```c
 * simonDisplay.c

#include <stdio.h>
#include "simonDisplay.h"
#include "supportFiles/display.h"
#include "supportFiles/utils.h"

//**********CODE FROM PROFESSOR**********//
#define TOUCH_PANEL_ANALOG_PROCESSING_DELAY_IN_MS 60 // in ms
#define MAX_STR 255
#define TEXT_SIZE 2
#define TEXT_VERTICAL_POSITION 0
#define TEXT_HORIZONTAL_POSITION (DISPLAY_HEIGHT/2)
#define INSTRUCTION_LINE_1 "Touch and release to start the Simon demo."
#define INSTRUCTION_LINE_2 "Demo will terminate after %d touches."
#define DEMO_OVER_MESSAGE_LINE_1 "Simon demo terminated"
#define DEMO_OVER_MESSAGE_LINE_2 "after %d touches."
#define TEXT_VERTICAL_POSITION 0 // Start at the far left.
#define ERASE_THE_SQUARE true  // drawSquare() erases if this is passed in.
#define DRAW_THE_SQUARE false  // drawSquare() draws the square if this is passed in.

//***********MY_VAR**********//
#define DIVIDE_HALF 2 // too cut the screen width and height in half
#define CENTER_BUTTEN_X 50 //offset from the side in the x direction
#define CENTER_BUTTEN_Y 30 //offset from the side in the y direction
#define BOX_HEIGHT 160  // the height of the box that flashes
#define BOX_WIDTH 120       //the width of the bow that flashed
#define HOME_POSITION 0     //the start postion 0,0




int8_t simonDisplay_computeRegionNumber(int16_t x, int16_t y){
    if(x < (DISPLAY_WIDTH/DIVIDE_HALF)){             //see if screen was touched on the left half
        if(y < (DISPLAY_HEIGHT/DIVIDE_HALF)){        //see if screen was touched on the top half
            return SIMON_DISPLAY_REGION_0;           //return button 0
        }
        else{                                       //screen was touched on the the bottom half
            return SIMON_DISPLAY_REGION_2;          //return button 2
        }
    }
    else{                                           //screen was touched on the right half
        if(y < (DISPLAY_HEIGHT/DIVIDE_HALF)){       //see if screen was touched on the top half
            return SIMON_DISPLAY_REGION_1;          //return button 1
        }
        else{                                       //screen was touched on the the bottom half
            return SIMON_DISPLAY_REGION_3;          //return button 3
        }
    }
}

// Draws a colored "button" that the user can touch.
```

```
58 // The colored button is centered in the region but does not fill the region.
59 void simonDisplay_drawButton(uint8_t regionNumber){
60     switch (regionNumber){
61     case SIMON_DISPLAY_REGION_0: // fills button #0 in with red
62         display_fillRect(CENTER_BUTTEN_X, CENTER_BUTTEN_Y, SIMON_DISPLAY_BUTTON_WIDTH,
   SIMON_DISPLAY_BUTTON_HEIGHT, DISPLAY_RED);
63         break;                       // exits the case
64     case SIMON_DISPLAY_REGION_1: // fills button #1 in with yellow
65         display_fillRect((DISPLAY_WIDTH/DIVIDE_HALF)+CENTER_BUTTEN_X, CENTER_BUTTEN_Y,
   SIMON_DISPLAY_BUTTON_WIDTH, SIMON_DISPLAY_BUTTON_HEIGHT, DISPLAY_YELLOW);
66         break;                       // exits the case
67     case SIMON_DISPLAY_REGION_2: // fills button #2 in with blue
68         display_fillRect(CENTER_BUTTEN_X,
   (DISPLAY_HEIGHT/DIVIDE_HALF)+CENTER_BUTTEN_Y, SIMON_DISPLAY_BUTTON_WIDTH,
   SIMON_DISPLAY_BUTTON_HEIGHT, DISPLAY_BLUE);
69         break;                       // exits the case
70     case SIMON_DISPLAY_REGION_3: // fills button #3 in with green
71         display_fillRect((DISPLAY_WIDTH/DIVIDE_HALF)+CENTER_BUTTEN_X,
   (DISPLAY_HEIGHT/DIVIDE_HALF)+CENTER_BUTTEN_Y, SIMON_DISPLAY_BUTTON_WIDTH,
   SIMON_DISPLAY_BUTTON_HEIGHT, DISPLAY_GREEN);
72         break;                       // exits the case
73     }
74 };
75 // Convenience function that draws all of the buttons.
