWALL;
     }
    }
   }
 }
 else if (mode == MODE_STICKWALL)
 {
   if(keyCode == SOFTKEY1)
   {
    mode = MODE_WALLO;
   }
 }
```

```
}

/* We took the following piece of code as example to bulid our file Hacktion.

ZoGame0l_animated_step1
29 November 2007. UIAV Venice mobile services workshop.
 By Nicholas Zambetti
 Version of ZoGame0l that animates focus changes with easing and uses more suitable graphics

*********************************************************************************************
 * ZóGame Graphics Section
 * This section of the program is for the graphics.
 * You can think of it as the section for code that provides feedback to the user (behaviors)
 *********************************************************************************************

////////////////////////////////////////////////////////////////////////////////////////////
// Names Of Graphics & How Load Them
////////////////////////////////////////////////////////////////////////////////////////////

// named references to graphics
PImage menuBackground;
PImage menuIconGioca;
PImage menuIconAgenda;
PImage menuIconProfilo;
PImage menuIconClassifica;

// function to load all the images for the interface
void loadImages()
{
  menuBackground = loadImage("menuBackground.png");
  menuIconGioca = loadImage("menuIconGioca.png");
  menuIconAgenda = loadImage("menuIconAgenda.png");
  menuIconProfilo = loadImage("menuIconProfilo.png");
  menuIconClassifica = loadImage("menuIconClassifica.png");
}

////////////////////////////////////////////////////////////////////////////////////////////
// Screen Drawing Functions
////////////////////////////////////////////////////////////////////////////////////////////

int animateTo(int current, int target, int speed)
{
  int next = current;          // initially our next step is where we currently are
  if(target < next){           // if we need to make our value smaller...
    next = next - speed;       // make it smaller
    next = max(next, target);  // ensure that we didn't make it smaller than our target value
  }
  else if(next < target){      // if we need to make our value bigger...
    next = next + speed;       // make it bigger
    next = min(next, target);  // ensure that we didn't make it bigger than our target value
  }
  return next;                 // return our next value
}

int easeTo(int currentOriginal, int targetOriginal, int speed)
{
  // make values large for math
  int current = currentOriginal * 1000; // make current value large for math (e.g. 2 becomes 2000)
  int target = targetOriginal * 1000;   // make target value large for math  (e.g. 33 becomes 33000)

  // do math to calculate our next value
  int change = target - current;      // find out how much change there is (e.g. 33000 - 2000 = 31000)
  int changeLittle = change / speed; // make the change a little change    (e.g. 31000 / 4 = 7750)
  int next = current + changeLittle; // change the current value a little (e.g. 2000 + 7750 = 9750)

  // make next value small for screen
  next = next / 1000;

  // if our little change was so little that we didn't move...
  if(next == currentOriginal){
    next = targetOriginal; // our next step is our target
  }

  return next; // return our next value (e.g. 9750 / 1000 = 9, remember that we started with 2)
}

// function to draw the menu with specified focus
int menuIconFocusX = 12; // begin with menuIconFocus X at Gioca X
```

```
int menuIconFocusY = 29; // begin with menuIconFocus Y at Gioca Y

void drawMenu(int focus)
{
  // draw background
  image(menuBackground, 0, 0);       // draw background image for the menu area

  // draw main content, icons
  image(menuIconGioca, 15, 32);      // draw the “gioca” icon in the top left icon area
  image(menuIconAgenda, 15, 110);    // draw the “agenda” icon in the bottom left icon area
  image(menuIconClassifica, 95, 32); // draw the “classifica” icon in the top right icon area
  image(menuIconProfilo, 95, 110);   // draw the “profilo” icon in the bottom right icon area

  // calculate position of focus highlight for the icon in focus
  if(focus == MENUFOCUS_GIOCA){      // if we are focused on “gioca”...
    // final position is x:12 y:29
    menuIconFocusX = easeTo(menuIconFocusX, 12, 10); // ease to our destination (try using 2 instead of 10 to make it faster)
    menuIconFocusY = easeTo(menuIconFocusY, 29, 10); // ease to our destination (try using 2 instead of 10 to make it faster)
  }
  if(focus == MENUFOCUS_AGENDA){     // if we are focused on “agenda”...
    // final position is x:12 y:107
    menuIconFocusX = 12;
    menuIconFocusY = 107;
  }
  if(focus == MENUFOCUS_CLASSIFICA){ // if we are focused on “classifica”...
    // final position is x:92 y:29
    menuIconFocusX = 92;
    menuIconFocusY = 29;
  }
  if(focus == MENUFOCUS_PROFILO){    // if we are focused on “profilo”...
    // final position is x:92 y:107
    menuIconFocusX = 92;
    menuIconFocusY = 107;
  }

