

```
/*
  Kyo
  11 December 2009
  created by Carla Piazza based on the code fragments prepared for
  the interaction design studio courses in the IUAV Visual and Multimedia
  Communication graduate programme by David Mellis, Vinay Ventrakamen
  and Nicholas Zambetti 2005-09
  see: www.interaction-venice.com/tools/programming/mobile-processing/
```

Special thanks to Nicholas Zambetti for his precious help programming Kyo application.

```
IUAV IxD Lab1 2009/2010
by Gillian Crampton Smith and Philip Tabor
*/
```

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

```
////////////////////////////////////
// Animation Functions, Provide next value given the current value, target value, and speed
////////////////////////////////////
```

```
// linear animation function (relative)
// e.g. easeIn(x, 100, 2);
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)
}

// eased in animation function (relative), somewhat hack but ok for simple animation
// e.g. easeIn(x, 100, 900);
int easeIn(int current, int target, int speed)
{
    int change = target - current; // record how far we are away
    int next = current;           // initially our next step is where we currently are
    if(target < next){           // if we need to make our value smaller...
        change = target - change; // flip how far we are away
        speed += change;          // divide by zeroes are bad
        speed /= change;          // divide
        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...
        speed += change;          // divide by zeroes are bad
        speed /= change;          // divide
        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
}

```

```

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

```

```

//Font
PFont font;

```

```

//Named references to images
PImage menu_start;
PImage menu_arrows;
PImage menu_selections;
PImage menu_options;
PImage menu_comment;
PImage screen_present;
PImage screen_past;
PImage screen_logout;
PImage screen_options;
PImage screen_unfold;
PImage screen_send_1;
PImage step_send_contacts;
PImage contact_selection;
PImage step_send_confirm;
PImage step_send_sending;
PImage step_send_sent;
PImage sending_text;
PImage screen_save;
PImage screen_homo_saving;
PImage screen_saved;
PImage memory1;
PImage memory2;

```

```
PImage memory3;  
PImage memory4;  
PImage memory1_pict;  
PImage memory2_video;  
PImage memory3_text;  
PImage memory4_audio;  
PImage unfolding_1;  
PImage origami_saving;
```

```
int whichOrigami = 0;
```

```
PImage [] origami = new PImage [100]; //Image array  
int numFrames = 31; // number of animation frames  
int frame = 0; // the frame to display  
int wait = 0;
```

```
PImage [] sending = new PImage [19]; //Image array  
int numFramesSending = 19; // the number of animation frames  
int frameSendingLoop = 3; // the frame to display  
int frameSending = 0;
```

```
int menu_selections_x = 0;  
int menu_selections_y = -15;  
int menu_options_x = 0; //current x position  
int menu_options_x_target = 0; //final x position (the x value/position that it has to reach at the  
end of the movement)  
int menu_options_y = -15;  
int menu_comment_x=0;  
int menu_comment_y = -15;
```

```
int origami_saving_xTarget = -25; //final x position  
int origami_saving_yTarget = 20; //final y position  
int origami_saving_x = -25; //starting x point position  
int origami_saving_y = 20; //starting y point position
```

```
int word_HOW = 0;  
int words_HOWTotal = 6; //total number of words per category HOW  
int word_WHERE = 0;  
int words_WHERETotal = 13; //total number of words per category WHERE  
int word_DO = 0;  
int words_DOTotal = 5; //total number of words per category DO  
int word_WHAT = 0;  
int words_WHATTotal = 19; //total number of words per category WHAT  
int word_WHEN = 0;  
int words_WHENTotal = 15; //total number of words per category WHEN
```

```
//Set of words in which kyo selects randomly one among (per category)
```

```
String[] words_HOW = {  
  "slowly|", "without shoes|", "on your own|", "blind fold|","smiling|", "with a friend|"};
```

```
String[] words_WHERE = {  
  "|Piazza San Marco|", "|street corner|", "|wherever you want|", "|Zattere|","|bench|","|boat|",  
  "|house|","|Ponte dei Sospiri|", "|not far from here|", "|under an arch|", "|Accademia|","|altana|","|  
Rialto|"};
```

```
String[] words_DO = {  
  "|smell|", "|taste|", "|touch|", "|listen|","|see|"};
```

```
String[] words_WHAT = {
```

```
"|clouds|", "|rain|", "|moon|", "|cinnamon|","|sun|","|wind|","|basil|","|air|","|chocolate|","|stone|",  
"|sponge|","|jam|","|poem|","|honey|","|waves|","|laughter|", "|breath|","|spritz|","|bubble|";
```

```
String[] words_WHEN = {  
    "|whenever you want", "|today", "|full moon night", "|sunset","|windy day","|tomorrow","|sunrise",  
    "|starry sky",  
    "|cloudy day", "|sad day", "|happy day", "|rainy day", "|sunny day", "|now", "|in half an hour"};
```

```
////////////////////////////////////  
//Function to unload all the images (to let the programm run faster)  
////////////////////////////////////
```

```
void unloadAll(){ // put all these in garbage  
    unloadStart();  
    unloadMenu();  
    unloadPresent();  
    unloadOrigami();  
    unloadPast();  
    unloadLogout();  
    unloadMemories();  
    unloadOptions();  
    unloadComment();  
    unloadSend();  
    unloadContacts();  
    unloadSaved();  
    unloadSending();
```

```
    (Runtime.getRuntime()).gc(); // call garbage collector  
}
```

