```
////////////////////////////////////////////////////////////////////////////////////
// Dèrive version 0.29a
// An application made for Nokia E60 mobile phone
// by Giuseppe Cosmai, Valeria Donati and Maria Tasca
// with the help of Christian Palino, Vinay Ventrakamen and Nicholas Zambetti
// IUAV Interaction Design Programme 2008/2009
// by Gillian Crampton Smith and Philip Tabor
// http://www.interaction-venice.com
////////////////////////////////////////////////////////////////////////////////////


/**************************************************************************
**********************
 * Graphics Section - Code that provides feedback to the user (behaviors)

**************************************************************************
**********************/

// Named references to images for the core GUI  (core 1 & 2)
PImage core1;  // splash, menu bg, empty bg
PImage core2;  // transparencies (high fog and frozen)
PImage menuoptions;  //txt  inserire in core 2
PImage fog;
PImage softBg;

//softkeys and toolbars
PImage tools;     // toolbass and softkeys texts
PImage menufocusbar;


// path stuff
PImage pathTxt1;   //txt
PImage pathTxt2;   //txt
PImage pathIcn;    //icons
PImage pathImg;    //txt
PImage lmfocusbarV;
PImage lmfocusbarH;
PImage selBall;

// upload stuff (single pieces)
PImage lastBar;
PImage upBar;
PImage baloon1;
PImage baloon2;


// upload txt
PImage upFil;   //testo
PImage chSense;
```

```
PImage upTxt;

// senses
PImage senses;
PImage senseTxt;
PImage arrows;


// Named references to images for navigation e maps
PImage mapBg;
PImage pathNav;
PImage feelings;


// Function to load all the images
void loadImages()
{
  core1 = loadImage("core1.jpg");
  core2 = loadImage("core2.png");
  tools = loadImage("tools.png");
  menuoptions = loadImage("menuoptions.png");
  menufocusbar = loadImage("menufocusbar.png");
  fog = loadImage("fog.png");
  softBg = loadImage("softBg.png");
  senses = loadImage("senses.png");
  senseTxt = loadImage("senseTxt.png");

}
// Function to load all the path images
void loadPathImages()
{
  pathTxt1 = loadImage("pathTxt1.png");
  pathTxt2 = loadImage("pathTxt2.png");
  pathIcn = loadImage("pathIcn.png");
  pathImg = loadImage("pathImg.jpg");

  lmfocusbarV = loadImage("lmfocusV.png");
  lmfocusbarH = loadImage("lmfocusH.png");
  chSense = loadImage("chSense.png");
  arrows = loadImage("arrows.png");
  pathNav = loadImage("pathNav.jpg");
  selBall = loadImage("selBall.png");
  baloon1 = loadImage("baloon1.png");
  baloon2 = loadImage("baloon2.png");
}

// Function to load all the upload screen images
void loadUpImages()
{
  upTxt = loadImage("upTxt.png");
```

```
  chSense = loadImage("chSense.png");
  arrows = loadImage("arrows.png");
}

// Function to load all the maps screen images
void loadMapImages()
{
  feelings = loadImage("feelings.png");
  mapBg = loadImage("mapBg.jpg");
}

void unloadImages()
{
  core1 = null;
  core2 = null;
  tools = null;
  menuoptions = null;
  menufocusbar = null;
}
// Function to unload all the path screen images
void unloadPathImages()
{
  pathTxt1 = null;
  pathTxt2 = null;
  pathIcn = null;
  pathImg = null;

  lmfocusbarV = null;
  lmfocusbarH = null;
  chSense = null;

  selBall = null;
  baloon1 = null;
  baloon2 = null;
}

// Function to load all the upload screen images
void unloadUpImages()
{
  upTxt = null;
  chSense = null;
  arrows = null;
}

// Function to unload all the maps screen images
void unloadMapImages()
{
  feelings = null;
  mapBg = null;
}
```

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

// Function to draw the menu screen
int menuFocusYPosition = 55;
void drawMenu()
{
  // draw menu background
  image(core1, width * 1, 0, width, height, 0, 0);


  // draw focus bar
  if(menuFocus == MENU_OPTION_PATH){
    menuFocusYPosition = easeOut(menuFocusYPosition, 55, 2);
  }
  else if(menuFocus == MENU_OPTION_RANDOM){
    menuFocusYPosition = easeOut(menuFocusYPosition, 110, 2);
  }
  else if(menuFocus == MENU_OPTION_MAPS){
    menuFocusYPosition = easeOut(menuFocusYPosition, 164, 2);
  }
  else if(menuFocus == MENU_OPTION_MYMAPS){
    menuFocusYPosition = easeOut(menuFocusYPosition, 219, 2);
  }
  else if(menuFocus == MENU_OPTION_UPLOAD){
    menuFocusYPosition = easeOut(menuFocusYPosition, 270, 2);
  }
  else if(menuFocus == MENU_OPTION_HELP){
    menuFocusYPosition = easeOut(menuFocusYPosition, 325, 2);
  }
  image(menufocusbar, 0, menuFocusYPosition);