76 void simonDisplay_drawAllButtons(){
77     simonDisplay_drawButton(SIMON_DISPLAY_REGION_0); // draw the button in position #0
78     simonDisplay_drawButton(SIMON_DISPLAY_REGION_1); // draw the button in position #1
79     simonDisplay_drawButton(SIMON_DISPLAY_REGION_2); // draw the button in position #2
80     simonDisplay_drawButton(SIMON_DISPLAY_REGION_3); // draw the button in position #3
81 };
82 // Convenience function that erases all of the buttons.
83 void simonDisplay_eraseAllButtons(){ //uses the same code from simonDisplay_drawButton
   but changes the color to black
84     display_fillRect(CENTER_BUTTEN_X, CENTER_BUTTEN_Y, SIMON_DISPLAY_BUTTON_WIDTH,
   SIMON_DISPLAY_BUTTON_HEIGHT, DISPLAY_BLACK);
85     display_fillRect((DISPLAY_WIDTH/DIVIDE_HALF)+CENTER_BUTTEN_X, CENTER_BUTTEN_Y,
   SIMON_DISPLAY_BUTTON_WIDTH, SIMON_DISPLAY_BUTTON_HEIGHT, DISPLAY_BLACK);
86     display_fillRect(CENTER_BUTTEN_X, (DISPLAY_HEIGHT/DIVIDE_HALF)+CENTER_BUTTEN_Y,
   SIMON_DISPLAY_BUTTON_WIDTH, SIMON_DISPLAY_BUTTON_HEIGHT, DISPLAY_BLACK);
87     display_fillRect((DISPLAY_WIDTH/DIVIDE_HALF)+CENTER_BUTTEN_X,
   (DISPLAY_HEIGHT/DIVIDE_HALF)+CENTER_BUTTEN_Y, SIMON_DISPLAY_BUTTON_WIDTH,
   SIMON_DISPLAY_BUTTON_HEIGHT, DISPLAY_BLACK);
88 };
89
90 // Draws a bigger square that completely fills the region.
91 // If the erase argument is true, it draws the square as black background to "erase"
   it.
92 void simonDisplay_drawSquare(uint8_t regionNo, bool erase){
93     if (erase){
94         switch (regionNo){
95         case SIMON_DISPLAY_REGION_0:   // fills box #0 in with black
96             display_fillRect(HOME_POSITION, HOME_POSITION, BOX_HEIGHT, BOX_WIDTH,
   DISPLAY_BLACK);
97             break;                      // exits the case
98         case SIMON_DISPLAY_REGION_1:   // fills box #1 in with black
99             display_fillRect(BOX_HEIGHT, HOME_POSITION, BOX_HEIGHT, BOX_WIDTH,
   DISPLAY_BLACK);
100            break;                      // exits the case
```

```
101         case SIMON_DISPLAY_REGION_2:   // fills box #2 in with black
102             display_fillRect(0, BOX_WIDTH, BOX_HEIGHT, BOX_WIDTH, DISPLAY_BLACK);
103             break;                      // exits the case
104         case SIMON_DISPLAY_REGION_3:   // fills box #3 in with black
105             display_fillRect(BOX_HEIGHT, BOX_WIDTH,BOX_HEIGHT, BOX_WIDTH,
    DISPLAY_BLACK);
106             break;                      // exits the case
107         }
108     }
109     else{ // if the erase is false then fill the box with the right color
110         switch (regionNo){
111         case SIMON_DISPLAY_REGION_0:   // fills box #0 in with red
112             display_fillRect(HOME_POSITION, HOME_POSITION, BOX_HEIGHT, BOX_WIDTH,
    DISPLAY_RED);
113             break;                      // exits the case
114         case SIMON_DISPLAY_REGION_1:   // fills box #1 in with yellow
115             display_fillRect(BOX_HEIGHT, HOME_POSITION, BOX_HEIGHT, BOX_WIDTH,
    DISPLAY_YELLOW);
116             break;                      // exits the case
117         case SIMON_DISPLAY_REGION_2:   // fills box #2 in with blue
118             display_fillRect(HOME_POSITION, BOX_WIDTH, BOX_HEIGHT, BOX_WIDTH,
    DISPLAY_BLUE);
119             break;                      // exits the case
120         case SIMON_DISPLAY_REGION_3:   // fills box #3 in with green
121             display_fillRect(BOX_HEIGHT, BOX_WIDTH,BOX_HEIGHT, BOX_WIDTH,
    DISPLAY_GREEN);
122             break;                      // exits the case
123         }
124     }
125 };
126
127 // Runs a brief demonstration of how buttons can be pressed and squares lit up to
    implement the user
128 // interface of the Simon game. The routine will continue to run until the touchCount
    has been reached, e.g.,
129 // the user has touched the pad touchCount times.
