```c
 * simonControl.c


#include "verifySequence_runTest.h"
#include "flashSequence.h"
#include "buttonHandler.h"
#include "simonDisplay.h"
#include "simonControl.h"
#include "supportFiles/display.h"
#include "supportFiles/utils.h"
#include "globals.h"
#include <stdio.h>
#include <stdint.h>
#include "../Lab2_switch_button/buttons.h"



#define START_MSG "TOUCH TO START"                                  // text for the start message
#define SIMON_MSG "SIMON"                                           // text for the start message
#define YAY_MSG "Yay!"                                              // text for the sequence completed message
#define NEW_L_MSG "Touch for new level"                             // text for the new level message
#define LONG_MSG "Longest Sequence: "                               // text for the longest sequence

#define START_MSG_X 70                                              // coordinates for the start message x position
#define START_MSG_Y 140                                             // coordinates for the start message y position

#define SIMON_MSG_X 80                                              // coordinates for the simon message x position
#define SIMON_MSG_Y 90                                              // coordinates for the simon message y position

#define YAY_MSG_X 100                                               // coordinates for the yay message x position
#define YAY_MSG_Y 100                                               // coordinates for the yay message y position

#define NEW_L_MSG_X 50                                              // coordinates for the new level message x position
#define NEW_L_MSG_Y YAY_MSG_Y                                       // coordinates for the new level message y position

#define LONG_MSG_X 50                                               // coordinates for the longest sequence message x position
#define LONG_MSG_Y YAY_MSG_Y                                        // coordinates for the longest sequence message y position

#define NUM_MSG_X 280                                               // coordinates for the number on the longest sequence message x position
#define NUM_MSG_Y YAY_MSG_Y                                         // coordinates for the number on the longest sequence message y position

```

```
47 #define FALSE 0                                    // define False as
   0 or reset
48 #define TRUE 1                                     // define true as
   1 for raising flags
49
50 #define TEXT_SIZE_L 5                              // larger text
   size
51 #define TEXT_SIZE_S 2                              // small text size
52 #define WAIT_TIME 40                               // time to for
   messages to appear
53 #define WAIT_TIME_S 1                              // time to give an
   extra tick in a state
54 #define INIT_LEVEL 4                               // the starting
   level
55
56
57 uint8_t myArray[GLOBALS_MAX_FLASH_SEQUENCE];       // the array that
   holds the sequence
58 uint8_t currLevel = INIT_LEVEL;                    // the current
   level = the difficulty of the sequence
59 uint8_t currIter = 1;                              // always start
   the level with the first box appearing of the sequence
60 uint8_t randNum = 0;                               // number used to
   generate the random numbers
61
62 //**********COUNTERS**********/
63 uint8_t yayCounter = 0;                            // counter for yay
   message wait
64 uint8_t newLevelCounter = 0;                       // counter for new
   level message wait
65 uint8_t longestSeqCounter = 0;                     // counter for
   longest sequence message wait
66 uint8_t fdCounter = 0;                             // counter for
   flash enable wait
67 uint8_t vdCounter = 0;                             // counter for
   verify enable wait
68
69 //**********FLAGS**********/
70 uint8_t initFlag = 0;                              // flag for when
   the start screen needs to be printed
71 uint8_t nextFlag = 0;                              // flag to send
   the SM on to the next level with out reseting
72
73
74
75
76 enum control_st_m{
77     control_init_st,                               // the
   initializing state
78     control_touch_st,                              // the first touch
   state off of the start screen state
79     control_setIterLength_st,                      // the state that
   sets the iteration length for that level state
80     control_flash_enable_st,                       // the enable
   flash sequence state
81     control_flash_disable_st,                      // the disable
   flash sequence state
82     control_verify_enable,                         // the enable
```

```
   verify sequence state
 83    control_verify_disable,                                  // the disable
   verify sequence state
 84    control_yay_st,                                          // the print yay
   message state
 85    control_new_level_st,                                    // the print new
   level message state
 86    control_longest_sq_st                                    // the print
   longest sequence message state
 87}controlCurrent = control_init_st;                           // initializes the
   first state to the control_init_st
 88
 89void randNumGen(){                                           // function that
   generates a random sequence
 90    srand(randNum);                                          // seeds the
   number generator
 91    for(uint8_t i = 0; i < currLevel; i++){                  // for loop to
   fill the array with random nmbers
 92        myArray[i] = rand() % (INIT_LEVEL);                  // filling the
   array
 93    }
 94}
 95
 96void simonControl_tick(){                                    // Simon control
   state machine
 97    switch(controlCurrent){
 98    case control_init_st:                                    // Moore state
   action for state #1
 99        if(!initFlag){
100            display_setCursor(START_MSG_X, START_MSG_Y);      // set the cursor
   for the text
101            display_setTextColor(DISPLAY_WHITE);             // set the color for
   the text
102            display_setTextSize(TEXT_SIZE_S);                // set the text
