

# aliens.c

```

/*
 * aliens.c
 * Taylor Cowley and Andrew Okazaki
 */

#include <stdio.h>
#include "platform.h"
#include "xparameters.h"
#include "xaxivdma.h"
#include "xio.h"
#include "time.h"
#include "unistd.h"
#include "util.h"
#include <stdbool.h>
#include <stdint.h>
#include "bunkers.h"
#include "tank.h"           // required to tell if our bullets hit him.
#include "interface.h"      //required to update the score
#define ALIEN_HEIGHT 8      // Aliens are 8 pixels tall
#define ALIEN_WIDTH 12      // Aliens are 12 pixels wide
#define ALIEN_COLUMNS 11    // 11 columns of aliens
#define BULLET_HEIGHT 5     // Bullets are 5 pixels tall
#define TOP_TOTAL 11        // 11 aliens in top group
#define LOC_ALIEN_ONE 50    // Pixel where the first alien is
#define MIDDLE_TOTAL 22     // There are 22 total middle aliens
#define BOTTOM_TOTAL 22      // There are 22 total bottom aliens
#define ALIEN_NUM_BULLETS 4 // Aliens can have up to 4 bullets at a time
#define ALIEN_NUM_BULLET_TYPES 2 // Aliens have 2 types of bullets to choose from
#define BAD_ADDRESS -1      // Nothing exists at screen address -1
#define MOVE_DOWN_PIXELS 15 // When the aliens move down, they do so 15 pixels
#define LEFT_BOUNDARY 11    // Aliens cannot go more left than this
#define RIGHT_BOUNDARY 307  // Aliens cannot go more right than this
#define BULLET_COL_OFFSET 6 // Bullets appear 11 more right than their alien
#define BULLET_ROW_OFFSET 11 // Bullets appear more down than their alien
#define SCREEN_LENGTH 320   // Our screen is 320 pixels wide
#define SCREEN_HEIGHT 240   // Our screen is 240 pixels tall
#define SCREEN_RES_X 640    // Our screen RESOLUTION is 640 pixels wide
#define SCREEN_RES_Y 480    // Our screen RESOLUTION is 480 pixels tall
#define WHITE 0xFFFFFFFF    // These
#define BLACK 0x00000000    // are colors
#define RED 0xFF0000
#define WORD_WIDTH 12
#define TOP_POINTS 40        // top alien amount of points given if killed
#define MIDDLE_POINTS 20     // middle alien amount of points given if killed
#define BOTTOM_POINTS 10      // bottom alien amount of points given if killed

// Packs each horizontal line of the figures into a single 32 bit word.
#define
packWord32(b31,b30,b29,b28,b27,b26,b25,b24,b23,b22,b21,b20,b19,b18,b17,b16,b15,b14,b13,b12,
,b11,b10,b9,b8,b7,b6,b5,b4,b3,b2,b1,b0) \
    ((b31 << 31) | (b30 << 30) | (b29 << 29) | (b28 << 28) | (b27 << 27) | (b26 <<
26) | (b25 << 25) | (b24 << 24) | \
        (b23 << 23) | (b22 << 22) | (b21 << 21) | (b20 << 20) | (b19 << 19) | (b18
<< 18) | (b17 << 17) | (b16 << 16) | \
        (b15 << 15) | (b14 << 14) | (b13 << 13) | (b12 << 12) | (b11 << 11) | (b10
<< 10) | (b9 << 9 ) | (b8 << 8 ) | \
        (b7 << 7 ) | (b6 << 6 ) | (b5 << 5 ) | (b4 << 4 ) | (b3 << 3 ) | (b2
<< 2 ) | (b1 << 1 ) | (b0 << 0 ) )

```

# aliens.c

```

#define packword12(b11,b10,b9,b8,b7,b6,b5,b4,b3,b2,b1,b0) \
    ((b11 << 11) | (b10 << 10) | (b9 << 9) | (b8 << 8) | (b7 << 7) | (b6 << 6
    ) \
    | (b5 << 5) | (b4 << 4) | (b3 << 3) | (b2 << 2) | (b1 << 1) |
    (b0 << 0) )

// -----
// The following static const ints define the aliens
// We have 3 types of aliens with 2 poses each
const int deadAlien[ALIEN_HEIGHT] =
{
    packword12(0,0,0,0,0,1,0,1,0,0,0,0),
    packword12(0,1,0,1,0,0,0,1,0,0,1,0),
    packword12(0,0,1,0,0,1,0,0,0,1,0,0),
    packword12(0,0,0,0,0,0,0,0,0,0,0,0),
    packword12(0,1,1,1,0,1,0,1,1,1,0,0),
    packword12(0,0,0,0,0,0,0,0,0,0,1,0),
    packword12(0,0,1,0,0,1,0,0,1,0,0,0),
    packword12(0,1,0,1,0,1,0,0,0,1,0,0) };
static const int32_t alien_top_in_12x8[ALIEN_HEIGHT] = {
    packword12(0,0,0,0,0,1,1,0,0,0,0,0),
    packword12(0,0,0,0,1,1,1,1,0,0,0,0),
    packword12(0,0,0,1,1,1,1,1,1,0,0,0),
    packword12(0,0,1,1,0,1,1,0,1,1,0,0),
    packword12(0,0,1,1,1,1,1,1,1,1,0,0),
    packword12(0,0,0,1,0,1,1,0,1,0,0,0),
    packword12(0,0,1,0,0,0,0,0,0,1,0,0),
    packword12(0,0,0,1,0,0,0,0,1,0,0,0) };
static const int32_t alien_top_out_12x8[ALIEN_HEIGHT] = {
    packword12(0,0,0,0,0,1,1,0,0,0,0,0),
    packword12(0,0,0,0,1,1,1,1,0,0,0,0),
    packword12(0,0,0,1,1,1,1,1,1,0,0,0),
    packword12(0,0,1,1,0,1,1,0,1,1,0,0),
    packword12(0,0,1,1,1,1,1,1,1,1,0,0),
    packword12(0,0,0,0,1,0,0,1,0,0,0,0),
    packword12(0,0,0,1,0,1,1,0,1,0,0,0),
    packword12(0,0,0,1,0,1,0,0,1,0,1,0) };
static const int32_t alien_middle_in_12x8[ALIEN_HEIGHT] = {
    packword12(0,0,0,1,0,0,0,0,0,1,0,0),
    packword12(0,0,0,0,1,0,0,0,0,1,0,0),
    packword12(0,0,0,0,1,1,1,1,1,1,0,0),
    packword12(0,0,1,1,0,1,1,1,0,1,1,0),
    packword12(0,1,1,1,1,1,1,1,1,1,1,1),
    packword12(0,1,1,1,1,1,1,1,1,1,1,1),
    packword12(0,1,0,1,0,0,0,0,0,1,0,1),
    packword12(0,0,0,0,1,1,0,1,1,0,0,0) };
static const int32_t alien_middle_out_12x8[] = {
    packword12(0,0,0,1,0,0,0,0,0,1,0,0),
    packword12(0,1,0,0,1,0,0,0,1,0,0,1),
    packword12(0,1,0,1,1,1,1,1,1,1,0,1),
    packword12(0,1,1,1,0,1,1,1,0,1,1,1),
    packword12(0,1,1,1,1,1,1,1,1,1,1,1),
    packword12(0,0,1,1,1,1,1,1,1,1,1,0),
    packword12(0,0,0,1,0,0,0,0,0,1,0,0),
    packword12(0,0,1,0,0,0,0,0,0,0,1,0) };
static const int32_t alien_bottom_in_12x8[ALIEN_HEIGHT] = {
    packword12(0,0,0,0,1,1,1,1,0,0,0,0),
    packword12(0,1,1,1,1,1,1,1,1,1,1,0),

