Physical Computing

Rebecca Schill
Yves-Cédric Bauwelinckx

November 23, 2018

Contents

Chapter 1

Mini arcade

1.1 Code

```
#include "U8glib.h"

/* Create an instance for the SSD1306 OLED display in SPI mode
 * connection scheme:
 *   D0=sck=Pin 13
 *   D1=mosi=Pin 11
 *   CS=Pin 2
 *   DC=A0=Pin 3
 *   Reset=Pin 4
 */

U8GLIB_SSD1306_128X64 u8g(13, 11, 2, 3, 4);

const int FSR_PIN = A0;
const int FSR_PIN2 = A1;


//////////////////////////////
//////////////////////////////
//////////////////////////////
//////////////////////////////
//////////////////////////////
// MAIN ARCADE STUFF //
//////////////////////////////
//////////////////////////////
//////////////////////////////
//////////////////////////////
//////////////////////////////
```

```
// 0 = MENU, 1 = SPACE INVADER, 2 = PONG
int fase = 0;

void draw_menu() {
  u8g.drawStr(20, 20, "CHOOSE YOUR GAME");
  u8g.drawStr(5, 40, "SPACE INVADER");
  u8g.drawStr(100, 40, "PONG");
}

void play_menu() {
  if(analogRead(FSR_PIN) > 350) {
    fase = 1;
  } else if(analogRead(FSR_PIN2) > 350) {
    fase = 2;
  }
}

void setup() {

  u8g.setFont(u8g_font_6x10);
  u8g.setColorIndex(1); // Instructs the display to draw with a pixel on.
  pinMode(FSR_PIN, INPUT);
  pinMode(FSR_PIN2, INPUT);
  Serial.begin(9600);
  randomSeed(analogRead(0));
  // make first appearance of obstacle random (SI)
  respawnObstacle();
}

void loop() {

  u8g.firstPage();
  do {
    switch(fase) {
      case 0: draw_menu(); break;
      case 1: draw_SI(); break;
      case 2: draw_pong(); break;
    }
  } while( u8g.nextPage() );
    switch(fase) {
      case 0: play_menu(); break;
      case 1: play_SI(); break;
      case 2: play_pong(); break;
  }
}
```

```
////////////////////////////////
////////////////////////////////
////////////////////////////////
////////////////////////////////
////////////////////////////////
// SI //
////////////////////////////////
////////////////////////////////
////////////////////////////////
////////////////////////////////
////////////////////////////////


const uint8_t rocket_left[]U8G_PROGMEM =
{ 0x80, 0x18F, 0x6F, 0x1F, 0x1F, 0x3F, 0x1F, 0x1F, 0x6F, 0x18F, 0x80};
const uint8_t rocket_right[] U8G_PROGMEM =
{0x0, 0xC0, 0xF0, 0xFC, 0xEE, 0xF3, 0xEE, 0xFC, 0xF0, 0xC0, 0x0};


const int obs_w = 12;
const int obs_h = 11;
int obs_x = 120;
int obs_y = 25;
int max_speed = 6;
const uint8_t comet_left[] U8G_PROGMEM
{ 0x1F, 0x3F, 0x5E, 0x77, 0xA4, 0xA5, 0x66, 0x72, 0xFD, 0xBF, 0x5F };
const uint8_t comet_right[] U8G_PROGMEM
{ 0x20, 0x90, 0xC0, 0x60, 0xA0, 0xE0, 0xE0, 0xA0, 0xE0, 0xC0, 0x0 };


const uint8_t fighter_left[] U8G_PROGMEM
{ 0x20, 0x40, 0xC0, 0xC7, 0xCF, 0xF8, 0xF8, 0xCF, 0xC7, 0xC0, 0x40, 0x20};
const uint8_t fighter_right[] U8G_PROGMEM
{ 0x20, 0x10, 0x18, 0x18, 0x98, 0xF8, 0xF8, 0x98, 0x18, 0x18, 0x10, 0x20};


int user_x = 20;
int user_y = 20;

// -10 default for putting it of screen
int projectile_x = 200;
int projectile_y = 200;
int projectile_w = 4;
boolean proj_av = true;
```

```
//hitbox of rocket
int HB_0[] = {user_x, user_y};
int HB_1[] = {user_x+9, user_y+1};
int HB_2[] = {user_x+15, user_y+5};
int HB_3[] = {user_x, user_y+10};
int HB_4[] = {user_x+9, user_y+9};

int pass_nb = 0;
int score = 0;

// after how many seconds the game should speed up
int speedup_time = 5;

boolean game_over_SI = false;
int game_over_timer = 0;

void play_SI() {
  hitDetection_SI();

  if(analogRead(FSR_PIN2) > 200) {
    shoot();
  }

  int speed_y = calculateUserSpeed();
  int box_speed = getBoxSpeed(pass_nb);
  moveBox(-box_speed,0);
  moveUser(0, speed_y);
  moveProjectile(2,0);
  if(game_over_SI) {
    game_over_timer++;
    if((analogRead(FSR_PIN) > 200 ||
       analogRead(FSR_PIN2) > 200) &&
       game_over_timer > 20){
        reset_SI();
      // give time for menu to show up