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

```
//Function to load the images for the Log-On menu
```

```
void loadStart()  
{  
    menu_start = loadImage ("menu_start.png");  
}
```

```
// Function to unload the Log-On menu
```

```
void unloadStart()  
{  
    menu_start = null;  
    (Runtime.getRuntime()).gc(); //call garbage collector  
}
```

```
void enterStart()
```

```
{  
    unloadAll();  
    loadStart();  
    screenMode = SCREEN_START;  
}
```

```
// Function to draw the Log-On menu
```

```
void drawStart()  
{  
    background(176,167,144);  
    image (menu_start, 0, -15);  
}
```

```
// Function to load all the images for the main Menu
```

```

void loadMenu()
{
    menu_selections = loadImage("menu_selections.png");
    menu_arrows = loadImage("menu_arrows.png");
}

// Function to unload the main Menu
void unloadMenu()
{
    menu_selections = null;
    menu_arrows = null;
    (Runtime.getRuntime()).gc(); //call garbage collector
}

void enterMenu(){
    unloadAll();
    loadMenu();
    screenMode= SCREEN_MENU;
}

// Function to draw the main Menu
void drawMenu()
{
    background(176,167,144);
    image (menu_selections, menu_selections_x, menu_selections_y);
    // 151x13

    if(menuFocus == MENU_OPTION_PAST){
        menu_selections_x = easeOut(menu_selections_x, 0, 10);
        image (menu_arrows, 151-8, 0, 8, 13, 180, 215); // right arrow
    }

    if(menuFocus == MENU_OPTION_PRESENT){
        menu_selections_x = easeOut(menu_selections_x, -230, 10);
        image (menu_arrows, 0, 0, 8, 13, 47, 215); // left arrow
        image (menu_arrows, 151-8, 0, 8, 13, 180, 215); // right arrow
    }

    if(menuFocus == MENU_OPTION_LOGOUT){
        menu_selections_x = easeOut(menu_selections_x, -480, 10);
        image (menu_arrows, 0, 0, 8, 13, 47, 215); // left arrow
    }
}

//////////FOCUS PRESENT//////////////////////////////////////
// Function to load all the images for the Present
void loadPresent(){
    screen_unfold = loadImage ("unfold.png");
    font = loadFont("CourierNewPSMT-17.mv/w");
    textFont(font);
    textAlign(CENTER);
}

// Function to unload the Present
void unloadPresent(){
    screen_unfold = null; // put this in garbage
    font = null;// and this too
    (Runtime.getRuntime()).gc(); // call garbage collector
}

```

```

void enterPresent(){
  unloadAll();
  loadPresent();
  screenMode = SCREEN_PRESENT;
  whichOrigami = random(0, 2);
  unloadOrigami();
  if(whichOrigami == 0){
    loadOrigamiA();
  }
  else{
    loadOrigamiB();
  }
}

```

```

void drawPresent(){
  background(176,167,144);
  image(screen_unfold, 0, 0);
  image (screen_present,0,0);
  //How to draw Origami
}

```

// Function to load all the origamiA

```

void loadOrigamiA(){
  screen_present = loadImage("1a.png");
  screen_options= loadImage("screen_options.png");
  numFrames = 31; //number of animation frames
  origami [0] = loadImage ("1a.png");
  origami [1] = loadImage ("2a.png");
  origami [2] = loadImage ("3a.png");
  origami [3] = loadImage ("4a.png");
  origami [4] = loadImage ("5a.png");
  origami [5] = loadImage ("6a.png");
  origami [6] = loadImage ("7a.png");
  origami [7] = loadImage ("8a.png");
  origami [8] = loadImage ("9a.png");
  origami [9] = loadImage ("10a.png");
  origami [10] = loadImage ("11a.png");
  origami [11] = loadImage ("12a.png");
  origami [12] = loadImage ("13a.png");
  origami [13] = loadImage ("14a.png");
  origami [14] = loadImage ("15a.png");
  origami [15] = loadImage ("16a.png");
  origami [16] = loadImage ("17a.png");
  origami [17] = loadImage ("18a.png");
  origami [18] = loadImage ("19a.png");
  origami [19] = loadImage ("20a.png");
  origami [20] = loadImage ("21a.png");
  origami [21] = loadImage ("22a.png");
  origami [22] = loadImage ("23a.png");
  origami [23] = loadImage ("24a.png");
  origami [24] = loadImage ("25a.png");
  origami [25] = loadImage ("26a.png");
  origami [26] = loadImage ("27a.png");
  origami [27] = loadImage ("28a.png");
  origami [28] = loadImage ("29a.png");
  origami [29] = loadImage ("30a.png");
  origami [30] = loadImage ("31a.png");
}

