

# aliens.c

```

1 /*
2  * aliens.c
3  * Taylor Cowley and Andrew Okazaki
4  */
5
6 #include <stdio.h>
7 #include "platform.h"
8 #include "xparameters.h"
9 #include "xaxivdma.h"
10 #include "xio.h"
11 #include "time.h"
12 #include "unistd.h"
13 #include <stdbool.h>
14 #include <stdint.h>
15 #define ALIEN_HEIGHT 8          // Aliens are 8 pixels tall
16 #define ALIEN_COLUMNS 11       // 11 columns of aliens
17 #define TOP_TOTAL 11           // 11 aliens in top group
18 #define LOC_ALIEN_ONE 50       // Pixel where the first alien is
19 #define MIDDLE_TOTAL 22        // There are 22 total middle aliens
20 #define BOTTOM_TOTAL 22         // There are 22 total bottom aliens
21 #define ALIEN_NUM_BULLETS 4    // Aliens can have up to 4 bullets at a time
22 #define ALIEN_NUM_BULLET_TYPES 2 // Aliens have 2 types of bullets to choose from
23 #define BAD_ADDRESS -1         // Nothing exists at screen address -1
24 #define MOVE_DOWN_PIXELS 15    // When the aliens move down, they do so 15 pixels
25 #define LEFT_BOUNDARY 11       // Aliens cannot go more left than this
26 #define RIGHT_BOUNDARY 307     // Aliens cannot go more right than this
27 #define BULLET_COL_OFFSET 6    // Bullets appear 11 more right than their alien
28 #define BULLET_ROW_OFFSET 11   // Bullets appear more down than their alien
29 #define SCREEN_LENGTH 320      // Our screen is 320 pixels wide
30 #define SCREEN_HEIGHT 240      // Our screen is 240 pixels tall
31 #define SCREEN_RES_X 640       // Our screen RESOLUTION is 640 pixels wide
32 #define SCREEN_RES_Y 480       // Our screen RESOLUTION is 480 pixels tall
33 #define WHITE 0xFFFFFFFF        // These
34 #define BLACK 0x00000000        // are colors
35 #define WORD_WIDTH 12
36
37 // Packs each horizontal line of the figures into a single 32 bit word.
38 #define packword12(b11,b10,b9,b8,b7,b6,b5,b4,b3,b2,b1,b0) \
39     ((b11 << 11) | (b10 << 10) | (b9 << 9) | (b8 << 8) | (b7 << 7) | (b6 <<
40     6) | \
41     | (b5 << 5) | (b4 << 4) | (b3 << 3) | (b2 << 2) | (b1 << 1) |
42     | (b0 << 0) )
43
44 // -----
45 // The following static const ints define the aliens
46 // We have 3 types of aliens with 2 poses each
47 static const int32_t alien_top_in_12x8[ALIEN_HEIGHT] = {
48     packword12(0,0,0,0,0,0,1,1,0,0,0,0),
49     packword12(0,0,0,0,0,1,1,1,1,0,0,0),
50     packword12(0,0,0,1,1,1,1,1,1,0,0,0),
51     packword12(0,0,1,1,0,1,1,0,1,1,0,0),
52     packword12(0,0,1,1,1,1,1,1,1,1,0,0),
53     packword12(0,0,0,1,0,1,1,0,1,0,0,0),
54     packword12(0,0,1,0,0,0,0,0,0,1,0,0),
55     packword12(0,0,0,1,0,0,0,0,1,0,0,0) };
56 static const int32_t alien_top_out_12x8[ALIEN_HEIGHT] = {
57     packword12(0,0,0,0,0,0,1,1,0,0,0,0),
58     packword12(0,0,0,0,1,1,1,1,0,0,0,0),

