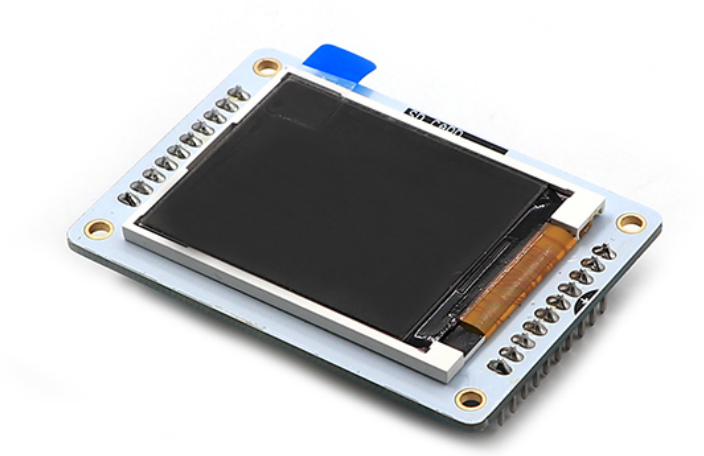
ARDUINO ESPLORA LCD

User Manual



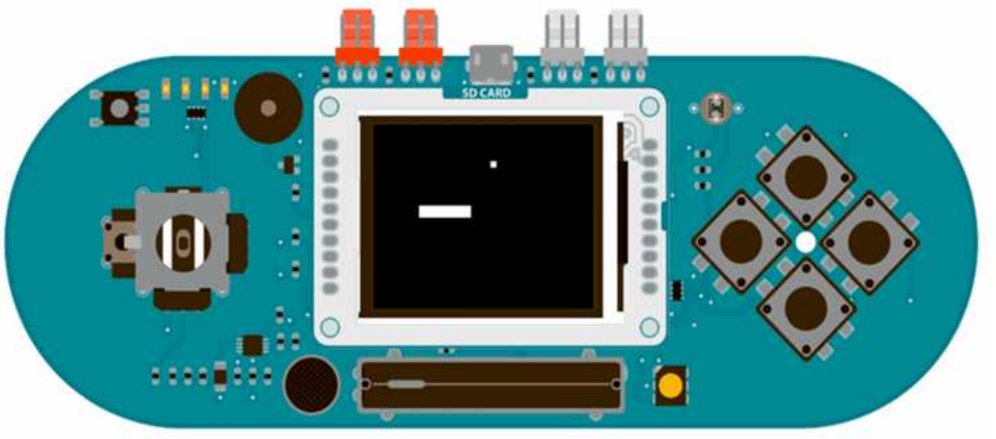
Esplora TFT Pong:

This sketch is a very basic implementation of pong for the TFT screen with an Arduino Esplora. This version of the game creates a rectangular platform that can move in two directions, and a ball that bounces against the edges of the screen as well as the movable platform. The slider on the Esplora controls the speed of the ball bouncing.

The example demonstrates collision detection between objects on the screen, as well as how to quickly update images without erasing the entire screen every loop()

Hardware Required:

- Arduino Esplora
- Arduino TFT screen



Circuit:

Attach the TFT screen to the socket on your Esplora, with the label "SD Card" facing up.

Code:

To use the screen you must first include the SPI and TFT libraries. Don't forget to include the Esplora library as well.

```
#include <Esplora.h>
#include <TFT.h>
#include <SPI.h>
```

Set up the variables for the ball and paddle x & y positions, the ball's direction, and the previous locations of the ball and paddle.

```
int paddleX = 0;
int paddleY = 0;
int oldPaddleX, oldPaddleY;
int ballDirectionX = 1;
int ballDirectionY = 1;

int ballX, ballY, oldBallX, oldBallY;
```

In setup(), start serial communication, initialize the display and clear the screen's background.

```
void setup() {
  Serial.begin(9600);
  // initialize the display
  EsploraTFT.begin();
  // set the background the black
  EsploraTFT.background(0,0,0);
}
```

loop() will hold the code for reading the joystick position, erasing the paddle's previous position, and drawing it in it's new location.

```
void loop() {
  // save the width and height of the screen
  int myWidth = EsploraTFT.width();
  int myHeight = EsploraTFT.height();

  // map the paddle's location to the joystick's position
  paddleX = map(Esplora.readJoystickX(), 512, -512, 0, myWidth) - 20/2;
  paddleY = map(Esplora.readJoystickY(), -512, 512, 0, myHeight) - 5/2;
  Serial.print(paddleX);
  Serial.print(" ");
  Serial.println(paddleY);

  // set the fill color to black and erase the previous
  // position of the paddle if different from present
  EsploraTFT.fill(0,0,0);
  if (oldPaddleX != paddleX || oldPaddleY != paddleY) {
    EsploraTFT.rect(oldPaddleX, oldPaddleY, 20, 5);
  }
```

```
  // draw the paddle on screen, save the current position
  // as the previous.
  EsploraTFT.fill(255,255,255);
  EsploraTFT.rect(paddleX, paddleY, 20, 5);
```