```

// Function to load all the origamiB

```

void loadOrigamiB(){

```

```

screen_present = loadImage("1b.png");
screen_options= loadImage("screen_options.png");
numFrames = 31; //number of animation frames
origami [0] = loadImage ("1b.png");
origami [1] = loadImage ("2b.png");
origami [2] = loadImage ("3b.png");
origami [3] = loadImage ("4b.png");
origami [4] = loadImage ("5b.png");
origami [5] = loadImage ("6b.png");
origami [6] = loadImage ("7b.png");
origami [7] = loadImage ("8b.png");
origami [8] = loadImage ("9b.png");
origami [9] = loadImage ("10b.png");
origami [10] = loadImage ("11b.png");
origami [11] = loadImage ("12b.png");
origami [12] = loadImage ("13b.png");
origami [13] = loadImage ("14b.png");
origami [14] = loadImage ("15b.png");
origami [15] = loadImage ("16b.png");
origami [16] = loadImage ("17b.png");
origami [17] = loadImage ("18b.png");
origami [18] = loadImage ("19b.png");
origami [19] = loadImage ("20b.png");
origami [20] = loadImage ("21b.png");
origami [21] = loadImage ("22b.png");
origami [22] = loadImage ("23b.png");
origami [23] = loadImage ("24b.png");
origami [24] = loadImage ("25b.png");
origami [25] = loadImage ("26b.png");
origami [26] = loadImage ("27b.png");
origami [27] = loadImage ("28b.png");
origami [28] = loadImage ("29b.png");
origami [29] = loadImage ("30b.png");
origami [30] = loadImage ("31b.png");
}

// Function to unload all the origami A-B
void unloadOrigami(){
  // loop through images and put them in garbage (null)
  int i = 0;
  while(i < 100){
    origami[i] = null;
    i = i + 1;
  }
  screen_options = null; // oh, put this in garbage too
  screen_present = null;
  (Runtime.getRuntime()).gc();// call garbage collector
}

void enterOrigami(){
  screenMode = SCREEN_ORIGAMI;
  frame = 0;
}

// Function to draw all the origami A-B
void drawOrigami(){
  if(whichOrigami == 0){
    drawOrigamiA();
  }
  else{
    drawOrigamiB();
  }
}

```

```

}
}

void drawOrigamiA(){

  if(0 < wait){
    wait = wait - 1;

  }
  else if (frame < numFrames){
    framerate (10);
    background(176,167,144);
    image (origami [frame],0,0);
    frame = frame + 1;

    if(frame == 7){
      wait = 5; //if frame value is equal to 7, please wait and..
      word_HOW = random(words_HOWTotal-1); //..draw a word choosing, randomly, among
"words_HOWTotal" category
      text(words_HOW[word_HOW], width/2, 250);
    }
    if(frame == 13){
      wait = 5; //if frame value is equal to 13, please wait and..
      word_WHERE = random(words_WHERETotal-1); //..draw a word choosing, randomly, among
"words_WHERETotal" category
      text(words_WHERE[word_WHERE], width/2, 250);
    }
    if(frame == 19){
      wait = 5; //if frame value is equal to 19, please wait and..
      word_DO = random(words_DOTotal-1); //..draw a word choosing, randomly, among "words_
DOTotal" category
      text(words_DO[word_DO], width/2, 250);
    }
    if(frame == 25){
      wait = 5; //if frame value is equal to 25, please wait and..
      word_WHAT = random(words_WHATTotal-1); //..draw a word choosing, randomly, among
"words_WHATTotal" category
      text(words_WHAT[word_WHAT], width/2, 250);
    }
    if(frame == 31){
      wait = 15; //if frame value is equal to 31, please wait and..
      word_WHEN = random(words_WHENTotal-1); //..draw a word choosing, randomly, among
"words_WHENTotal" category
      text(words_WHEN[word_WHEN], width/2, 250);
      image (screen_options,0,0);
    }
  }
  else{
    framerate (40); //number of frames per second
  }
}

```

```

void drawOrigamiB(){

  if(0 < wait){
    wait = wait - 1;

  }
  else if (frame < numFrames){
    framerate (10); //number of frames per second
    background(176,167,144);
  }
}

