```
/*
Cinomadic is an interactive system,
accessed by mobile phone,
which allows people to discover the projection sites of an
itinerant and irreverent open air cinema around Venice.
*/

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

//Names for each possible screen

//Master Modes
int SECTION_LOGO = 0;
int SECTION_MAINMENU = 1;
int SECTION_EVENTS = 2;
int SECTION_SEARCH = 3;
int SECTION_MYEVENTS = 4;
int SECTION_TORCH = 5;

//Main Menu Focus Options
int MAINMENU_EVENTS = 0;
int MAINMENU_SEARCH = 1;
int MAINMENU_MYEVENTS = 2;
int MAINMENU_TORCH = 3;


//MyEvents Screen Modes
int MYEVENTS_MENU = 0;
int MYEVENTS_DETAIL = 1;

//MyEvents Menu Focus Options
int MYEVENTS_MENU_EVENT0 = 0;
int MYEVENTS_MENU_EVENT1 = 1;
int MYEVENTS_MENU_EVENT2 = 2;
int MYEVENTS_MENU_EVENT3 = 3;

//MyEvents Detail Mode
int MYEVENTS_DETAIL_EVENT0 = 0;
int MYEVENTS_DETAIL_EVENT1 = 1;
int MYEVENTS_DETAIL_EVENT2 = 2;
int MYEVENTS_DETAIL_EVENT3 = 3;

//Search mode
int SCREEN_POPUP = 0;
int SCREEN_MAPPA = 1;
int SCREEN_FUMETTO = 0;

//Torch Mode
int TORCH_OPENING = 0;
int TORCH_INFO = 1;
int TORCH_MAP = 2;

int TORCH_MAP_STAGE0 = 0;
int TORCH_MAP_STAGE1 = 1;
int TORCH_MAP_STAGE2 = 2;
int TORCH_MAP_STAGE3 = 3;
int TORCH_MAP_STAGE4 = 4;
```

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

int sectionMode =  SECTION_LOGO;
int mainmenuFocus = MAINMENU_EVENTS;
int eventsMode = 0; // events_images 1st array index
int eventsModeMovie = 0; // events_images 2nd array index
int myeventsMode = MYEVENTS_MENU;
int myeventsmenuFocus = MYEVENTS_MENU_EVENT0;
int myeventsdetailMode = MYEVENTS_DETAIL_EVENT0;


// Torch variables
int torchMode = TORCH_OPENING;
int torchMapMode = TORCH_MAP_STAGE0;


//Search variables
int searchMode = SCREEN_POPUP;
int searchFocus = SCREEN_FUMETTO;

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


void setup()
{
  myPhone = new Phone(this); // Creates a phone controller
  myPhone.fullscreen(); // Use the entire screen
  loadFonts();
  loadImages();
  updateLightPosition();
  initializeTorchVars();
  initializeEventSchedules();
  initializeEventsVars();
}


void draw()
{
  if((sectionMode == SECTION_EVENTS)||(sectionMode == SECTION_MYEVENTS)){
    loadAllEventsImages();
  }
  else if(sectionMode == SECTION_TORCH){
    loadTorchImages();
  }else{
    unloadAllEventsImages();
    unloadTorchImages();
  }

  if(sectionMode == SECTION_LOGO){
    drawSectionLogo();
  }
  else if(sectionMode == SECTION_MAINMENU){
    drawSectionMainmenu();
  }
  else if(sectionMode == SECTION_EVENTS){
    drawSectionEvents();
  }
  else if(sectionMode == SECTION_SEARCH){
    drawSectionSearch();
  }
  else if(sectionMode == SECTION_MYEVENTS){
    drawSectionMyevents();
  }
  else if(sectionMode == SECTION_TORCH){
    drawSectionTorch();
  }
}

////////////////////////////////////////////////////////////////////////////////////////////
// Initialization functions
////////////////////////////////////////////////////////////////////////////////////////////

void initializeEventSchedules(){
  events_schedules[0][0] = "Monday\n20:30 - 23:30";
  events_schedules[0][1] = "Tuesday\n20:30 - 23:30";
  events_schedules[0][2] = "Friday\n21:00 - 23:30";
  events_schedules[0][3] = "Saturday\n21:00 - 23:30";

  events_schedules[1][0] = "Tuesday\n20:00 - 23:00";
  events_schedules[1][1] = "Wednsday\n20:30 - 23:30";
  events_schedules[1][2] = "Friday\n20:30 - 23:30";
  events_schedules[1][3] = "Sunday\n21:00 - 00:00";

  events_schedules[2][0] = "Wednesday\n22:00 - 00:00";
  events_schedules[2][1] = "Friday\n20:30 - 23:30";
  events_schedules[2][2] = "Saturday\n20:30 - 23:30";
  events_schedules[2][3] = "Sunday\n22:00 - 00:00";

