```
/*******************************
 * first SETUP everything needed
 *******************************/

import processing.phone.*;     // import phone library to be able to go fullscreen
Phone myLG;                    // my phone's name reference

int fps = 5;


void setup()
{
  framerate(fps);

  myLG = new Phone(this); // create new phone instance/controller
  myLG.fullscreen();      // go full screen

    loadImages();

}



/*******************************
 * CONSTANTS - mode, focus, options
 *******************************/

int MODE_SPLASH = 0;
int MODE_CLOUDS_IN = 1;
int MODE_HAIKU = 2;
int MODE_CLOUDS_OUT = 3;
int MODE_MENU = 4;
int MODE_WRITEHAIKU = 5;
int MODE_SAVEHAIKU = 6;
int MODE_READHAIKU = 7;
int MODE_HAIKU_1 = 21;
int MODE_HAIKU_2 = 22;
int MODE_HAIKU_3 = 23;

int MENUFOCUS_WRITE = 0;
int MENUFOCUS_SAVE = 1;
int MENUFOCUS_READ = 2;

int READMENUFOCUS_1 = 11;
int READMENUFOCUS_2 = 12;
int READMENUFOCUS_3 = 13;
int READMENUFOCUS_4 = 14;
int READMENUFOCUS_5 = 15;
```

```
/******************************
 * VARIABLES
 *******************************/

int mode = MODE_SPLASH;            // splash screen at the start, for a few seconds

int menuFocus = MENUFOCUS_WRITE;    // menu screen, initial focus on write option

int readFocus = READMENUFOCUS_1; // read menu screen, initial focus on first haiku in the list

int splashTimeout = fps * 2;        // time duration of the splash screen
int cloudsInTimeout = fps * 3;
int haikuTimeout = fps * 2;
int cloudsOutTimeout = fps * 3;



/******************************
 * define GRAPHICS and load them
 *******************************/

PImage splashScreen;

PImage cloudsIn;
PImage cloudsOut;
PImage clearSky;

PImage haikuLine_1;
PImage haikuLine_2;
PImage haikuLine_3;

PImage menuBackground;
PImage menuitemWrite;
PImage menuitemSave;
PImage menuitemRead;

PImage writeHaiku;
PImage saveHaiku;
PImage readHaikuBackground;
PImage readHaiku_1;
PImage readHaiku_2;
PImage readHaiku_3;
PImage readHaiku_4;
PImage readHaiku_5;


void loadImages(){
  splashScreen = loadImage("splashScreen.png");
  cloudsIn = loadImage("cloudsIn.png");
  cloudsOut = loadImage("cloudsOut.png");
  clearSky = loadImage("clearSky.png");

  menuBackground = loadImage("menuBackground.png");
  menuitemWrite = loadImage("menuitemWrite.png");
  menuitemSave = loadImage("menuitemSave.png");
  menuitemRead = loadImage("menuitemRead.png");

  haikuLine_1 = loadImage("haikuLine_1.png");
  haikuLine_2 = loadImage("haikuLine_2.png");
  haikuLine_3 = loadImage("haikuLine_3.png");


  writeHaiku = loadImage("writeHaiku.png");
  saveHaiku = loadImage("saveHaiku.png");
  readHaikuBackground = loadImage("readHaikuBackground.png");
  readHaiku_1 = loadImage("readHaiku_1.png");
  readHaiku_2 = loadImage("readHaiku_2.png");
  readHaiku_3 = loadImage("readHaiku_3.png");
  readHaiku_4 = loadImage("readHaiku_4.png");
  readHaiku_5 = loadImage("readHaiku_5.png");

}



/*******************************
 * SCREEN drawing functions
 *******************************/

void drawSplash(){
  image(splashScreen,0,0);
}

void drawCloudsIn(){
  image(cloudsIn,0,0);
}

void drawHaikuLine_1(){
  image(clearSky,0,0);
  image(haikuLine_1,0,0);
}

void drawHaikuLine_2(){
  image(clearSky,0,0);
  image(haikuLine_2,0,0);
}
```

```
void drawHaikuLine_3(){
  image(clearSky,0,0);
  image(haikuLine_3,0,0);
}

void drawCloudsOut(){
  image(cloudsOut,0,0);
}

void drawMenu(int focus){
  if(focus == MENUFOCUS_WRITE){
    image(menuBackground,0,0);
    image(menuitemWrite,0,0);
  }
  else if(focus == MENUFOCUS_SAVE){
    image(menuBackground,0,0);
    image(menuitemSave,0,0);
  }
  else if(focus == MENUFOCUS_READ){
    image(menuBackground,0,0);
    image(menuitemRead,0,0);
  }
}

void drawWriteHaiku(){
  image(writeHaiku,0,0);
}

void drawSaveHaiku(){
  image(saveHaiku,0,0);
}

void drawReadHaiku(int readFocus){
  if(readFocus == READMENUFOCUS_1){
    image(readHaikuBackground,0,0);
    image(readHaiku_1,0,0);
  }
  else if(readFocus == READMENUFOCUS_2){
    image(readHaikuBackground,0,0);
    image(readHaiku_2,0,0);
  }
  else if(readFocus == READMENUFOCUS_3){
    image(readHaikuBackground,0,0);
    image(readHaiku_3,0,0);
  }
  else if(readFocus == READMENUFOCUS_4){
    image(readHaikuBackground,0,0);
    image(readHaiku_4,0,0);
```

```
 }
 else if(readFocus == READMENUFOCUS_5){
   image(readHaikuBackground,0,0);
   image(readHaiku_5,0,0);
 }

}
/*******************************
 * DRAWING function
 *******************************/


void draw(){                    // happens repeatedly (according to framerate)...