```

# aliens.c

```

        packword12(1,1,1,1,1,1,1,1,1,1,1,1),
        packword12(1,1,1,0,0,1,1,0,0,1,1,1),
        packword12(1,1,1,1,1,1,1,1,1,1,1,1),
        packword12(0,0,1,1,1,0,0,1,1,1,0,0),
        packword12(0,1,1,0,0,1,1,0,0,1,1,0),
        packword12(0,0,1,1,0,0,0,0,1,1,0,0) };
static const int32_t alien_bottom_out_12x8[] = {
    packword12(0,0,0,0,1,1,1,1,0,0,0,0),
    packword12(0,1,1,1,1,1,1,1,1,1,1,0),
    packword12(1,1,1,1,1,1,1,1,1,1,1,1),
    packword12(1,1,1,0,0,1,1,0,0,1,1,1),
    packword12(1,1,1,1,1,1,1,1,1,1,1,1),
    packword12(0,0,0,1,1,0,0,1,1,0,0,0),
    packword12(0,0,1,1,0,1,1,0,1,1,0,0),
    packword12(1,1,0,0,0,0,0,0,0,0,1,1) };
// End of the const ints that define the alien pixels
// -----

// -----
// These are our internal methods, used only by ourselves
// Draws the aliens on the screen - top, middle, and bottom aliens
void build_top(uint32_t * framePointer, const int32_t alien_middle[], bool erase);
void build_middle(uint32_t * framePointer, const int32_t alien_middle[], bool erase);
void build_bottom(uint32_t * framePointer, const int32_t alien_bottom[], bool
forceUpdate);
// Fire a bullet from either a top, middle, or bottom alien
int32_t fire_bottom(uint32_t * framePointer, int32_t r);
int32_t fire_middle(uint32_t * framePointer, int32_t r);
int32_t fire_top(uint32_t * framePointer, int32_t r);
// Checks to see whether our aliens are currently capable of shooting
bool can_aliens_shoot();
// Draws a bullet on the screen
void draw_bullet(uint32_t * framePointer, int32_t bullet, uint32_t color);
// We like our aliens black
void aliens_blacken(uint32_t * framePointer, uint32_t row, uint32_t col);
// Have the aliens destroyed us?
void aliens_detect_game_over();
// End internal method declarations
// -----

// These structs hold all of our aliens.
struct top { // Struct for our top aliens
    int32_t row;
    int32_t col; bool alive; // alien has row, column, and alive?
    bool exploding;
} top[TOP_TOTAL];

struct middleAlien { // Struct for our middle aliens
    int32_t row;
    int32_t col; bool alive; // alien has row, column, and alive?
    bool exploding;
} middleAlien[MIDDLE_TOTAL];

struct bottomAlien { // Struct for our bottom aliens
    int32_t row;
    int32_t col; bool alive; // alien has row, column, and alive?
    bool exploding;
} bottomAlien[MIDDLE_TOTAL];

```

# aliens.c

```
// aliens can have two types of bullet: cross and lightning
// cross 0 and 3 are identical
typedef enum {
    cross0, cross1, cross2, cross3, lightning0, lightning1
} bullet_type;
struct alien_bullet { // Struct that holds our aliens' bullets
    int32_t row;
    int32_t col; bool alive; // Bullets have coordinates and alive?
    bullet_type bullet_type; // Bullets also have a type.
} alien_bullet[ALIEN_NUM_BULLETS];

int32_t alien_count; // a count of how many aliens are alive
int32_t how_many.aliens_left;
uint32_t * frame; // framePointer
//initialize all of the aliens by setting values contained in struct's and printing
aliens to the screen
void aliens_init(uint32_t * framePointer) {
#define ALIEN_TOP_ROW_INIT 30 // Where
#define ALIEN_MIDDLE_ROW_INIT 45 // the
#define ALIEN_MIDDLE2_ROW_INIT 60 // aliens
#define ALIEN_BOTTOM_ROW_INIT 75 // are
#define ALIEN_BOTTOM2_ROW_INIT 90 // initialized to
#define ALIEN_SPACING 15 // Spacing between aliens
    //local variables, loc is the starting location of alien one on the screen
    int32_t i, loc = LOC_ALIEN_ONE;
    frame = framePointer;

    //loops through one row of aliens
    for (i = 0; i < ALIEN_COLUMNS; i++) {
        top[i].row = ALIEN_TOP_ROW_INIT; //set the row of alien tops to 30
        top[i].col = loc; //sets the column of alien tops
        top[i].alive = true; //sets the alien is alive flag
        top[i].exploding = false;

        middleAlien[i].row = ALIEN_MIDDLE_ROW_INIT; //middle aliens
        middleAlien[i].col = loc; //sets column of first row of middle aliens
        middleAlien[i].alive = true; //sets first row of middle aliens to alive
        middleAlien[i].exploding = false;
        middleAlien[i + ALIEN_COLUMNS].row = ALIEN_MIDDLE2_ROW_INIT; //sets middle
        middleAlien[i + ALIEN_COLUMNS].col = loc; //sets column second row middle
        middleAlien[i + ALIEN_COLUMNS].alive = true; //sets second row middle alive
        middleAlien[i + ALIEN_COLUMNS].exploding = false;

        bottomAlien[i].row = ALIEN_BOTTOM_ROW_INIT; //sets bottom aliens
        bottomAlien[i].col = loc; //sets column of first row of bottom aliens
        bottomAlien[i].alive = true; //sets first row of bottom aliens to alive
        bottomAlien[i].exploding = false;
        bottomAlien[i + ALIEN_COLUMNS].row = ALIEN_BOTTOM2_ROW_INIT; //bottom
        bottomAlien[i + ALIEN_COLUMNS].col = loc; //sets column second row bottom
        bottomAlien[i + ALIEN_COLUMNS].alive = true; //sets second row bottom alive
        bottomAlien[i + ALIEN_COLUMNS].exploding = false;
        loc += ALIEN_SPACING; //controls the column spacing in-between alien
    }

    //now that structs are built draw top, middle, and bottom aliens to screen
    build_tops(framePointer, alien_top_in_12x8, false); // Top
    build_middle(framePointer, alien_middle_in_12x8, false); // Middle
}
```

