```
/*****************************************************************/
/*****************************************************************/
/***                                                           ***/
/***  Space Invaders Simulator                                 ***/
/***                                                           ***/
/***  ECE 4180 Developer Team: Space Pirates                   ***/
/***     Members: David Gaspard, Zhe Cheng Lee                 ***/
/***                                                           ***/
/*****************************************************************/
/*****************************************************************/


/*****************************************************************/
/*                                                               */
/* Libaries                                                      */
/*                                                               */
/*****************************************************************/
#include "mbed.h"
#include "NokiaLCD.h"
#include "SIgame.h"
#include "rtos.h"
#include "SDFileSystem.h"
#include "wave_player.h"

/*****************************************************************/
/*                                                               */
/* Classes and Global Variables                                  */
/*                                                               */
/*****************************************************************/
NokiaLCD lcd(p5, p7, p8, p9, NokiaLCD::LCD6610); // mosi, sclk, cs, rst, type
DigitalIn button(p20); // Pushbutton for firing laser
AnalogIn horiz(p15); // From joystick for steering ship
SDFileSystem sd(p11, p12, p13, p14, "sd"); //SD card
AnalogOut DACout(p18);//Speaker out
wave_player waver(&DACout);//Wave player

int state;
int doneski;//done playing start music?

OBJECT wave[NUM_ALIEN_ROWS][NUM_ALIEN_COLS],
       wave_old[NUM_ALIEN_ROWS][NUM_ALIEN_COLS];

/*****************************************************************/
/*                                                               */
/* Prototypes                                                    */
/*                                                               */
/*****************************************************************/
void start();
int game();
void lose();
void win();
void updateScreen(OBJECT ship, OBJECT ship_old, POINT ylaser, POINT ylaser_old,
                  POINT elaser[], POINT elaser_old[], int lives_rem);
```

```
void playstart(void const *args);
void startmusic(void const *args);


/*************************************************************/
/*                                                           */
/* Main program                                              */
/*                                                           */
/*************************************************************/
int main()
{
    doneski=0;
    Thread thread(playstart);
    Thread startmu(startmusic);
    state = START;
    button.mode(PullUp);

    while (1) { // Game loop
        switch (state) {
            case START:
                //Thread playmusic(playstart);
                start(); // Start game
                state = GAME;
                break;

            case GAME:
                startmu.terminate();
                doneski=1;
                state = game(); // Actual game
                break;

            case LOSE:
                lose(); // Losing screen
                state = START;
                break;

            case WIN:
                win(); // Winning screen
                state = START;
                break;
        }
    }
}


/*************************************************************/
/*                                                           */
/* Game-State Procedures                                     */
/*                                                           */
/*************************************************************/
void start()
{
```

```
    /******
     * start
     *     Starts up game
     ******/
    lcd.cls();
    lcd.locate(0,4);
    lcd.printf("SPACE INVADERS");
    lcd.locate(0,10);
    lcd.printf("PRESS BUTTON TO CONTINUE");
    WAIT4FULLPUSH(button);
}


int game()
{
    /******
     * game
     *     Handles the main gameplay. Prepares for win state if player destroys all
     *     aliens and lose state if player loses all lives
     ******/
    POINT ylaser, ylaser_old, elaser[ELASER_CAP], elaser_old[ELASER_CAP];
    OBJECT ship, ship_old, *sh_alien, *frontline[NUM_ALIEN_COLS];
    int time, lives_rem, als_rem, i;