  events_schedules[3][0] = "Monday\n20:00 - 22:30";
  events_schedules[3][1] = "Wednsday\n20:00 - 22:30";
  events_schedules[3][2] = "Thursday\n20:30 - 23:00";
  events_schedules[3][3] = "Saturday\n20:00 - 22:30";

  events_schedules[4][0] = "Tuesday\n20:00 - 23:00";
  events_schedules[4][1] = "Wednsday\n20:30 - 23:30";
  events_schedules[4][2] = "Friday\n21:00 - 00:00";
  events_schedules[4][3] = "Sunday\n20:00 - 23:00";

  events_schedules[5][0] = "Monday\n20:00 - 23:00";
  events_schedules[5][1] = "Wednsday\n20:00 - 23:00";
  events_schedules[5][2] = "Thuersday\n21:30 - 00:30";
  events_schedules[5][3] = "Saturday\n21:30 - 00:30";

  events_schedules[6][0] = "Tuesday\n20:00 - 00:00";
  events_schedules[6][1] = "Friday\n22:30 - 00:30";
  events_schedules[6][2] = "Saturday\n21:00 - 01:00";
  events_schedules[6][3] = "Sunday\n20:00 - 00:00";
}

void initializeTorchVars(){
  int i;
  for(i = 0; i < 100; ++i){
    mapConfettiPositions0[i][0] = random(0, width);
    mapConfettiPositions0[i][1] = random(-height, 0);
  }
  for(i = 0; i < 6; ++i){
    mapConfettiPositions1[i][0] = random(0, width);
    mapConfettiPositions1[i][1] = random(-height, 0);
  }
}
```

```
void initializeEventsVars(){
  events_descriptions[0][0] = "I Short-film\n\n-A beautiful day-\nJonny Jolly\nThailand 2008\n3'33''";
  events_images_counts[0] = 1;

  events_descriptions[1][0] = "I Short-film\n\n-Passage-\nMichael Capone\nUSA 2006\n5'45''";
  events_images_counts[1] = 1;

  events_descriptions[2][0] = "I Short-film\n\n-Wally-\nMarcus Jung\nCanada 2008\n15'00''";
  events_descriptions[2][1] = "II Short-film\n\n-You and me-\nJohn Mc'Bill\nGreat Britain\n2008\n";
  events_descriptions[2][2] = "III Short-film\n\n-Cicci & Ciocci-\nMaria Poltrona\nItaly 2007\n32'50''";
  events_descriptions[2][3] = "IV Short-film\n\n-Mix-\nFloris De Maria\nRomania 2006\n20'30''";
  events_images_counts[2] = 4;

  events_descriptions[3][0] = "I Short-film\n\n-Manneh!!-\nStefan Hauser,\nJonas Wolf\nGermany 
2008\n20'30''";
  events_images_counts[3] = 1;

  events_descriptions[4][0] = "I Short-film\n\n-Zu Zu-\nPiyarporn\nTubkong\nCambogia 2008\n5'10''";
  events_images_counts[4] = 1;

  events_descriptions[5][0] = "I Short-film\n\n-Lola-\nJekaterina\nMilovaz\nEstonia 2006\n9'15''";
  events_images_counts[5] = 1;

  events_descriptions[6][0] = "I Short-film\n\n-La vie-\nDomenique\nRouge\nFrance 2006\n10'15''";
  events_images_counts[6] = 1;
}

void collectGarbage(){
  //System.gc();
  Runtime r = Runtime.getRuntime();
  r.gc();
}


//GRAPHICS

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

PFont futura12MediumWhite;
PFont futura16BoldWhite;
PFont futura12MediumBlack;

////////////////////////////////////////////////////////////////
// LOGO
PImage logo;

////////////////////////////////////////////////////////////////
// MAIN MENU
PImage mainmenu_background;
PImage mainmenu_options;
PImage mainmenu_focusbar1;
PImage mainmenu_focusbar2;
PImage mainmenu_focusbar3;
PImage mainmenu_focusbar4;

////////////////////////////////////////////////////////////////
// EVENTS
PImage events_background;
int[] events_images_counts = new int[7];
```

```
PImage[][] events_images = new PImage[7][4];
String[][] events_descriptions = new String[7][4];
String[][] events_schedules = new String[7][4];



////////////////////////////////////////////////////////////
// MY EVENTS
PImage myevents_menu_background;
PImage myevents_menu_details;
PImage myevents_menu_focusbar;
PImage myevents_detail_background;
PImage myevents_detail_background1;
PImage myevents_detail_background2;
PImage myevents_detail_background3;
PImage myevents_search_saved;

////////////////////////////////////////////////////////////
// SEARCH
PImage search_mappa;
PImage search_stella;
PImage search_popup;
PImage search_fumetto;

////////////////////////////////////////////////////////////
// TORCH
//PImage torch;
//PImage torch from Suzana
PImage openingBackground;
PImage openingAnimation[] = new PImage[5];