  // draw menu options text
  image(menuoptions, 0, 0);
}



//////////////////////////////////////////////////
// Function to draw screen path
//////////////////////////////////////////////////

void drawScreenPath()
{

  if(pathMode > 0 && pathMode != 30 && pathAsking == true && pathAsked ==
false){
    prevPathMode = pathMode;
    pathMode = 30;
```

```
}

switch(pathMode) {

case 0:                 // inizio
  println(pathMode);
  drawEmptyBg();
  image(pathTxt1, width * 0, 0, width, height, 0, 0);
  m = millis();
  if (timeSet == false){
    time = m + 3000;
    timeSet = true;
  }
  if(m > time){
    m = 0;
    time = 0;
    timeSet = false;
    pathMode++;
  }
  break;

case 1:                 // select landmark 1
  drawEmptyBg();
  drawLmFocus();
  softCode = 5;
  drawLowBar();
  println(pathMode);
  break;

case 2:                 // select landmark 2
  drawEmptyBg();
  drawLmFocus2();
  softCode = 5;
  drawLowBar();
  println(pathMode);
  break;

case 3:                 // select sense

  drawEmptyBg();
  drawSenses();
  softCode = 5;
  drawLowBar();
  println(pathMode);
  break;

case 4:                 // summary
  drawEmptyBg();
  image(pathTxt1, width * 2, 0, width, height, 0, 0);
  softCode = 4;
```

```
  drawLowBar();
  println(pathMode);
  break;

case 5:                         // path watch
  drawNavBg();
  if(toolbar == true){
    image(core2, width * 0, 0, width, height, 0, 0);    //freeeze!
  }
  if(rPressed == true){
    println("pressed");
    m = millis();
    if (timeSet == false){
      time = m + 2000;
      timeSet = true;
    }
    println(m);
    println(time);
    if(m > time){
      m = 0;
      time = 0;
      timeSet = false;
      rPressed = false;
      pathMode = 6;
    }
  }
  softCode = 1;
  drawUpperBar();
  drawLowBar();
  drawHelpers();
  println(pathMode);
  break;

case 6:                       // generate and control balls
  drawNavBg();
  drawBubb();
  noStroke();
  image(selBall, selX-30, selY-30);          // selezione
  if(toolbar == true){
    image(core2, width * 0, 0, width, height, 0, 0);    //freeeze!
  }
  softCode = 1;
  drawUpperBar();
  drawLowBar();
  println(toolCode);
  break;

case 7:                         // tagcloud
  image(pathImg, width * 0, 0, width, height, 0, 0);
  softCode = 3;
```

```
  drawLowBar();
  println(pathMode);
  break;

 case 8:                        // exploration
  drawNavBg();
  drawExplo();
  if(toolbar == true){
    image(core2, width * 0, 0, width, height, 0, 0);    //freeeze!
  }
  softCode = 1;
  drawUpperBar();
  drawLowBar();
  println(pathMode);
  break;

 case 9:                        // start touching
  image(pathImg, width * 1, 0, width, height, 0, 0);
  m = millis();
  if (timeSet == false){
    time = m + 4000;
    timeSet = true;
  }
  println(m);
  println(time);
  if(m > time){
    m = 0;
    time = 0;
    timeSet = false;
    pathMode = 10;
  }
  println(pathMode);
  break;

 case 10:                       // comment
  drawEmptyBg();
  image(pathTxt2, width * 0, 0, width, height, 0, 150);
  image(pathTxt2, width * 1, 0, width, height, 0, 0);
  softCode = 4;
  drawLowBar();
  println(pathMode);
  break;

 case 11:                       // hint
  drawEmptyBg();
  image(pathTxt2, width * 2, 0, width, height, 0, 0);
  softCode = 4;
  drawLowBar();
  println(pathMode);
  break;
```

```
  case 12:                      // feed
    drawEmptyBg();
    how = textInput("how did you felt?", "", 16);
    pathMode = 13;
    println(pathMode);
    break;

  case 13:                      // intensity
    drawEmptyBg();
    drawIntensity();
    softCode = 4;
    drawLowBar();
    println(pathMode);
    break;

  case 14:                      // feed
    drawEmptyBg();
    println(pathMode);
    break;



  case 30:                    // continue question
    drawEmptyBg();
    image(pathTxt1, width * 1, 0, width, height, 0, 0);
    softCode = 4;
    drawLowBar();
    println(pathMode);
    break;

  case 20:
    break;