   size to small
103            display_println(START_MSG);                      // print the start
   message
104            display_setCursor(SIMON_MSG_X, SIMON_MSG_Y);     // set the cursor
   for the text
105            display_setTextColor(DISPLAY_WHITE);             // set the color for
   the text
106            display_setTextSize(TEXT_SIZE_L);                // set the text
   size to large
107            display_println(SIMON_MSG);                      // print simon on
   the screen
108            if(!initFlag){
109                initFlag = TRUE;                             // raise the flag
   saying that it has been initialized
110            }
111        }
112        randNum++;                                           // get a new
   random number for the sequence generator
113        break;
114    case control_touch_st:                                     // Moore state
   action for state #2
115        break;
116    case control_setIterLength_st:
117        globals_setSequenceIterationLength(currIter);        // set the
```

```
   iteration length to the current number number in the the array sequence
118         break;
119     case control_flash_enable_st:                                   // Moore state
   action for state #3
120         fdCounter++;                                                // increment the
   counter so the state is active for at least one tick
121         flashSequence_enable();                                       // enable the
   flash sequence SM
122         break;
123     case control_flash_disable_st:                                  // Moore state
   action for state #4
124         fdCounter = FALSE;                                            // reset the
   counter
125         flashSequence_disable();                                      // disable the
   flash sequence SM
126         break;
127     case control_verify_enable:                                     // Moore state
   action for state #5
128         vdCounter++;                                                // increment the
   counter so the state is active for at least one tick
129         verifySequence_enable();                                     // enable the
   verify seq SM
130         break;
131     case control_verify_disable:                                    // Moore state
   action for state #6
132         vdCounter = FALSE;                                            // reset the
   counter
133         verifySequence_disable();                                     // disable the
   verify seq SM
134         break;
135     case control_yay_st:                                            // Moore state
   action for state #7
136         yayCounter++;                                                 // increment
   the counter
137         break;
138     case control_new_level_st:                                      // Moore state
   action for state #8
139         newLevelCounter++;                                                  //
   increment the counter
140         break;
141     case control_longest_sq_st:                                     // Moore state
   action for state #9
142         longestSeqCounter++;                                                //
   increment the counter
143         break;
144     }
145     switch(controlCurrent){
146     case control_init_st:                                           // Mealy
   transition state action for state #1
147         if(nextFlag){
148             randNumGen();                                           // fill the
   array with random numbers
149             globals_setSequence(myArray,currLevel);                 // set the
   sequence
150             controlCurrent = control_setIterLength_st;
151         }
152         else if(display_isTouched()){
153             randNumGen();
```

```
154             globals_setSequence(myArray,currLevel);
155             controlCurrent = control_touch_st;
156         }
157         break;
158     case control_touch_st:                                      // Mealy
    transition state action for state #2
159         if(!display_isTouched()){
160             // print out the start messages
161             display_setCursor(START_MSG_X, START_MSG_Y);            // set the cursor
    for the text
162             display_setTextColor(DISPLAY_BLACK);
    // erase the text
163             display_setTextSize(TEXT_SIZE_S);                       // set the text
    size to small
164             display_println(START_MSG);
165             display_setCursor(SIMON_MSG_X, SIMON_MSG_Y);            // set the cursor
    for the text
166             display_setTextColor(DISPLAY_BLACK);
    // erase the text
167             display_setTextSize(TEXT_SIZE_L);                       // set the text
    size to large
168             display_println(SIMON_MSG);
169             controlCurrent = control_setIterLength_st;
170         }
171         break;
172     case control_setIterLength_st:                              // Mealy
    transition state action for state #3
173         controlCurrent = control_flash_enable_st;
174         break;
175     case control_flash_enable_st:                               // Mealy
    transition state action for state #4
176         if(flashSequence_isComplete()&& (fdCounter > WAIT_TIME_S)){   //if the flash
    SM is done then move to next state
177             controlCurrent = control_flash_disable_st;
178         }
179         break;
180     case control_flash_disable_st:                              // Mealy
    transition state action for state #5
181         controlCurrent = control_verify_enable;
182         break;
183     case control_verify_enable:                                 // Mealy
    transition state action for state #6
184         if(verifySequence_isComplete() && (vdCounter > WAIT_TIME_S)){ //if the verify
    sequence is done move to next state
185             controlCurrent = control_verify_disable;
186         }
187         break;
188     case control_verify_disable:                                // Mealy
    transition state action for state #7
189         if(currIter == currLevel){                                // if the
    level and the iteration match then they won
190             // print yay msg