```

```

image (origami [frame],0,0);
frame = frame + 1;

if(frame == 7){
    wait = 5;//if frame value is equal to 7, please wait and..
    word_HOW = random(words_HOWTotal-1); //..draw a word choosing, randomly, among
"words_HOWTotal" category
    text(words_HOW[word_HOW], width/2, 250);
}
if(frame == 13){
    wait = 5; //if frame value is equal to 13, please wait and..
    word_WHERE = random(words_WHERETotal-1); //..draw a word choosing, randomly, among
"words_WHERETotal" category
    text(words_WHERE[word_WHERE], width/2, 250);
}
if(frame == 19){
    wait = 5; //if frame value is equal to 19, please wait and..
    word_DO = random(words_DOTotal-1);//..draw a word choosing, randomly, among "words_
DOTotal" category
    text(words_DO[word_DO], width/2, 250);
}
if(frame == 25){
    wait = 5; //if frame value is equal to 25, please wait and..
    word_WHAT = random(words_WHATTotal-1); //..draw a word choosing, randomly, among
"words_WHATTotal" category
    text(words_WHAT[word_WHAT], width/2, 250);
}
if(frame == 31){
    wait = 15; //if frame value is equal to 31, please wait and..
    word_WHEN = random(words_WHENTotal-1);//..draw a word choosing, randomly, among
"words_WHENTotal" category
    text(words_WHEN[word_WHEN], width/2, 250);
    image (screen_options,0,0);
}
}
else{
    framerate (40); //number of frames per second
}
}

```

```

//////////focusOptions
//Function to load all the images for the Options menu
void loadOptions(){
    menu_options = loadImage ("menu_options.png");
    menu_arrows = loadImage("menu_arrows.png");
}

```

```

void enterOptions(){
    unloadAll();
    loadOptions();
    screenMode = SCREEN_OPTIONS;
    menu_options_x = menu_options_x_target;
}

```

```

//Function to unload the Options menu
void unloadOptions(){
    menu_options = null;
    menu_arrows = null;
    // call garbage collector
}

```

```

(Runtime.getRuntime()).gc();
}

//Function to draw the Options menu
void drawOptions(){
  background(176,167,144);
  image (menu_options, menu_options_x, menu_options_y);
  // 151x13

  if(optionsFocus == MENU_OPTION_COMMENT){
    menu_options_x_target = 0;
    image (menu_arrows, 151-8, 0, 8, 13, 180, 215); // right arrow
  }
  if(optionsFocus == MENU_OPTION_SEND){
    menu_options_x_target = -230;
    image (menu_arrows, 0, 0, 8, 13, 47, 215); // left arrow
    image (menu_arrows, 151-8, 0, 8, 13, 180, 215); // right arrow
  }
  if(optionsFocus == MENU_OPTION_SAVE){
    menu_options_x_target = -480;
    image (menu_arrows, 0, 0, 8, 13, 47, 215); // left arrow
  }
  menu_options_x = easeOut(menu_options_x, menu_options_x_target, 10);
  framerate (40);
}

////////////////////COMMENT
//Function to load all the images of the Comment menu
void loadComment(){
  menu_arrows = loadImage("menu_arrows.png");
  menu_comment=loadImage("menu_comment.png");
}
void enterComment(){
  screenMode= SCREEN_COMMENT;
  unloadAll();
  loadComment();
}

//Function to unload the Comment menu
void unloadComment(){
  menu_arrows = null; //put these in garbage
  menu_comment= null;
  (Runtime.getRuntime()).gc(); //call the garbage collector
}

//Function to draw the Comment menu
void drawComment(){
  background(176,167,144);
  image (menu_comment, menu_comment_x, menu_comment_y);
  // 151x13

  if(focusComment == OPTION_TEXT){
    menu_comment_x = easeOut(menu_comment_x, 0, 10);
    image (menu_arrows, 151-8, 0, 8, 13, 180, 215); // right arrow
  }
  if(focusComment == OPTION_PICT){
    menu_comment_x = easeOut(menu_comment_x, -230, 10);
  }
}

```

```

    image (menu_arrows, 0, 0, 8, 13, 47, 215); // left arrow
    image (menu_arrows, 151-8, 0, 8, 13, 180, 215); // right arrow

}
if(focusComment == OPTION_AUDIO){
    menu_comment_x = easeOut(menu_comment_x, -480, 10);
    image (menu_arrows, 0, 0, 8, 13, 47, 215); // left arrow
    image (menu_arrows, 151-8, 0, 8, 13, 180, 215); // right arrow
}
if(focusComment == OPTION_VIDEO){
    menu_comment_x = easeOut(menu_comment_x, -710, 10);
    image (menu_arrows, 0, 0, 8, 13, 47, 215); // left arrow
}
framerate (40);
}

////////////////////SEND
//Function to load all the images of the Send screen
void loadSend(){
    screen_send_1 = loadImage ("screen_send_1.png");

}

void enterSend(){
    screenMode = SCREEN_SEND;
    unloadAll();
    loadSend();
}

//Function to unload the Send screen
void unloadSend(){
    screen_send_1 = null; //put this in garbage
    (Runtime.getRuntime()).gc(); //call the garbage collector
}

//Function to draw the Send screen
void drawSend(){
    background(176,167,144);
    image (screen_send_1,0,0);
}
////////////////////focus Contacts////////////////////////////////////
//Function to load all the images for the Contacts screen
void loadContacts(){
    step_send_contacts = loadImage ("contacts.png");
    contact_selection = loadImage ("contact_selection.png");
}

void enterContacts(){
    screenMode = SCREEN_CONTACTS;
    optionsContacts = OPTION_CONTACTS_1;
    unloadAll();
    loadContacts();
}

//Function to unload the Contact screen
void unloadContacts(){
    step_send_contacts = null;
    contact_selection = null;
    (Runtime.getRuntime()).gc();
}