      delay(500);
    }
  } else {score += box_speed;}

  delay(20);
  pass_nb++;
}

void draw_SI(){
  if(game_over_SI) {
```

```
    u8g.drawStr(35, 32, "GAME OVER");
    update_score(52, 42);
  } else {
    // normal game
    u8g.drawBitmapP(user_x, user_y, 1, 11, rocket_left);
    u8g.drawBitmapP(user_x + 8, user_y, 1, 11, rocket_right);
    // obstacle sprite
    u8g.drawBitmapP(obs_x, obs_y, 1, 11, fighter_left);
    u8g.drawBitmapP(obs_x + 8, obs_y, 1, 11, fighter_right);

    // projectile
    u8g.drawLine(projectile_x, projectile_y,
    projectile_x+projectile_w, projectile_y);
    update_score(80, 10);
  }
}


void update_score(int x, int y) {
  enum {BufSize=12}; // If a is short use a smaller number, eg 5 or 6
  char buf[BufSize];
  snprintf (buf, BufSize, "%d", score);
  u8g.drawStr(x, y, buf);
}


void moveUser(int sx, int sy) {
  user_x += sx;
  user_y = bound_y(user_y - sy, 11);
  update_hitbox();
}

void update_hitbox() {
  HB_0[0] = user_x;
  HB_1[0] = user_x+9;
  HB_2[0] = user_x+15;
  HB_3[0] = user_x;
  HB_4[0] = user_x+9;

  HB_0[1] = user_y;
  HB_1[1] = user_y+1;
  HB_2[1] = user_y+5;
  HB_3[1] = user_y+10;
  HB_4[1] = user_y+9;
}
```

```
void moveBox(int sx, int sy) {
  obs_x += sx;
  obs_y += sy;

  if (obs_x < 0) {
    respawnObstacle();
    pass_nb++;
  }
}

void moveProjectile(int sx, int sy) {
  projectile_x += sx;
  projectile_y += sy;
  if (projectile_x > 127) {
    projectile_x = -10000000;
    proj_av = true;
  }
}

void respawnObstacle() {
  obs_x = 127;
  obs_y = random(0, 63-obs_h);
}


void shoot() {
  if(!proj_av) {
    return;
  }
  projectile_x = user_x + 16;
  projectile_y = user_y + 6;
  proj_av = false;
}

// works with coordinates at bottom
int bound_y_triangle(int y, int height) {
  if (y > 63) {
    y = 63;
  } else if(y < height) {
    y = height;
  }
  return y;
}

// works with coordinates at top
int bound_y(int y, int height) {
```

```
  if (y > 63-height) {
    y = 63 - height;
  } else if(y < 0) {
    y = 0;
  }
  return y;
}

boolean pointInBox(int point[], int box_x_upperleft,
int box_y_upperleft, int box_width, int box_height) {
  int x = point[0];
  int y = point[1];
  if (
      (x >= box_x_upperleft && x <= (box_x_upperleft + box_width))
      &&
      (y >= box_y_upperleft && y <= (box_y_upperleft + box_height))
      ) {
    return true;
  }
  return false;
}

void hitDetectionUser() {
  // hitdetection user and obstacle
  if (pointInBox(HB_0, obs_x, obs_y, obs_w, obs_h) ||
      pointInBox(HB_1, obs_x, obs_y, obs_w, obs_h) ||
      pointInBox(HB_2, obs_x, obs_y, obs_w, obs_h) ||
      pointInBox(HB_3, obs_x, obs_y, obs_w, obs_h) ||
      pointInBox(HB_4, obs_x, obs_y, obs_w, obs_h)){
        game_over_SI = true;
        Serial.println("crashed");
  }
}

void hitDetectionProjectile() {
    // hitdetection projectile and obstacle
  int proj_coo[] = {projectile_x+projectile_w, projectile_y};
  if (pointInBox(proj_coo, obs_x, obs_y, obs_w, obs_h)) {
    respawnObstacle();
    proj_av = true;
    projectile_x = -100000;
    score += 100;
  }
}

void hitDetection_SI() {
```

```
  hitDetectionUser();
  hitDetectionProjectile();
}

int getBoxSpeed(int pass_nb) {
  return min(+1 + (pass_nb/(speedup_time*50)), +max_speed);
}

int calculateUserSpeed() {
  int pressure = analogRead(FSR_PIN);
  int speed_y = pressure/150;

  if(analogRead(FSR_PIN) < 200) {
    speed_y = -(200 - pressure)/100;
  }
  return speed_y;
 }

void reset_SI() {

  respawnObstacle();

  user_x = 20;
  user_y = 20;

  // -10 default for putting it of screen
  projectile_x = 200;
  projectile_y = 200;
  proj_av = true;

  pass_nb = 0;
  score = 0;

  // after how many seconds the game should speed up
  speedup_time = 5;
  game_over_SI = false;
  game_over_timer = 0;
  fase = 0;
};


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