Save the paddle's current location as the previous location, so the next time through you can check if it has moved.

```
oldPaddleX = paddleX;
  oldPaddleY = paddleY;
```

At the end of loop(), read the slider's position to determine the speed of the ball. You'll call a custom function namedmoveBall() to update the ball's position.

```
int ballSpeed = map(Esplora.readSlider(), 0, 1023, 0, 80)+1;
  if (millis() % ballSpeed < 2) {
    moveBall();
  }
}
```

moveBall() will update the ball's position, erase its previous location, and draw it in the new spot. It will also check to make sure it does not go off the screen, reversing direction when it hits the sides. This also calls a second custom function named inPaddle() which checks for intersections of the ball and paddle.

```
void moveBall() {
  if (ballX > EsploraTFT.width() || ballX < 0) {
    ballDirectionX = -ballDirectionX;
  }
  if (ballY > EsploraTFT.height() || ballY < 0) {
    ballDirectionY = -ballDirectionY;
  }
  if (inPaddle(ballX, ballY, paddleX, paddleY, 20, 5)) {
    ballDirectionY = -ballDirectionY;
  }

  ballX += ballDirectionX;
  ballY += ballDirectionY;

  EsploraLCD.fill(0,0,0);

  if (oldBallX != ballX || oldBallY != ballY) {
    EsploraTFT.rect(oldBallX, oldBallY, 5, 5);
  }

  EsploraLCD.fill(255,255,255);
  EsploraLCD.rect(ballX, ballY, 5, 5);

  oldBallX = ballX;
  oldBallY = ballY;
}
```

inPaddle() check to see if the paddle and ball occupy the same space. If so, it returns TRUE, which reverses the ball's direction in moveBall().

```
boolean inPaddle(int x, int y, int rectX, int rectY, int rectWidth, int rectHeight) {
  boolean result = false;

  if ((x >= rectX && x <= (rectX + rectWidth)) &&
    (y >= rectY && y <= (rectY + rectHeight))) {
    result = true;
  }
  return result;
}
```

The complete sketch is below :