```

```

//Function to draw all the elements of the Contacts screen
void drawContacts(){
  background(176,167,144);
  image (step_send_contacts,0,0);

  if (optionsContacts == OPTION_CONTACTS_1){
    image (contact_selection,25,110);
  }
  else if (optionsContacts == OPTION_CONTACTS_2){
    image (contact_selection,25,134);
  }
  else if (optionsContacts == OPTION_CONTACTS_3){
    image (contact_selection,25,158);
  }
  else if (optionsContacts == OPTION_CONTACTS_4){
    image (contact_selection,25,181);
  }
  else if (optionsContacts == OPTION_CONTACTS_5){
    image (contact_selection,25,204);
  }
  else if (optionsContacts == OPTION_CONTACTS_6){
    image (contact_selection,25,229);
  }
}
/////////////////////////////////////////focusCONFIRM/////////////////////////////////////////
//Function to load the images of the Confirm screen
void loadConfirm(){
  step_send_confirm = loadImage ("confirm.png");
}

void enterConfirm(){
  screenMode = SCREEN_CONFIRM;
  unloadAll();
  loadConfirm();
}

//Function to unload the Confirm screen
void unloadConfirm(){
  step_send_confirm = null; //put it in the garbage
  (Runtime.getRuntime()).gc(); //call the garbage collector!
}

//Function to draw the Confirm screen
void drawConfirm(){
  background(176,167,144);
  image (step_send_confirm,0,0);
}

/////////////////////////////////////////focusSENDING/////////////////////////////////////////
//Function to load all the images for the Sending screen
void loadSending(){

  sending [0] = loadImage ("0s.png");
  sending [1] = loadImage ("1s.png");
  sending [2] = loadImage ("2s.png");
  sending [3] = loadImage ("3s.png");
  sending [4] = loadImage ("4s.png");
  sending [5] = loadImage ("5s.png");
  sending [6] = loadImage ("6s.png");
  sending [7] = loadImage ("7s.png");
  sending [8] = loadImage ("8s.png");
}

```

```

sending [9] = loadImage ("9s.png");
sending [10] = loadImage ("10s.png");
sending [11] = loadImage ("11s.png");
sending [12] = loadImage ("12s.png");
sending [13] = loadImage ("13s.png");
sending [14] = loadImage ("14s.png");
sending [15] = loadImage ("15s.png");
sending [16] = loadImage ("16s.png");
sending [17] = loadImage ("17s.png");
sending [18] = loadImage ("18s.png");
step_send_sent = loadImage ("sent.png");
sending_text = loadImage ("sending_text.png");
}

```

```

void enterSending(){
  screenMode = SCREEN_SENDING;
  frameSending = 0;
  frameSendingLoop = 3;
  unloadAll();
  loadSending();
}

```

```

//Function to unload all the Sending images
void unloadSending(){
  int i = 0;
  while(i < 19){
    sending[i] = null; //put this in garbage
    i = i + 1;
  }
  step_send_sent = null; //and these as well
  sending_text = null;
  (Runtime.getRuntime()).gc(); //call the garbage collector
}

```

```

//Function to draw the Sending screen
void drawSending(){
  if(0 < frameSendingLoop){
    if (frameSending < numFramesSending){ //if the value of the frame is minor than the value of
the total numbers of frames then..
      framerate (20); //frames per second
      background(176,167,144);
      image (sending [frameSending],0,0); //then...draw this
      frameSending = frameSending + 1;
      image (sending_text,0,0);
    }
    else{
      frameSending = 0;
      frameSendingLoop = frameSendingLoop - 1;
    }
  }
  if(0 == frameSendingLoop){
    if (frameSending == 0){
      background(176,167,144);
      image (step_send_sent,0,0);
      framerate (40);
    }
  }
}
}
}
}

```

//////////////////////////////////////SAVE

```

//Function to load the Save screen
void loadSave(){
  screen_save = loadImage ("screen_save_question.png");
}

void enterSave(){
  screenMode = SCREEN_SAVE;
  unloadAll();
  loadSave();
}

//Function to unload Save images
void unloadSave(){
  screen_save = null;//put it in garbage
  (Runtime.getRuntime()).gc(); //the garbage collector is coming
}

//Function to draw Save screen
void drawSave(){
  background(176,167,144);
  image (screen_save,0,0);
}
////////////////////////////////////////focusSAVING////////////////////////////////////////
//Function to load all Saved screen elements
void loadSaved(){
  origami_saving = loadImage ("origami_saving.png");
  screen_homo_saving = loadImage ("homo_saved.png");
  screen_saved = loadImage ("screen_saved_arrows.png");
}

void enterSaved(){
  screenMode = SCREEN_SAVED;
  origami_saving_x = -25;
  origami_saving_y = 20;
  origami_saving_xTarget = 80;
  origami_saving_yTarget = 75;
  unloadAll();
  loadSaved();
}

//Function to unload Saved screen
void unloadSaved(){
  origami_saving = null; //put all these (the null ones) in garbage
  screen_homo_saving = null;
  screen_saved = null;
  (Runtime.getRuntime()).gc(); //oh,oh, the garbage collector is coming!
}

//Function to draw the Saved screen elements
void drawSaved(){
  background (176,167,144);
  image (screen_homo_saving,0,0);
  origami_saving_x= easeOut (origami_saving_x, origami_saving_xTarget, 2);
  origami_saving_y= easeOut (origami_saving_y, origami_saving_yTarget, 7);
  image (origami_saving, origami_saving_x, origami_saving_y);

  if (origami_saving_y == origami_saving_yTarget){
    image (screen_saved,0,0);
  }
}