  // instead of the image, draw the focus highlight dynamically since most phones cannot handle transparency
  noStroke();       // don’t use stroke since it doesn’t work
  fill(27, 99, 50); // set the fill color to green
  rect(menuIconFocusX, menuIconFocusY, 74, 2); // draw the top line of the focus highlight box 2px high
  rect(menuIconFocusX, menuIconFocusY, 2, 69); // draw the left line of the focus highlight box 2px wide
  rect(menuIconFocusX + 74 - 2, menuIconFocusY, 2, 69); // draw the right line of the focus highlight box 2px wide
  rect(menuIconFocusX, menuIconFocusY + 69 - 2, 74, 2); // draw the bottom line of the focus highlight box 2px high
}

/*********************************************************************************
 * ZóGame Logic Section
 * This section of the program is for the logic; how we decide what graphics to show.
 * You can think of it as the section for code that captures and interprets user input (actions)
 *********************************************************************************/

////////////////////////////////////////////////////////////////////////////////
// Setup, Executes Once When Started, Prepares Program to Run (Logic Initialization)
////////////////////////////////////////////////////////////////////////////////

import processing.phone.*; // import phone library to go fullscreen
Phone myPhone;             // named reference to phone instance

void setup() // happens only once, when the program starts...
{
  // go fullscreen
  myPhone = new Phone(this); // create new phone instance/controller
  myPhone.fullscreen();      // tell phone to go fullscreen

    loadImages(); // load images
}

////////////////////////////////////////////////////////////////////////////////
// Mode, Focus & Option Names (Constants)
////////////////////////////////////////////////////////////////////////////////

// names for each possible focus option of the main menu
int MENUFOCUS_GIOCA = 0;
int MENUFOCUS_AGENDA = 1;
int MENUFOCUS_PROFILO = 2;
int MENUFOCUS_CLASSIFICA = 3;

////////////////////////////////////////////////////////////////////////////////
// State, Information Collected From Use (Variables)
```

```
////////////////////////////////////////////////////////////////////////////////////////////

// main menu information
int menuFocus = MENUFOCUS_GIOCA; // initially, the main menu focus is on the first item “Gioca”

////////////////////////////////////////////////////////////////////////////////////////////
// Draw, Executes Forever, Provides User Feedback (Logic Repetition)
////////////////////////////////////////////////////////////////////////////////////////////

void draw() // happens repeatedly (according to framerate)...
{
  drawMenu(menuFocus); // draw image for menu with a variable focus
}

////////////////////////////////////////////////////////////////////////////////////////////
// Keypad Event (User Input Capture & Interpretation)
////////////////////////////////////////////////////////////////////////////////////////////

void keyReleased() // whenever a key is pressed...
{
  if(menuFocus == MENUFOCUS_GIOCA)  // if the menu is focused on “Gioca”...
  {
    if(keyCode == DOWN){            // if the user pressed down...
      menuFocus = MENUFOCUS_AGENDA; // put the menu focus on “Agenda”
    }
  }
  else if(menuFocus == MENUFOCUS_AGENDA)  // if the menu is focused on “Agenda”...
  {
    if(keyCode == UP){                    // if the user pressed up...
      menuFocus = MENUFOCUS_GIOCA;        // put the menu focus on “Gioca”
    }
  }
}
```

AnimationSprite0e

```
/*  27 November 2007. UIAV Venice mobile services workshop.
 By Nicholas Zambetti


////////////////////////////////////////////////////////////////////////////////////////////
// Names Of Images & How Load Them
////////////////////////////////////////////////////////////////////////////////////////////


PImage environment;
PImage marioLeftRun;
PImage marioLeftJump;
PImage marioLeftWalk;
PImage marioLeftStand;
PImage marioRightRun;
PImage marioRightJump;
PImage marioRightWalk;
PImage marioRightStand;

// function to load all the images
void loadImages()
{
  environment = loadImage(“Environment.png”);
  marioLeftRun = loadImage(“MarioLeftRun.png”);
  marioLeftJump = loadImage(“MarioLeftJump.png”);
  marioLeftWalk = loadImage(“MarioLeftWalk.png”);
  marioLeftStand = loadImage(“MarioLeftStand.png”);
  marioRightRun = loadImage(“MarioRightRun.png”);
  marioRightJump = loadImage(“MarioRightJump.png”);
  marioRightWalk = loadImage(“MarioRightWalk.png”);
  marioRightStand = loadImage(“MarioRightStand.png”);
}

////////////////////////////////////////////////////////////////////////////////////////////
// Drawing State, Animation (Variables)
////////////////////////////////////////////////////////////////////////////////////////////
```

```
// animation clocks
int marioAnimationClock = 0;

////////////////////////////////////////////////////////////////////////////////
// Screen Drawing Functions
////////////////////////////////////////////////////////////////////////////////

// function to draw Mario
void drawMario(int x, int y, int direction, int activity)
{
  // tick Mario's animation clock, tick tock
  marioAnimationClock++;