```

# aliens.c

```

57     packword12(0,0,0,1,1,1,1,1,1,0,0,0),
58     packword12(0,0,1,1,0,1,1,0,1,1,0,0),
59     packword12(0,0,1,1,1,1,1,1,1,1,0,0),
60     packword12(0,0,0,0,1,0,0,1,0,0,0,0),
61     packword12(0,0,0,1,0,1,1,0,1,0,0,0),
62     packword12(0,0,1,0,1,0,0,1,0,1,0,0) };
63 static const int32_t alien_middle_in_12x8[ALIEN_HEIGHT] = {
64     packword12(0,0,0,1,0,0,0,0,0,1,0,0),
65     packword12(0,0,0,0,1,0,0,0,1,0,0,0),
66     packword12(0,0,0,1,1,1,1,1,1,1,0,0),
67     packword12(0,0,1,1,0,1,1,1,0,1,1,0),
68     packword12(0,1,1,1,1,1,1,1,1,1,1,1),
69     packword12(0,1,1,1,1,1,1,1,1,1,1,1),
70     packword12(0,1,0,1,0,0,0,0,0,1,0,1),
71     packword12(0,0,0,0,1,1,0,1,1,0,0,0) };
72 static const int32_t alien_middle_out_12x8[] = {
73     packword12(0,0,0,1,0,0,0,0,0,1,0,0),
74     packword12(0,1,0,0,1,0,0,0,1,0,0,1),
75     packword12(0,1,0,1,1,1,1,1,1,1,0,1),
76     packword12(0,1,1,1,0,1,1,1,0,1,1,1),
77     packword12(0,1,1,1,1,1,1,1,1,1,1,1),
78     packword12(0,0,1,1,1,1,1,1,1,1,1,0),
79     packword12(0,0,0,1,0,0,0,0,0,1,0,0),
80     packword12(0,0,1,0,0,0,0,0,0,0,1,0) };
81 static const int32_t alien_bottom_in_12x8[ALIEN_HEIGHT] = {
82     packword12(0,0,0,0,1,1,1,1,0,0,0,0),
83     packword12(0,1,1,1,1,1,1,1,1,1,1,0),
84     packword12(1,1,1,1,1,1,1,1,1,1,1,1),
85     packword12(1,1,1,0,0,1,1,0,0,1,1,1),
86     packword12(1,1,1,1,1,1,1,1,1,1,1,1),
87     packword12(0,0,1,1,1,0,0,1,1,1,0,0),
88     packword12(0,1,1,0,0,1,1,0,0,1,1,0),
89     packword12(0,0,1,1,0,0,0,0,1,1,0,0) };
90 static const int32_t alien_bottom_out_12x8[] = {
91     packword12(0,0,0,0,1,1,1,1,0,0,0,0),
92     packword12(0,1,1,1,1,1,1,1,1,1,1,0),
93     packword12(1,1,1,1,1,1,1,1,1,1,1,1),
94     packword12(1,1,1,0,0,1,1,0,0,1,1,1),
95     packword12(1,1,1,1,1,1,1,1,1,1,1,1),
96     packword12(0,0,0,1,1,0,0,1,1,0,0,0),
97     packword12(0,0,1,1,0,1,1,0,1,1,0,0),
98     packword12(1,1,0,0,0,0,0,0,0,0,1,1) };
99 // End of the const ints that define the alien pixels
100 // -----
101
102 // -----
103 // These are our internal methods, used only by ourselves
104 // Draws the aliens on the screen - top, middle, and bottom aliens
105 void build_tops(uint32_t * framePointer, const int32_t alien_top[]);
106 void build_middle(uint32_t * framePointer, const int32_t alien_middle[]);
107 void build_bottom(uint32_t * framePointer, const int32_t alien_bottom[]);
108 // Fire a bullet from either a top, middle, or bottom alien
109 int32_t fire_bottom(uint32_t * framePointer, int32_t r);
110 int32_t fire_middle(uint32_t * framePointer, int32_t r);
111 int32_t fire_top(uint32_t * framePointer, int32_t r);
112 // Checks to see whether our aliens are currently capable of shooting
113 bool can.aliens.shoot();
114 // Draws a bullet on the screen