  }
}



//////////////////////////////////////////////////
// Function to draw screen random
//////////////////////////////////////////////////

void drawScreenRandom()
{
  drawEmptyBg();
}

//////////////////////////////////////////////////
// Function to draw screen upload
```

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

void drawScreenUpload()
{

 switch(upMode) {

 case 0:
  println(upMode);
  upMode++;
  break;

 case 1:                    // select sense
  drawEmptyBg();
  drawSenses();
  softCode = 5;   //back
  drawLowBar();
  println(upMode);
  break;

 case 2:                    // post a comment


  drawEmptyBg();
  upBar = loadImage("upBar.png");
  drawUpBar();
  image(upTxt, width * 0, 0, width, height, 0, 0);
  PFont font;
  font = loadFont("Verdana-20.mvlw");
  textFont(font);
  text(upComm, 55, 260);
  text(upSource, 55, 125);
  println(upMode);
  break;

 case 3:                    // post a comment
  upBar = null;
  if(upY == 181){
    upComm = textInput("post a comment", "Nice!", 20);
  }
  else{
    upSource = textInput("the source", "door", 9);
  }
  upMode++;
  break;

 case 4:                    // duration
  lastBar = loadImage("lastBar.png");
  drawEmptyBg();
  image(lastBar, 0, 116);
```

```
    image(upTxt, width * 1, 0, width, height, 0, 0);
    softCode = 5;
    drawLowBar();
    println(upMode);
    break;

  case 5:                   // summary
    drawEmptyBg();
    lastBar = null;
    image(upTxt, width * 2, 0, width, height, 0, 0);
    softCode = 4;
    drawLowBar();
    println(upMode);

    //upMode++;
    break;

  case 6:                   // done
    drawEmptyBg();
    image(upTxt, width * 3, 0, width, height, 0, 0);
    println(upMode);
    softCode = 6;
    drawLowBar();
    //upMode++;
    break;
  }
}

//////////////////////////////////////////////////
// Function to draw screen maps
//////////////////////////////////////////////////

void drawScreenMaps()
{
  drawMap();
}

//////////////////////////////////////////////////
// Function to draw screen mymaps
//////////////////////////////////////////////////

void drawScreenMyMaps()
{
  drawMap();
}

//////////////////////////////////////////////////
// Function to draw screen help
//////////////////////////////////////////////////
```

```
void drawScreenHelp()
{
  drawEmptyBg();
}

////////////////////////////////////////////////////
// Function to draw fog
////////////////////////////////////////////////////

void drawFog()
{
  if(screenMode != SCREEN_SPLASH){
    image(fog, f, 0);
    f = f - 2;
    if (f < (-704)){
      f = 0;
    }
  }
}

////////////////////////////////////////////////////
// Function to draw empty background
////////////////////////////////////////////////////

void drawEmptyBg()
{
  image(core1, width * 2, 0, width, height, 0, 0);
}

////////////////////////////////////////////////////
// Function to draw upper toolbar
////////////////////////////////////////////////////

void drawUpperBar()
{
  image(core2, width * 1, 0, width, height, 0, 0);

  if(toolbar == true){
    if(toolFocus == 1){
      image(tools, 0, 40 * 9, width, 40, 0, 0);
    }
    else if(toolFocus == 2){
      image(tools, 0, 40 * 8, width, 40, 0, 0);
    }
  }
  if(toolCode == 1){
    image(tools, 0, 40 * 6, width, 40, 0, 0);
  }
  else if(toolCode == 2){
    image(tools, 0, 40 * 7, width, 40, 0, 0);
```

```
  }
}

//////////////////////////////////////////////////
// Function to draw lower toolbar
//////////////////////////////////////////////////

void drawLowBar()
{
  image(softBg, 0, height - 150);
  image(tools, 0, 40 * (softCode - 1), width, 40, 0, height - 40);
}

//////////////////////////////////////////////////
// Function to draw sense selection
//////////////////////////////////////////////////

void drawSenses()
{
  image(chSense, 0, 55);
  if(senseCode == 1){
    image(senses, 0, 160 * 0, width, 160, 0, 125);
    image(senseTxt, 0, 31 * 0, width, 31, 0, 304);

  }
  else if(senseCode == 2){
    image(senses, 0, 160 * 1, width, 160, 0, 125);
    image(senseTxt, 0, 31 * 1, width, 31, 0, 304);
  }
  else if(senseCode == 3){
    image(senses, 0, 160 * 2, width, 160, 0, 125);
    image(senseTxt, 0, 31 * 2, width, 31, 0, 304);
  }
  else if(senseCode == 4){
    image(senses, 0, 160 * 3, width, 160, 0, 125);
    image(senseTxt, 0, 31 * 3, width, 31, 0, 304);
  }
  else if(senseCode == 5){
    image(senses, 0, 160 * 4, width, 160, 0, 125);
    image(senseTxt, 0, 31 * 4, width, 31, 0, 304);
  }
  if(rPressed == true){
    image(arrows, 0, 31 * 0, width, 31, 0, 304);
  }
  else if(lPressed == true){
    image(arrows, 0, 31 * 1, width, 31, 0, 304);
  }
  else if(rPressed == false && lPressed == false){
    image(arrows, 0, 31 * 2, width, 31, 0, 304);
  }
```

```
}

//////////////////////////////////////////////////
// Function to draw nav background
//////////////////////////////////////////////////

void drawNavBg()
{
  if(pathMode == 5){
    if (toolCode == 2){
      image(pathNav, width * 1, 0, width, height, 0, 0);
    }
    else{
      background(255);
      image(pathNav, width * 0, 0, width, height, 0, 0);
    }
  }
  else if(pathMode == 6){
    if (toolCode == 2){
      image(pathNav, width * 3, 0, width, height, 0, 0);
    }
    else{
      background(255);
      image(pathNav, width * 2, 0, width, height, 0, 0);
    }
  }
  else if(pathMode == 8){
    background(255);
    if (toolCode == 2){
      image(pathNav, width * 5, 0, width, height, 0, 0);
    }
    else{
      background(255);
      image(pathNav, width * 4, 0, width, height, 0, 0);
    }
  }
}

//////////////////////////////////////////////////
// Function to draw bubbles
//////////////////////////////////////////////////

void drawBubb()
{
  for(int i = 0; i < balls.length; i++){
    balls[i].render();
  }
}

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