PImage infoArrowScrollUp;
PImage infoArrowScrollDown;
PImage infoMask;
PImage infoTextScroll;

PImage mapBackground0;
PImage mapBackground1;
PImage mapBackground2;
PImage mapHighlightArrows0;
PImage mapHighlightArrows1;
PImage mapHighlightArrows2;
PImage mapZoomBackground0;
PImage mapZoomBackground1;
PImage mapYou;
PImage mapUsers0;
PImage mapUsers1;
PImage mapUsers2;
PImage mapZoomUsers0;
PImage mapZoomUsers1;
PImage mapConfetti0;
PImage mapConfetti1;

int mapConfettiPositions0[][] = new int[100][2];
int mapConfettiPositions1[][] = new int[6][2];

////////////////////////////////////////////////////////////

void loadFonts(){
  futura12MediumBlack = loadFont("FuturaBT-Medium-12.mvlw", color(0));
  futura12MediumWhite = loadFont("FuturaBT-Medium-12.mvlw", color(255));
  futura16BoldWhite = loadFont("FuturaBT-Bold-16.mvlw", color(255));
```

```
}

void loadMainMenuImages(){
  mainmenu_background = loadImage("mainmenu_background.png");
  mainmenu_options = loadImage("mainmenu_options.png");
  mainmenu_focusbar1 = loadImage("mainmenu_focusbar1.png");
  mainmenu_focusbar2 = loadImage("mainmenu_focusbar2.png");
  mainmenu_focusbar3 = loadImage("mainmenu_focusbar3.png");
  mainmenu_focusbar4 = loadImage("mainmenu_focusbar4.png");
}

void loadMyEventsImages(){
  myevents_menu_background = loadImage("myevents_menu_background.png");
  myevents_menu_details = loadImage("myevents_menu_details.png");
  myevents_menu_focusbar = loadImage("myevents_menu_focusbar.png");
  myevents_detail_background = loadImage("myevents_detail_background.png");
  myevents_detail_background1 = loadImage("myevents_detail_background1.png");
  myevents_detail_background2 = loadImage("myevents_detail_background2.png");
  myevents_detail_background3 = loadImage("myevents_detail_background3.png");
  myevents_search_saved = loadImage ("myevents_search_saved.png");
}

void unloadMyEventsImages(){
  myevents_menu_background = null;
  myevents_menu_details = null;
  myevents_menu_focusbar = null;
  myevents_detail_background = null;
  myevents_detail_background1 = null;
  myevents_detail_background2 = null;
  myevents_detail_background3 = null;
  myevents_search_saved = null;
  collectGarbage();
}

void loadEventsImages(){
  events_background = loadImage("events_background.png");

  events_images[0][0] = loadImage("events_image00.png");
  events_images[1][0] = loadImage("events_image10.png");

  events_images[2][0] = loadImage("events_image20.png");
  events_images[2][1] = loadImage("events_image21.png");
  events_images[2][2] = loadImage("events_image22.png");
  events_images[2][3] = loadImage("events_image23.png");

  events_images[3][0] = loadImage("events_image30.png");

  events_images[4][0] = loadImage("events_image40.png");

  events_images[5][0] = loadImage("events_image50.png");

  events_images[6][0] = loadImage("events_image60.png");
}

void unloadEventsImages(){
  events_background = null;
  int i = 0;
  int j = 0;
  for(i = 0; i < 7; i++){
    for(j = 0; j < events_images_counts[i]; j++){
      events_images[i][j] = null;
    }
  }
  collectGarbage();
}

boolean eventsLoadedImages = false;

void loadAllEventsImages(){
  if(true == eventsLoadedImages){
    return;
  }
  loadEventsImages();
  loadMyEventsImages();
  eventsLoadedImages = true;
}

void unloadAllEventsImages(){
  if(false == eventsLoadedImages){
    return;
  }
  unloadEventsImages();
  unloadMyEventsImages();
  eventsLoadedImages = false;
}

boolean torchLoadedImages = false;

void loadTorchImages(){
  if(true == torchLoadedImages){
    return;
  }
  openingBackground = loadImage("openingBackground.png");
  openingAnimation[0] = loadImage("openingAnimation0.png");
  openingAnimation[1] = loadImage("openingAnimation1.png");
  openingAnimation[2] = loadImage("openingAnimation2.png");
  openingAnimation[3] = loadImage("openingAnimation3.png");
  openingAnimation[4] = loadImage("openingAnimation4.png");

  infoMask = loadImage("infoMask.png");
  infoTextScroll = loadImage("infoTextScroll.png");
  infoArrowScrollUp = loadImage("infoArrowScrollUp.png");
  infoArrowScrollDown = loadImage("infoArrowScrollDown.png");