```

# aliens.c

```

115 void draw_bullet(uint32_t * framePointer, int32_t bullet, uint32_t color);
116 // Draws a pixel on the screen.
117 void aliens_draw_pixel(uint32_t *framePointer, uint32_t row, uint32_t col,
118     uint32_t color);
119 // End internal method declarations
120 // -----
121
122 // These structs hold all of our aliens.
123 struct top { // Struct for our top aliens
124     int32_t row;
125     int32_t col; bool alive; // alien has row, column, and alive?
126 } top[TOP_TOTAL];
127
128 struct middleAlien { // Struct for our middle aliens
129     int32_t row;
130     int32_t col; bool alive; // alien has row, column, and alive?
131 } middleAlien[MIDDLE_TOTAL];
132
133 struct bottomAlien { // Struct for our bottom aliens
134     int32_t row;
135     int32_t col; bool alive; // alien has row, column, and alive?
136 } bottomAlien[MIDDLE_TOTAL];
137
138 // aliens can have two types of bullet: cross and lightning
139 // cross 0 and 3 are identical
140 typedef enum {
141     cross0, cross1, cross2, cross3, lightning0, lightning1
142 } bullet_type;
143 struct alien_bullet { // Struct that holds our aliens' bullets
144     int32_t row;
145     int32_t col; bool alive; // Bullets have coordinates and alive?
146     bullet_type bullet_type; // Bullets also have a type.
147 } alien_bullet[ALIEN_NUM_BULLETS];
148
149 int32_t alien_count; // a count of how many aliens are alive
150
151 /*
152  * Draws a pixel on the screen. To compensate for our double-resolution screen,
153  * it must draw 4 real pixels for every in-came pixel.
154  */
155 void aliens_draw_pixel(uint32_t *framePointer, uint32_t row, uint32_t col,
156     uint32_t color) {
157 #define DRAW_PIXEL_ROW_MULTIPLIER 1280 // 640 * 2 for screen doubling
158 #define DRAW_PIXEL_ROW 640 // one row offset
159 #define DRAW_PIXEL_DOUBLE 2 // for doubling
160 // We draw 4 pixels for every 1 small-screen pixel
161 framePointer[row * DRAW_PIXEL_ROW_MULTIPLIER + col * DRAW_PIXEL_DOUBLE]
162     = color;
163 framePointer[row * DRAW_PIXEL_ROW_MULTIPLIER + col * DRAW_PIXEL_DOUBLE + 1]
164     = color;
165 framePointer[row * DRAW_PIXEL_ROW_MULTIPLIER + DRAW_PIXEL_ROW + col
166     * DRAW_PIXEL_DOUBLE] = color;
167 framePointer[row * DRAW_PIXEL_ROW_MULTIPLIER + DRAW_PIXEL_ROW + col
168     * DRAW_PIXEL_DOUBLE + 1] = color;
169 }
170
171 // initialize all of the aliens by setting values contained in struct's and printing
    aliens to the screen

```

# aliens.c

```

172 void aliens_init(uint32_t * framePointer) {
173 #define ALIEN_TOP_ROW_INIT 30           // Where
174 #define ALIEN_MIDDLE_ROW_INIT 45        // the
175 #define ALIEN_MIDDLE2_ROW_INIT 60       // aliens
176 #define ALIEN_BOTTOM_ROW_INIT 75        // are
177 #define ALIEN_BOTTOM2_ROW_INIT 90       // initialized to
178 #define ALIEN_SPACING 15                // Spacing between aliens
179     //local variables, loc is the starting location of alien one on the screen
180     int32_t i, loc = LOC_ALIEN_ONE;
181
182     //loops through one row of aliens
183     for (i = 0; i < ALIEN_COLUMNS; i++) {
184         top[i].row = ALIEN_TOP_ROW_INIT; //set the row of alien tops to 30
185         top[i].col = loc; //sets the column of alien tops
186         top[i].alive = true; //sets the alien is alive flag
187
188         middleAlien[i].row = ALIEN_MIDDLE_ROW_INIT; //middle aliens
189         middleAlien[i].col = loc; //sets column of first row of middle aliens
190         middleAlien[i].alive = true; //sets first row of middle aliens to alive
191         middleAlien[i + ALIEN_COLUMNS].row = ALIEN_MIDDLE2_ROW_INIT; //sets middle
192         middleAlien[i + ALIEN_COLUMNS].col = loc; //sets column second row middle
193         middleAlien[i + ALIEN_COLUMNS].alive = true; //sets second row middle alive
194
195         bottomAlien[i].row = ALIEN_BOTTOM_ROW_INIT; //sets bottom aliens
196         bottomAlien[i].col = loc; //sets column of first row of bottom aliens
197         bottomAlien[i].alive = true; //sets first row of bottom aliens to alive
198         bottomAlien[i + ALIEN_COLUMNS].row = ALIEN_BOTTOM2_ROW_INIT; //bottom
199         bottomAlien[i + ALIEN_COLUMNS].col = loc; //sets column second row bottom
200         bottomAlien[i + ALIEN_COLUMNS].alive = true; //sets second row bottom alive
201         loc += ALIEN_SPACING; //controls the column spacing in-between alien
202     }
203
204     //now that structs are built draw top, middle, and bottom aliens to screen
205     build_tops(framePointer, alien_top_in_12x8); // Top
206     build_middle(framePointer, alien_middle_in_12x8); // Middle
207     build_bottom(framePointer, alien_bottom_in_12x8); // Bottom
208 }
209
210 // Draws the top aliens on the screen
211 void build_tops(uint32_t * framePointer, const int32_t alien_top[]) {
212     int32_t row, col, i; // initialize variables
213     for (i = 0; i < TOP_TOTAL; i++) { //loop through top column of aliens
214         for (row = 0; row < ALIEN_HEIGHT; row++) { //loop top aliens' pixels row
215             int32_t currentRow = row + top[i].row; // current pixel row of alien
216             for (col = 0; col < WORD_WIDTH; col++) { //loop alien's pixel col
217                 int32_t currentCol = col + top[i].col; //current col of alien
218                 if ((alien_top[row] & (1 << (WORD_WIDTH - col - 1)))
219                     && top[i].alive) {
220                     // If our alien is alive and has a pixel there, draw it
221                     aliens_draw_pixel(framePointer, currentRow, currentCol,
222                                         WHITE);
223                 } else { // If not, erase it.
224                     aliens_draw_pixel(framePointer, currentRow, currentCol,
225                                         BLACK);
226                 }
227             }
228         }
229     }
}