```
// Function to draw landmark focusbar
////////////////////////////////////////////////////

int lmYPosition = 54;
int lmXPosition = -300;
void drawLmFocus()
{

  // draw first lm focus bar
  if(lmFocus == 0){
    lmYPosition = easeOut(lmYPosition, 54, 2);
  }
  else if(lmFocus == 1){
    lmYPosition = easeOut(lmYPosition, 82, 2);
  }
  else if(lmFocus == 2){
    lmYPosition = easeOut(lmYPosition, 110, 2);
  }
  else if(lmFocus == 3){
    lmYPosition = easeOut(lmYPosition, 137, 2);
  }
  else if(lmFocus == 4){
    lmYPosition = easeOut(lmYPosition, 165, 2);
  }
  else if(lmFocus == 5){
    lmYPosition = easeOut(lmYPosition, 194, 2);
    lmXPosition = -300;
  }

  // draw seconfd lm focus bar
  if(lmFocus == 6){
    lmXPosition = easeOut(lmXPosition, 33, 2);
  }
  else if(lmFocus == 7){
    lmXPosition = easeOut(lmXPosition, 191, 2);
  }


  image(lmfocusbarV, 0, lmYPosition);
  image(lmfocusbarH, lmXPosition, 287);

  // draw landmarks options text
  image(pathIcn, width * 0, 0, width, height, 0, 0);
}

////////////////////////////////////////////////////
// Function to draw landmark focusbar in second page
////////////////////////////////////////////////////

int lmYPosition2 = 70;
```

```
int lmXPosition2 = 33;
void drawLmFocus2()
{

  // draw seconfd lm focus bar
  if(lmFocus2 == 1){
    lmXPosition2 = 33;
    lmYPosition2 = 70;
  }
  else if(lmFocus2 == 2){
    lmXPosition2 = 191;
    lmYPosition2 = 70;
  }
  else if(lmFocus2 == 3){
    lmXPosition2 = 33;
    lmYPosition2 = 162;
  }
  else if(lmFocus2 == 4){
    lmXPosition2 = 191;
    lmYPosition2 = 162;
  }
  else if(lmFocus2 == 5){
    lmXPosition2 = 33;
    lmYPosition2 = 254;
  }
  else if(lmFocus2 == 6){
    lmXPosition2 = 191;
    lmYPosition2 = 254;
  }

  image(lmfocusbarH, lmXPosition2, lmYPosition2);

  // draw landmarks options text
  image(pathIcn, width * 1, 0, width, height, 0, 0);
}

//////////////////////////////////////////////////////
// Function to draw help baloons in the path view
//////////////////////////////////////////////////////

void drawHelpers()
{
  m = millis();
  if (timeSet == false && helped == false){
    baloon = 1;
    time = m + 2000;
    timeSet = true;
  }
  if(m > time && helped == false){
    baloon++;
```

```
    time = m + 2000;
  }
  if(baloon == 3){
    baloon = 0;
    m = 0;
    time = 0;
    timeSet = false;
    helped = true;
    baloon1 = null;
    baloon2 = null;
  }


  if(baloon == 1){
    image(baloon1, 0, 0);
  }
  else if(baloon == 2){
    image(baloon2, 0, 0);
  }

}


//////////////////////////////////////////////////
// Function to draw exploration
//////////////////////////////////////////////////

void drawExplo()
{
  PImage navBal1;
  PImage navBal2;
  navBal1 = loadImage("navBal1.png");
  navBal2 = loadImage("navBal2.png");
  if(animate == 1){
    elX = easeOut(elX, 315, 2);
  }
  else if(animate == 2){
    elY = easeOut(elY, 190, 2);
  }
  if(elX < 310){
    image (navBal1,0,0);
  }
  else if(elX > 312){
    image (navBal2,0,0);
  }
  if(elY < 200){
    myPhone.vibrate(2500);
    pathMode = 9;
  }
  fill(0);
```

```
  noStroke();
  ellipse(elX, elY, 38, 38);
}


//////////////////////////////////////////////////
// Function to draw intensity
//////////////////////////////////////////////////

void drawIntensity()
{
  colorMode(HSB, 100);
  noStroke();
  fill(0, 98, 45);
  ellipse(width/2, height-120, 26, 26);
  fill(0, 98, 75);
  ellipse(width/2, height-200, 36, 36);
  fill(0, 98, 105);
  ellipse(width/2, height-280, 46, 46);
  colorMode(RGB, 255);
  image(selBall, (width/2)-30, feedY);          // selezione
  println(feedY);
  image(pathTxt2, width * 3, 0, width, height, 0, 0);
}