130
131 // I used a busy-wait delay (utils_msDelay) that uses a for-loop and just blocks until
    the time has passed.
132 // When you implement the game, you CANNOT use this function as we discussed in class.
    Implement the delay
133 // using the non-blocking state-machine approach discussed in class.
134 void simonDisplay_runTest(uint16_t touchCount){
135     display_init();         // Always initialize the display.
136     char str[MAX_STR];      // Enough for some simple printing.
137     uint8_t regionNumber = 0;   // Convenience variable.
138     uint16_t touches = 0;   // Terminate when you receive so many touches.
139     // Write an informational message and wait for the user to touch the LCD.
140     display_fillScreen(DISPLAY_BLACK);              // clear the screen.
141     display_setCursor(TEXT_VERTICAL_POSITION, TEXT_HORIZONTAL_POSITION); // move to
    the middle of the screen.
142     display_setTextSize(TEXT_SIZE);                 // Set the text size for the
    instructions.
143     display_setTextColor(DISPLAY_RED, DISPLAY_BLACK);   // Reasonable text color.
144     sprintf(str, INSTRUCTION_LINE_1);                   // Copy the line to a buffer.
145     display_println(str);                               // Print to the LCD.
146     display_println();                                  // new-line.
147     sprintf(str, INSTRUCTION_LINE_2, touchCount);       // Copy the line to a buffer.
```

```
148     display_println(str);                                   // Print to the LCD.
149     while (!display_isTouched());       // Wait here until the screen is touched.
150     while (display_isTouched());        // Now wait until the touch is released.
151     display_fillScreen(DISPLAY_BLACK);  // Clear the screen.
152     simonDisplay_drawAllButtons();      // Draw all of the buttons.
153     bool touched = false;         // Keep track of when the pad is touched.
154     int16_t x, y;                     // Use these to keep track of coordinates.
155     uint8_t z;                        // This is the relative touch pressure.
156     while (touches < touchCount) {  // Run the loop according to the number of touches
    passed in.
157         if (!display_isTouched() && touched) {          // user has stopped touching
    the pad.
158             simonDisplay_drawSquare(regionNumber, ERASE_THE_SQUARE);  // Erase the
    square.
159             simonDisplay_drawButton(regionNumber);        // DISPLAY_REDraw the
    button.
160             touched = false;                  // Released the touch, set touched to
    false.
161         }
162         else if (display_isTouched() && !touched) {   // User started touching the
    pad.
163             touched = true;                               // Just touched the pad, set
    touched = true.
164             touches++;                                    // Keep track of the number
    of touches.
165             display_clearOldTouchData();                  // Get rid of data from
    previous touches.
166             // Must wait this many milliseconds for the chip to do analog processing.
167             utils_msDelay(TOUCH_PANEL_ANALOG_PROCESSING_DELAY_IN_MS);
168             display_getTouchedPoint(&x, &y, &z);          // After the wait, get the
    touched point.
169             regionNumber = simonDisplay_computeRegionNumber(x, y);// Compute the
    region number, see above.
170             simonDisplay_drawSquare(regionNumber, DRAW_THE_SQUARE);  // Draw the
    square (erase = false).
171         }
172     }
173     // Done with the demo, write an informational message to the user.
174     display_fillScreen(DISPLAY_BLACK);        // clear the screen.
175     // Place the cursor in the middle of the screen.
176     display_setCursor(TEXT_VERTICAL_POSITION, TEXT_HORIZONTAL_POSITION);
177     display_setTextSize(TEXT_SIZE); // Make it readable.
178     display_setTextColor(DISPLAY_RED, DISPLAY_BLACK);  // red is foreground color,
    black is background color.
179     sprintf(str, DEMO_OVER_MESSAGE_LINE_1);    // Format a string using sprintf.
180     display_println(str);                      // Print it to the LCD.
181     sprintf(str, DEMO_OVER_MESSAGE_LINE_2, touchCount);  // Format the rest of the
    string.
182     display_println(str);  // Print it to the LCD.
183     };
184
185
186
```