```

# aliens.c

```

230 }
231
232 // Draws the middle aliens to the screen
233 void build_middle(uint32_t * framePointer, const int32_t alien_middle[]) {
234     int32_t row, col, i; // declare our variables
235     for (i = 0; i < MIDDLE_TOTAL; i++) { // Looping through all the middle aliens
236         for (row = 0; row < ALIEN_HEIGHT; row++) { // Pixel y
237             int32_t currentRow = row + middleAlien[i].row; // current pixel row
238             for (col = 0; col < WORD_WIDTH; col++) { // Pixel x
239                 int32_t currentCol = col + middleAlien[i].col; // current col alien
240                 if ((alien_middle[row] & (1 << (WORD_WIDTH - col - 1)))
241                     && middleAlien[i].alive) {
242                     // If our alien is alive and has a pixel there, draw it
243                     aliens_draw_pixel(framePointer, currentRow, currentCol,
244                                     WHITE);
245                 } else { // Otherwise, erase it.
246                     aliens_draw_pixel(framePointer, currentRow, currentCol,
247                                     BLACK);
248                 }
249             }
250         }
251     }
252 }
253
254 // Draws the bottom aliens to the screen
255 void build_bottom(uint32_t * framePointer, const int32_t alien_bottom[]) {
256     int32_t row, col, i; // Declare vars
257     for (i = 0; i < BOTTOM_TOTAL; i++) { // Looping through all the bottom aliens
258         for (row = 0; row < ALIEN_HEIGHT; row++) { // looping through y pixels
259             int32_t currentRow = row + bottomAlien[i].row; // current row
260             for (col = 0; col < WORD_WIDTH; col++) { // looping through x pixels
261                 int32_t currentCol = col + bottomAlien[i].col; // current col
262                 if ((alien_bottom[row] & (1 << (WORD_WIDTH - col - 1)))
263                     && bottomAlien[i].alive) {
264                     // If our alien is alive and has a pixel here, draw it
265                     aliens_draw_pixel(framePointer, currentRow, currentCol,
266                                     WHITE);
267                 } else { // otherwise, erase it.
268                     aliens_draw_pixel(framePointer, currentRow, currentCol,
269                                     BLACK);
270                 }
271             }
272         }
273     }
274 }
275
276 // Does the needful to move the aliens left
277 void aliens_left(uint32_t * framePointer) {
278     int32_t i, row; // Declare loop vars
279     for (i = 0; i < MIDDLE_TOTAL; i++) { // Move every single alien LEFT
280         if (i < TOP_TOTAL) {
281             top[i].col--;
282         } // Move the top aliens LEFT
283         middleAlien[i].col--; // Move the middle aliens LEFT
284         bottomAlien[i].col--; // Move the bottom aliens LEFT
285     }
286     if (alien_count == 0) { // If aliens are out, make them in
287         alien_count = 1;