//////////////////////////////////////////////////
// Function to draw intensity
//////////////////////////////////////////////////

void drawScreenSplash()
{
  image(core1, width * 0, 0, width, height, 0, 0);
  m = millis();
  if (timeSet == false){
    time = m + 2000;
    timeSet = true;
  }
  if(m > time){
    m = 0;
    time = 0;
    timeSet = false;
    screenMode = SCREEN_MENU;
  }
}

//////////////////////////////////////////////////
// Function to draw upload comment bar
//////////////////////////////////////////////////

void drawUpBar()
```

```
{
  image(upBar, 0,  upY);
}

////////////////////////////////////////////////
// Function to draw map
////////////////////////////////////////////////

void drawMap()
{
   image(mapBg, 0, 0);
   if(redGrowing == true){
   for(m = 3; m <= 5; m++){
   image(feelings, width * 0, 0, width, height, 0, 0);  //red
   if(m == 5){
   redGrowing = false;
   }
   }
   }else{
   for(m = 5; m >= 0; m--){
   image(feelings, width * 0, 0, width, height, 0, 0);  //red
   if(m == 0){
   redGrowing = true;
   }
   }
   }
}


/*************************************************************************
**********************
 * Animation Section - Code that performs animation calculations

*************************************************************************
**********************/

// linear animation function (relative)
// e.g. animateTo(x, 100, 10);
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
}

// ease out animation function (relative)
// e.g. easeIn(x, 100, 10);
int easeOut(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)
}


/*****************************************************************************
**********************
 * Feelings pointer's classess

*****************************************************************************
**********************/

class Ball
{
  int _x = 0;
  int _y = 0;
  int _radius = 10;
  color _color;
  boolean _selected = false;
```

```
  Ball(int theX, int theY, int theRadius, color theColor){
    _x = theX;
    _y = theY;
    _radius = theRadius;
    _color = theColor;
  }

  void setPosition(int theX, int theY){
    _x = theX;
    _y = theY;
  }

  void setRadius(int theRadius){
    _radius = theRadius;
  }

  void setColor(color theColor){
    _color = theColor;
  }

  void select(){
    _selected = true;
  }

  void unselect(){
    _selected = false;
  }

  void render(){
    noStroke();
    fill(_color);
    if(_selected){
      selX = _x;
      selY = _y;
    }
    ellipse(_x, _y, _radius, _radius);
  }
}


/**********************************************************************
**********************
 * Logic Section - Code that captures and interprets user input

**********************************************************************
**********************/

////////////////////////////////////////////////////////////////////////////////
// Libraries - Additional libraries that your program requires
////////////////////////////////////////////////////////////////////////////////
```

```
import processing.phone.*;
Phone myPhone; // Named reference to your phone

////////////////////////////////////////////////////////////////////////////////
// Screens - Named states of the program to make code more readable (Constants)
////////////////////////////////////////////////////////////////////////////////

// Names for each possible screen
int SCREEN_MENU = 0;
int SCREEN_PATH = 1;
int SCREEN_RANDOM = 2;
int SCREEN_UPLOAD = 3;
int SCREEN_MAPS = 4;
int SCREEN_MYMAPS = 5;
int SCREEN_HELP = 6;
int SCREEN_SPLASH = 7;

// Names for each possible menu focus selection
int MENU_OPTION_PATH = 0;
int MENU_OPTION_RANDOM = 1;
int MENU_OPTION_UPLOAD = 2;
int MENU_OPTION_MAPS = 3;
int MENU_OPTION_MYMAPS = 4;
int MENU_OPTION_HELP = 5;

////////////////////////////////////////////////////////////////////////////////
// State - Information collected from use (Variables)
////////////////////////////////////////////////////////////////////////////////

int screenMode = SCREEN_SPLASH;  // Overall mode of the application, start with
the menu
int menuFocus = MENU_OPTION_PATH;  // Focus state of the menu
int lmFocus = 0;  // Focus state of the lm selection
int lmFocus2 = 1;  // Focus state of the lm second page selection


////////////////////////////////////////////////////////////////////////////////
// Modes - States in the various screens
////////////////////////////////////////////////////////////////////////////////

// time/keys variables
int time = 0;
int m = 0;
int animate = 0;
int elX = 55;
int elY = 283;
int upY = 52;
int feedY = 106;
int pathMode = 0;
```

```
int prevPathMode = 0;
boolean pathLoaded = false;
boolean uploadLoaded = false;
boolean rPressed = false;
boolean lPressed = false;
boolean timeSet = false;
boolean pathAsking = false;
boolean pathAsked = false;
boolean helped = false;
boolean redGrowing = true;
int f = 0;
int baloon = 0;
int randMode = 0;
int upMode = 0;
int mapsMode = 0;
int mymapsMode = 0;
int helpMode = 0;
int selX = 0;
int selY = 0;
int softCode = 0;
int senseCode = 1;
int toolCode = 1;
int toolFocus = 0;
boolean toolbar = false;
int selectedBall = 0;
boolean rendered = false;
Ball[] balls = new Ball[6];

String how = "";
String upComm = "_";
String upSource = "_";