```

```
//////////FOCUS PAST////////////////////////////////////
```

```
//Function to load all the Past images
```

```
void loadPast(){  
    screen_past= loadImage("screen_past.png");  
    memory1= loadImage ("memory1.png");  
    memory2= loadImage ("memory2.png");  
    memory3= loadImage ("memory3.png");  
    memory4= loadImage ("memory4.png");  
}
```

```
void enterPast(){  
    screenMode = SCREEN_PAST;  
    unloadAll();  
    loadPast();  
}
```

```
//Function to unload all the past images
```

```
void unloadPast(){  
    screen_past= null; //Throw it away  
    memory1= null; //and this too  
    memory2= null; //and this as well  
    memory3= null; //also this please  
    memory4= null; //and it too  
    (Runtime.getRuntime()).gc(); // call garbage collector  
}
```

```
//Function to draw Past screen
```

```
void drawPast(){  
    background(176,167,144);  
  
    if(focusPast == OPTION_MEMORIES_1){  
        image (memory1,0,0);  
    }  
    if(focusPast == OPTION_MEMORIES_2){  
        image (memory2,0,0);  
    }  
    if(focusPast == OPTION_MEMORIES_3){  
        image (memory3,0,0);  
    }  
    if(focusPast == OPTION_MEMORIES_4){  
        image (memory4,0,0);  
    }  
}
```

```
//Function to load all the Memories images
```

```
void loadMemories(){  
  
    memory1_pict = loadImage ("memory1_pict.png");  
    memory2_video = loadImage ("memory2_video.png");  
    memory3_text = loadImage ("memory3_text.png");  
    memory4_audio = loadImage ("memory4_audio.png");  
}
```

```
void enterMemories(){  
    screenMode = SCREEN_MEMORIES_1;  
    unloadAll();  
    loadMemories();  
}
```

```

}

//Function to unload all the Memories images
void unloadMemories(){
    memory1_pict = null; //put all these in garbage
    memory2_video = null;
    memory3_text = null;
    memory4_audio = null;
    (Runtime.getRuntime()).gc(); // call garbage collector
}

```

```

//Function to draw Memories screen
void drawMemories(){
    background (176,167,144);
    if(focusPast == OPTION_MEMORIES_1){
        image (memory1_pict,0,0);
    }
    if(focusPast == OPTION_MEMORIES_2){
        image (memory2_video,0,0);
    }
    if(focusPast == OPTION_MEMORIES_3){
        image (memory3_text,0,0);
    }
    if(focusPast == OPTION_MEMORIES_4){
        image (memory4_audio,0,0);
    }
}

```

```

////////////////////LOG OUT////////////////////////////////////
//Function to load all the images for Log Out screen
void loadLogout(){
    screen_logout= loadImage("exit.png");
}

```

```

void enterLogout(){
    screenMode = SCREEN_LOGOUT;
    unloadAll();
    loadLogout();
}

```

```

//Function to unload images of the Log Out screen
void unloadLogout(){
    screen_logout= null; //put this in garbage
    (Runtime.getRuntime()).gc(); // call garbage collector
}

```

```

//Function to draw Log Out screen
void drawLogout(){
    background(176,167,144);
    image(screen_logout, 0, 0);
}

```

```

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

```

```
////////////////////////////////////  
// Libraries - For special control of the phone  
////////////////////////////////////
```