```
////////////////////////////////////////////////////////////////
// PONG //
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////

// player variables
int player1_x = 5;
int player1_y = 30;

int player2_x = 115;
int player2_y = 30;

const int player_h = 15;
const int player_w = 3;

int ball_x = 60;
int ball_y = 30;
const int ball_side = 3;
int ball_speed_x = 1;
int ball_speed_y = 1;

boolean player1_wins = false;
boolean player2_wins = false;
boolean game_over_pong = false;

void play_pong() {
  // player1 controls
  int player1_speed = calculatePlayerSpeed(FSR_PIN);
  int player2_speed = calculatePlayerSpeed(FSR_PIN2);

  movePlayer1(0, player1_speed);
  movePlayer2(0, player2_speed);
  moveBall();

  if(game_over_pong) {
    game_over_timer++;
    if((analogRead(FSR_PIN) > 200 ||
        analogRead(FSR_PIN2) > 200)
        &&
        game_over_timer > 30) {
          reset_pong();
          // give time for menu to show up
```

```
            delay(500);
        }
    }

    delay(7);
}

void draw_pong() {
    if(player1_wins) {
        u8g.drawStr(27, 32, "PLAYER 1 WINS");
    } else if (player2_wins) {
        u8g.drawStr(27, 32, "PLAYER 2 WINS");
    } else {
        // normal situation
        u8g.drawBox(player1_x, player1_y, player_w, player_h);
        u8g.drawBox(player2_x, player2_y, player_w, player_h);
        u8g.drawBox(ball_x, ball_y, ball_side, ball_side);
    }
}


void movePlayer1(int sx, int sy) {
    player1_x += sx;
    player1_y = bound_y_player(player1_y + sy, player_h);
}

void movePlayer2(int sx, int sy) {
    player2_x += sx;
    player2_y = bound_y_player(player2_y + sy, player_h);
}

void moveBall() {
    bound_x_ball();
    bound_y_ball();
}


// works with coordinates at top
int bound_y_player(int y, int height) {
    if (y > 63-height) {
        y = 63 - height;
    } else if(y < 0) {
        y = 0;
    }
    return y;
}
```

```
void bound_x_ball() {
  if(hitDetection_pong()) {
    ball_speed_x = - ball_speed_x;
  } else if (ball_x < 2) {
    player2_wins = true;
    game_over_pong = true;
  } else if (ball_x > 127) {
    player1_wins = true;
    game_over_pong = true;
  }
  ball_x += ball_speed_x;
}

void bound_y_ball() {
  if(ball_y > 63 - ball_side) {
    ball_y = 63 - ball_side;
    ball_speed_y = -ball_speed_y;
  } else if (ball_y < ball_side) {
    ball_y = ball_side;
    ball_speed_y = -ball_speed_y;
  } ball_y += ball_speed_y;
}

boolean hitDetection_pong() {
  return hitDetection_Player1() ||
  hitDetection_Player2();
}

boolean hitDetection_Player1() {
  return pointInBox(ball_x, ball_y + ball_side,
  player1_x, player1_y, player_w, player_h) ||
         pointInBox(ball_x, ball_y,
         player1_x, player1_y, player_w, player_h);
}

boolean hitDetection_Player2() {
  boolean result =
  pointInBox(ball_x+ball_side, ball_y + ball_side,
  player2_x, player2_y, player_w, player_h) ||
  pointInBox(ball_x+ball_side, ball_y,
  player2_x, player2_y, player_w, player_h);
  return result;
}

boolean pointInBox(int x, int y, int box_x_upperleft, int box_y_upperleft, int box_wi
```

```
  if (
      (x >= box_x_upperleft && x <= (box_x_upperleft + box_width))
      &&
      (y >= box_y_upperleft && y <= (box_y_upperleft + box_height))
      ) {
        return true;
  }
  return false;
}

int calculatePlayerSpeed(int PIN) {

  int pressure = analogRead(PIN);
  int speed_y = -1;

  if(analogRead(PIN) < 200) {
    speed_y = 1;
  }
  return speed_y;
}

void reset_pong() {
    // player variables
    player1_x = 5;
    player1_y = 30;

    player2_x = 115;
    player2_y = 30;

    ball_x = 60;
    ball_y = 30;
    ball_speed_x = 1;
    ball_speed_y = 1;

    player1_wins = false;
    player2_wins = false;
    game_over_pong = false;
    game_over_timer = 0;
    fase = 0;
}
```