```

# aliens.c

```

288     build_tops(framePointer, alien_top_in_12x8); // Draw top aliens
289     build_middle(framePointer, alien_middle_in_12x8); // Draw mid aliens
290     build_bottom(framePointer, alien_bottom_in_12x8); // Draw bot aliens
291 } else { // And vice versa
292     alien_count = 0;
293     build_tops(framePointer, alien_top_out_12x8); // Draw top aliens
294     build_middle(framePointer, alien_middle_out_12x8); // Draw mid aliens
295     build_bottom(framePointer, alien_bottom_out_12x8); // Draw bot aliens
296 }
297
298 for (row = 0; row < ALIEN_HEIGHT; row++) { // For all the alien Y pixels
299     for (i = 0; i < MIDDLE_TOTAL; i++) { // For every alien
300         // Erase them for the middle and bottom aliens - top is skinnier
301         aliens_draw_pixel(framePointer, row + bottomAlien[i].row,
302             WORD_WIDTH + bottomAlien[i].col, BLACK);
303         aliens_draw_pixel(framePointer, row + middleAlien[i].row,
304             WORD_WIDTH + middleAlien[i].col, BLACK);
305     }
306 }
307 }
308 }
309
310 // Does the needful to move the aliens right
311 void aliens_right(uint32_t * framePointer) {
312     int32_t i, row; // Declare loop vars
313     for (i = 0; i < MIDDLE_TOTAL; i++) { // Move every single alien RIGHT
314         if (i < 11) {
315             top[i].col += 1;
316         } // Move top aliens RIGHT
317         middleAlien[i].col += 1; // Move middle aliens RIGHT
318         bottomAlien[i].col += 1; // Move bottom aliens RIGHT
319     }
320
321     if (alien_count == 0) { // If aliens are out, make them in
322         alien_count = 1;
323         build_tops(framePointer, alien_top_in_12x8); // Draw top aliens
324         build_middle(framePointer, alien_middle_in_12x8); // Draw mid aliens
325         build_bottom(framePointer, alien_bottom_in_12x8); // Draw bot aliens
326     } else { // And vice versa
327         alien_count = 0;
328         build_tops(framePointer, alien_top_out_12x8); // Draw top aliens
329         build_middle(framePointer, alien_middle_out_12x8); // Draw mid aliens
330         build_bottom(framePointer, alien_bottom_out_12x8); // Draw bot aliens
331     }
332
333     for (row = 0; row < ALIEN_HEIGHT; row++) { // For all the alien Y pixels
334         for (i = 0; i < MIDDLE_TOTAL; i++) { // For every alien
335             // Erase that column of pixels for mid and bottom. Top not necessary
336             aliens_draw_pixel(framePointer, row + bottomAlien[i].row,
337                 bottomAlien[i].col - 1, BLACK); // Notice it's col-1 bottom
338             aliens_draw_pixel(framePointer, row + middleAlien[i].row,
339                 middleAlien[i].col, BLACK);
340         }
341     }
342 }
343
344 // Does the needful when aliens hit the left rail
345 void hit_left_rail(uint32_t * framePointer) {

```

# aliens.c

```

346 // First we erase the entire top row of alien pixels for moving down.
347 int32_t col, row, i; // declare loop vars
348 for (row = 0; row < ALIEN_HEIGHT; row++) { // Go through alien pixels Y
349     for (col = 0; col < WORD_WIDTH; col++) { // Go through alien pixels X
350         if (((alien_top_out_l2x8[row] | alien_top_in_l2x8[row]) & (1
351             << (WORD_WIDTH - col - 1)))) { // if pixel exists here
352             for (i = 0; i < TOP_TOTAL; i++) { // ERASE IT!
353                 aliens_draw_pixel(framePointer, row + top[i].row,
354                     col + top[i].col, BLACK);
355             }
356         }
357     }
358 }
359 for (i = 0; i < MIDDLE_TOTAL; i++) { // For all the aliens, move them down
360     if (i < TOP_TOTAL) {
361         top[i].row += MOVE_DOWN_PIXELS;
362     } // Move top aliens down
363     middleAlien[i].row += MOVE_DOWN_PIXELS; // Move mid aliens down
364     bottomAlien[i].row += MOVE_DOWN_PIXELS; // Move bot aliens down
365 }
366 for (row = 0; row < ALIEN_HEIGHT; row++) { // Now to erase pixels on left side
367     for (i = 0; i < MIDDLE_TOTAL; i++) { // For all the middle aliens
368         aliens_draw_pixel(framePointer, row + middleAlien[i].row,
369             middleAlien[i].col, BLACK); // Erase the pixels on the left
370     }
371 }
372 }
373
374 // Does the needful when aliens hit the right rail
375 void hit_right_rail(uint32_t * framePointer) {
376     // First we erase the entire top row of alien pixels for moving down
377     int32_t col, row, i; // Declare loop vars
378     for (row = 0; row < ALIEN_HEIGHT; row++) { // Go through alien pixels Y
379         for (col = 0; col < WORD_WIDTH; col++) { // Go through alien pixels X
380             if (((alien_top_out_l2x8[row] | alien_top_in_l2x8[row]) & (1
381                 << (WORD_WIDTH - col - 1)))) { // if pixel exists here
382                 for (i = 0; i < TOP_TOTAL; i++) { // Erase it!
383                     aliens_draw_pixel(framePointer, row + top[i].row,
384                         col + top[i].col, BLACK);
385                 }
386             }
387         }
388     }
389     for (i = 0; i < MIDDLE_TOTAL; i++) { // For all the aliens, move them down
390         if (i < TOP_TOTAL) {
391             top[i].row += MOVE_DOWN_PIXELS;
392         } // Move top aliens down
393         middleAlien[i].row += MOVE_DOWN_PIXELS; // Move mid aliens down
394         bottomAlien[i].row += MOVE_DOWN_PIXELS; // Move bot aliens down
395     }
396     for (row = 0; row < ALIEN_HEIGHT; row++) { // Now to erase pixels on the right
side
397         for (i = 0; i < TOP_TOTAL; i++) { // Erase the pixels on the right
398             aliens_draw_pixel(framePointer, row + top[i].row,
399                 WORD_WIDTH - 1 + top[i].col, BLACK);
400         }
401     }
402 }