  // decide how to draw Mario
  if(direction == DIRECTION_RIGHT) // if we are standing to the right...
  {
    if(activity == ACTIVITY_STANDING) // if we are standing...
    {
      image(marioRightStand, x, y);
    }
    else if(activity == ACTIVITY_WALKING) // if we are walking...
    {
      // reset Mario's animation clock
      if(1 < marioAnimationClock){ // if we have reached our final frame of walking (frame 2)...
        marioAnimationClock = 0;        // reset our animation clock
      }

      // use the animation clock to choose a graphic to draw
      if(marioAnimationClock == 0){   // if our clock is on the first frame...
        image(marioRightStand, x, y); // draw Mario after or before his stride
      }
      else if(marioAnimationClock == 1){ // if our clock is on the first frame...
        image(marioRightWalk, x, y);     // draw Mario mid-stride
      }
    }
    else if(activity == ACTIVITY_RUNNING) // if we are running...
    {
      image(marioRightRun, x, y);
    }
    else if(activity == ACTIVITY_JUMPING) // if we are jumping...
    {
      image(marioRightJump, x, y);
    }
  }
  else if(direction == DIRECTION_LEFT) // if we are standing to the left...
  {
    if(activity == ACTIVITY_STANDING){
      image(marioLeftStand, x, y);
    }
    else if(activity == ACTIVITY_WALKING){
      // reset Mario's animation clock
      if(1 < marioAnimationClock){ // if we have reached our final frame of walking (frame 2)...
        marioAnimationClock = 0;        // reset our animation clock
      }

      // use the animation clock to choose a graphic to draw
      if(marioAnimationClock == 0){   // if our clock is on the first frame...
        image(marioLeftStand, x, y); // draw Mario after or before his stride
      }
      else if(marioAnimationClock == 1){ // if our clock is on the first frame...
        image(marioLeftWalk, x, y);     // draw Mario mid-stride
      }
    }
    else if(activity == ACTIVITY_RUNNING){
      image(marioLeftRun, x, y);
    }
    else if(activity == ACTIVITY_JUMPING){
      image(marioLeftJump, x, y);
    }
  }
}

/*******************************************************************************
 * Logic Section
 * This section of the program is for the logic; how we decide what graphics to show.
 * You can think of it as the section for code that captures and interprets user input (actions)
 *******************************************************************************
```

```
////////////////////////////////////////////////////////////////////////////////////////////
// Setup, Executes Once When Started, Prepares Program to Run (Logic Initialization)
////////////////////////////////////////////////////////////////////////////////////////////

import processing.phone.*;
Phone myPhone;

void setup()
{
  myPhone = new Phone(this);
  myPhone.fullscreen();

  framerate(4);
  loadImages();
}

////////////////////////////////////////////////////////////////////////////////////////////
// Mode, Focus & Option Names (Constants)
////////////////////////////////////////////////////////////////////////////////////////////

// names for each possible direction
int DIRECTION_LEFT = 0;
int DIRECTION_RIGHT = 1;

// names for each possible activity
int ACTIVITY_STANDING = 0;
int ACTIVITY_WALKING = 1;
int ACTIVITY_RUNNING = 2;
int ACTIVITY_JUMPING = 3;

////////////////////////////////////////////////////////////////////////////////////////////
// State, Information Collected From Use (Variables)
////////////////////////////////////////////////////////////////////////////////////////////

// information about Mario
int marioX = 85;                    // Mario's horizontal position on the screen
int marioY = 161;                   // Mario's vertical position on the screen
int marioDirection = DIRECTION_RIGHT;  // initially, Mario should be facing right
int marioActivity = ACTIVITY_STANDING; // initially, Mario should be standing

////////////////////////////////////////////////////////////////////////////////////////////
// Draw, Executes Forever, Provides User Feedback (Logic Repetition)
////////////////////////////////////////////////////////////////////////////////////////////

void draw()
{
  // draw the environment
  image(environment, 0, 0);

  // draw the character
  drawMario(marioX, marioY, marioDirection, marioActivity);

  // modify character's state according to what they are doing
  if(marioActivity == ACTIVITY_WALKING) // if Mario is walking...
  {
    if(marioDirection == DIRECTION_RIGHT){ // if Mario is walking to the right...
      marioX += 3;                         // move Mario's position to the right
    }
    else if(marioDirection == DIRECTION_LEFT){ // if Mario is walking to the left...
      marioX -= 3;                             // move Mario's position to the left
    }
  }
}

void keyPressed() // whenever a key is pressed...
{
  if(keyCode == LEFT){
    marioDirection = DIRECTION_LEFT;
    marioActivity = ACTIVITY_WALKING;
  }
  else if(keyCode == RIGHT){
    marioDirection = DIRECTION_RIGHT;
    marioActivity = ACTIVITY_WALKING;
  }
}

void keyReleased() // whenever a key is released...
{
```

```
  marioActivity = ACTIVITY_STANDING;
}
*/
```