CODES ARE FOUND IN SOFTWARE, so no need to copy them

```
/*

  Esplora TFT Pong

  This example for the Esplora with an Arduino TFT screen reads
  the value of the joystick to move a rectangular platform
  on the x and y axes. The platform can intersect with a ball
  causing it to bounce. The Esplora's slider adjusts the speed
  of the ball.

 This example code is in the public domain.
 Created by Tom Igoe December 2012
 Modified 15 April 2013 by Scott Fitzgerald

 http://www.arduino.cc/en/Tutorial/EsploraTFTPong

 */

#include <Esplora.h>
#include <TFT.h>          // Arduino LCD library
#include <SPI.h>

// variables for the position of the ball and paddle
int paddleX = 0;
int paddleY = 0;
int oldPaddleX, oldPaddleY;
int ballDirectionX = 1;
int ballDirectionY = 1;

int ballX, ballY, oldBallX, oldBallY;

void setup() {
  Serial.begin(9600);
  // initialize the display
  EsploraTFT.begin();
  // set the background the black
  EsploraTFT.background(0, 0, 0);
}
```

```
void loop() {
  // save the width and height of the screen
  int myWidth = EsploraTFT.width();
  int myHeight = EsploraTFT.height();

  // map the paddle's location to the joystick's position
  paddleX = map(Esplora.readJoystickX(), 512, -512, 0, myWidth) - 20 / 2;
  paddleY = map(Esplora.readJoystickY(), -512, 512, 0, myHeight) - 5 / 2;
  Serial.print(paddleX);
  Serial.print(" ");
  Serial.println(paddleY);

  // set the fill color to black and erase the previous
  // position of the paddle if different from present
  EsploraTFT.fill(0, 0, 0);

  if (oldPaddleX != paddleX || oldPaddleY != paddleY) {
    EsploraTFT.rect(oldPaddleX, oldPaddleY, 20, 5);
  }

  // draw the paddle on screen, save the current position
  // as the previous.
  EsploraTFT.fill(255, 255, 255);
  EsploraTFT.rect(paddleX, paddleY, 20, 5);
  oldPaddleX = paddleX;
  oldPaddleY = paddleY;

  // read the slider to determinde the speed of the ball
  int ballSpeed = map(Esplora.readSlider(), 0, 1023, 0, 80) + 1;
  if (millis() % ballSpeed < 2) {
    moveBall();
  }
}


// this function determines the ball's position on screen
void moveBall() {
  // if the ball goes offscreen, reverse the direction:
  if (ballX > EsploraTFT.width() || ballX < 0) {
    ballDirectionX = -ballDirectionX;
  }


  if (ballY > EsploraTFT.height() || ballY < 0) {
    ballDirectionY = -ballDirectionY;
  }

  // check if the ball and the paddle occupy the same space on screen
  if (inPaddle(ballX, ballY, paddleX, paddleY, 20, 5)) {
    ballDirectionY = -ballDirectionY;
  }
  // update the ball's position
  ballX += ballDirectionX;
  ballY += ballDirectionY;
```

```
  // erase the ball's previous position
  EsploraTFT.fill(0, 0, 0);

  if (oldBallX != ballX || oldBallY != ballY) {
    EsploraTFT.rect(oldBallX, oldBallY, 5, 5);
  }

  // draw the ball's current position
  EsploraTFT.fill(255, 255, 255);

  EsploraTFT.rect(ballX, ballY, 5, 5);

  oldBallX = ballX;
  oldBallY = ballY;

}

// this function checks the position of the ball
// to see if it intersects with the paddle
boolean inPaddle(int x, int y, int rectX, int rectY, int rectWidth, int rectHeight) {
  boolean result = false;
  if ((x >= rectX && x <= (rectX + rectWidth)) &&
      (y >= rectY && y <= (rectY + rectHeight))) {
    result = true;
  }

  return result;
}
```

