```c
/*
 * ticTacToeControl.c
 *
 *  Created on: Jun 2, 2015
 *      Author: Taylor Cowley
 */

#include "ticTacToeControl.h"
#include "supportFiles/display.h"
#include "buttons.h"

//calculates to see if any moves have already been made
bool ticTacToeControl_boardIsEmpty(minimax_board_t *board);

//Holds the current state of the machine
ticTacToeControl_state_t currentState = init_state;


//Ticks the current state of ticTacToe and returns the next state
void ticTacToeControl_tick() {
    //holds the timer for the touch screen wait and player selection wait
    static int32_t countdown_timer;

    //Holds the current state of the board for minimax
    static minimax_board_t board;

    //tells us who is playing what
    static bool CPU_is_x;

    //First, we perform the state action
    switch(currentState){
    case init_state:        //we start here
        ticTacToeDisplay_init();    //init the screen
        minimax_initBoard(&board);  //init the board (to empty)
        ticTacToeControl_print_status("touch screen quick or you have to play O!"); //notify the
user of the rules
        break;
    case choose_players:    //we just wait...
        //We subtract 1 from the timer unless it is already 0
        countdown_timer = countdown_timer - 1;
        break;
    case choose_players_chosen: //now we wait for them to lift their finger
        //so we do nothing
        break;
    case CPU_turn:          //The computer's turn
        uint8_t row;    //where minimax will store the move
        uint8_t col;    //where minimax will store the move
        if(ticTacToeControl_boardIsEmpty(&board)){  //this is the first move
            //hard-coded first move right in the middle.
            board.squares[TICTACTOECONTROL_CENTER][TICTACTOECONTROL_CENTER] =
MINIMAX_PLAYER_SQUARE;
            //display the move on the board
            CPU_is_x ? ticTacToeDisplay_drawX(TICTACTOECONTROL_CENTER, TICTACTOECONTROL_CENTER)
                    : ticTacToeDisplay_drawO(TICTACTOECONTROL_CENTER, TICTACTOECONTROL_CENTER);
        } else {
            //computes the computer's move and gives it to us in row and col
            //true means the computer is currently playing
```

```
            minimax_computeNextMove(&board, true, &row, &col);
            //we make the computer's move on the board.
            board.squares[row][col] = MINIMAX_PLAYER_SQUARE;
            //display the move on the board
            CPU_is_x ? ticTacToeDisplay_drawX(row, col) : ticTacToeDisplay_drawO(row, col);
        }
        break;
    case player_turn:           //The player's turn
        break;      //we wait for a touch.
    case player_touch_wait_state://Waiting for the touch sensors to cool
        //We subtract 1 from the timer unless it is already 0
        countdown_timer = countdown_timer > 0 ? countdown_timer - 1 : 0;
        break;
    case game_over:         //The end of the game!
        buttons_init();//prep the buttons
        int16_t endGame_score;      //get the end game score
        endGame_score = minimax_computeBoardScore(&board, true);    //get the score
        if(endGame_score == 0) {    //Catsgame! Print that to screen
            ticTacToeControl_print_status("CATSGAME!\n\rPush a button for new game");
        } else if(endGame_score > 0){//X wins! Print that to screen
            ticTacToeControl_print_status("X WINS!!!\n\rPush a button for new game");
        } else {                    //O wins! Print that to screen
            ticTacToeControl_print_status("O WINS!!!\n\rPush a button for new game");
        }

        break;
    case end_game:          //the end credits. wait for new game.
        break;  //(do nothing)
    default:                //a grave error; print it!
        printf("A grave error has happened; invalid state.\n\r");
        break;
    }

    //Now, we update to the next state and perform mealy outputs
    switch(currentState){
    case init_state:        //we start here
        countdown_timer = TICTACTOECONTROL_PLAYER_TIMEOUT;
        currentState = choose_players;      //start with the computer's turn
        break;
    case choose_players:
        if(countdown_timer <= 0){
            ticTacToeDisplay_init();    //init the screen
            currentState = CPU_turn;    //timed out! CPU's turn
            CPU_is_x = true;            //CPU is x and goes first
        }
        if(display_isTouched()){        //they touched
            ticTacToeDisplay_init();    //init the screen
            currentState = choose_players_chosen;   //they are playing X!
        }
        break;
    case choose_players_chosen:
        if(!display_isTouched()){       //wait until they lift their finger
            ticTacToeDisplay_init();    //init the screen
            currentState = player_turn; //they play!
            CPU_is_x = false;           //player is x and goes first
        }
        break;
```

```
    case CPU_turn:          //The computer's turn
        //test to see if game is over. We need to hand that function a score.
        if(minimax_isGameOver(minimax_computeBoardScore(&board, true))){
            currentState = game_over;   //game is over! go to end game
        } else {                        //not game over
            currentState = player_turn;     //CPU took his turn, player's turn now
        }
        break;
    case player_turn:           //The player's turn
        if(display_isTouched()){    //hey! X made his move! now to wait.
            display_clearOldTouchData();                        //reset for good measure
            currentState = player_touch_wait_state;                 //move to next state
            countdown_timer = TICTACTOECONTROL_TOUCH_COOLDOWN;  //rev the timer
        }
        break;
    case player_touch_wait_state://Waiting for the touch sensors to cool
        //We check countdown timer to make sure we've waited long enough
        //We check touched because we wait for the player to lift her finger
        if(countdown_timer <= 0 && !display_isTouched()){
            //the player has made her move! Time to do things and move to next state!
            uint8_t row, col;   //for storing the move
            //retrive the player's move
            ticTacToeDisplay_touchScreenComputeBoardRowColumn(&row, &col);
            //save the move on the board
            board.squares[row][col] = MINIMAX_OPPONENT_SQUARE;
            //Display the move on the board
            CPU_is_x ? ticTacToeDisplay_drawO(row, col) : ticTacToeDisplay_drawX(row, col);
            //test to see if game is over. We need to hand that function a score.
            if(minimax_isGameOver(minimax_computeBoardScore(&board, false))){
                currentState = game_over;       //game is over! go to end game
            } else {                            //not game over
                currentState = CPU_turn;            //game is not over! go to CPU's turn
            }
        }
        break;
    case game_over:         //the end of the game!
        currentState = end_game;    //now for the end credits.
    case end_game:          //wait for new game.
        if(buttons_read()){
            //if any button is pushed, reset the game
            currentState = init_state;
        }
        break;
    default:                //a grave error; print it!
        printf("A grave error has happened; invalid state.\n\r");
        break;
    }



}

//calculates to see if any moves have already been made
bool ticTacToeControl_boardIsEmpty(minimax_board_t *board){
    for (int8_t row = 0; row < MINIMAX_BOARD_ROWS; row++) { //cycle through the row
        for (int8_t col = 0; col < MINIMAX_BOARD_COLUMNS; col++) { //cycle through the column
            if(board->squares[row][col] != MINIMAX_EMPTY_SQUARE){   //check if empty square
```

```
                return false;   //if any square is not empty, the board is not empty
            }
        }
    }
    return true;                //all squares are empty; the board is empty
}

//This function preps the screen to display a status
void ticTacToeControl_print_status(const char str[]){
    //These are good status colors
    display_setTextColor(DISPLAY_CYAN, DISPLAY_WHITE);
    //Set the proper text size
    display_setTextSize(TICTACTOECONTROL_STATUS_TEXT_SIZE);
    //Set the cursor
    display_setCursor(TICTACTOECONTROL_STATUS_TEXT_X, TICTACTOECONTROL_STATUS_TEXT_Y);
    //We are ready to print! print to the screen
    display_println(str);
}
```