# aliens.c

```

    build_bottom(framePointer, alien_bottom_in_12x8, false); // Bottom

    how_many.aliens_left = TOP_TOTAL + MIDDLE_TOTAL + BOTTOM_TOTAL;
}

// Draws the top aliens on the screen
void build_tops(uint32_t * framePointer, const int32_t alien_top[], bool erase) {
    uint32_t color = erase ? BLACK : WHITE ;
    int32_t row, col, i; // initialize variables
    for (i = 0; i < TOP_TOTAL; i++) { //loop through top column of aliens
        for (row = 0; row < ALIEN_HEIGHT; row++) { //loop top aliens' pixels row
            int32_t currentRow = row + top[i].row; // current pixel row of alien
            for (col = 0; col < WORD_WIDTH; col++) { //loop alien's pixel col
                int32_t currentCol = col + top[i].col; //current col of alien
                if ((alien_top[row] & (1 << (WORD_WIDTH - col - 1)))
                    && top[i].alive) {
                    // If our alien is alive and has a pixel there, draw it
                    util_draw_pixel(framePointer, currentRow, currentCol,
                                    color);
                } else if(top[i].alive){ // otherwise, erase it.
                    util_draw_pixel(framePointer, currentRow, currentCol, BLACK);
                } else if(top[i].exploding){
                    top[i].exploding = false;
                    aliens_blacken(framePointer, currentRow, currentCol);
                }
            }
        }
    }
}

// Draws the middle aliens to the screen
void build_middle(uint32_t * framePointer, const int32_t alien_middle[], bool erase) {
    uint32_t color = erase ? BLACK : WHITE ;
    int32_t row, col, i; // declare our variables
    for (i = 0; i < MIDDLE_TOTAL; i++) { // Looping through all the middle aliens
        for (row = 0; row < ALIEN_HEIGHT; row++) { // Pixel y
            int32_t currentRow = row + middleAlien[i].row; //current pixel row
            for (col = 0; col < WORD_WIDTH; col++) { // Pixel x
                int32_t currentCol = col + middleAlien[i].col; // current col alien
                if ((alien_middle[row] & (1 << (WORD_WIDTH - col - 1)))
                    && middleAlien[i].alive) {
                    // If our alien is alive and has a pixel there, draw it
                    util_draw_pixel(framePointer, currentRow, currentCol,
                                    color);
                } else if(middleAlien[i].alive){ // otherwise, erase it.
                    util_draw_pixel(framePointer, currentRow, currentCol, BLACK);
                } else if(middleAlien[i].exploding){
                    middleAlien[i].exploding = false;
                    aliens_blacken(framePointer, currentRow, currentCol);
                }
            }
        }
    }
}

// Draws the bottom aliens to the screen
void build_bottom(uint32_t * framePointer, const int32_t alien_bottom[], bool erase) {
    int32_t row, col, i; // Declare vars

```

# aliens.c

```

uint32_t color = erase ? BLACK : WHITE ;
for (i = 0; i < BOTTOM_TOTAL; i++) { // Looping through all the bottom aliens
    for (row = 0; row < ALIEN_HEIGHT; row++) { // looping through y pixels
        int32_t currentRow = row + bottomAlien[i].row; // current row
        for (col = 0; col < WORD_WIDTH; col++) { // looping through x pixels
            int32_t currentCol = col + bottomAlien[i].col; // current col
            if ((alien_bottom[row] & (1 << (WORD_WIDTH - col - 1)))
                && bottomAlien[i].alive) {
                // If our alien is alive and has a pixel here, draw it
                util_draw_pixel(framePointer, currentRow, currentCol,
                                color);
            } else if(bottomAlien[i].alive){ // otherwise, erase it.
                util_draw_pixel(framePointer, currentRow, currentCol, BLACK);
            } else if(bottomAlien[i].exploding){
                bottomAlien[i].exploding = false;
                aliens_blacken(framePointer, currentRow, currentCol);
            }
        }
    }
}

// Draws a big, black, rectangle over an alien
void aliens_blacken(uint32_t * framePointer, uint32_t row, uint32_t col){
    int32_t r, c;
    for(r=0;r<ALIEN_HEIGHT;r++){
        for(c=0;c<ALIEN_WIDTH;c++){
            util_draw_pixel(framePointer, r+row, c+col, BLACK);
        }
    }
}

// Does the needful to move the aliens left
void aliens_left(uint32_t * framePointer) {
    int32_t i, row; // Declare loop vars
    for (i = 0; i < MIDDLE_TOTAL; i++) { // Move every single alien LEFT
        if (i < TOP_TOTAL) {
            top[i].col--;
        } // Move the top aliens LEFT
        middleAlien[i].col--; // Move the middle aliens LEFT
        bottomAlien[i].col--; // Move the bottom aliens LEFT
    }
    if (alien_count == 0) { // If aliens are out, make them in
        alien_count = 1;
        build_tops(framePointer, alien_top_in_12x8, false); // Draw top aliens
        build_middle(framePointer, alien_middle_in_12x8, false); // Draw mid aliens
        build_bottom(framePointer, alien_bottom_in_12x8, false); // Draw bot aliens
    } else { // And vice versa
        alien_count = 0;
        build_tops(framePointer, alien_top_out_12x8, false); // Draw top aliens
        build_middle(framePointer, alien_middle_out_12x8, false); // Draw mid aliens
        build_bottom(framePointer, alien_bottom_out_12x8, false); // Draw bot aliens
    }

    for (row = 0; row < ALIEN_HEIGHT; row++) { // For all the alien Y pixels
        for (i = 0; i < MIDDLE_TOTAL; i++) { // For every alien
            // Erase them for the middle and bottom aliens - top is skinnier
            if(bottomAlien[i].alive){