191             display_setCursor(YAY_MSG_X, YAY_MSG_Y);            // set the cursor for
    the text
192             display_setTextColor(DISPLAY_WHITE);                // set the color for
    the text
193             display_setTextSize(TEXT_SIZE_L);                       // set the text
    size to large
```

```
194             display_println(YAY_MSG);
195             controlCurrent = control_yay_st;
196         }
197         else if(verifySequence_isTimeOutError()|| verifySequence_isUserInputError()){
    //if there was an user error of a time out error go to end
198             //print longest sequence msg
199             display_setCursor(LONG_MSG_X, LONG_MSG_Y);            // set the cursor
    for the text
200             display_setTextColor(DISPLAY_WHITE);                  // set the color for
    the text
201             display_setTextSize(TEXT_SIZE_S);                       // set the text
    size to small
202             display_println(LONG_MSG);
203             display_setCursor(NUM_MSG_X, NUM_MSG_Y);            // set the cursor for
    the text
204             display_println(currLevel);
205             controlCurrent = control_longest_sq_st;
206         }
207         else{
208             currIter++;                                         // increment the
    iteration and go throught the flash and verify again
209             controlCurrent = control_setIterLength_st;
210         }
211         break;
212     case control_yay_st:                                          // Mealy
    transition state action for state #8
213         if(yayCounter >= WAIT_TIME){                            // time to wait for
    the print yay
214             // black yay
215             display_setCursor(YAY_MSG_X, YAY_MSG_Y);            // set the cursor for
    the text
216             display_setTextColor(DISPLAY_BLACK);
    // erase the text
217             display_setTextSize(TEXT_SIZE_L);                       // set the text
    size to large
218             display_println(YAY_MSG);
219             // print new level
220             display_setCursor(NEW_L_MSG_X, NEW_L_MSG_Y);            // set the cursor
    for the text
221             display_setTextColor(DISPLAY_WHITE);                  // set the color for
    the text
222             display_setTextSize(TEXT_SIZE_S);                       // set the text
    size to small
223             display_println(NEW_L_MSG);
224             controlCurrent = control_new_level_st;
225         }
226         break;
227     case control_new_level_st:                                    // Mealy
    transition state action for state #9
228         if(newLevelCounter >= WAIT_TIME){                       // time to wait
    for the new level msg
229             // black new level
230             display_setCursor(NEW_L_MSG_X, NEW_L_MSG_Y);            // set the cursor
    for the text
231             display_setTextColor(DISPLAY_BLACK);
    // erase the text
232             display_setTextSize(TEXT_SIZE_S);                       // set the text
    size to small
```

```
233             display_println(NEW_L_MSG);
234             // print longSQ
235             display_setCursor(LONG_MSG_X, LONG_MSG_Y);             // set the cursor
    for the text
236             display_setTextColor(DISPLAY_WHITE);                   // set the color for
    the text
237             display_setTextSize(TEXT_SIZE_S);                        // set the text
    size to small
238             display_println(LONG_MSG);
239             display_setCursor(NUM_MSG_X, NUM_MSG_Y);             // set the cursor for
    the text
240             display_println(currLevel);
241             controlCurrent = control_longest_sq_st;
242         }
243         if(display_isTouched()){                                     // if screen is
    touch go to harder level
244             // black new level
245             display_setCursor(NEW_L_MSG_X, NEW_L_MSG_Y);             // set the cursor
    for the text
246             display_setTextColor(DISPLAY_BLACK);
    // erase the text
247             display_setTextSize(TEXT_SIZE_S);                        // set the text
    size to small
248             display_println(NEW_L_MSG);
249             // Initialize var
250             currLevel++;                                                 // increment
    level
251             currIter = TRUE;                                             //reset VAR
252             nextFlag = TRUE;                                            //raise the
    next flag
253             yayCounter = FALSE;                                            //reset VAR
254             newLevelCounter = FALSE;                                          //reset
    VAR
255             longestSeqCounter = FALSE;
    //reset VAR
256             controlCurrent = control_init_st;
257         }
258         break;
259     case control_longest_sq_st:                                    // Mealy
    transition state action for state #10
260         if(longestSeqCounter >= WAIT_TIME){
261             // black longSQ
262             display_setCursor(LONG_MSG_X, LONG_MSG_Y);             // set the cursor
    for the text
263             display_setTextColor(DISPLAY_BLACK);
    // erase the text
264             display_setTextSize(TEXT_SIZE_S);                        // set the text
    size to small
265             display_println(LONG_MSG);
266             display_setCursor(NUM_MSG_X, NUM_MSG_Y);             // set the cursor for
    the text
267             display_println(currLevel);
268             // Initialize var
269             currLevel = INIT_LEVEL;                                 //reset VAR
270             currIter = TRUE;                                      //reset VAR
271             initFlag = FALSE;                                      //reset VAR
272             nextFlag = FALSE;                                      //reset VAR
273             yayCounter = FALSE;                                      //reset VAR
```

```
274             newLevelCounter = FALSE;                                //reset VAR
275             longestSeqCounter = FALSE;                               //reset VAR
276             controlCurrent = control_init_st;
277         }
278         break;
279     }
280 }
281
282
```