////////////////////////////////////////////////////////////////////////////////////////
// Events - Timed/Triggered events to keep the program running (Execution)
////////////////////////////////////////////////////////////////////////////////////////

// Setup - Executes only once, prepares program to run (Logic Initialization)
void setup()
{
  myPhone = new Phone(this); // Creates a phone controller
  myPhone.fullscreen(); // Use the entire screen

    loadImages();
    colorMode(HSB, 100);
    for(int i = 0; i < balls.length; i++){
      balls[i] = new Ball(random(40, width-40), random(80, height-80), random(15,
50), color(0, 98, (random(45,100))));
      balls[selectedBall].select();
    }
```

```
    colorMode(RGB, 255);
}

// Draw - Executes forever, provides user feedback (Logic Repetition)
void draw()
{
  if(screenMode == SCREEN_MENU){
    drawMenu();
  }
  else if(screenMode == SCREEN_SPLASH){
    drawScreenSplash();
  }
  else if(screenMode == SCREEN_PATH){
    if(pathLoaded == false){
    loadPathImages();
    pathLoaded = true;
    }
    uploadLoaded = false;
    drawScreenPath();
  }   else if(screenMode == SCREEN_RANDOM){
    drawScreenRandom();
  }
  else if(screenMode == SCREEN_UPLOAD){
    if(pathLoaded == false){
    loadUpImages();
    uploadLoaded = true;
    }
    pathLoaded = false;
    drawScreenUpload();
  }
 else if(screenMode == SCREEN_MAPS){
    loadMapImages();
    drawScreenMaps();
  }
 else if(screenMode == SCREEN_MYMAPS){
    loadMapImages();
    drawScreenMyMaps();
  }
 else if(screenMode == SCREEN_HELP){
    drawScreenHelp();
    }
    drawFog();
}


/**********************************************************************
**********************
 * Key Commands Section - Code to capture user's input
```

```
**********************************************************************
**********************/

// KeyPressed - Executes whenever a key is pressed, captures user input (User Input)
void keyPressed()
{

  // if showing menu
  if(screenMode == SCREEN_MENU){
    // if focused on option PATH
    if(menuFocus == MENU_OPTION_PATH){
      // if pressed down
      if(keyCode == DOWN){
        menuFocus = MENU_OPTION_RANDOM; // move focus to option RAND
      }
      // if pressed in
      else if(keyCode == FIRE){
        if (pathMode != 0 && pathAsked == false) {
          pathAsking = true;
          println("vabbe");
        }
        screenMode = SCREEN_PATH; // show screen PATH
      }
    }
    // if focused on option RANDOM
    else if(menuFocus == MENU_OPTION_RANDOM){
      // if pressed up
      if(keyCode == UP){
        menuFocus = MENU_OPTION_PATH; // move focus to option PATH
      }
      // if pressed down
      else if(keyCode == DOWN){
        menuFocus = MENU_OPTION_MAPS; // move focus to option UPLOAD
      }
      // if pressed in
      else if(keyCode == FIRE){
        screenMode = SCREEN_RANDOM; // show screen RANDOM
      }
    }
    // if focused on option MAPS
    else if(menuFocus == MENU_OPTION_MAPS){
      // if pressed up
      if(keyCode == UP){
        menuFocus = MENU_OPTION_RANDOM; // move focus to option RAND
      }
      // if pressed down
      else if(keyCode == DOWN){
        menuFocus = MENU_OPTION_MYMAPS; // move focus to option MAPS
      }
```

```
  // if pressed in
  else if(keyCode == FIRE){
    screenMode = SCREEN_MAPS; // show screen B
  }
}
// if focused on option MYMAPS
else if(menuFocus == MENU_OPTION_MYMAPS){
  // if pressed up
  if(keyCode == UP){
    menuFocus = MENU_OPTION_MAPS; // move focus to option UPLOAD
  }
  // if pressed down
  else if(keyCode == DOWN){
    menuFocus = MENU_OPTION_UPLOAD; // move focus to option MYMAPS
  }
  // if pressed in
  else if(keyCode == FIRE){
    screenMode = SCREEN_MYMAPS; // show screen B
  }
}
// if focused on option UPLOAD
else if(menuFocus == MENU_OPTION_UPLOAD){
  // if pressed up
  if(keyCode == UP){
    menuFocus = MENU_OPTION_MYMAPS; // move focus to option MAPS
  }
  // if pressed down
  else if(keyCode == DOWN){
    menuFocus = MENU_OPTION_HELP; // move focus to option HELP
  }
  // if pressed in
  else if(keyCode == FIRE){
    screenMode = SCREEN_UPLOAD; // show screen B
  }
}
// if focused on option HELP
else if(menuFocus == MENU_OPTION_HELP){
  // if pressed up
  if(keyCode == UP){
    menuFocus = MENU_OPTION_UPLOAD; // move focus to option MYMAPS
  }
  // if pressed in
  else if(keyCode == FIRE){
    screenMode = SCREEN_HELP; // show screen C
  }
}
// if pressed right softkey (while focused on any option)
if(keyCode == SOFTKEY2){
  exit(); // exit application
}
```