```

# aliens.c

```

        util_draw_pixel(framePointer, row + bottomAlien[i].row,
                        WORD_WIDTH + bottomAlien[i].col, BLACK);
    }
    if(middleAlien[i].alive){
        util_draw_pixel(framePointer, row + middleAlien[i].row,
                        WORD_WIDTH + middleAlien[i].col, BLACK);
    }
}

// Here we loop through every single dang alien and see if they hit the dang bunkers
for (i = 0; i < MIDDLE_TOTAL; i++) { // Move every single alien LEFT
    if (i < TOP_TOTAL) {
        if(top[i].alive){
            bunkers_detect_collision(top[i].row,top[i].col,true);
            bunkers_detect_collision(top[i].row+ALIEN_HEIGHT/2,top[i].col,true);
            bunkers_detect_collision(top[i].row+ALIEN_HEIGHT,top[i].col,true);
        }
    } // Move the top aliens LEFT

    if(middleAlien[i].alive){
        bunkers_detect_collision(middleAlien[i].row,middleAlien[i].col,true);
        bunkers_detect_collision(middleAlien[i].row+ALIEN_HEIGHT/2,middleAlien[i].col,
true);
        bunkers_detect_collision(middleAlien[i].row+ALIEN_HEIGHT,middleAlien[i].col,t
rue);
    }

    if(bottomAlien[i].alive){
        bunkers_detect_collision(bottomAlien[i].row,bottomAlien[i].col,true);
        bunkers_detect_collision(bottomAlien[i].row+ALIEN_HEIGHT/2,bottomAlien[i].col,
true);
        bunkers_detect_collision(bottomAlien[i].row+ALIEN_HEIGHT,bottomAlien[i].col,t
rue);
    }
}

// Does the needful to move the aliens right
void aliens_right(uint32_t * framePointer) {
    int32_t i, row; // Declare loop vars
    for (i = 0; i < MIDDLE_TOTAL; i++) { // Move every single alien RIGHT
        if (i < 11) {
            top[i].col += 1;
        } // Move top aliens RIGHT
        middleAlien[i].col += 1; // Move middle aliens RIGHT
        bottomAlien[i].col += 1; // Move bottom aliens RIGHT
    }

    if (alien_count == 0) { // If aliens are out, make them in
        alien_count = 1;
        build_tops(framePointer, alien_top_in_12x8, false); // Draw top aliens
        build_middle(framePointer, alien_middle_in_12x8, false); // Draw mid aliens
        build_bottom(framePointer, alien_bottom_in_12x8, false); // Draw bot aliens
    } else { // And vice versa
        alien_count = 0;
    }
}

```

# aliens.c

```

    build_tops(framePointer, alien_top_out_12x8, false); // Draw top aliens
    build_middle(framePointer, alien_middle_out_12x8, false); // Draw mid aliens
    build_bottom(framePointer, alien_bottom_out_12x8, false); // Draw bot aliens
}

for (row = 0; row < ALIEN_HEIGHT; row++) { // For all the alien Y pixels
    for (i = 0; i < MIDDLE_TOTAL; i++) { // For every alien
        // Erase that column of pixels for mid and bottom. Top not necessary
        if(bottomAlien[i].alive){
            util_draw_pixel(framePointer, row + bottomAlien[i].row,
                            bottomAlien[i].col - 1, BLACK); // Notice it's col-1 bottom
        }
        if(middleAlien[i].alive){
            util_draw_pixel(framePointer, row + middleAlien[i].row,
                            middleAlien[i].col, BLACK);
        }
    }
}

// Here we loop through every single dang alien and see if they hit the dang bunkers
for (i = 0; i < MIDDLE_TOTAL; i++) { // Move every single alien LEFT
    if (i < TOP_TOTAL) {
        if(top[i].alive){
            bunkers_detect_collision(top[i].row,top[i].col+ALIEN_WIDTH,true);
        }
    } // Move the top aliens LEFT

    if(middleAlien[i].alive){
        bunkers_detect_collision(middleAlien[i].row,middleAlien[i].col+ALIEN_WIDTH,true);
    }

    if(bottomAlien[i].alive){
        bunkers_detect_collision(bottomAlien[i].row,bottomAlien[i].col+ALIEN_WIDTH,true);
    }
}

// Does the needful when aliens hit the left rail
void hit_left_rail(uint32_t * framePointer) {
    // Erase ALL the aliens.
    build_tops(framePointer, alien_bottom_out_12x8, true);
    build_middle(framePointer, alien_bottom_out_12x8, true);
    build_bottom(framePointer, alien_bottom_out_12x8, true);

    // First we erase the entire top row of alien pixels for moving down.
    int32_t col, row, i; // declare loop vars
    for (row = 0; row < ALIEN_HEIGHT; row++) { // Go through alien pixels Y
        for (col = 0; col < WORD_WIDTH; col++) { // Go through alien pixels X
            if (((alien_top_out_12x8[row] | alien_top_in_12x8[row]) & (1
                << (WORD_WIDTH - col - 1)))) { // if pixel exists here
                for (i = 0; i < TOP_TOTAL; i++) { // ERASE IT!
                    util_draw_pixel(framePointer, row + top[i].row,
                                    col + top[i].col, BLACK);
                }
            }
        }
    }
}

