```
// BMP-loading example specifically for the TFTLCD breakout board.
// If using the Arduino shield, use the tftbmp_shield.pde sketch instead!
// If using an Arduino Mega make sure to use its hardware SPI pins, OR make
// sure the SD library is configured for 'soft' SPI in the file Sd2Card.h.

// Writing by JC slatkin - smartpoker.jimdo.com - French WebSite
// version : 1.0
// date : 30/12/2017
// Hardware : Arduino Uno + Screen TFT 2.4
// Description :  Reproduce a 3D effect on a Button as Windows


// If you want more memory, add comment in  "void debug_message()"
// 65% f global variable busy with debug_message without comment
// 52 % with debug_message with comment

#include <Adafruit_GFX.h>    // Core graphics library
#include <TftSpfd5408.h> // Hardware-specific library
#include <TouchScreen.h>
#include <SD.h>
#include <SPI.h>

// The control pins for the LCD can be assigned to any digital or
// analog pins...but we'll use the analog pins as this allows us to
// double up the pins with the touch screen (see the TFT paint example).
#define LCD_CS A3 // Chip Select goes to Analog 3
#define LCD_CD A2 // Command/Data goes to Analog 2
#define LCD_WR A1 // LCD Write goes to Analog 1
#define LCD_RD A0 // LCD Read goes to Analog 0

// When using the BREAKOUT BOARD only, use these 8 data lines to the LCD:
// For the Arduino Uno, Duemilanove, Diecimila, etc.:
//   D0 connects to digital pin 8  (Notice these are
//   D1 connects to digital pin 9   NOT in order!)
//   D2 connects to digital pin 2
//   D3 connects to digital pin 3
//   D4 connects to digital pin 4
//   D5 connects to digital pin 5
//   D6 connects to digital pin 6
//   D7 connects to digital pin 7
// For the Arduino Mega, use digital pins 22 through 29
// (on the 2-row header at the end of the board).

// For Arduino Uno/Duemilanove, etc
//  connect the SD card with DI going to pin 11, DO going to pin 12 and SCK going to pin 13
(standard)
//  Then pin 10 goes to CS (or whatever you have set up)
#define SD_CS 10     // Set the chip select line to whatever you use (10 doesnt conflict with the
library)

// In the SD card, place 24 bit color BMP files (be sure they are 24-bit!)
// There are examples in the sketch folder
```

```
// our TFT wiring
TftSpfd5408 tft(LCD_CS, LCD_CD, LCD_WR, LCD_RD, A4);

// Assign human-readable names to some common 16-bit color values:
#define BLACK 0x0000
#define BLUE 0x001F
#define RED 0xF800
//#define GREEN 0x07E0
//#define CYAN 0x07FF
//#define MAGENTA 0xF81F
//#define YELLOW 0xFFE0
#define WHITE 0xFFFF
#define GREY1 0xEF7D
#define GREY2 0xBDD7

// ***** Define the background color and the inverse color
int background_color=BLACK;
int inverse_background_color=WHITE;

#define MINPRESSURE 10
#define MAXPRESSURE 1000

#define BOXSIZE 40
#define PENRADIUS 3
int oldcolor, currentcolor;
// valeur usuellement constaté lors des calibrations 150,930
// X correspond au coté le plus petit de l'ecran
// Y correspond au coté le plus grand de l'ecran
#define TS_MINX 150
#define TS_MINY 150
#define TS_MAXX 930
#define TS_MAXY 930

//position de l'image+dimension de l'image après fonction de mapping
int BOX_H=26;
int BOX_W=96;
int BOXX_MIN=1;
int BOXY_MIN=1;
// ajoute des pixels au controle du toucher car certains boutons peuvent
// etre trop petit
#define BOXSEC_X 10
#define BOXSEC_Y 10


#define YP A3 // must be an analog pin, use "An" notation!
#define XM A2 // must be an analog pin, use "An" notation!