  mapBackground0 = loadImage("mapBackground0.png");
  mapBackground1 = loadImage("mapBackground1.png");
  mapBackground2 = loadImage("mapBackground2.png");
  mapHighlightArrows0 = loadImage("mapHighlightArrows0.png");
  mapHighlightArrows1 = loadImage("mapHighlightArrows1.png");
  mapHighlightArrows2 = loadImage("mapHighlightArrows2.png");
  mapZoomBackground0 = loadImage("mapZoomBackground0.png");
  mapZoomBackground1 = loadImage("mapZoombackground1.png");
  mapYou = loadImage("mapYou.png");
  mapUsers0 = loadImage("mapUsers0.png");
  mapUsers1 = loadImage("mapUsers1.png");
  mapUsers2 = loadImage("mapUsers2.png");
  mapZoomUsers0 = loadImage("mapZoomUsers0.png");
  mapZoomUsers1 = loadImage("mapZoomUsers1.png");
  mapConfetti0 = loadImage("mapConfetti0.png");
  mapConfetti1 = loadImage("mapConfetti1.png");

  torchLoadedImages = true;
}
```

```
void unloadTorchImages(){
  if(false == torchLoadedImages){
    return;
  }
  openingBackground = null;
  openingAnimation[0] = null;
  openingAnimation[1] = null;
  openingAnimation[2] = null;
  openingAnimation[3] = null;
  openingAnimation[4] = null;

  infoMask = null;
  infoTextScroll = null;
  infoArrowScrollUp = null;
  infoArrowScrollDown = null;

  mapBackground0 = null;
  mapBackground1 = null;
  mapBackground2 = null;
  mapHighlightArrows0 = null;
  mapHighlightArrows1 = null;
  mapHighlightArrows2 = null;
  mapZoomBackground0 = null;
  mapZoomBackground1 = null;
  mapYou = null;
  mapUsers0 = null;
  mapUsers1 = null;
  mapUsers2 = null;
  mapZoomUsers0 = null;
  mapZoomUsers1 = null;
  mapConfetti0 = null;
  mapConfetti1 = null;

  collectGarbage();
  torchLoadedImages = false;
}

void loadSearchImages(){
  search_mappa = loadImage ("search_mappa.png");
  search_stella = loadImage ("search_stella.png");
  search_popup = loadImage ("search_popup.png");
  search_fumetto = loadImage ("search_fumetto.png");
}

// Function to load all the images
void loadImages(){
  logo = loadImage("logo.png");
  loadMainMenuImages();
  loadSearchImages();
}

int menuFocusYPosition = 51;
// search variables
//ellipse x and y coordinates
int ellipseX = 125;
int ellipseY = 200;
//triangle x1 and x2 coordinates
int a = 120;
int a1 = 130;
// triangle y coordinate
int b = 200;
//triangle2 x coordinate
```

```
int c = 125;
//triangle2 y1 and y2 coordinates
int d1 = 195;
int d2 = 205;
// stella coordinates
int stellaX = 120;
int stellaY = 210;
// distance of the spotlight
int distance = 0;

int fumetto1x = 100;
int fumetto2y = 100;

/////////////////////////////////////////////////////////////

void drawSectionLogo (){
  image (logo, 0,0);
}

void drawSectionMainmenu()
{
// draw menu background
  image(mainmenu_background, 0, 0);

// draw focus bar
  if(mainmenuFocus == MAINMENU_EVENTS){
    image(mainmenu_focusbar1, 0, menuFocusYPosition);
  }
  else if(mainmenuFocus == MAINMENU_SEARCH){
    image(mainmenu_focusbar2, 0, menuFocusYPosition);
  }
  else if(mainmenuFocus == MAINMENU_MYEVENTS){
    image(mainmenu_focusbar3, 0, menuFocusYPosition);
  }
  else if(mainmenuFocus == MAINMENU_TORCH){
    image(mainmenu_focusbar4, 0, menuFocusYPosition);
  }
// draw menu options text
  image(mainmenu_options, 0, 0);
}

void drawSectionEvents(){
  image(events_background, 0, 0);

// draw the event image
  image(events_images[eventsMode][eventsModeMovie], 54, 57);// draw the event description
  textFont(futura12MediumBlack);
  text(events_descriptions[eventsMode][eventsModeMovie], 131, 186);

// draw the schedule text
  textFont(futura12MediumWhite);
  text(events_schedules[eventsMode][0], 25, 186);
  text(events_schedules[eventsMode][1], 25, 212);
  text(events_schedules[eventsMode][2], 25, 238);
  text(events_schedules[eventsMode][3], 25, 264);

// draw title
  textFont(futura16BoldWhite);
  String eventNumber = "" + (eventsMode + 1);
  text(eventNumber, 115, 43);
}
```