```
}

// if showing screen path

if(screenMode == SCREEN_PATH){

  // if pressed rigth soft key
  if(keyCode == SOFTKEY2){
    if(pathMode == 30){
      pathAsked = true;
      pathMode = prevPathMode;
    }
    else if(pathMode == 1){
      pathMode = 1;
    }
    else if(pathMode == 4){
      pathMode = 5;
    }
    else if(pathMode == 7){
      pathMode = 8;
    }
    else if(pathMode == 10){
      pathMode = 12;
    }
    else if(pathMode == 11){
      pathMode = 12;
    }
    else if(pathMode == 12){
      pathMode = 13;
    }
    else{
      pathAsked = false;
      screenMode = SCREEN_MENU; // show menu
    }
  }

  //left
  if(keyCode == SOFTKEY1){
    if(pathAsking == true && pathAsked == false){
      pathMode = 0;
      pathAsked = true;
    }
    else if(pathMode == 1){
      screenMode = SCREEN_MENU; // show menu
    }
    else if(pathMode == 10){
      pathMode = 11;
```

```
  }
  else if(pathMode == 11){
    pathMode = 6;
  }
  else if(pathMode == 12){
    pathMode = 13;
  }
  else{
    pathMode--;
  }
}

//up
if(keyCode == UP){
  if(pathMode == 1){        // landmark1
    lmFocus--;
    if (lmFocus == 5){
      lmXPosition = -300;
    }
    if (lmFocus == 5){
      lmYPosition = 194;
    }
  }
  else if(pathMode == 2){
    lmFocus2--;
    if (lmFocus2 == 0){
      lmFocus2 = 6;
    }
  }
  else if(pathMode == 5){
    if (toolbar == false){
      toolbar = true;
      toolFocus = 1;
    }
  }
  else if(pathMode == 6){
    if (toolbar == false){
      toolbar = true;
      toolFocus = 1;
    }
  }
  else if(pathMode == 8){
    if (toolbar == false){
      toolbar = true;
      toolFocus = 1;
    }
  }
  else if(pathMode == 13){
    feedY = feedY - 80;
    if (feedY < height-310){
```

```
      feedY = height-150;
    }
  }
}

//down
if(keyCode == DOWN){
  if(pathMode == 1){        //landmark2
    lmFocus++;
    if (lmFocus == 6){
      lmYPosition = 600;
      lmXPosition = 33;
    }
  }
  else if(pathMode == 2){
    lmFocus2++;
    if (lmFocus2 == 7){
      lmFocus2 = 1;
    }
  }
  else if(pathMode == 5){
    if (toolbar == true){
      toolbar = false;
    }
  }
  else if(pathMode == 6){
    if (toolbar == true){
      toolbar = false;
    }
  }
  else if(pathMode == 8){
    if (toolbar == true){
      toolbar = false;
    }
  }
  else if(pathMode == 13){
    feedY = feedY + 80;
    if (feedY > height-150){
      feedY = height-310;
    }
  }
}

//LEFT
if(keyCode == LEFT){
  if(pathMode == 1){        // landmark1
    lmFocus--;
    if (lmFocus == 5){
      lmXPosition = -300;
    }
```

```
  if (lmFocus == 5){
    lmYPosition = 194;
  }
}
else if(pathMode == 2){
  lmFocus2--;
  if (lmFocus2 == 0){
    lmFocus2 = 6;
  }
}
else if(pathMode == 3){
  lPressed = true;
  senseCode--;
  if (senseCode == 0){
    senseCode = 5;
  }
}
else if(pathMode == 5){
  if(toolbar == true){
    toolFocus--;
    if (toolFocus == 0){
      toolFocus = 2;
    }
  }
}
else if(pathMode == 6){
  if(toolbar == false){
    if(selectedBall > 0){
      balls[selectedBall].unselect();
      selectedBall--;
      balls[selectedBall].select();
    }
    else{
      balls[selectedBall].unselect();
      selectedBall = (balls.length - 1);
      balls[selectedBall].select();
    }
  }
  else{
    toolFocus--;
    if (toolFocus == 0){
      toolFocus = 2;
    }
  }
}
else if(pathMode == 8){
  if(toolbar == true){
    toolFocus--;
    if (toolFocus == 0){
      toolFocus = 2;
```

```
      }
    }
  }

}

//RIGHT
if(keyCode == RIGHT){
  if(pathMode == 1){        //landmark2
    lmFocus++;
    if (lmFocus == 6){
      lmYPosition = 600;
      lmXPosition = 33;
    }
  }
  else if(pathMode == 2){
    lmFocus2++;
    if (lmFocus2 == 7){
      lmFocus2 = 1;
    }
  }
  else if(pathMode == 3){
    rPressed = true;
    senseCode++;
    if (senseCode == 6){
      senseCode = 1;
    }
  }
  else if(pathMode == 5){
    if(toolbar == false){
      rPressed = true;
    }
    else{
      toolFocus--;
      if (toolFocus == 0){
        toolFocus = 2;
      }
    }
  }
  else if(pathMode == 6){
    if(toolbar == false){
      if(selectedBall < (balls.length - 1)){
        balls[selectedBall].unselect();
        selectedBall++;
        balls[selectedBall].select();
      }
      else{
        balls[selectedBall].unselect();
        selectedBall = 0;
        balls[selectedBall].select();
```