Esplora TFT Snake:

```
/*

  Esplora TFT Snake

  this is an open source version of the classic snake game.

  Press one of the switches to change the direction of the

  snake and move the slider to make the game faster or slower.

  The score is in the upper right corner of the screen.

  You earn 9 point for each beacon the snake eats. Enjoy :)

*/

// first of all we need to include the libraries

#include <Esplora.h>

#include <TFT.h>          // Arduino LCD library

#include <SPI.h>


// then we have to choice the resolution of our game.
const int scale = 4;

int xScreen = EsploraTFT.width()/scale;

int yScreen = EsploraTFT.height()/scale;
```

```
// some global variables
const int snakeInitialLength = 12;
const int snakeMaxLength = 300; // max length of the snake
int snakeLength = snakeInitialLength; // length of the snake
int positions[2][snakeMaxLength]; // positions of the snake pieces
int beacon[2]; // poition of the beacon
int directionSnake = 0; // the direction where the snake moves
unsigned long lastTimeMoved; // last time in wich time snake moved
unsigned long score = (snakeLength - snakeInitialLength) * 9; //score
unsigned long lastScore = score; // score at the previous time
const int scorePosition = 135; // position of the score in the screen

void setup() {
 // we need to initialize the display
 EsploraTFT.begin();
 // and to select the color of the background
 EsploraTFT.background(0,0,0); // black for an higher contrast

 // initialize the position of the snake
 const int startX = 8; // position where the snake starts
 const int startY = 5;
 // draw these positions
 for(int ii = 0; ii < snakeLength; ii++) {
  positions[0][ii] = startX + ii;
  positions[1][ii] = startY;
  // draw some boxes to figure where is the snake
  EsploraTFT.fill(255, 255, 255); // drain in white
  EsploraTFT.rect(scale*positions[0][ii], scale*positions[1][ii],
   scale, scale); // the snake dimension are proportional to scale
}
delay(2000);
for(int jj = 0; jj < snakeLength ; jj++) {
 Serial.println(positions[0][jj]);
 Serial.println(positions[1][jj]);
 Serial.println();
}
```

```
 // to make the position of the beacon random
 // use random seed with temperature
 randomSeed((long)Esplora.readTemperature(DEGREES_C)*100);
 // place the beacon
 placeBeacon();
 Serial.println(beacon[0]);
Serial.println(beacon[1]);
// write the score
EsploraTFT.stroke(255, 255, 255);
// convert score into String
String scoreString = String(score);
// convert string into char array
char charScore [4];
scoreString.toCharArray(charScore, 4);
// print it
EsploraTFT.text(charScore, scorePosition, 0);
EsploraTFT.noStroke();
}
void loop() {
 // the snake moves accordingly to the value read from potentiometer
 int slider = Esplora.readSlider();
 // map this value in a value of time
 unsigned int timeDelay = map(slider, 0, 1023, 50, 1000);
 if(millis() - lastTimeMoved > timeDelay) {
  // let the snake move
  moveTheSnake();
  lastTimeMoved = millis();
 }
 // read the buttons to change directions
 if (Esplora.readButton(SWITCH_DOWN) == LOW && directionSnake != 3) {
  // if the snake is going up it can't go down
  // so we use that AND funciotn
  directionSnake = 1;
 }
if (Esplora.readButton(SWITCH_LEFT) == LOW && directionSnake != 0) {
 // if the snake is going up it can't go down
 // so we use that AND funciotn
 directionSnake = 2;
```

```
}
if (Esplora.readButton(SWITCH_UP) == LOW && directionSnake != 1) {
  // if the snake is going up it can't go down
  // so we use that AND funciotn
  directionSnake = 3;
}
if (Esplora.readButton(SWITCH_RIGHT) == LOW && directionSnake != 2) {
  // if the snake is going up it can't go down
  // so we use that AND funciotn
  directionSnake = 0;
}

// see it we hurt another part of the snake
for(int kk = 0; kk < snakeLength - 1 ; kk++) {
  if(positions[0][kk] == positions[0][snakeLength - 1] &&
    positions[1][kk] == positions[1][snakeLength - 1]) {
      // draw a blue blackground
      EsploraTFT.background(0, 100, 150);
      // write in white
      EsploraTFT.stroke(255, 255, 255);
      // select the text size
      EsploraTFT.setTextSize(2);
      // "you Lose"
      EsploraTFT.text("You Lose", 0, 0);
      EsploraTFT.text("Press Reset", 0, 30);
      EsploraTFT.text("To Restart", 0, 60);
      // stop the game
      while(true) {
      }
    }
}

// update the score
score = (snakeLength - snakeInitialLength) * 9;
// if it is different from previous one
```

```
  if(score != lastScore) {
    // delete the old one
    EsploraTFT.stroke(0, 0, 0);
    // convert score into String
    String scoreString = String(lastScore);
    // convert string into char array
    char charScore [4];
    scoreString.toCharArray(charScore, 4);
    // print it
    EsploraTFT.text(charScore, scorePosition, 0);
    // print the new one
    EsploraTFT.stroke(255, 255, 255);
    // convert score into String
    scoreString = String(score);
    // convert string into char array
    scoreString.toCharArray(charScore, 4);
    // print it
    EsploraTFT.text(charScore, scorePosition, 0);
    // save the value of last score
    lastScore = score;
    // we don't want stroke no more
    EsploraTFT.noStroke();
  }


  // a little pause for stability
  delay(10);
}

void moveTheSnake() {
  // move the snake according to the direction
  switch(directionSnake) {
    case 0: { // move right
      if(beaconIsEaten()) { // if we ate the beacon
        snakeLength++; // the snake increases in length