```
void drawSectionSearch(){
  if(searchMode == SCREEN_POPUP){
    drawMappa();
    drawPopup();
  }
  else if(searchMode == SCREEN_MAPPA){
    updateLightPosition();
    drawMappa();
  }
  else if (searchMode ==SCREEN_FUMETTO){
    drawFumetto();
  }
}
/////////////////////////////////////////////////////////////////////////////////////////////////
// SEARCH FUNTIONS TO drawSectionSearch

void drawMappa (){
  image (search_mappa,0,0);
  drawFascio();
  drawStella();
  drawFumetto();
}

void drawPopup (){
  image (search_popup,0,0);
}

void drawStella (){
  image (search_stella,stellaX,stellaY);
}

void drawFascio(){
  fill (50,255,255);
  if( ellipseX < 0 && ellipseX > 180 && ellipseY < 100&&  ellipseY >340){
    fill(50, 255, 255);
  }

  if( ellipseX > 80 && ellipseY < 100){
    fill(50, 255, 255);
  }
  else if( ellipseX > 100 && ellipseX < 150 && ellipseY < 150&& ellipseY > 115 &&50 < calculateDistance
(fumetto1x, fumetto2y, ellipseX, ellipseY)){
    fill(255,0,0);
  }
  else if (distance > 30){
    fill(50, 50, 255);
  }
  noStroke();
  drawTriangle();
  drawTriangle2();
  drawEllipse();
}

void drawEllipse (){
  ellipse(ellipseX, ellipseY,10,10);
}

void drawTriangle(){
  triangle(128,220,a,b,a1,b);
}

void drawTriangle2(){
  triangle(128,220,c,d1,c,d2);
}

boolean showingFumetto = false;
void drawFumetto(){
  if(ellipseX > 110 && ellipseX < 150 && ellipseY < 140 && ellipseY > 120 && 50 <
calculateDistance(fumetto1x, fumetto2y, ellipseX, ellipseY) ){
    showingFumetto = true;
    image (search_fumetto,0,0);
  }
}

void updateLightPosition(){
  int speed = 2;
  if(keyIsPressed){
    //move the light(ellipse and triangle and triangle2)
    switch (keyCode) {
    case UP:
      ellipseY = max(25, ellipseY - speed);
      b = max(25, b -speed);
      d1 = max(20, d1 -speed);
      d2 = max(30, d2 -speed);
      break;
    case DOWN:
      ellipseY = min(height - 50, ellipseY + speed);
      b = min(height - 50, b + speed);
      d1 = min(height - 55, d1 + speed);
      d2 = min(height - 45, d2 + speed);
      break;
    case LEFT:
      ellipseX = max(5, ellipseX - speed);
      a = max(0, a - speed);
      a1 = max(10, a1 - speed);
      c = max(5, c - speed);
      break;
    case RIGHT:
      ellipseX = min(width - 5, ellipseX + speed);
      a = min(width - 10, a + speed);
      a1 = min(width, a1 + speed);
      c = min(width-5, c + speed);
      break;
    }
  }
  distance = calculateDistance(ellipseX, ellipseY, stellaX, stellaY);
}

//////////////////////////////////////////////////////////////////////////////////////////

void drawSectionMyevents(){
  if(myeventsMode == MYEVENTS_MENU){
    image(myevents_menu_background, 0, 0);

    if(myeventsmenuFocus == MYEVENTS_MENU_EVENT0){
      image(myevents_menu_focusbar, 0, 4);
    }
    else if(myeventsmenuFocus == MYEVENTS_MENU_EVENT1){
      image(myevents_menu_focusbar, 0, 34);
    }
    else if(myeventsmenuFocus == MYEVENTS_MENU_EVENT2){
      image(myevents_menu_focusbar, 0, 64);
    }
    else if(myeventsmenuFocus == MYEVENTS_MENU_EVENT3){
      image(myevents_menu_focusbar, 0, 94);
    }
    // draw section_myevents options text
    image(myevents_menu_details, 24, 41);
  }
  else if(myeventsMode == MYEVENTS_DETAIL){
    if(myeventsdetailMode == MYEVENTS_DETAIL_EVENT0){
      image(myevents_detail_background, 0, 0);
    }
    else if(myeventsdetailMode == MYEVENTS_DETAIL_EVENT1){
      image (myevents_detail_background1,0,0);
    }
    else if(myeventsdetailMode == MYEVENTS_DETAIL_EVENT2){
      image (myevents_detail_background2, 0, 0);
    }
    else if(myeventsdetailMode == MYEVENTS_DETAIL_EVENT3){
      image (myevents_detail_background3, 0, 0);
    }
  }
}