```
      }
    }
    else{
      toolFocus++;
      if (toolFocus == 3){
        toolFocus = 1;
      }
    }
  }
  else if(pathMode == 8){
    if(toolbar == false){
      animate++;
    }
    else{
      toolFocus--;
      if (toolFocus == 0){
        toolFocus = 2;
      }
    }
  }
}

//FIRE
if(keyCode == FIRE){
  if(pathMode == 1){        //landmark2
    if (lmFocus <= 5){
      pathMode = 2;
    }
    else{
      pathMode = 3;
    }
  }
  else if(pathMode == 2){
    pathMode = 3;
  }
  else if(pathMode == 3){
    pathMode = 4;
  }
  else if(pathMode == 4){
    pathMode = 5;
  }
  else if(pathMode == 5){
    if(toolbar == true){
      if (toolFocus == 2){
        if(toolCode == 1){
          toolCode = 2;
          toolbar = false;
        }
        else{
          toolCode = 1;
```

```
        toolbar = false;
       }
     }
     /* else{
      // upload sense
      }*/
   }
 }
 else if(pathMode == 6){
   if(toolbar == false){
     pathMode = 7;
   }
   else{
     if (toolFocus == 2){
       if(toolCode == 1){
         toolCode = 2;
         toolbar = false;
       }
       else{
         toolCode = 1;
         toolbar = false;
       }
     }
     /* else{
      // upload sense
      }*/
   }
 }
 else if(pathMode == 7){
   pathMode = 8;
 }
 else if(pathMode == 8){
   if(toolbar == true){
     if (toolFocus == 2){
       if(toolCode == 1){
         toolCode = 2;
         toolbar = false;
       }
       else{
         toolCode = 1;
         toolbar = false;
       }
     }
     /* else{
      // upload sense
      }*/
   }
 }
 else if(pathMode == 12){
   pathMode = 13;
```

```
    }

  }

}

// if showing screen random


if(screenMode == SCREEN_RANDOM){
  // if pressed left soft key
  if(keyCode == SOFTKEY2){
    screenMode = SCREEN_MENU; // show menu
  }
}


// if showing screen upload


if(screenMode == SCREEN_UPLOAD){

  // if pressed left soft key
  if(keyCode == SOFTKEY2){
    screenMode = SCREEN_MENU; // show menu
  }
  if(upMode == 2){        //sel senses
    upMode = 3;
    }

  if(keyCode == LEFT){

    if(upMode == 1){
      lPressed = true;
      senseCode--;
      if (senseCode == 0){
        senseCode = 5;
      }
    }
    else if(upMode == 2){
      if(upY == 181){
        upY = 52;
      }else{
      upY = 181;
      }
    }
  }

  if(keyCode == RIGHT){
```

```
  //upload screen
  if(upMode == 1){
    rPressed = true;
    senseCode++;
    if (senseCode == 6){
      senseCode = 1;
    }
  }
  else if(upMode == 2){
    if(upY == 181){
      upY = 52;
    }else{
    upY = 181;
    }
  }
}

if(keyCode == DOWN){
  if(upMode == 2){
    if(upY == 181){
      upY = 52;
    }else{
    upY = 181;
    }
  }
}

if(keyCode == UP){

  if(upMode == 2){
    if(upY == 181){
      upY = 52;
    }else{
    upY = 181;
    }
  }
}

 if(keyCode == FIRE){
  if(upMode == 1){        //sel senses
  upMode = 2;
  }
  else if(upMode == 4){
  upMode = 5;
  }
  else if(upMode ==5){        //sel senses
  upMode = 6;
  }
  else if(upMode == 6){        //sel senses
  screenMode = SCREEN_MENU;
```

```
    }

   }
  }

  // if showing screen maps


  if(screenMode == SCREEN_MAPS){
   // if pressed left soft key
   if(keyCode == SOFTKEY2){
    screenMode = SCREEN_MENU; // show menu
   }
  }


  // if showing screen mymaps


  if(screenMode == SCREEN_MYMAPS){
   // if pressed left soft key
   if(keyCode == SOFTKEY2){
    screenMode = SCREEN_MENU; // show menu
   }
  }


  // if showing screen help


  if(screenMode == SCREEN_HELP){
   // if pressed left soft key
   if(keyCode == SOFTKEY2){
    screenMode = SCREEN_MENU; // show menu
   }
  }
}


// KeyReleased - Executes whenever a key is released, captures user input (User
Input)
void keyReleased()
{

  if(keyCode == RIGHT){
   if(pathMode == 3 || pathMode == 5){
    rPressed = false;
    timeSet = false;
   }
   if(upMode == 1){
```

```
      rPressed = false;
    }
  }


  if(keyCode == LEFT){
    if(pathMode == 3){
      lPressed = false;
    }
    else if(upMode == 1){
      lPressed = false;
    }
  }

}
```