```

# aliens.c

```

403
404 // moves the aliens and detects wall boundries and direction changes too!
405 void aliens_move(uint32_t * framePointer) {
406     static int32_t flag;
407     int32_t i, j;
408     for (i = 0; i < ALIEN_COLUMNS; i++) { // Go through every alien column
409         // And see if any alien in that column is alive and has hit left
410         if (top[i].alive || middleAlien[i].alive || middleAlien[i
411             + ALIEN_COLUMNS].alive || bottomAlien[i].alive || bottomAlien[i
412             + ALIEN_COLUMNS].alive) {
413             if (top[i].col == LEFT_BOUNDRY) { // If an alien has hit side
414                 flag = 1; // Set the flag that we've hit the side
415                 hit_left_rail(framePointer); // Call hit_rail.
416             }
417         }
418     }
419     for (j = ALIEN_COLUMNS - 1; j >= 0; j--) { // Now to check to see
420         if (top[j].alive || middleAlien[j].alive || middleAlien[j
421             + ALIEN_COLUMNS].alive || bottomAlien[j].alive || bottomAlien[j
422             + ALIEN_COLUMNS].alive) {
423             if (top[j].col == RIGHT_BOUNDRY) { // if an alien has hit right.
424                 flag = 0; // false
425                 hit_right_rail(framePointer); // we have hit the right rail
426             }
427         }
428     }
429     if (flag == 1) { // if we are moving right
430         aliens_right(framePointer); // go right
431     } else { // we are actually going left
432         aliens_left(framePointer); // so go left
433     }
434 }
435
436 // Kills a random alien
437 // Currently has a bug that if the last alien dies, infinite loop
438 void aliens_kill(uint32_t * framePointer) {
439     int32_t r = rand() % 55; // Get a random number
440
441     if (r < TOP_TOTAL) { // If we have killed a top
442         if (!top[r].alive) { // Already dead!
443             aliens_kill(framePointer); // Try again
444         } else {
445             top[r].alive = false; // kill the alien
446             build_tops(framePointer, alien_top_in_12x8); // redraw aliens
447         }
448     } else if (r < (TOP_TOTAL + MIDDLE_TOTAL)) { // if we have killed a mid
449         if (!middleAlien[r - TOP_TOTAL].alive) { // Already dead!
450             aliens_kill(framePointer); // try again
451         } else {
452             middleAlien[r - TOP_TOTAL].alive = false; // kill alien
453             build_middle(framePointer, alien_middle_in_12x8); // redraw aliens
454         }
455     } else { // we have killed a bot
456         if (!bottomAlien[r - (TOP_TOTAL + MIDDLE_TOTAL)].alive) { // Already dead!
457             aliens_kill(framePointer); // Try again
458         } else {
459             bottomAlien[r - (TOP_TOTAL + MIDDLE_TOTAL)].alive = false; // Kill alien
460             build_bottom(framePointer, alien_bottom_in_12x8); // redraw aliens