////////////////////////////////////////////////////////////////////////////////////////////////
// Torch functions

void drawSectionTorch(){
  if(torchMode == TORCH_OPENING){
    drawOpening();
  }
  else if(torchMode == TORCH_INFO){
    drawInfo();
  }
  else if(torchMode == TORCH_MAP){
    drawMap();
  }
}

boolean drawOpeningAnimationDirection = true;
int drawOpeningAnimationClock = 0;
void drawOpening(){
  image(openingBackground, 0, 0);
  image(openingAnimation[drawOpeningAnimationClock], 0, 0);
  if(drawOpeningAnimationDirection){
    drawOpeningAnimationClock++;
    if(drawOpeningAnimationClock > 3){
      drawOpeningAnimationDirection = false;
    }
  }
  else{
    drawOpeningAnimationClock--;
    if(drawOpeningAnimationClock < 1){
      drawOpeningAnimationDirection = true;
    }
  }
}

int drawInfoTextPosition = 4;
void drawInfo(){
  int bottomScrollPosition = 240 - 470 - 5;
  if(torchKeyIsPressed){
    if(keyCode == UP){
      drawInfoTextPosition += 10;
    }
    else if(keyCode == DOWN){
      drawInfoTextPosition -= 10;
    }
    drawInfoTextPosition = constrain(drawInfoTextPosition, bottomScrollPosition, 4);
  }
  image(infoTextScroll, 0, drawInfoTextPosition);
  image(infoMask, 0, 0);
  if(drawInfoTextPosition < 4){
    image(infoArrowScrollUp, 33, 64);
  }
  if(drawInfoTextPosition != bottomScrollPosition){
    image(infoArrowScrollDown, 33, 290);
  }
}

boolean mapArrowsEnabled = false;
int mapYouX = 110;
int mapYouY = 164;

void drawMap(){
  if(torchMapMode == TORCH_MAP_STAGE0){
    image(mapBackground0, 0, 0);
    image(mapUsers0, 0, 0);
    if(mapArrowsEnabled){
      image(mapHighlightArrows0, 0, 0);
    }
  }
  else if(torchMapMode == TORCH_MAP_STAGE1){
    image(mapBackground1, 0, 0);
    image(mapUsers1, 0, 0);
    if(mapArrowsEnabled){
      image(mapHighlightArrows1, 0, 0);
    }
  }
  else if(torchMapMode == TORCH_MAP_STAGE2){
    image(mapBackground2, 0, 0);
    image(mapUsers2, 0, 0);
    if(mapArrowsEnabled){
      image(mapHighlightArrows2, 0, 0);
    }
  }
  else if(torchMapMode == TORCH_MAP_STAGE3){
    image(mapZoomBackground0, 0, 0);
    image(mapZoomUsers0, 0, 0);
  }
  else if(torchMapMode == TORCH_MAP_STAGE4){
    image(mapZoomBackground1, 0, 0);
    image(mapZoomUsers1, 0, 0);
    //image(mapYou, 110, 164); // start
    //image(mapYou, 150, 120); // end
    mapYouX = animateTo(mapYouX,150, 2);
    mapYouY = animateTo(mapYouY,120, 2);
    image(mapYou, mapYouX, mapYouY); // start
    if((mapYouX == 150)&&(mapYouY == 120)){
      drawMapConfetti();
    }
  }
}

boolean finished = false;
void drawMapConfetti(){
  if(finished){
    return;
  }
  finished = true;
  int i;
  for(i = 0; i < 100; ++i){
    image(mapConfetti0, mapConfettiPositions0[i][0], mapConfettiPositions0[i][1]);
    mapConfettiPositions0[i][1] += random(5, 15);
    if(mapConfettiPositions0[i][1] < height){
      finished = false;
    }
  }
  for(i = 0; i < 6; ++i){
    image(mapConfetti1, mapConfettiPositions1[i][0], mapConfettiPositions1[i][1]);
    mapConfettiPositions1[i][1] += random(5, 15);
    if(mapConfettiPositions1[i][1] < height){
      finished = false;
    }
  }
}
/*
// Animation Functions ////////////////////////////////////////////////////////////////////////////////////////////////////

// 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)
}
*/
```

```
//BEHAVIORS

void keyPressedLogo (){
  if (keyCode == SOFTKEY2){
    sectionMode =  SECTION_MAINMENU;
  }
}