```



aliens.c

```

    }
}

}

for (i = 0; i < MIDDLE_TOTAL; i++) { // For all the aliens, move them down
    if (i < TOP_TOTAL) {
        top[i].row += MOVE_DOWN_PIXELS;
    } // Move top aliens down
    middleAlien[i].row += MOVE_DOWN_PIXELS; // Move mid aliens down
    bottomAlien[i].row += MOVE_DOWN_PIXELS; // Move bot aliens down
}

for (row = 0; row < ALIEN_HEIGHT; row++) { // Now to erase pixels on left side
    for (i = 0; i < MIDDLE_TOTAL; i++) { // For all the middle aliens
        util_draw_pixel(framePointer, row + middleAlien[i].row,
            middleAlien[i].col, BLACK); // Erase the pixels on the left
    }
}

}

// Does the needful when aliens hit the right rail
void hit_right_rail(uint32_t * framePointer) {
    // Erase ALL the aliens.
    build_tops(framePointer, alien_bottom_out_12x8, true);
    build_middle(framePointer, alien_bottom_out_12x8, true);
    build_bottom(framePointer, alien_bottom_out_12x8, true);

    // First we erase the entire top row of alien pixels for moving down
    int32_t col, row, i; // Declare loop vars
    for (row = 0; row < ALIEN_HEIGHT; row++) { // Go through alien pixels Y
        for (col = 0; col < WORD_WIDTH; col++) { // Go through alien pixels X
            if (((alien_top_out_12x8[row] | alien_top_in_12x8[row]) & (1
                << (WORD_WIDTH - col - 1)))) { // if pixel exists here
                for (i = 0; i < TOP_TOTAL; i++) { // Erase it!
                    util_draw_pixel(framePointer, row + top[i].row,
                        col + top[i].col, BLACK);
                }
            }
        }
    }

    for (i = 0; i < MIDDLE_TOTAL; i++) { // For all the aliens, move them down
        if (i < TOP_TOTAL) {
            top[i].row += MOVE_DOWN_PIXELS;

            // Move top aliens down
            middleAlien[i].row += MOVE_DOWN_PIXELS; // Move mid aliens down
            bottomAlien[i].row += MOVE_DOWN_PIXELS; // Move bot aliens down

            aliens_detect_game_over();
        }
    }

    for (row = 0; row < ALIEN_HEIGHT; row++) { // Now to erase pixels on the right side
        for (i = 0; i < TOP_TOTAL; i++) { // Erase the pixels on the right
            util_draw_pixel(framePointer, row + top[i].row,
                WORD_WIDTH - 1 + top[i].col, BLACK);
        }
    }
}
}

```

# aliens.c

```

// detects if teh aliens win
void aliens_detect_game_over(){
#define WINLINE 210 - ALIEN_HEIGHT
    if (bottomAlien[ALIEN_COLUMNS].row >= WINLINE ){ // if the aliens are to low end the
game
        int i;
        for(i=ALIEN_COLUMNS;i<ALIEN_COLUMNS+ALIEN_COLUMNS;i++){
            if(bottomAlien[i].alive){
                interface_game_over(); // end game put up game over screen
                return;
            }
        }
        if (bottomAlien[0].row >= WINLINE ){ // if the aliens are to low end the game
            for(i=0;i<ALIEN_COLUMNS;i++){
                if(bottomAlien[i].alive){
                    interface_game_over(); // end game put up game over screen
                    return;
                }
            }
        }
        if (middleAlien[ALIEN_COLUMNS].row >= WINLINE ){ // if the aliens are to low
end the game
            int i;
            for(i=ALIEN_COLUMNS;i<ALIEN_COLUMNS+ALIEN_COLUMNS;i++){
                if(middleAlien[i].alive){
                    interface_game_over(); // end game put up game over screen
                    return;
                }
            }
        }
        if (middleAlien[0].row >= WINLINE ){ // if the aliens are to low end the
game
            for(i=0;i<ALIEN_COLUMNS;i++){
                if(middleAlien[i].alive){
                    interface_game_over(); // end game put up game over screen
                    return;
                }
            }
        }
        if (top[0].row >= WINLINE ){ // if the aliens are to low end the game
            for(i=0;i<ALIEN_COLUMNS;i++){
                if(top[i].alive){
                    interface_game_over(); // end game put up game over screen
                    return;
                }
            }
        }
    }
}

// moves the aliens and detects wall boundries and direction changes too!
void aliens_move(uint32_t * framePointer) {
    static int32_t flag;
    int32_t i, j;
    for (i = 0; i < ALIEN_COLUMNS; i++) { // Go through every alien column
        // And see if any alien in that column is alive and has hit left
        if (top[i].alive || middleAlien[i].alive || middleAlien[i
+ ALIEN_COLUMNS].alive ||

```

# aliens.c

```

bottomAlien[i].alive || bottomAlien[i
+ ALIEN_COLUMNS].alive) {
    if (top[i].col == LEFT_BOUNDARY) { // If an alien has hit side
        flag = 1; // Set the flag that we've hit the side
        hit_left_rail(framePointer); // Call hit_rail.
    }
}
}
for (j = ALIEN_COLUMNS - 1; j >= 0; j--) { // Now to check to see
    if (top[j].alive || middleAlien[j].alive || middleAlien[j
+ ALIEN_COLUMNS].alive ||
bottomAlien[j].alive || bottomAlien[j
+ ALIEN_COLUMNS].alive) {
    if (top[j].col == RIGHT_BOUNDARY) { // if an alien has hit right.
        flag = 0; // false
        hit_right_rail(framePointer); // we have hit the right rail
    }
}
if (flag == 1) { // if we are moving right
    aliens_right(framePointer); // go right
} else { // we are actually going left
    aliens_left(framePointer); // so go left
}
}

// Kills a random alien
// Currently has a bug that if the last alien dies, infinite loop
void aliens_kill(uint32_t * framePointer) {
    int32_t r = rand() % 55; // Get a random number

    if (r < TOP_TOTAL) { // If we have killed a top
        if (!top[r].alive) { // Already dead!
            aliens_kill(framePointer); // Try again
        } else {
            top[r].alive = false; // kill the alien
            build_tops(framePointer, alien_top_in_12x8, false); // redraw aliens
        }
    } else if (r < (TOP_TOTAL + MIDDLE_TOTAL)) { // if we have killed a mid
        if (!middleAlien[r - TOP_TOTAL].alive) { // Already dead!
            aliens_kill(framePointer); // try again
        } else {
            middleAlien[r - TOP_TOTAL].alive = false; // kill alien
            build_middle(framePointer, alien_middle_in_12x8, false); // redraw aliens
        }
    } else { // we have killed a bot
        if (!bottomAlien[r - (TOP_TOTAL + MIDDLE_TOTAL)].alive) { // Already dead!
            aliens_kill(framePointer); // Try again
        } else {
            bottomAlien[r - (TOP_TOTAL + MIDDLE_TOTAL)].alive = false; // Kill alien
            build_bottom(framePointer, alien_bottom_in_12x8, false); // redraw aliens
        }
    }
}

// Returns true if aliens can shoot- that is, if there exists a top alive alien