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

```
////////////////////////////////////  
// State - Named modes of the program to make code more readable  
////////////////////////////////////
```

```
// Names for each possible screen  
final int SCREEN_START = 0;  
final int SCREEN_MENU = 1;  
final int SCREEN_PRESENT = 2;  
final int SCREEN_PAST = 3;  
final int SCREEN_LOGOUT = 4;  
final int SCREEN_MEMORIES_1 = 5;  
final int SCREEN_ORIGAMI = 6;  
final int SCREEN_OPTIONS = 7;  
final int SCREEN_COMMENT = 8;  
final int SCREEN_SEND = 9;  
final int SCREEN_SAVE = 10;  
final int SCREEN_CONTACTS = 11;  
final int SCREEN_CONFIRM = 12;  
final int SCREEN_SENDING = 13;  
final int SCREEN_SAVED = 14;
```

```
// Names for each possible menu focus selection  
final int MENU_OPTION_PAST = 0;  
final int MENU_OPTION_PRESENT = 1;  
final int MENU_OPTION_LOGOUT = 2;
```

```
// Names for each possible menu focus options  
final int MENU_OPTION_COMMENT = 0;  
final int MENU_OPTION_SEND = 1;  
final int MENU_OPTION_SAVE = 2;
```

```
//Names for each possible menu focus memories  
final int OPTION_MEMORIES_1 = 0;  
final int OPTION_MEMORIES_2 = 1;  
final int OPTION_MEMORIES_3 = 2;  
final int OPTION_MEMORIES_4 = 3;
```

```
//Names for each menu contacts options  
final int OPTION_CONTACTS_1 = 0;  
final int OPTION_CONTACTS_2 = 1;  
final int OPTION_CONTACTS_3 = 2;  
final int OPTION_CONTACTS_4 = 3;  
final int OPTION_CONTACTS_5 = 4;  
final int OPTION_CONTACTS_6 = 5;
```

```
//Names for each possible menu options' comment menu  
final int OPTION_TEXT = 0;  
final int OPTION_PICT = 1;  
final int OPTION_AUDIO = 2;  
final int OPTION_VIDEO = 3;
```

```
int screenMode = SCREEN_START;
```

```
int menuFocus = MENU_OPTION_PAST;
int focusPast = OPTION_MEMORIES_1;
int optionsFocus = MENU_OPTION_COMMENT;
int focusComment = OPTION_TEXT;
int optionsContacts = OPTION_CONTACTS_1;
```

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

  enterStart();
}
```

```
/////////////////////////////////////////////////////////////////
// Draw - Executes forever, provides user feedback (Logic Repetition)
/////////////////////////////////////////////////////////////////
```

```
void draw(){
  if(screenMode == SCREEN_START){
    drawStart();
  }
  else if(screenMode == SCREEN_MENU){
    drawMenu();
  }
  else if(screenMode == SCREEN_PRESENT){
    drawPresent();
  }

  else if(screenMode == SCREEN_PAST){
    drawPast();
  }
  else if(screenMode == SCREEN_LOGOUT){
    drawLogout();
  }
  else if(screenMode == SCREEN_MEMORIES_1){
    drawMemories();
  }
  else if(screenMode == SCREEN_ORIGAMI){
    drawOrigami();
  }
  else if(screenMode == SCREEN_OPTIONS){
    drawOptions();
  }
  else if(screenMode == SCREEN_COMMENT){
    drawComment();
  }
  else if(screenMode == SCREEN_SEND){
    drawSend();
  }
  else if(screenMode == SCREEN_CONTACTS){
    drawContacts();
  }
  else if(screenMode == SCREEN_CONFIRM){
    drawConfirm();
  }
}
```



```

/*****
*****
* KeyPressed Section - Code that captures user's input
*****
*****/

void keyPressedScreenStart(){
  switch(keyCode){
    case FIRE:
      enterMenu();
      break;
  }
}

void keyPressedScreenMenu(){
  switch(menuFocus){
    case MENU_OPTION_PAST:
      switch(keyCode){
        case RIGHT:
          menuFocus = MENU_OPTION_PRESENT;
          break;
        case FIRE:
          enterPresent();
          break;
      }
      break;
    case MENU_OPTION_PRESENT:
      switch(keyCode){
        case RIGHT:
          menuFocus = MENU_OPTION_LOGOUT;
          break;
        case LEFT:
          menuFocus = MENU_OPTION_PAST;
          break;
        case FIRE:
          enterPast();
          break;
      }
      break;
    case MENU_OPTION_LOGOUT:
      switch(keyCode){
        case LEFT:
          menuFocus = MENU_OPTION_PRESENT;
          break;
        case FIRE:
          enterLogout();
          break;
      }
      break;
  }
}

void keyPressedScreenPast(){
  switch(focusPast){
    case OPTION_MEMORIES_1:
      switch (keyCode){
        case RIGHT:
          focusPast = OPTION_MEMORIES_2;
          break;

```

```

case FIRE:
    enterMemories();
    break;
case SOFTKEY2:
    enterMenu();
    break;

}
break;
case OPTION_MEMORIES_2:
    switch (keyCode){
    case LEFT:
        focusPast = OPTION_MEMORIES_1;
        break;
    case RIGHT:
        focusPast = OPTION_MEMORIES_3;
        break;
    case FIRE:
        enterMemories();
        break;
    case SOFTKEY2:
        enterMenu();
        break;
    }
    break;
case OPTION_MEMORIES_3:
    switch (keyCode){
    case LEFT:
        focusPast = OPTION_MEMORIES_2;
        break;
    case RIGHT:
        focusPast = OPTION_MEMORIES_4;
        break;
    case FIRE:
        enterMemories();
        break;
    case SOFTKEY2:
        enterMenu();
        break;
    }
    break;
case OPTION_MEMORIES_4:
    switch (keyCode){
    case LEFT:
        focusPast = OPTION_MEMORIES_3;
        break;
    case FIRE:
        enterMemories();
        break;
    case SOFTKEY2:
        enterMenu();
        break;
    }
}
}
}
void keyPressedScreenPresent(){
    switch(keyCode){
    case SOFTKEY1:
        enterOrigami();
        break;
    case SOFTKEY2:

```