```



aliens.c

```

461     }
462 }
463 }
464
465 // Returns true if aliens can shoot- that is, if there exists a top alive alien
466 bool can_aliens_shoot() {
467     int32_t i; // Declare loop variable
468     for (i = 0; i < TOP_TOTAL; i++) { // Look at all the top aliense
469         if (top[i].alive) { // If there exists a single alive top alien
470             return true; // We have an alive alien!
471         }
472     }
473     return false; // All the top aliens are dead; we cannot shoot
474 }
475
476 // Fires a bullet from a random alien
477 void alien_missile(uint32_t * framePointer) {
478     if (!can_aliens_shoot()) { // The aliens can't even shoot! Don't even try.
479         return;
480     }
481
482     int32_t r = rand() % ALIEN_COLUMNS; // Get a random column
483     int32_t bullet_address = BAD_ADDRESS; // Initialize the address
484     do { // Keep trying to shoot
485         bullet_address = fire_bottom(framePointer, r);
486     } while (bullet_address == BAD_ADDRESS); // until we get a good address
487
488     // We have a bullet address! now to make it alive and draw it.
489     int32_t i;
490     for (i = 0; i < ALIEN_NUM_BULLETS; i++) {
491         if (alien_bullet[i].alive) { // If we already have a living bullet
492             continue; // Go on to the next one
493         } else { // We have a dead bullet spot- let's alive a bullet here!
494             alien_bullet[i].alive = true;
495             // Randomly choose a bullet type
496             alien_bullet[i].bullet_type
497                 = rand() % ALIEN_NUM_BULLET_TYPES ? cross0 : lightning0;
498             // TODO: This math can be simplified
499             alien_bullet[i].col = bullet_address % SCREEN_RES_X; // Set address
500             alien_bullet[i].row = bullet_address / SCREEN_RES_X; // of bullet
501             draw_bullet(framePointer, i, WHITE); // And draw it!
502             return;
503         }
504     }
505 }
506
507 // Draws the selected bullet to the screen
508 void draw_bullet(uint32_t * framePointer, int32_t bullet, uint32_t color) {
509     #define PIXEL_LINE_1 1 // These
510     #define PIXEL_LINE_2 2 // defines
511     #define PIXEL_LINE_3 3 // only
512     #define PIXEL_LINE_4 4 // have
513     #define PIXEL_LEFT -1 // meaning
514     #define PIXEL_RIGHT 1 // in this function, so I put them here
515     uint32_t row = alien_bullet[bullet].row; // Current row
516     uint32_t col = alien_bullet[bullet].col; // and column where to draw
517     switch (alien_bullet[bullet].bullet_type) {
518     case cross0: // Cross0 and cross 3 are identically drawn

```

# aliens.c

```

519  case cross3: // The only difference is in the state machine where they go
520      // 5 pixels down in a line
521      aliens_draw_pixel(framePointer, row, col, color);
522      aliens_draw_pixel(framePointer, row + PIXEL_LINE_1, col, color);
523      aliens_draw_pixel(framePointer, row + PIXEL_LINE_2, col, color);
524      aliens_draw_pixel(framePointer, row + PIXEL_LINE_3, col, color);
525      aliens_draw_pixel(framePointer, row + PIXEL_LINE_4, col, color);
526
527      // Crossbar on the cross - right in the middle
528      aliens_draw_pixel(framePointer, row + PIXEL_LINE_2, col + PIXEL_RIGHT,
529          color);
530      aliens_draw_pixel(framePointer, row + PIXEL_LINE_2, col + PIXEL_LEFT,
531          color);
532      break;
533  case cross1:
534      // 5 pixels down in a line
535      aliens_draw_pixel(framePointer, row, col, color);
536      aliens_draw_pixel(framePointer, row + PIXEL_LINE_1, col, color);
537      aliens_draw_pixel(framePointer, row + PIXEL_LINE_2, col, color);
538      aliens_draw_pixel(framePointer, row + PIXEL_LINE_3, col, color);
539      aliens_draw_pixel(framePointer, row + PIXEL_LINE_4, col, color);
540
541      // Crossbar on the cross- on the lower one
542      aliens_draw_pixel(framePointer, row + PIXEL_LINE_3, col + PIXEL_RIGHT,
543          color);
544      aliens_draw_pixel(framePointer, row + PIXEL_LINE_3, col + PIXEL_LEFT,
545          color);
546      break;
547  case cross2:
548      // 5 pixels down in a line
549      aliens_draw_pixel(framePointer, row, col, color);
550      aliens_draw_pixel(framePointer, row + PIXEL_LINE_1, col, color);
551      aliens_draw_pixel(framePointer, row + PIXEL_LINE_2, col, color);
552      aliens_draw_pixel(framePointer, row + PIXEL_LINE_3, col, color);
553      aliens_draw_pixel(framePointer, row + PIXEL_LINE_4, col, color);
554
555      // Crossbar on the cross- on the upper one
556      aliens_draw_pixel(framePointer, row + PIXEL_LINE_1, col + PIXEL_RIGHT,
557          color);
558      aliens_draw_pixel(framePointer, row + PIXEL_LINE_1, col + PIXEL_LEFT,
559          color);
560      break;
561  case lightning0:
562      // 5 pixels down - starting left then right, then going back left
563      aliens_draw_pixel(framePointer, row, col + PIXEL_LEFT, color);
564      aliens_draw_pixel(framePointer, row + PIXEL_LINE_1, col, color);
565      aliens_draw_pixel(framePointer, row + PIXEL_LINE_2, col + PIXEL_RIGHT,
566          color);
567      aliens_draw_pixel(framePointer, row + PIXEL_LINE_3, col, color);
568      aliens_draw_pixel(framePointer, row + PIXEL_LINE_4, col + PIXEL_LEFT,
569          color);
570      break;
571  case lightning1:
572      // 5 pixels down - starting right then left, then back right
573      aliens_draw_pixel(framePointer, row, col + PIXEL_RIGHT, color);
574      aliens_draw_pixel(framePointer, row + PIXEL_LINE_1, col, color);
575      aliens_draw_pixel(framePointer, row + PIXEL_LINE_2, col + PIXEL_LEFT,
576          color);