void keyPressedSectionMainmenu(){
  if  (mainmenuFocus == MAINMENU_EVENTS){ // if focused in optionA=events
    if (keyCode == RIGHT){ // if pressed RIGHT
      mainmenuFocus = MAINMENU_SEARCH; // moves focus to optionB=search
    }
    else if (keyCode == SOFTKEY2){ // IF FIRE is pressed
      sectionMode = SECTION_EVENTS; // shows events
    }
  }
  else if  (mainmenuFocus == MAINMENU_SEARCH) { // if focused in optionA=events
    if (keyCode == LEFT){ // if pressed RIGHT
      mainmenuFocus = MAINMENU_EVENTS; // moves focus to optionB=search
    }
    else if (keyCode == RIGHT){ // if pressed RIGHT
      mainmenuFocus = MAINMENU_MYEVENTS; // moves focus to optionB=search
    }
    else if (keyCode == SOFTKEY2){ // IF FIRE is pressed
      sectionMode = SECTION_SEARCH; // shows events
    }
  }
  else if (mainmenuFocus == MAINMENU_MYEVENTS){
    if (keyCode == LEFT){
      mainmenuFocus = MAINMENU_SEARCH;
    }
    else if (mainmenuFocus == MAINMENU_MYEVENTS){
      if (keyCode == RIGHT){
        mainmenuFocus = MAINMENU_TORCH;
      }
      else if (mainmenuFocus == MAINMENU_MYEVENTS) {
        if (keyCode == SOFTKEY2){
          sectionMode = SECTION_MYEVENTS;
        }
      }
    }
  }
  else if (keyCode == SOFTKEY2){
    sectionMode = SECTION_TORCH;
  }
  else if (mainmenuFocus == MAINMENU_TORCH) {
    if (keyCode == LEFT){
      mainmenuFocus = MAINMENU_MYEVENTS;
    }
  }
}

void keyPressedSectionEvents(){
  if (keyCode == SOFTKEY1){
    sectionMode = SECTION_MAINMENU;
  }
  // events_images.length is 7
  // our last item in the array is 6
  // (events_images.length - 1) gives us 6
  if(keyCode == LEFT){
    if (0 == eventsMode){
      eventsMode = 6;
    }
    else{
      --eventsMode;
    }
    eventsModeMovie = 0;
  }
  if(keyCode == RIGHT){
    if (eventsMode == 6){
      eventsMode = 0;
    }
    else{
      ++eventsMode;
    }
    eventsModeMovie = 0;
  }
  if(keyCode == UP){
    if (0 == eventsModeMovie){
      eventsModeMovie = (events_images_counts[eventsMode] - 1);
    }
    else{
      --eventsModeMovie;
    }
  }
  if(keyCode == DOWN){
    if (eventsModeMovie == (events_images_counts[eventsMode] - 1)){
      eventsModeMovie = 0;
    }
    else{
      ++eventsModeMovie;
    }
  }
}


////////////////////////////////////////////////////////////
//search functions

void keyPressedSectionSearch(){
  keyIsPressed = true; // record that the key is pressed

  if(searchMode == SCREEN_POPUP){
    if(keyCode == SOFTKEY2){
      searchMode = SCREEN_MAPPA; // show screen A
    }
    else if (keyCode == SOFTKEY1){
      sectionMode =  SECTION_MAINMENU;
    }
  }
  else if (searchMode == SCREEN_MAPPA){
    if(keyCode == SOFTKEY2){
      if(showingFumetto){
        sectionMode = SECTION_EVENTS;
        eventsMode = 0;
        eventsModeMovie = 0;
      }
      else{
        searchMode = SCREEN_POPUP;
      }
    }
    else if(keyCode == SOFTKEY1){
      sectionMode = SECTION_MAINMENU;
    }
  }
}


// funtion updatelightposition in void setup()
int calculateDistance(int x1, int y1, int x2, int y2){
  int dist2X = itofp(sq(x1 - x2));
  int dist2Y = itofp(sq(y1 - y2));
  return fptoi(sqrt(dist2X + dist2Y));
}

////////////////////////////////////////////////////////////

void keyPressedSectionMyevents(){
  if (myeventsMode == MYEVENTS_MENU){

    //if focus on the 1'event = EVENT0
    if (myeventsmenuFocus == MYEVENTS_MENU_EVENT0){
      if (keyCode == DOWN){
        myeventsmenuFocus = MYEVENTS_MENU_EVENT1;
      }
      else if(keyCode == SOFTKEY2){
        myeventsMode = MYEVENTS_DETAIL;
        myeventsdetailMode = MYEVENTS_DETAIL_EVENT0;
      }
    }

    //if focus on the 1'event = EVENT1
    else  if(myeventsmenuFocus == MYEVENTS_MENU_EVENT1){
      if (keyCode == DOWN){
        myeventsmenuFocus = MYEVENTS_MENU_EVENT2;
      }
      else if (keyCode == UP){
        myeventsmenuFocus = MYEVENTS_MENU_EVENT0;
      }
      else if(keyCode == SOFTKEY2){
        myeventsMode = MYEVENTS_DETAIL;
        myeventsdetailMode = MYEVENTS_DETAIL_EVENT1;
      }
    }

    //if focus on the 1'event = EVENT2
    else  if(myeventsmenuFocus == MYEVENTS_MENU_EVENT2){
      if (keyCode == DOWN){
        myeventsmenuFocus = MYEVENTS_MENU_EVENT3;
      }
      else if (keyCode == UP){
        myeventsmenuFocus = MYEVENTS_MENU_EVENT1;
      }
      else if(keyCode == SOFTKEY2){
        myeventsMode = MYEVENTS_DETAIL;
        myeventsdetailMode = MYEVENTS_DETAIL_EVENT2;
      }
    }