 if(mode == MODE_SPLASH){

   if(0 < splashTimeout){          // draws splash screen first
     splashTimeout--;
     drawSplash();
   }
   else{
     mode = MODE_CLOUDS_IN;
   }
 }


 else if(mode == MODE_CLOUDS_IN){

   if(0 < cloudsInTimeout){          // draws clouds-in screen
     cloudsInTimeout--;
     drawCloudsIn();

   }
   else{
     mode = MODE_HAIKU_1;
     cloudsInTimeout = fps * 3;
   }
 }


 else if(mode == MODE_HAIKU_1){

   if(0 < haikuTimeout){          // draws haiku screen 1
     drawHaikuLine_1();
     haikuTimeout--;

   }
 else{
  mode = MODE_HAIKU_2;
  haikuTimeout = fps * 2;
 }
}


else if(mode == MODE_HAIKU_2){

 if(0 < haikuTimeout){          // draws haiku screen 2
  drawHaikuLine_2();
  haikuTimeout--;

 }
 else{
  mode = MODE_HAIKU_3;
  haikuTimeout = fps * 2;
 }
}


else if(mode == MODE_HAIKU_3){

 if(0 < haikuTimeout){          // draws haiku screen 3
  drawHaikuLine_3();
  haikuTimeout--;

 }
 else{
  mode = MODE_CLOUDS_OUT;
  haikuTimeout = fps * 2;
 }
}


else if(mode == MODE_CLOUDS_OUT){

 if(0 < cloudsOutTimeout){          // draws clouds-out screen
  cloudsOutTimeout--;
  drawCloudsOut();

 }
 else{
  mode = MODE_CLOUDS_IN;
  cloudsOutTimeout = fps * 3;
 }
}
```

```
 else if(mode == MODE_MENU){
   drawMenu(menuFocus);            // then goes to main menu
 }


 else if(mode == MODE_WRITEHAIKU){
   drawWriteHaiku();             // draws write-haiku screen
 }


 else if(mode == MODE_SAVEHAIKU){
   drawSaveHaiku();             // draws save-haiku screen
 }

 else if(mode == MODE_READHAIKU){
   drawReadHaiku(readFocus);      // draws read-haiku screen
 }

} // close of draw



/*******************************
 * USER INPUT from keypad
 *******************************/

void keyPressed(){

 if((mode == MODE_CLOUDS_OUT)||(mode == MODE_CLOUDS_IN)){

   mode = MODE_MENU;
 }


 if(mode == MODE_MENU){

   if(keyCode == SOFTKEY1){
     mode = MODE_CLOUDS_IN;
   }

   if(menuFocus == MENUFOCUS_WRITE){
     if(keyCode == SOFTKEY2){
       mode = MODE_WRITEHAIKU;
     }
     else if(keyCode == RIGHT){
       menuFocus = MENUFOCUS_SAVE;
     }
```

```
    else if(keyCode == LEFT){
      menuFocus = MENUFOCUS_READ;
    }
  }

  else if(menuFocus == MENUFOCUS_SAVE){
    if(keyCode == SOFTKEY2){
      mode = MODE_SAVEHAIKU;
    }
    else if(keyCode == RIGHT){
      menuFocus = MENUFOCUS_READ;
    }
    else if(keyCode == LEFT){
      menuFocus = MENUFOCUS_WRITE;
    }
  }

  else if(menuFocus == MENUFOCUS_READ){
    if(keyCode == SOFTKEY2){
      mode = MODE_READHAIKU;
    }
    else if(keyCode == RIGHT){
      menuFocus = MENUFOCUS_WRITE;
    }
    else if(keyCode == LEFT){
      menuFocus = MENUFOCUS_SAVE;
    }
  }

}


if((mode == MODE_WRITEHAIKU)||(mode == MODE_SAVEHAIKU)){

  if(keyCode == SOFTKEY1){
    mode = MODE_MENU;
  }
}


if(mode == MODE_READHAIKU){

  if(keyCode == SOFTKEY1){
    mode = MODE_MENU;
  }

  if(readFocus == READMENUFOCUS_1){
    if(keyCode == UP){
        readFocus = READMENUFOCUS_5;
      }

      else if(keyCode == DOWN){
        readFocus = READMENUFOCUS_2;
      }
    }

    else if(readFocus == READMENUFOCUS_2){
      if(keyCode == UP){
        readFocus = READMENUFOCUS_1;
      }
      else if(keyCode == DOWN){
        readFocus = READMENUFOCUS_3;
      }
    }

    else if(readFocus == READMENUFOCUS_3){
      if(keyCode == UP){
        readFocus = READMENUFOCUS_2;
      }
      else if(keyCode == DOWN){
        readFocus = READMENUFOCUS_4;
      }
    }

    else if(readFocus == READMENUFOCUS_4){
      if(keyCode == UP){
        readFocus = READMENUFOCUS_3;
      }
      else if(keyCode == DOWN){
        readFocus = READMENUFOCUS_5;
      }
    }

    else if(readFocus == READMENUFOCUS_5){
      if(keyCode == UP){
        readFocus = READMENUFOCUS_4;