```

# aliens.c

```

577     aliens_draw_pixel(framePointer, row + PIXEL_LINE_3, col, color);
578     aliens_draw_pixel(framePointer, row + PIXEL_LINE_4, col + PIXEL_RIGHT,
579         color);
580     break;
581 }
582
583 }
584
585 // This sees if our bottom alien at index r is alive to shoot
586 int32_t fire_bottom(uint32_t * framePointer, int32_t r) {
587     if (!bottomAlien[r + ALIEN_COLUMNS].alive) { // If the very bottom alien is dead
588         if (!bottomAlien[r].alive) { // AND the second row alien is also dead
589             return fire_middle(framePointer, r); // Try to make a higher alien shoot
590         }
591         // This is the starting coordinate of the bullet.
592         return (bottomAlien[r].row + BULLET_COL_OFFSET + 1) * SCREEN_RES_X
593             + (BULLET_COL_OFFSET + bottomAlien[r].col);
594     }
595     // The very bottom alien is alive and needs to shoot
596     // Time to return the starting position of the bullet!
597     return (bottomAlien[r + ALIEN_COLUMNS].row + BULLET_COL_OFFSET + 1)
598         * SCREEN_RES_X + (BULLET_COL_OFFSET + bottomAlien[r
599             + ALIEN_COLUMNS].col);
600 }
601 }
602
603 // This sees if either middle alien at index r is alive to shoot
604 int32_t fire_middle(uint32_t * framePointer, int32_t r) {
605     if (!middleAlien[r + ALIEN_COLUMNS].alive) { // If the very bottom (middle) alien
606         // is dead
607         if (!middleAlien[r].alive) { // AND the second row (middle) alien is dead
608             return fire_top(framePointer, r); // Top row alien has to fire
609         }
610         // This is the starting coordinate of the bullet
611         return (middleAlien[r].row + BULLET_COL_OFFSET) * SCREEN_RES_X
612             + (BULLET_COL_OFFSET + middleAlien[r].col);
613     }
614     // The bottom alien is alive and needs to fire
615     // This is the starting coordinate of the bullet
616     return (middleAlien[r + ALIEN_COLUMNS].row + BULLET_COL_OFFSET)
617         * SCREEN_RES_X + (BULLET_COL_OFFSET + middleAlien[r
618             + ALIEN_COLUMNS].col);
619 }
620
621 // This sees to see if our top alien at index r is alive to shoot
622 int32_t fire_top(uint32_t * framePointer, int32_t r) {
623     if (!top[r].alive) { // Our top alien is dead.
624         return BAD_ADDRESS; // We failed to fire a missile! return -1
625     }
626     // Our alien is alive!
627     return (top[r].row + BULLET_COL_OFFSET) * SCREEN_RES_X
628         + (BULLET_COL_OFFSET + top[r].col); // Return good address
629 }
630
631 // Updates alien bullets. erases previous one, increments type, and redraws.
632 void aliens_update_bullets(uint32_t * framePointer) {

```

aliens.c

```
633     int32_t i; // Declare loop var
634     for (i = 0; i < ALIEN_NUM_BULLETS; i++) { // Cycle through all bullets
635         if (alien_bullet[i].row > SCREEN_HEIGHT) { // If bullet off screen
636             alien_bullet[i].alive = false; // kill it
637         } else if (alien_bullet[i].alive) { // If bullet is alive
638             draw_bullet(framePointer, i, BLACK); // erase to prep redraw
639
640             switch (alien_bullet[i].bullet_type) { // Increment bullet type
641                 case cross0: // mid, going down
642                     alien_bullet[i].bullet_type = cross1; // bar go down
643                     break;
644                 case cross1: // down
645                     alien_bullet[i].bullet_type = cross3; // bar go mid
646                     break;
647                 case cross2: // up
648                     alien_bullet[i].bullet_type = cross0; // bar go down
649                     break;
650                 case cross3: // mid, going up
651                     alien_bullet[i].bullet_type = cross2; // bar go up
652                     break;
653                 case lightning0: // left lightning
654                     alien_bullet[i].bullet_type = lightning1; // go right
655                     break;
656                 case lightning1: // right lightning
657                     alien_bullet[i].bullet_type = lightning0; // go left
658                     break;
659             }
660             alien_bullet[i].row++; // Move bullet down
661             draw_bullet(framePointer, i, WHITE); // redraw bullet
662         }
663     }
664 }
665
```