```

# aliens.c

```

bool can_alien_shoot() {
    int32_t i; // Declare loop variable
    for (i = 0; i < TOP_TOTAL; i++) { // Look at all the top aliense
        if (top[i].alive) { // If there exists a single alive top alien
            return true; // We have an alive alien!
        }
    }
    return false; // All the top aliens are dead; we cannot shoot
}

// Fires a bullet from a random alien
void alien_missile(uint32_t * framePointer) {
#define TRY_TO_SHOOT_TIMES 3
    if (!can_alien_shoot()) { // The aliens can't even shoot! Don't even try.
        return;
    }

    int32_t r = rand() % ALIEN_COLUMNS; // Get a random column
    int32_t bullet_address = BAD_ADDRESS; // Initialize the address

    int32_t trying = 0; // Try several times to shoot
    do { // Keep trying to shoot
        bullet_address = fire_bottom(framePointer, r);
    } while (bullet_address == BAD_ADDRESS && (trying++ < TRY_TO_SHOOT_TIMES)); // until
    we get a good address

    if(bullet_address == BAD_ADDRESS){ // We tried 3 times to shoot
        return; // But failed! :(
    }

    // We have a bullet address! now to make it alive and draw it.
    int32_t i;
    for (i = 0; i < ALIEN_NUM_BULLETS; i++) {
        if (alien_bullet[i].alive) { // If we already have a living bullet
            continue; // Go on to the next one
        } else { // We have a dead bullet spot- let's alive a bullet here!
            alien_bullet[i].alive = true;
            // Randomly choose a bullet type
            alien_bullet[i].bullet_type
            = rand() % ALIEN_NUM_BULLET_TYPES ? cross0 : lightning0;
            // TODO: This math can be simplified
            alien_bullet[i].col = bullet_address % SCREEN_RES_X; // Set address
            alien_bullet[i].row = bullet_address / SCREEN_RES_X; // of bullet
            draw_bullet(framePointer, i, WHITE); // And draw it!
            return;
        }
    }
}

// Draws the selected bullet to the screen
void draw_bullet(uint32_t * framePointer, int32_t bullet, uint32_t color) {
#define PIXEL_LINE_1 1 // These
#define PIXEL_LINE_2 2 // defines
#define PIXEL_LINE_3 3 // only
#define PIXEL_LINE_4 4 // have
#define PIXEL_LEFT -1 // meaning
#define PIXEL_RIGHT 1 // in this function, so I put them here
    uint32_t row = alien_bullet[bullet].row; // Current row

```

## aliens.c

```
uint32_t col = alien_bullet[bullet].col; // and column where to draw
switch (alien_bullet[bullet].bullet_type) {
case cross0: // Cross0 and cross 3 are identically drawn
case cross3: // The only difference is in the state machine where they go
    // 5 pixels down in a line
    util_draw_pixel(framePointer, row, col, color);
    util_draw_pixel(framePointer, row + PIXEL_LINE_1, col, color);
    util_draw_pixel(framePointer, row + PIXEL_LINE_2, col, color);
    util_draw_pixel(framePointer, row + PIXEL_LINE_3, col, color);
    util_draw_pixel(framePointer, row + PIXEL_LINE_4, col, color);

    // Crossbar on the cross - right in the middle
    util_draw_pixel(framePointer, row + PIXEL_LINE_2, col + PIXEL_RIGHT,
        color);
    util_draw_pixel(framePointer, row + PIXEL_LINE_2, col + PIXEL_LEFT,
        color);
    break;
case cross1:
    // 5 pixels down in a line
    util_draw_pixel(framePointer, row, col, color);
    util_draw_pixel(framePointer, row + PIXEL_LINE_1, col, color);
    util_draw_pixel(framePointer, row + PIXEL_LINE_2, col, color);
    util_draw_pixel(framePointer, row + PIXEL_LINE_3, col, color);
    util_draw_pixel(framePointer, row + PIXEL_LINE_4, col, color);

    // Crossbar on the cross- on the lower one
    util_draw_pixel(framePointer, row + PIXEL_LINE_3, col + PIXEL_RIGHT,
        color);
    util_draw_pixel(framePointer, row + PIXEL_LINE_3, col + PIXEL_LEFT,
        color);
    break;
case cross2:
    // 5 pixels down in a line
    util_draw_pixel(framePointer, row, col, color);
    util_draw_pixel(framePointer, row + PIXEL_LINE_1, col, color);
    util_draw_pixel(framePointer, row + PIXEL_LINE_2, col, color);
    util_draw_pixel(framePointer, row + PIXEL_LINE_3, col, color);
    util_draw_pixel(framePointer, row + PIXEL_LINE_4, col, color);

    // Crossbar on the cross- on the upper one
    util_draw_pixel(framePointer, row + PIXEL_LINE_1, col + PIXEL_RIGHT,
        color);
    util_draw_pixel(framePointer, row + PIXEL_LINE_1, col + PIXEL_LEFT,
        color);
    break;
case lightning0:
    // 5 pixels down - starting left then right, then going back left
    util_draw_pixel(framePointer, row, col + PIXEL_LEFT, color);
    util_draw_pixel(framePointer, row + PIXEL_LINE_1, col, color);
    util_draw_pixel(framePointer, row + PIXEL_LINE_2, col + PIXEL_RIGHT,
        color);
    util_draw_pixel(framePointer, row + PIXEL_LINE_3, col, color);
    util_draw_pixel(framePointer, row + PIXEL_LINE_4, col + PIXEL_LEFT,
        color);
    break;
case lightning1:
    // 5 pixels down - starting right then left, then back right
    util_draw_pixel(framePointer, row, col + PIXEL_RIGHT, color);
```