      }
      else if(keyCode == DOWN){
        readFocus = READMENUFOCUS_1;
      }
    }
  }
}
```

```
void keyReleased(){
  if(mode == MODE_SAVEHAIKU){
    mode = MODE_MENU;
    menuFocus = MENUFOCUS_SAVE;
  }
}

// This builds on the Mobile Processing code developed for the IUAV Interaction Design
// Programme by David Mellis, Vinay Ventrakamen and Nicholas Zambetti 2005-07. See
// www.interaction-venice.com/resources/?page_id=5.

// In our prototype we inspired from this code written by Nicholas:

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

// named references to graphics
PImage menuGioca;
PImage menuAgenda;
PImage menuProfilo;
PImage menuClassifica;
PImage giocaSestiereCannaregio;

// function to load all the images for the interface
void loadImages()
{
  // menu graphics
  menuGioca = loadImage("menuGioca.png");
  menuAgenda = loadImage("menuAgenda.png");
  menuProfilo = loadImage("menuProfilo.png");
  menuClassifica = loadImage("menuClassifica.png");

  // gioca sestiere graphics
  giocaSestiereCannaregio = loadImage("giocaSestiereCannaregio.png");
}

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

// function to draw the menu with specified focus
void drawMenu(int focus)
{
  if(focus == MENUFOCUS_GIOCA){
    image(menuGioca, 0, 0);
  }
  else if(focus == MENUFOCUS_AGENDA){
    image(menuAgenda, 0, 0);
  }
  else if(focus == MENUFOCUS_PROFILO){
    image(menuProfilo, 0, 0);
  }
  else if(focus == MENUFOCUS_CLASSIFICA){
    image(menuClassifica, 0, 0);
  }
}

// function to draw the Sestiere selector in the Gioca section
void drawGiocaSestiere()
{
  image(giocaSestiereCannaregio, 0, 0);
}

/**********************************************************************************************
 * 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 mode
// tip: one mode for each screen is easy to program for a demo but less easy to make into a real
// application
//   one mode per logical group/flow of screens (e.g. registration, menu) is difficult to program
// but easier to make real
int MODE_MENU = 0;
int MODE_GIOCASESTIERE = 1;

// 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 mode = MODE_MENU; // initially, we are showing the main menu

// 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)...
{
 if(mode == MODE_MENU){ // if we are in "Menu" mode...
  drawMenu(menuFocus); // draw image for menu with a variable focus
 }
 else if(mode == MODE_GIOCASESTIERE){ // if we are in "Sestiere" mode...
  drawGiocaSestiere();          // draw image for the sestiere selector
 }
}

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

void keyReleased() // whenever a key is pressed...
{
 if(mode == MODE_MENU) // if we are at the main menu...
 {
  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(keyCode == RIGHT){          // if the user pressed right...
        menuFocus = MENUFOCUS_CLASSIFICA; // put the menu focus on "Classifica"
      }
      else if(keyCode == FIRE){    // if the user pressed fire/center...
        mode = MODE_GIOCASESTIERE; // put the menu focus on "Classifica"
      }
    }
    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"
      }
      else if(keyCode == RIGHT){       // if the user pressed right...
        menuFocus = MENUFOCUS_PROFILO; // put the menu focus on "Profilo"
      }
    }
  }
  else if(mode == MODE_GIOCASESTIERE) // if we are at the sestiere selector...
  {
    if(keyCode == '1'){  // if the user pressed "Indietro"...
      mode = MODE_MENU;  // bring the user back to the main menu
    }
  }
}

*/

// and he help us even to create the timer :

/*

  if(mode == MODE_SPLASH){

    if(0 < splashTimeout){           // draws splash screen first
      splashTimeout--;
      drawSplash();
    }
    else{
      mode = MODE_CLOUDS_IN;
    }
  }

*/
```