#define YM 9 // can be a digital pin
#define XP 8 // can be a digital pin
TouchScreen ts = TouchScreen(XP, YP, XM, YM, 300);

boolean Onthebutton=false;
```

```
boolean previous=false;
boolean debug=true;
/* Mode debug - convention name
 *  On the console, name : value means name is not calculated
 *  name = value  means name is calculated
 */

 int rotation=0;  // default value
 int H1,H2,W1,W2;
int PRESSURE_PRECISION=10;
int L1,L2;

char filename[18]="btn_ok2.bmp";
int  bmpWidth, bmpHeight;   // W+H in pixels

void setup()
{
  Serial.begin(9600);
  tft.reset();
  tft.begin(0x9341);
  Serial.print(F("Initializing SD card..."));
  if (!SD.begin(SD_CS)) {
    Serial.println(F("failed!"));
    return;
  }
  Serial.println(F("OK!"));

  // the rotation change the localisation of the image
  // Rotation =0, image in Min_x=25 Max_x=75 , Min_y=8, Max_x=75, Max_y=150
  // Rotation=1 image in Min_x=250, max_x=300, Min_y=15,mac_y=75
   // Rotation=2 image in Min_x=204, max_x=300, Min_y=230,mac_y=250
   // Rotation=3 image in Min_x=8, max_x=75, Min_y=190,mac_y=250
  // These values depends on the shape of the button
  // to Determine theses values, use the debug mode and tap on the screen

  // change this value if you want
  rotation=0;


  tft.setRotation(rotation);
  tft.fillScreen(background_color);
  bmpDraw(filename, 0, 0);
  BOX_W = bmpWidth;
  BOX_H = bmpHeight;
  delay(200);
  Border3D_OFF();
  //BOXX_MIN, et BOX_H doivent etre ajusté en fonction de la valeur de SetRotation
  // je dessine 4 traits pour creer un effet 3 D et je changerais la couleur pour l'effet enfoncé
  /*  0,0------------------BOX_W-10,0 (White or Black) ( 2ndtrait )
      |  1,1-------------BOX_W-11   (gris clair ou gris foncé) (3nd trait)
      |   |
      |   |
```

```
    |  |
    |  1,BOX_H-11 (3eme trait)
    0,BOX_H-10 (1er trait)


  // DrawFastVLine(origin_x,origin_y,lenght, color)
  // how react this fonction with Setrotation ?


     tft.drawFastVLine(BOXX_MIN,BOXY_MIN, BOX_H-2,RED); //1er trait
     tft.drawFastHLine(BOXY_MIN,BOXY_MIN, BOX_W-10,RED); //2nd trait
     tft.drawFastVLine(BOXX_MIN+1,BOXY_MIN+1, BOX_H-3,RED); //3eme trait
     tft.drawFastHLine(BOXY_MIN+1,BOXY_MIN+1, BOX_W-11,RED); //4eme trait

 // Later when i will be above the bmp, i will just change the color
*/
  message_debug(1);
} //End Setup

void loop()
{

digitalWrite(13, HIGH);
 TSPoint p = ts.getPoint();
 digitalWrite(13, LOW);
 // if sharing pins, you'll need to fix the directions of the touchscreen pins
 //pinMode(XP, OUTPUT);
 pinMode(XM, OUTPUT);
 pinMode(YP, OUTPUT);
 //pinMode(YM, OUTPUT);
 // we have some minimum pressure we consider 'valid'
 // pressure of 0 means no pressing!