```

    enterMenu();
    break;
}
}
void keyPressedScreenMemories(){
    switch(keyCode){
    case SOFTKEY2:
        enterMenu();
        break;
    case SOFTKEY1:
        enterPast();
        break;
    }
}
void keyPressedScreenOrigami(){
    switch(keyCode){
    case SOFTKEY1:
        enterOptions();
        break;
    case SOFTKEY2:
        enterMenu();
        break;
    }
}
void keyPressedScreenOptions(){
    switch(optionsFocus){
    case MENU_OPTION_COMMENT:
        switch(keyCode){
        case RIGHT:
            optionsFocus = MENU_OPTION_SEND;
            break;
        case FIRE:
            enterComment();
        }
        break;
    case MENU_OPTION_SEND:
        switch(keyCode){
        case RIGHT:
            optionsFocus = MENU_OPTION_SAVE;
            break;
        case LEFT:
            optionsFocus = MENU_OPTION_COMMENT;
            break;
        case FIRE:
            enterSend();
        }
        break;
    case MENU_OPTION_SAVE:
        switch(keyCode){
        case LEFT:
            optionsFocus = MENU_OPTION_SEND;
            break;
        case FIRE:
            enterSave();
        }
    }
}
void keyPressedScreenComment(){
    switch(focusComment){
    case OPTION_TEXT:
        switch(keyCode){

```

```

case RIGHT:
    focusComment = OPTION_PICT;
    break;
case SOFTKEY2:
    enterOptions();
}
break;
case OPTION_PICT:
    switch(keyCode){
case RIGHT:
    focusComment = OPTION_AUDIO;
    break;
case LEFT:
    focusComment = OPTION_TEXT;
    break;
case SOFTKEY2:
    enterOptions();
}
break;
case OPTION_AUDIO:
    switch(keyCode){
case RIGHT:
    focusComment = OPTION_VIDEO;
    break;
case LEFT:
    focusComment = OPTION_PICT;
    break;
case SOFTKEY2:
    enterOptions();
}
break;
case OPTION_VIDEO:
    switch(keyCode){
case LEFT:
    focusComment = OPTION_AUDIO;
    break;
case SOFTKEY2:
    enterOptions();
}
}
}
}

```

```

void keyPressedScreenSave(){
    switch(keyCode){
case SOFTKEY2:
    enterOptions();
    break;
case FIRE:
    enterSaved();
}
}

```

```

void keyPressedScreenSend(){
    switch(keyCode){
case SOFTKEY1:
    enterContacts();
    break;
case SOFTKEY2:
    enterOptions();
    break;
}
}
}

```

```

void keyPressedScreenContacts(){
  switch(optionsContacts){
  case OPTION_CONTACTS_1:
    switch (keyCode){
    case DOWN:
      optionsContacts = OPTION_CONTACTS_2;
      break;
    case SOFTKEY2:
      enterOptions();
      break;
    }
    break;
  case OPTION_CONTACTS_2:
    switch (keyCode){
    case DOWN:
      optionsContacts = OPTION_CONTACTS_3;
      break;
    case UP:
      optionsContacts = OPTION_CONTACTS_1;
      break;
    }
    break;
  case OPTION_CONTACTS_3:
    switch (keyCode){
    case DOWN:
      optionsContacts = OPTION_CONTACTS_4;
      break;
    case UP:
      optionsContacts = OPTION_CONTACTS_2;
      break;
    }
    break;
  case OPTION_CONTACTS_4:
    switch (keyCode){
    case DOWN:
      optionsContacts = OPTION_CONTACTS_5;
      break;
    case UP:
      optionsContacts = OPTION_CONTACTS_3;
      break;
    }
    break;
  case OPTION_CONTACTS_5:
    switch (keyCode){
    case DOWN:
      optionsContacts = OPTION_CONTACTS_6;
      break;
    case UP:
      optionsContacts = OPTION_CONTACTS_4;
      break;
    case FIRE:
      enterConfirm();
      break;
    }
    break;
  case OPTION_CONTACTS_6:
    switch (keyCode){
    case UP:
      optionsContacts = OPTION_CONTACTS_5;
      break;
    }
  }
}

```

```
    break;
  }
}
```

```
void keyPressedScreenConfirm(){
  switch(keyCode){
  case SOFTKEY2:
    enterContacts();
    break;
  case FIRE:
    enterSending();
  }
}
```

```
void keyPressedScreenSending(){
  switch(keyCode){
  case SOFTKEY1:
    enterOptions();
    break;
  case SOFTKEY2:
    enterMenu();
  }
}
```

```
void keyPressedScreenSaved(){
  switch(keyCode){
  case SOFTKEY1:
    enterOptions();
    break;
  case SOFTKEY2:
    enterMenu();
  }
}
```