# aliens.c

```

        util_draw_pixel(framePointer, row + PIXEL_LINE_1, col, color);
        util_draw_pixel(framePointer, row + PIXEL_LINE_2, col + PIXEL_LEFT,
            color);
        util_draw_pixel(framePointer, row + PIXEL_LINE_3, col, color);
        util_draw_pixel(framePointer, row + PIXEL_LINE_4, col + PIXEL_RIGHT,
            color);
        break;
    }
}

// This sees if our bottom alien at index r is alive to shoot
int32_t fire_bottom(uint32_t * framePointer, int32_t r) {
    if (!bottomAlien[r + ALIEN_COLUMNS].alive) { // If the very bottom alien is dead
        if (!bottomAlien[r].alive) { // AND the second row alien is also dead
            return fire_middle(framePointer, r); // Try to make a higher alien shoot it
        } else { // the bottom alien is dead, but the second-row one is alive
            // This is the starting coordinate of the bullet.
            return (bottomAlien[r].row + BULLET_COL_OFFSET + 1) * SCREEN_RES_X
                + (BULLET_COL_OFFSET + bottomAlien[r].col);
        }
    } else { // The very bottom alien is alive and needs to shoot
        // Time to return the starting position of the bullet!
        return (bottomAlien[r + ALIEN_COLUMNS].row + BULLET_COL_OFFSET + 1)
            * SCREEN_RES_X + (BULLET_COL_OFFSET + bottomAlien[r
                + ALIEN_COLUMNS].col);
    }
}

// This sees if either middle alien at index r is alive to shoot
int32_t fire_middle(uint32_t * framePointer, int32_t r) {
    if (!middleAlien[r + ALIEN_COLUMNS].alive) { // If the very bottom (middle) alien is
    dead
        if (!middleAlien[r].alive) { // AND the second row (middle) alien is dead
            return fire_top(framePointer, r); // Top row alien has to fire
        } else { // the bottom alien is dead, but the second-row one is alive
            // This is the starting coordinate of the bullet
            return (middleAlien[r].row + BULLET_COL_OFFSET) * SCREEN_RES_X
                + (BULLET_COL_OFFSET + middleAlien[r].col);
        }
    } else { // The bottom alien is alive and needs to fire
        // This is the starting coordinate of the bullet
        return (middleAlien[r + ALIEN_COLUMNS].row + BULLET_COL_OFFSET)
            * SCREEN_RES_X + (BULLET_COL_OFFSET + middleAlien[r
                + ALIEN_COLUMNS].col);
    }
}

// This sees to see if our top alien at index r is alive to shoot
int32_t fire_top(uint32_t * framePointer, int32_t r) {
    if (!top[r].alive) { // Our top alien is dead.
        return BAD_ADDRESS; // We failed to fire a missile! return -1
    } else { // Our alien is alive!
        return (top[r].row + BULLET_COL_OFFSET) * SCREEN_RES_X
            + (BULLET_COL_OFFSET + top[r].col); // Return good address
    }
}

```

# aliens.c

```
// Updates alien bullets. erases previous one, increments type, and redraws.
void aliens_update_bullets(uint32_t * framePointer) {
    int32_t i; // Declare loop var
    for (i = 0; i < ALIEN_NUM_BULLETS; i++) { // Cycle through all bullets
        if (alien_bullet[i].row > SCREEN_HEIGHT) { // If bullet off screen
            alien_bullet[i].alive = false; // kill it
        } else if (alien_bullet[i].alive) { // If bullet is alive
            draw_bullet(framePointer, i, BLACK); // erase to prep redraw
            if(tank_detect_collision(alien_bullet[i].row+BULLET_HEIGHT,
alien_bullet[i].col)){
                alien_bullet[i].alive = false;
                continue;
            }
            if(bunkers_detect_collision(alien_bullet[i].row +
BULLET_HEIGHT,alien_bullet[i].col, false)){
                alien_bullet[i].alive = false;
                continue;
            }
            if(alien_bullet[i].row == 220){
                alien_bullet[i].alive = false;
                continue;
            }

            switch (alien_bullet[i].bullet_type) { // Increment bullet type
                case cross0: // mid, going down
                    alien_bullet[i].bullet_type = cross1; // bar go down
                    break;
                case cross1: // down
                    alien_bullet[i].bullet_type = cross3; // bar go mid
                    break;
                case cross2: // up
                    alien_bullet[i].bullet_type = cross0; // bar go down
                    break;
                case cross3: // mid, going up
                    alien_bullet[i].bullet_type = cross2; // bar go up
                    break;
                case lightning0:// left lightning
                    alien_bullet[i].bullet_type = lightning1; // go right
                    break;
                case lightning1:// right lightning
                    alien_bullet[i].bullet_type = lightning0; // go left
                    break;
            }
            alien_bullet[i].row++; // Move bullet down
            draw_bullet(framePointer, i, WHITE); // redraw bullet
        }
    }
}

void aliens_delete_bottom(uint32_t location){
    int32_t row, col; // Declare vars
    for (row = 0; row < ALIEN_HEIGHT; row++) { // looping through y pixels
        for (col = 0; col < WORD_WIDTH; col++) { // looping through x pixels
            if (alien_bottom_out_12x8[row] & (1 << (WORD_WIDTH - col - 1))) {
                // If our alien is alive and has a pixel here, draw it
                util_draw_pixel(frame, row + bottomAlien[location].row, col +
bottomAlien[location].col,
                    BLACK);
            }
        }
    }
}
```

