```c
/*
 * simonDisplay.c
 *
 *  Created on: Jun 4, 2015
 *      Author: Taylor Cowley
 */

#include "simonDisplay.h"

#define TOUCH_PANEL_ANALOG_PROCESSING_DELAY_IN_MS 60 // in ms
#define MAX_STR 255
#define TEXT_SIZE 2

//Return which region is correlated with that x and y value set
int8_t simonDisplay_computeRegionNumber(int16_t x, int16_t y){
      if (x < 0 || y < 0)       //if either x or y is negative
        return -1;              //the region does not exist
      if (x < SIMON_DISPLAY_HALF_WIDTH){        //we are in the left half
          if (y < SIMON_DISPLAY_HALF_HEIGHT)        //we are on top
              return SIMON_DISPLAY_REGION_0;        //top left
          else                                      //we are on bottom
              return 2;                             //bottom left
      } else {                                    //we are in the right half
          if (y < SIMON_DISPLAY_HALF_HEIGHT)        //we are on top
              return 1;                             //top right
          else                                      //we are on bottom
              return 3;                             //bottom right
      }
}

// Draws a colored "button" that the user can touch.
// The colored button is centered in the region but does not fill the region.
void simonDisplay_drawButton(uint8_t regionNumber){
    // API: fillRect(x, y, width, height)
    switch(regionNumber) {                  //button 0-4
    case SIMON_DISPLAY_BUTTON_2:            //BOTTOM LEFT
        display_fillRect(SIMON_DISPLAY_FOURTH_WIDTH -   SIMON_DISPLAY_BUTTON_WIDTH_HALF,    //t
  op left of button
                SIMON_DISPLAY_THREE_FOURTH_HEIGHT   -   SIMON_DISPLAY_BUTTON_WIDTH_HALF,
      //top left of button
                SIMON_DISPLAY_BUTTON_WIDTH,         //size of button
                SIMON_DISPLAY_BUTTON_HEIGHT,        //size of button
                DISPLAY_BLUE);              //BUTTON COLOR
        break;
    case SIMON_DISPLAY_BUTTON_0:            //TOP LEFT
        display_fillRect(SIMON_DISPLAY_FOURTH_WIDTH -   SIMON_DISPLAY_BUTTON_WIDTH_HALF,    //t
  op left of button
                SIMON_DISPLAY_FOURTH_HEIGHT         -   SIMON_DISPLAY_BUTTON_WIDTH_HALF,    //t
  op left of button
                SIMON_DISPLAY_BUTTON_WIDTH,         //size of button
                SIMON_DISPLAY_BUTTON_HEIGHT,        //size of button
                DISPLAY_RED);               //BUTTON COLOR
        break;
    case SIMON_DISPLAY_BUTTON_1:            //TOP RIGHT
        display_fillRect(SIMON_DISPLAY_THREE_FOURTH_WIDTH   -   SIMON_DISPLAY_BUTTON_WIDTH_HAL
  F,  //top left of button
                SIMON_DISPLAY_FOURTH_HEIGHT                 -   SIMON_DISPLAY_BUTTON_WIDTH_HAL
```

```
  F,  //top left of button
53                SIMON_DISPLAY_BUTTON_WIDTH,         //size of button
54                SIMON_DISPLAY_BUTTON_HEIGHT,        //size of button
55                DISPLAY_YELLOW);            //BUTTON COLOR
56        break;
57    case SIMON_DISPLAY_BUTTON_3:            //BOTTOM RIGHT
58        display_fillRect(SIMON_DISPLAY_THREE_FOURTH_WIDTH   -   SIMON_DISPLAY_BUTTON_WIDTH_HAL
  F,  //top left of button
59                SIMON_DISPLAY_THREE_FOURTH_HEIGHT           -   SIMON_DISPLAY_BUTTON_WIDTH_HAL
  F,  //top left of button
60                SIMON_DISPLAY_BUTTON_WIDTH,         //size of button
61                SIMON_DISPLAY_BUTTON_HEIGHT,        //size of button
62                DISPLAY_GREEN);             //BUTTON COLOR
63        break;
64    default:                                //THis is an error
65        printf("We can't draw a button! Trying to draw button %d\n\r",
  regionNumber);  //print the error
66        break;
67    }
68 }
69
70 // Convenience function that draws all of the buttons.
71 void simonDisplay_drawAllButtons(){
72     simonDisplay_drawButton(SIMON_DISPLAY_BUTTON_0);    //draw button 0
73     simonDisplay_drawButton(SIMON_DISPLAY_BUTTON_1);    //draw button 1
74     simonDisplay_drawButton(SIMON_DISPLAY_BUTTON_2);    //draw button 2
75     simonDisplay_drawButton(SIMON_DISPLAY_BUTTON_3);    //draw button 3
76 }
77
78 // Draws a bigger square that completely fills the region.
79 // If the erase argument is true, it draws the square as black background to "erase" it.