 if (p.z > MINPRESSURE && p.z < MAXPRESSURE)
 {
 // scale from TS_MIN, TS_MAX to tft.width, tft.height
    p.x = map(p.x, TS_MINX, TS_MAXX, tft.width(), 0);
    p.y = map(p.y, TS_MINY, TS_MAXY, tft.height(), 0);


  if (rotation==0)
  {
    H1=0;
    H2=BOX_H+2*PRESSURE_PRECISION;
    W1=0;
    W2=BOX_W+PRESSURE_PRECISION; // Test OK
  }
   if (rotation==1)
  {
    H1=0;
    H2=BOX_W;
    W1=tft.width()- (BOX_H+2*PRESSURE_PRECISION);
    W2=tft.width();
```

```
  }
  if (rotation==2)
  {
    H1=tft.height()-(+PRESSURE_PRECISION+BOX_H);
    H2=tft.height()+PRESSURE_PRECISION;
    W1=tft.width()-(+PRESSURE_PRECISION+BOX_W);
    W2=tft.width(); // Test OK

  }
     if (rotation==3)
  {
    H1=tft.height()-BOX_W;
    H2=tft.height()+PRESSURE_PRECISION;
    W1=0;
    W2=BOX_H+2*PRESSURE_PRECISION; // Test OK
  }
  BOXX_MIN=0;
  BOXY_MIN=0;
  L1=BOX_W-5;
  L2=BOX_H-1;

  message_debug(2);

if (p.y > H1 && p.y <H2)
{
   if (debug)
     {
      Serial.println("Condition 1 OK");
     }
   if (p.x > W1 && p.x <W2)
     {
     // je suis quelquepart au dessus du BMP
     // je change la couleur

     if (previous==false)
        { Border3D_ON();
          previous=true;
        }
     Onthebutton=true;
     message_debug(3);
     } //endif
} // sinon je ne suis pas sur le boutton
else
{
  if (Onthebutton)
  {
     // je sors du boutton, je remet a l'etat initial
     /*tft.drawFastHLine(BOXX_MIN,BOXY_MIN, L1,BLUE);
     tft.drawFastHLine(BOXX_MIN+1,BOXY_MIN+1, L1-1,BLUE);
     tft.drawFastVLine(BOXX_MIN,BOXY_MIN, L2,BLUE);
     tft.drawFastVLine(BOXX_MIN+1,BOXY_MIN+1,L2-1,BLUE);
     */
```

```
      Onthebutton=false;
      if (previous==true)
         { Border3D_OFF();
           previous=false;
         }
      if (debug) Serial.println("Onthebutton=false");
    } // end if
   // sinon j'etais deja hors du boutton, donc je ne fais rien

 } //end else
 }//End if  (p.z > MINPRESSURE && p.z < MAXPRESSURE
} // end loop

// This function opens a Windows Bitmap (BMP) file and
// displays it at the given coordinates.  It's sped up
// by reading many pixels worth of data at a time
// (rather than pixel by pixel).  Increasing the buffer
// size takes more of the Arduino's precious RAM but
// makes loading a little faster.  20 pixels seems a
// good balance.

#define BUFFPIXEL 20

void bmpDraw(char *filename, int x, int y) {

  File     bmpFile;
//  int      bmpWidth, bmpHeight;   // W+H in pixels
  uint8_t  bmpDepth;              // Bit depth (currently must be 24)
  uint32_t bmpImageoffset;        // Start of image data in file
  uint32_t rowSize;               // Not always = bmpWidth; may have padding
  uint8_t  sdbuffer[3*BUFFPIXEL]; // pixel in buffer (R+G+B per pixel)
  uint16_t lcdbuffer[BUFFPIXEL];  // pixel out buffer (16-bit per pixel)
  uint8_t  buffidx = sizeof(sdbuffer); // Current position in sdbuffer
  boolean  goodBmp = false;       // Set to true on valid header parse
  boolean  flip    = true;        // BMP is stored bottom-to-top
  int      w, h, row, col;
  uint8_t  r, g, b;
  uint32_t pos = 0, startTime = millis();
  uint8_t  lcdidx = 0;
  boolean  first = true;

  if((x >= tft.width()) || (y >= tft.height())) return;

  Serial.println();
  Serial.print(F("Loading image '"));
  Serial.print(filename);
  Serial.println('\'');
  // Open requested file on SD card
  bmpFile = SD.open(filename);

  // Parse BMP header
```

```
if(read16(bmpFile) == 0x4D42) { // BMP signature
  Serial.println(F("File size: ")); Serial.println(read32(bmpFile));
  (void)read32(bmpFile); // Read & ignore creator bytes
  bmpImageoffset = read32(bmpFile); // Start of image data
  Serial.print(F("Image Offset: ")); Serial.println(bmpImageoffset, DEC);
  // Read DIB header
  Serial.print(F("Header size: ")); Serial.println(read32(bmpFile));
  bmpWidth  = read32(bmpFile);
  bmpHeight = read32(bmpFile);
  if(read16(bmpFile) == 1) { // # planes -- must be '1'
    bmpDepth = read16(bmpFile); // bits per pixel
    Serial.print(F("Bit Depth: ")); Serial.println(bmpDepth);
    if((bmpDepth == 24) && (read32(bmpFile) == 0)) { // 0 = uncompressed

      goodBmp = true; // Supported BMP format -- proceed!