# aliens.c

```

    }
    if (alien_bottom_in_12x8[row] & (1 << (WORD_WIDTH - col - 1))) {
        // If our alien is alive and has a pixel here, draw it
        util_draw_pixel(frame, row + bottomAlien[location].row, col +
bottomAlien[location].col,
            BLACK);
    }
    if (deadAlien[row] & (1 << (WORD_WIDTH - col - 1))) {
        // If our alien is alive and has a pixel here, draw it
        util_draw_pixel(frame, row + bottomAlien[location].row, col +
bottomAlien[location].col,
            WHITE);
    }
}
}
}
}
void aliens_delete_top(uint32_t location){
    int32_t row, col; // Declare vars
    for (row = 0; row < ALIEN_HEIGHT; row++) { // looping through y pixels
        for (col = 0; col < WORD_WIDTH; col++) { // looping through x pixels
            if (alien_top_out_12x8[row] & (1 << (WORD_WIDTH - col - 1))) {
                // If our alien is alive and has a pixel here, draw it
                util_draw_pixel(frame, row + top[location].row, col + top[location].col,
                    BLACK);
            }
            if (alien_top_in_12x8[row] & (1 << (WORD_WIDTH - col - 1))) {
                // If our alien is alive and has a pixel here, draw it
                util_draw_pixel(frame, row + top[location].row, col + top[location].col,
                    BLACK);
            }
            if (deadAlien[row] & (1 << (WORD_WIDTH - col - 1))) {
                // If our alien is alive and has a pixel here, draw it
                util_draw_pixel(frame, row + top[location].row, col + top[location].col,
                    WHITE);
            }
        }
    }
}
void aliens_delete_middle(uint32_t location){
    int32_t row, col; // Declare vars
    for (row = 0; row < ALIEN_HEIGHT; row++) { // looping through y pixels
        for (col = 0; col < WORD_WIDTH; col++) { // looping through x pixels
            if (alien_middle_out_12x8[row] & (1 << (WORD_WIDTH - col - 1))) {
                // If our alien is alive and has a pixel here, draw it
                util_draw_pixel(frame, row + middleAlien[location].row, col +
middleAlien[location].col,
                    BLACK);
            }
            if (alien_middle_in_12x8[row] & (1 << (WORD_WIDTH - col - 1))) {
                // If our alien is alive and has a pixel here, draw it
                util_draw_pixel(frame, row + middleAlien[location].row, col +
middleAlien[location].col,
                    BLACK);
            }
            if (deadAlien[row] & (1 << (WORD_WIDTH - col - 1))) {
                // If our alien is alive and has a pixel here, draw it
                util_draw_pixel(frame, row + middleAlien[location].row, col +
middleAlien[location].col,

```



aliens.c

```

        WHITE);
    }
}

// Tank calls this to see if its bullet collides with an alien
bool aliens_detect_collision(uint32_t row,uint32_t col){
    if(row == (top[0].row + ALIEN_HEIGHT)){ //
        int i;
        for(i=0;i<ALIEN_COLUMNS;i++){
            if(top[i].alive && col > top[i].col && col < top[i].col + ALIEN_WIDTH){
                // The bullet has hit the bottom of our alien!
                interface_increment_score(TOP_POINTS);
                top[i].alive = false; // Kill the alien
                top[i].exploding = true;
                aliens_delete_top(i); // kill alien
                if(--how_many.aliens_left == 0){
                    interface_success();
                }
                return true; // We hit something!
            }
        }
    }
    if(row == (middleAlien[0].row + ALIEN_HEIGHT)){
        int i;
        for(i=0;i<ALIEN_COLUMNS;i++){
            if(middleAlien[i].alive &&
                col > middleAlien[i].col&&col<middleAlien[i].col+ALIEN_WIDTH){
                // The bullet has hit the bottom of our alien!
                interface_increment_score(MIDDLE_POINTS);
                middleAlien[i].alive = false; // Kill the alien
                middleAlien[i].exploding = true;
                aliens_delete_middle(i); // kill alien
                if(--how_many.aliens_left == 0){
                    interface_success();
                }
                return true; // We hit something!
            }
        }
    }
    if(row == (middleAlien[ALIEN_COLUMNS].row + ALIEN_HEIGHT)){
        int i;
        for(i=ALIEN_COLUMNS;i<ALIEN_COLUMNS+ALIEN_COLUMNS;i++){
            if(middleAlien[i].alive &&
                col>middleAlien[i].col && col < middleAlien[i].col + ALIEN_WIDTH){
                // The bullet has hit the bottom of our alien!
                interface_increment_score(MIDDLE_POINTS);
                aliens_delete_middle(i); // kill alien
                middleAlien[i].alive = false; // Kill the alien
                middleAlien[i].exploding = true;
                if(--how_many.aliens_left == 0){
                    interface_success();
                }
                return true; // We hit something!
            }
        }
    }
}

```

aliens.c

```

}
if(row == (bottomAlien[0].row + ALIEN_HEIGHT)){
    int i;
    for(i=0;i<ALIEN_COLUMNS;i++){
        if(bottomAlien[i].alive &&
           col > bottomAlien[i].col && col < bottomAlien[i].col + ALIEN_WIDTH){
            // The bullet has hit the bottom of our alien!
            interface_increment_score(BOTTOM_POINTS);
            aliens_delete_bottom(i); // kill alien
            bottomAlien[i].alive = false; // Kill the alien
            bottomAlien[i].exploding = true;
            if(--how_many.aliens_left == 0){
                interface_success();
            }
            return true; // We hit something!
        }
    }
}
if(row == (bottomAlien[ALIEN_COLUMNS].row + ALIEN_HEIGHT)){
    int i;
    for(i=ALIEN_COLUMNS;i<ALIEN_COLUMNS+ALIEN_COLUMNS;i++){
        if(bottomAlien[i].alive &&
           col > bottomAlien[i].col && col < bottomAlien[i].col + ALIEN_WIDTH){
            // The bullet has hit the bottom of our alien!
            interface_increment_score(BOTTOM_POINTS);
            aliens_delete_bottom(i); // kill alien
            bottomAlien[i].alive = false; // Kill the alien
            bottomAlien[i].exploding = true;
            if(--how_many.aliens_left == 0){
                interface_success();
            }
            return true; // We hit something!
        }
    }
}

// If we get here, the bullet is not at the row of any alien
return false; // No collision detected.
}

```