    float hper;
    bool reach, left, lcancel;

    // Set display screen
    lcd.cls();
    lcd.background(BLACK);

    // Initialize
    lives_rem = NUM_LIVES_START;
    als_rem = NUM_ALIENS_START;
    reach = left = lcancel = false;
    time = 0;
    summonWave(); // Starting locations of all alien
    memcpy(wave_old, wave, sizeof(wave)); // Make copy for aliens' old locations

    // Keep track of aliens in front of respective columns
    for (i = 0; i < NUM_ALIEN_COLS; i++)
        frontline[i] = &wave[NUM_ALIEN_ROWS][i];

    // Set up laser structures
    elaser[0].collide = elaser[1].collide = elaser[2].collide =
            elaser_old[0].collide = elaser_old[1].collide = elaser_old[2].collide
                                  = ylaser.collide = ylaser_old.collide = true; // No lasers
                                  on screen yet
    // Enemy lasers are white
    elaser[0].color = elaser[1].color = elaser[2].color = WHITE;
    ylaser.color = GREEN; // Player's laser is green

    // Set up player's ship
    ship = startShip();
```

```
    memcpy(&ship_old, &ship, sizeof(ship)); // Save copy for ship's old locations

    // Game goes on until player loses all lives, destroy all aliens, or allow
    // any alien to reach screen border on player's side
    while (!reach && lives_rem > 0 && als_rem > 0) {
        /* Laser Movement */
        /******************/
        // Player's laser is on screen
        if (!(ylaser.collide)) {
            if (ylaser.y >= 0)
                (ylaser.y)-= 3; // Laser travels up to aliens' side
            else
                ylaser.collide = true; // Laser vanishes at top of screen
        }

        // Enemy lasers is on screen
        for (i = 0; i < ELASER_CAP; i++) {
            if (elaser[i].y < SCREENHEIGHT) // Enemy lasers travel down to player's side
                (elaser[i].y)++;
            else // Enemy laser vanishes at bottom of screen
                elaser[i].collide = true;
        }

        /* Controls       */
        /******************/
        // Player firing laser
        if (button && ylaser.collide) {
            // Initialize laser's locations according to ship's
            ylaser.y = ship.y - 1;
            ylaser.x = ship.x + (ship.width >> 1);
            // Can fire next laser after current hits top of screen or other objects
            ylaser.collide = false;
        }

        // Player steering ship
        hper = horiz; // Read joystick's horizontal position.
        if (GOLEFT(hper) && ship.x > 0)
            (ship.x)--; // Ship moves left/

        if (GORIGHT(hper) && ship.x < SCREENWIDTH - 1)
            (ship.x)++; // Ship moves right

        /* Alien Activity */
        /******************/
        // Move wave of aliens. Update whether any alien reaches screen on
        // player's side
        reach = moveAlienWave(&left);

        // Enemy aliens firing lasers. Aliens fire lasers only after certain
        // time intervals or when one of their lasers canceled out with player's
        if (time >= RELOAD_TIME || lcancel) {
            // Choose random front alien to fire laser. Ignore any invalid ones
            // (those in columns that were completely wiped out have been set to
```

```
            // NULL)
            while ((sh_alien = frontline[rand() / (RAND_MAX/i + 1)]) == NULL);

            // Find enemy laser structure not in use for display
            if (elaser[0].collide)
                i = 0;
            else if (elaser[1].collide)
                i = 1;
            else
                i = 2;

            // Initialize laser's location according to chosen alien's
            elaser[i].y = sh_alien->y + 1;
            elaser[i].x = sh_alien->x + (ALIEN_WIDTH >> 1);
            elaser[i].collide = lcancel = false;
            time = 0; // Reset time for next enemy laser fire
        }

        /* Collisions      */
        /*******************/
        // Cancel out player's laser and enemy laser should they connect
        lcancel = twoLasersCollide(&ylaser, elaser);

        // Kill alien if player's laser hits one
        destroyAlien(&als_rem, &ylaser, frontline);

        // Check for any enemy laser touching ship
        for (i = 0; i < ELASER_CAP; i++) {
            if (!(elaser[i].collide) && elaser[i].y >= ship.y && elaser[i].y <=
                    ship.y + ship.height && elaser[i].x >= ship.x && elaser[i].x
                    <= ship.x + ship.width) {
                // Plyaer loses life if hit by laser
                lives_rem--;
                ship.killed = elaser[i].collide = true;
            }
        }