80 void simonDisplay_drawSquare(uint8_t regionNo, bool erase){
81
82        // API: fillRect(x, y, width, height)
83        switch(regionNo) {                  //button 0-4
84        case SIMON_DISPLAY_BUTTON_2:            //BOTTOM LEFT
85            display_fillRect(0,                 //Far left
86                    SIMON_DISPLAY_HALF_HEIGHT,  //Halfway down
87                    SIMON_DISPLAY_HALF_WIDTH,   //Half the screen
88                    SIMON_DISPLAY_HALF_HEIGHT,  //Half the screen
89                    erase ? DISPLAY_BLACK : DISPLAY_BLUE);  //either black or BUTTON COLOR
  depending on erase
90            break;
91        case SIMON_DISPLAY_BUTTON_0:            //TOP LEFT
92            display_fillRect(0,                 //Top
93                    0,                          //Far left
94                    SIMON_DISPLAY_HALF_WIDTH,   //Half the screen
95                    SIMON_DISPLAY_HALF_HEIGHT,  //Half the screen
96                    erase ? DISPLAY_BLACK : DISPLAY_RED);   //either black or BUTTON COLOR
  depending on erase
97            break;
98        case SIMON_DISPLAY_BUTTON_1:            //TOP RIGHT
99            display_fillRect(SIMON_DISPLAY_HALF_WIDTH,  //Middle
100                   0,                                   //Top
101                   SIMON_DISPLAY_HALF_WIDTH,    //Half the screen
102                   SIMON_DISPLAY_HALF_HEIGHT,   //Half the screen
103                   erase ? DISPLAY_BLACK : DISPLAY_YELLOW);    //either black or BUTTON COLOR
```

```
   depending on erase
104             break;
105         case SIMON_DISPLAY_BUTTON_3:            //BOTTOM RIGHT
106             display_fillRect(SIMON_DISPLAY_HALF_WIDTH,  //Middle
107                     SIMON_DISPLAY_HALF_HEIGHT,          //Middle
108                     SIMON_DISPLAY_HALF_WIDTH,   //Half the screen
109                     SIMON_DISPLAY_HALF_HEIGHT,  //Half the screen
110                     erase ? DISPLAY_BLACK : DISPLAY_GREEN); //either black or BUTTON COLOR
   depending on erase
111             break;
112         default:                                //THis is an error
113             printf("We can't draw a button! Trying to draw button %d\n\r",
   regionNo);  //print the error
114             break;
115         }
116 }
117
118
119 // Runs a brief demonstration of how buttons can be pressed and squares lit up to implement
   the user
120 // interface of the Simon game. The routine will continue to run until the touchCount has been
   reached, e.g.,
121 // the user has touched the pad touchCount times.
122
123 // I used a busy-wait delay (utils_msDelay) that uses a for-loop and just blocks until the
   time has passed.
124 // When you implement the game, you CANNOT use this function as we discussed in class.
   Implement the delay
125 // using the non-blocking state-machine approach discussed in class.
126 void simonDisplay_runTest(uint16_t touchCount) {
127   display_init();  // Always initialize the display.
128   char str[MAX_STR];   // Enough for some simple printing.
129   uint8_t regionNumber;
130   uint16_t touches = 0;
131   // Write an informational message and wait for the user to touch the LCD.
132   display_fillScreen(DISPLAY_BLACK);        // clear the screen.
133   display_setCursor(0, display_height()/2); //
134   display_setTextSize(TEXT_SIZE);
135   display_setTextColor(DISPLAY_RED, DISPLAY_BLACK);
136   sprintf(str, "Touch and release to start the Simon demo.");
137   display_println(str);
138   display_println();
139   sprintf(str, "Demo will terminate after %d touches.", touchCount);
140   display_println(str);
141   while (!display_isTouched());       // Wait here until the screen is touched.
142   while (display_isTouched());        // Now wait until the touch is released.
143   display_fillScreen(DISPLAY_BLACK);  // Clear the screen.
144   simonDisplay_drawAllButtons();      // Draw all of the buttons.
145   bool touched = false;           // Keep track of when the pad is touched.
146   int16_t x, y;               // Use these to keep track of coordinates.
147   uint8_t z;                  // This is the relative touch pressure.
148   while (touches < touchCount) {  // Run the loop according to the number of touches passed
   in.
149     if (!display_isTouched() && touched) {         // user has stopped touching the pad.
150       simonDisplay_drawSquare(regionNumber, true); // Erase the square.
151       simonDisplay_drawButton(regionNumber);       // DISPLAY_REDraw the button.
152       touched = false;                                                    // Released
```

```
   the touch, set touched to false.
153     } else if (display_isTouched() && !touched) {   // User started touching the pad.
154       touched = true;                               // Just touched the pad, set touched =
   true.
155       touches++;                                                                  // Keep
   track of the number of touches.
156       display_clearOldTouchData();  // Get rid of data from previous touches.
157       // Must wait this many milliseconds for the chip to do analog processing.
158       utils_msDelay(TOUCH_PANEL_ANALOG_PROCESSING_DELAY_IN_MS);
159       display_getTouchedPoint(&x, &y, &z);                  // After the wait, get the touched
   point.
160       regionNumber = simonDisplay_computeRegionNumber(x, y);// Compute the region number.
161       simonDisplay_drawSquare(regionNumber, false);     // Draw the square (erase = false).
162     }
163   }
164   // Done with the demo, write an informational message to the user.
165   display_fillScreen(DISPLAY_BLACK);        // clear the screen.
166   display_setCursor(0, display_height()/2); // Place the cursor in the middle of the screen.
167   display_setTextSize(2);                   // Make it readable.
168   display_setTextColor(DISPLAY_RED, DISPLAY_BLACK);  // red is foreground color, black is
   background color.
169   sprintf(str, "Simon demo terminated");    // Format a string using sprintf.
170   display_println(str);                     // Print it to the LCD.
171   sprintf(str, "after %d touches.", touchCount);  // Format the rest of the string.
172   display_println(str);  // Print it to the LCD.
173 }
174
```