      Serial.print(F("Image size: "));
      Serial.print(bmpWidth);
      Serial.print('x');
      Serial.println(bmpHeight);

      // BMP rows are padded (if needed) to 4-byte boundary
      rowSize = (bmpWidth * 3 + 3) & ~3;

      // If bmpHeight is negative, image is in top-down order.
      // This is not canon but has been observed in the wild.
      if(bmpHeight < 0) {
        bmpHeight = -bmpHeight;
        flip      = false;
      }

      // Crop area to be loaded
      w = bmpWidth;
      h = bmpHeight;
      if((x+w-1) >= tft.width())  w = tft.width()  - x;
      if((y+h-1) >= tft.height()) h = tft.height() - y;

      // Set TFT address window to clipped image bounds
      tft.setAddrWindow(x, y, x+w-1, y+h-1);

      for (row=0; row<h; row++) { // For each scanline...
        // Seek to start of scan line.  It might seem labor-
        // intensive to be doing this on every line, but this
        // method covers a lot of gritty details like cropping
        // and scanline padding.  Also, the seek only takes
        // place if the file position actually needs to change
        // (avoids a lot of cluster math in SD library).
        if(flip) // Bitmap is stored bottom-to-top order (normal BMP)
          pos = bmpImageoffset + (bmpHeight - 1 - row) * rowSize;
        else     // Bitmap is stored top-to-bottom
          pos = bmpImageoffset + row * rowSize;
        if(bmpFile.position() != pos) { // Need seek?
          bmpFile.seek(pos);
```

```
          buffidx = sizeof(sdbuffer); // Force buffer reload
        }

        for (col=0; col<w; col++) { // For each column...
          // Time to read more pixel data?
          if (buffidx >= sizeof(sdbuffer)) { // Indeed
            // Push LCD buffer to the display first
            if(lcdidx > 0) {
              tft.pushColors(lcdbuffer, lcdidx, first);
              lcdidx = 0;
              first  = false;
            }
            bmpFile.read(sdbuffer, sizeof(sdbuffer));
            buffidx = 0; // Set index to beginning
          }

          // Convert pixel from BMP to TFT format
          b = sdbuffer[buffidx++];
          g = sdbuffer[buffidx++];
          r = sdbuffer[buffidx++];
          lcdbuffer[lcdidx++] = tft.color565(r,g,b);
        } // end pixel
      } // end scanline
      // Write any remaining data to LCD
      if(lcdidx > 0) {
        tft.pushColors(lcdbuffer, lcdidx, first);
      }
      message_debug(4);
    } // end goodBmp
  }
 }

 bmpFile.close();
 if(!goodBmp) Serial.println(F("BMP format not recognized."));
}

// These read 16- and 32-bit types from the SD card file.
// BMP data is stored little-endian, Arduino is little-endian too.
// May need to reverse subscript order if porting elsewhere.