        updateScreen(ship, ship_old, ylaser, ylaser_old, elaser, elaser_old,
                     lives_rem); // Update gameplay screen
        ship.killed = false;
        time++; // Time passes

        // Update old locations to current
        memcpy(wave_old, wave, sizeof(wave));
        memcpy(elaser_old, elaser, sizeof(elaser));
        ylaser_old = ylaser;
        ship_old = ship;
    } // End of main gameplay loop


    // Game's outcomes
    if (als_rem) // Lost all lives or aliens reached your border
        return LOSE;
```

```
    else // Destroyed all aliens
        return WIN;
}


void win()
{
    /******
     * win
     *     Displays congratulation screen and prompt player to replay
     ******/
    // Congratulation screen
    lcd.cls();
    lcd.locate(0,3);
    lcd.printf("WELL DONE\n\rEARTHLING");
    lcd.locate(0,6);
    lcd.printf("THIS TIME YOU WIN");
    wait(2);

    // Prompt player to play game again
    lcd.cls();
    lcd.locate(0,10);
    lcd.printf("PRESS BUTTON TO PLAY AGAIN");
    WAIT4FULLPUSH(button);
}


void lose()
{
    /******
     * lose
     *     Displays game-over screen and prompts player to replay
     ******/
    // Game-over screen
    lcd.cls();
    lcd.locate(4,3);
    lcd.printf("YOU SUCK");
    wait(2);

    // Prompt player to play game again
    lcd.locate(0,10);
    lcd.printf("PRESS BUTTON TO PLAY AGAIN");
    WAIT4FULLPUSH(button);
}


/*************************************************************/
/*                                                           */
/* Game-Drawing Funntion                                     */
/*                                                           */
/*************************************************************/
void updateScreen(OBJECT ship, OBJECT ship_old, POINT ylaser, POINT ylaser_old,
                  POINT elaser[], POINT elaser_old[], int lives_rem)
```

```
{
    /******
     * updateScreen
     *     Updates game display on Nokia LCD screen
     ******/
    int r, c;

    // Old locations are drawn with black to erase previous images in display
    // when redrawing images
    wait(0.02);

    // Redraw surviving aliens from their positions
    for (c = 0; c < NUM_ALIEN_COLS; c++) {
        for (r = 0; r < NUM_ALIEN_ROWS; r++) {
            lcd.fill(wave_old[r][c].x, wave_old[r][c].y, wave_old[r][c].width,
                     wave_old[r][c].height, BLACK);
            // Aliens as white rectangles
            if (!(wave[r][c].killed)) // Don't draw destroyed aliens
                lcd.fill(wave[r][c].x, wave[r][c].y, wave[r][c].width,
                         wave[r][c].height, wave[r][c].color);
        }
    }

    // Redraw enemy lasers as single white pixels
    for (c = 0; c < ELASER_CAP; c++) {
        lcd.pixel(elaser_old[c].x, elaser_old[c].y, BLACK);
        // Don't draw lasers that hit bottom screen border or other objects
        if (!(elaser[c].collide))
            lcd.pixel(elaser[c].x, elaser[c].y, elaser[c].color);
    }

    // Redraw player's laser as single green pixel. Check that laser doesn't
    // hit top screen border or other objects
    lcd.pixel(ylaser_old.x, ylaser_old.y, BLACK);
    if (!(ylaser.collide))
        lcd.pixel(ylaser.x, ylaser.y, ylaser.color);

    if (ship.killed) { // Ship is destroyed from being hit by laser
        lcd.locate(0,15);
        lcd.printf("LIVES: %d", lives_rem);
        wait(2); // Game pauses for few seconds before player uses next life
        lcd.cls(); // Clear lives message
    }

    // Redraw player's ship as green rectangle
    lcd.fill(ship_old.x, ship_old.y, ship_old.width, ship_old.height, BLACK);
    lcd.fill(ship.x, ship.y, ship.width, ship.height, ship.color);
}

void playstart(void const *args)
{
    while(true) {
        FILE *wave_file;
```

```
        while(state==START) {
            Thread::wait(50);
        }

        while(doneski!=1) {
            Thread::wait(500);
        }

        while(state==GAME) {
            if(button==1) {

                wave_file=fopen("/sd/mample4.wav","r");


                waver.play(wave_file);
                fclose(wave_file);
            }
        }

        while(state==LOSE) {
            wave_file=fopen("/sd/mample3.wav","r");
            waver.play(wave_file);
            fclose(wave_file);
        }

        while(state==WIN) {
            wave_file=fopen("/sd/mample3.wav","r");
            waver.play(wave_file);
            fclose(wave_file);
        }
    }
}

void startmusic(void const *args)
{

    FILE *wave_file;
    wave_file=fopen("/sd/mample2.wav","r");
    waver.play(wave_file);
    fclose(wave_file);

}
```