```

```
    positions[0][snakeLength - 1] =
      (positions[0][snakeLength - 2] + 1) % xScreen;
    // we use the modulus so when the snake goes out from a part
    // it returns in the opposite one
    // the y is the same as before
    positions[1][snakeLength - 1] = positions[1][snakeLength - 2];
    // draw the new head of the snake
    drawSnakeHead();
   } else {
    // update positions
    updatePositions();
    // move also the terminal of the snake
    positions[0][snakeLength - 1] =
      (positions[0][snakeLength - 1] + 1) % xScreen;
    // draw the new head
    drawSnakeHead();
   }
   break;
  }

  case 1: { // move down
   if(beaconIsEaten()) { // if we ate the beacon
    snakeLength++; // the snake increases in length
    positions[1][snakeLength - 1] =
      (positions[1][snakeLength - 2] + 1) % yScreen;
    positions[0][snakeLength - 1] = positions[0][snakeLength - 2];
    // draw the new head of the snake
    drawSnakeHead();
   } else {
    // update positions
    updatePositions();
    // move also the terminal of the snake
    positions[1][snakeLength - 1] =
      (positions[1][snakeLength - 1] + 1) % yScreen;
    // draw the new head
```

```
    drawSnakeHead();
   }
   break;
  }

 case 2: { // move left
   if(beaconIsEaten()) { // if we ate the beacon
     snakeLength++; // the snake increases in length
     positions[0][snakeLength - 1] =
       (positions[0][snakeLength - 2] - 1);
     // if it goes utside replace on the other part
     if(positions[0][snakeLength - 1] <= 0) {
       positions[0][snakeLength - 1] = xScreen - 1;
     }
     positions[1][snakeLength - 1] = positions[1][snakeLength - 2];
     // draw the new head of the snake
     drawSnakeHead();
   } else {
     // update positions
     updatePositions();
     // move also the terminal of the snake
     positions[0][snakeLength - 1] =
       (positions[0][snakeLength - 1] - 1);
     if(positions[0][snakeLength - 1] <= 0) {
       positions[0][snakeLength - 1] = xScreen - 1;
     }
     // draw the new head
     drawSnakeHead();
   }
   break;
  }

 case 3: { // move up
   if(beaconIsEaten()) { // if we ate the beacon
     snakeLength++; // the snake increases in length
```

```
      positions[1][snakeLength - 1] =
        (positions[1][snakeLength - 2] - 1);
      if(positions[1][snakeLength - 1] <= 0) {
        positions[1][snakeLength - 1] = yScreen - 1;
      }
      positions[0][snakeLength - 1] = positions[0][snakeLength - 2];
      // draw the new head of the snake
      drawSnakeHead();
    } else {
      // update positions
      updatePositions();
      // move also the terminal of the snake
      positions[1][snakeLength - 1] =
        (positions[1][snakeLength - 1] - 1);
      if(positions[1][snakeLength - 1] <= 0) {
        positions[1][snakeLength - 1] = yScreen - 1;
      }
      // draw the new head
      drawSnakeHead();
    }
    break;
  }
 }
}

boolean beaconIsEaten() {
 // this function returns true if beacon is eated and
 // false if it is not
 if(positions[0][snakeLength - 1] == beacon[0] &&
  positions[1][snakeLength - 1] == beacon[1]) {
  Serial.println("Beacon Eated");
  // put the beacon in another place
  placeBeacon();
  return true;
 } else {
```

```
    return false;
  }
}

void updatePositions() {
   // debugging
  Serial.println("Snake is Moving");
  for(int jj = 0; jj < snakeLength ; jj++) {
   Serial.println(positions[0][jj]);
   Serial.println(positions[1][jj]);
   Serial.println();
  }
  // this function update the position of the snake except the
  // head of the snake, which position is user choice dependent
  // make the snake move
  // first we have to delete the tail of the snake
  EsploraTFT.fill(0, 0, 0); // the same color of the background
  EsploraTFT.rect(scale * positions[0][0], scale * positions[1][0],
    scale, scale);
  // next update positions
  for(int ii = 0; ii < snakeLength - 1; ii++) {
    // update the positions
    positions[0][ii] = positions[0][ii + 1];
    positions[1][ii] = positions[1][ii + 1];
  }
}

void drawSnakeHead() {
  // this function draw the new head of the snake
  EsploraTFT.fill(255, 255, 255);
  EsploraTFT.rect(scale*positions[0][snakeLength - 1],
    scale*positions[1][snakeLength - 1], scale, scale);
}

void placeBeacon() {
```

```
  // this function place the beacon where
  // there is not a snake part
  // repeat while we don't break it
  while (true) {
   // define an int for verification
   int flag = 0;
   // don't put the beacon on the external of the screen
   const int tollerance = 5;
   // locate the beacon in a random place
   beacon[0] = (int) random(tollerance, xScreen - tollerance);
   beacon[1] = (int) random(tollerance, yScreen - tollerance);
   // find if a part of the snake is on the beacon
   for(int jj = 0; jj < snakeLength; jj++) {
    // if they have the same x
    if(beacon[0] == positions[0][jj]) {
     // cheack the y
     if(beacon[1] == positions[1][jj]) {
      // increment the flag
      flag++;
     }
    }
   }
   // if the flag is 0 we don't have the beacon over
   // a part of the snake
   if(flag == 0) {
    // so we can go away
    break;
   }
  }

  // draw the beacon
  EsploraTFT.fill(0, 255, 0);
  EsploraTFT.rect(scale * beacon[0], scale * beacon[1], scale, scale);
}
```

How to open Software:

- Enter to http://www.ekt2.com/products/productdetails?ProductId=E5E3CE8B-683F-45A7-8BAC-534B680E2D0F

- Press the icon to start the download