uint16_t read16(File f) {
 uint16_t result;
 ((uint8_t *)&result)[0] = f.read(); // LSB
 ((uint8_t *)&result)[1] = f.read(); // MSB
 return result;
}

uint32_t read32(File f) {
 uint32_t result;
 ((uint8_t *)&result)[0] = f.read(); // LSB
 ((uint8_t *)&result)[1] = f.read();
 ((uint8_t *)&result)[2] = f.read();
```

```
  ((uint8_t *)&result)[3] = f.read(); // MSB
  return result;
}

void Border3D_OFF()
{

      tft.drawFastHLine(BOXX_MIN,BOXY_MIN, L1,inverse_background_color);
      tft.drawFastHLine(BOXX_MIN+1,BOXY_MIN+1,L1-1,inverse_background_color);
      tft.drawFastHLine(BOXX_MIN+2,BOXY_MIN+2, L1,GREY1);
      tft.drawFastVLine(BOXX_MIN,BOXY_MIN,L2-1,inverse_background_color);
      tft.drawFastVLine(BOXX_MIN,BOXY_MIN, L2,inverse_background_color);
      tft.drawFastVLine(BOXX_MIN+1,BOXY_MIN+1,L2-1,GREY1);

}

void Border3D_ON()
{

tft.drawRect(BOXX_MIN,BOXY_MIN,BOXX_MIN+L1,BOXY_MIN+L2,background_color);
      tft.drawRect(BOXX_MIN+1,BOXY_MIN+1,BOXX_MIN+L1-1,BOXY_MIN+L2-
1,background_color);
      tft.drawRect(BOXX_MIN+2,BOXY_MIN+2,BOXX_MIN+L1-2,BOXY_MIN+L2-2,GREY2);
}

void message_debug(int i)
 {
   // Add */ here if you want more memory
 /*
  if (debug)
    {
      switch (i)
       {
       case 1 :
                Serial.print (" filename : ");
                Serial.println (filename);
                Serial.print (" Rotation :");
                Serial.println(rotation);
                Serial.println(" Dimension de l'ecran");
                Serial.print ("width  :"  );
                Serial.println (tft.width());
                Serial.print ("height :  ");
                Serial.println(tft.height());
                Serial.print ("BOXY_MIN : ");
                Serial.println(BOXY_MIN);
                Serial.print ("BOX_H : ");
                Serial.println(BOX_H);
                Serial.print ("BOXX_MIN : ");
                Serial.println(BOXX_MIN);
                Serial.print ("BOX_W : ");
                Serial.println(BOX_W);
                Serial.print (" Pressure Précision : ");
```

```
            Serial.println (PRESSURE_PRECISION);
            break;
      case 2 :
            Serial.print ("W1,W2 = ");
            Serial.print (W1);
            Serial.print (" , ");
            Serial.print (W2);
            Serial.print ( " - Valeur mesurée de P.x = ");
            Serial.println(p.x);
            Serial.print("H1,H2 = ");
            Serial.print (" , ");
            Serial.print (H1);
            Serial.print (" , ");
            Serial.print (H2);
            Serial.print ( " - Valeur mesurée de P.y = ");
            Serial.println(p.y);
          break;
       case 3 :
            Serial.println("Onthebutton=true");
            Serial.print (" BOXX_MIN BOXYMIN : ");
            Serial.print (BOXX_MIN);
            Serial.print (" ," );
            Serial.println (BOXY_MIN);
            Serial.print ( "L1 (HLine), L2 (VLine) : ");
            Serial.print ( L1);
            Serial.print (" , ");
           Serial.println(L2);
          break;
      case 4 :
            Serial.print(F("Loaded in "));
            Serial.print(millis() - startTime);
            Serial.println(" ms");
            break;
      default : Serial.println (" Mode debug activé");
      } // end Switch


   } // End if debug
 // Add /* here if you want more memory
*/
 } //End Message debug
```