    //if focus on the 1'event = EVENT3
    else  if(myeventsmenuFocus == MYEVENTS_MENU_EVENT3){
      if (keyCode == UP){
        myeventsmenuFocus = MYEVENTS_MENU_EVENT2;
      }
      else if(keyCode == SOFTKEY2){
        myeventsMode = MYEVENTS_DETAIL;
        myeventsdetailMode = MYEVENTS_DETAIL_EVENT3;
      }
    }
    if (keyCode == SOFTKEY1){
      sectionMode =  SECTION_MAINMENU;
    }
  }

  else if (myeventsMode == MYEVENTS_DETAIL){
    if (myeventsdetailMode == MYEVENTS_DETAIL_EVENT0){
      if (keyCode == SOFTKEY1){
        myeventsMode = MYEVENTS_MENU ;
      }
    }

    // if focus on MYEVENTS_DETALI_EVENT1
    if (myeventsdetailMode == MYEVENTS_DETAIL_EVENT1){
      if (keyCode == SOFTKEY1){
        myeventsMode = MYEVENTS_MENU ;
      }
    }
    // if focus on MYEVENTS_DETAIL_EVENT2
    if (myeventsdetailMode == MYEVENTS_DETAIL_EVENT2){
      if (keyCode == SOFTKEY1){
        myeventsMode = MYEVENTS_MENU ;
      }
    }
    // if focus on MYEVENTS_DETAIL-EVENT3
    if (myeventsdetailMode == MYEVENTS_DETAIL_EVENT3){
      if (keyCode == SOFTKEY1){
        myeventsMode = MYEVENTS_MENU ;
      }
    }
  }
}

void keyPressedSectionTorch(){
  torchKeyIsPressed = true;
  if(torchMode == TORCH_OPENING){
    torchKeyPressedOpening();
  }
  else if(torchMode == TORCH_INFO){
    torchKeyPressedInfo();
  }
  else if(torchMode == TORCH_MAP){
    torchKeyPressedMap();
  }
}

boolean torchKeyIsPressed = false;

void torchKeyPressedOpening(){
  if(keyCode == SOFTKEY1){
    sectionMode =  SECTION_MAINMENU;
  }
  else if(keyCode == SOFTKEY2){
    torchMode = TORCH_INFO;
  }
}
void torchKeyPressedInfo(){
  if(keyCode == SOFTKEY1){
    torchMode = TORCH_OPENING;
  }
  else if(keyCode == SOFTKEY2){
    torchMode = TORCH_MAP;
  }
}

void torchKeyPressedMap(){
  if (keyCode == '0'){
    mapArrowsEnabled = !mapArrowsEnabled;
  }
  if(torchMapMode == TORCH_MAP_STAGE0){
    if (keyCode == UP){
      torchMapMode = TORCH_MAP_STAGE1;
    }
  }
  else if(torchMapMode == TORCH_MAP_STAGE1){
    if(keyCode == DOWN){
      torchMapMode = TORCH_MAP_STAGE0;
    }
    else if (keyCode == UP){
      torchMapMode = TORCH_MAP_STAGE2;
    }
  }
  else if(torchMapMode == TORCH_MAP_STAGE2){
    if(keyCode == DOWN){
      torchMapMode = TORCH_MAP_STAGE1;
    }
    else if (keyCode == '*'){
      torchMapMode = TORCH_MAP_STAGE3;
    }
  }
  else if(torchMapMode == TORCH_MAP_STAGE3){
    if(keyCode == DOWN){
      torchMapMode = TORCH_MAP_STAGE2;
    }
    else if (keyCode == UP){
      torchMapMode = TORCH_MAP_STAGE4;
    }
  }
  else if(torchMapMode == TORCH_MAP_STAGE4){
    if(keyCode == DOWN){
      torchMapMode = TORCH_MAP_STAGE3;
    }
    else if (keyCode == UP){
    }
  }
}


void keyPressed(){

  if(sectionMode == SECTION_LOGO){
    keyPressedLogo();
  }
  else if(sectionMode == SECTION_MAINMENU){
    keyPressedSectionMainmenu();
  }
  else if (sectionMode == SECTION_EVENTS){
    keyPressedSectionEvents();
  }
  else if (sectionMode == SECTION_SEARCH){
    keyPressedSectionSearch();
  }
  else if (sectionMode == SECTION_MYEVENTS){
    keyPressedSectionMyevents();
  }
  else if (sectionMode == SECTION_TORCH){
    keyPressedSectionTorch();
  }
}

////////////////////////////////////////////////////////////////////
// records whether or not the key is pressed
boolean keyIsPressed = false;

void keyReleased(){
  keyIsPressed = false; // record that the key is no longer pressed
  torchKeyIsPressed = false;
}
```