```c
1    #include "address_map_arm.h"
2    #include <stdbool.h>
3    #include <stdlib.h>
4    #include <stdio.h>
5
6    volatile int pixel_buffer_start; // global variable
7
8    void plot_pixel(int x, int y, short int line_color)
9    {
10       *(short int *)(pixel_buffer_start + (y << 10) + (x << 1)) = line_color;
11   }
12
13   // Swaps 2 numbers using the XOR operation
14   void swap(int * x, int * y)
15   {
16       int temp = *x;
17       *x = *y;
18       *y = temp;
19   }
20
21   void draw_line(int x1, int y1, int x2, int y2, short int colour)
22   {
23       // Check steepness of the line, if it is steep, it's better
24       // to move along the y-axis when drawing
25       bool is_steep = abs(y2-y1) > abs(x2-x1);
26       // If it is steep switch the x and y values
27       // the drawing loop will decide how the drawing will occur
28       if(is_steep) {
29           swap(&x1,&y1);
30           swap(&x2,&y2);
31       }
32
33       // We are going to increment from x1 to x2 so
34       // swap the endpoints if x1 > x2
35       if(x1 > x2) {
36           swap(&x1,&x2);
37           swap(&y1,&y2);
38       }
39
40       int deltax = x2-x1;
41       int deltay = abs(y2-y1);
42       int error = -(deltax/2);
43       int x,y,y_step;
44
45       // Figure out how y will be incremented
46       if(y1<y2) y_step = 1;
47       else y_step = -1;
48
49       for(x=x1,y=y1; x<=x2; x++) {
50           // If the line is steep the x and y values are swapped
51           if(is_steep) plot_pixel(y,x,colour);
52           else plot_pixel(x,y,colour);
53
54           // Check margin of error
55           error += deltay;
56           if(error>=0) {
57               y += y_step; // Increment y val
58               error -= deltax; // Reset error
59           }
60       }
61   }
62
63   // Draw black to every pixel on the screen
64   void clear_screen()
65   {
66       int x,y;
67       // The screen is 320x240
68       for(x=0; x<320; x++) {
69           for(y=0; y<240; y++) {
```

```
70                     plot_pixel(x,y,0x0000);
71             }
72         }
73     }
74
75     int main()
76     {
77         volatile int * pixel_ctrl_ptr = (int *)PIXEL_BUF_CTRL_BASE;
78         /* Read location of the pixel buffer from the pixel buffer controller */
79         pixel_buffer_start = *pixel_ctrl_ptr;
80
81         clear_screen();
82         draw_line(0, 0, 150, 150, 0x001F);    // this line is blue
83         draw_line(150, 150, 319, 0, 0x07E0);  // this line is green
84         draw_line(0, 239, 319, 239, 0xF800);  // this line is red
85         draw_line(319, 0, 0, 239, 0xF81F);    // this line is a pink color
86
87         return 0;
88     }
89
```

```c
1   #include "address_map_arm.h"
2   #include <stdbool.h>
3   #include <stdlib.h>
4   #include <stdio.h>
5
6   volatile int pixel_buffer_start; // global variable
7
8   void plot_pixel(int x, int y, short int line_color)
9   {
10      *(short int *)(pixel_buffer_start + (y << 10) + (x << 1)) = line_color;
11  }
12
13  // Swaps 2 numbers using the XOR operation
14  void swap(int * x, int * y)
15  {
16      int temp = *x;
17      *x = *y;
18      *y = temp;
19  }
20
21  void draw_line(int x1, int y1, int x2, int y2, short int colour)
22  {
23      // Check steepness of the line, if it is steep, it's better
24      // to move along the y-axis when drawing
25      bool is_steep = abs(y2-y1) > abs(x2-x1);
26      // If it is steep switch the x and y values
27      // the drawing loop will decide how the drawing will occur
28      if(is_steep) {
29          swap(&x1,&y1);
30          swap(&x2,&y2);
31      }
32
33      // We are going to increment from x1 to x2 so
34      // swap the endpoints if x1 > x2
35      if(x1 > x2) {
36          swap(&x1,&x2);
37          swap(&y1,&y2);
38      }
39
40      int deltax = x2-x1;
41      int deltay = abs(y2-y1);
42      int error = -(deltax/2);
43      int x,y,y_step;
44
45      // Figure out how y will be incremented
46      if(y1<y2) y_step = 1;
47      else y_step = -1;
48
49      for(x=x1,y=y1; x<=x2; x++) {
50          // If the line is steep the x and y values are swapped
51          if(is_steep) plot_pixel(y,x,colour);
52          else plot_pixel(x,y,colour);
53
54          // Check margin of error
55          error += deltay;
56          if(error>=0) {
57              y += y_step; // Increment y val
58              error -= deltax; // Reset error
59          }
60      }
61  }
62
63  // Draw black to every pixel on the screen
64  void clear_screen()
65  {
66      int x,y;
67      // The screen is 320x240
68      for(x=0; x<320; x++) {
69          for(y=0; y<240; y++) {
```

```c
                    plot_pixel(x,y,0x0000);
            }
        }
    }

    // Synchronizes the display with the VGA timing
    void wait_for_vsync()
    {
        volatile int * pixel_ctrl_ptr = (int *)PIXEL_BUF_CTRL_BASE;
        register int status;

        *pixel_ctrl_ptr = 1; // Start synchronization process

        // Keep waiting until the whole screen ahs been drawn
        do {
            status = *(pixel_ctrl_ptr + 3);
        } while((status & 0x01) != 0);
    }

    int main()
    {
        volatile int * pixel_ctrl_ptr = (int *)PIXEL_BUF_CTRL_BASE;
        /* Read location of the pixel buffer from the pixel buffer controller */
        pixel_buffer_start = *pixel_ctrl_ptr;

        clear_screen();

        // Infinitely loop
        int y = 0; // We are only moving the line's y-coordinate
        int y_step = 1; // Start by moving down
        while(1) {
            draw_line(0,y,319,y,0x001F); // Draw a blue line at new y coordinate

            wait_for_vsync(); // Draw the line at a rate of 60 pixels/second

            draw_line(0,y,319,y,0x0000); // Black line to "erase" previous line
            y += y_step; // Increment y
            // Bounce the line when it gets to the ends
            if(y==239) y_step = -1;
            else if(y==0) y_step = 1;
        }

        return 0;
    }
```

```c
1    #include "address_map_arm.h"
2    #include <stdbool.h>
3    #include <stdlib.h>
4    #include <stdio.h>
5    #include <time.h>
6
7    volatile int pixel_buffer_start; // global variable
8
9    void plot_pixel(int x, int y, short int line_color)
10   {
11       *(short int *)(pixel_buffer_start + (y << 10) + (x << 1)) = line_color;
12   }
13
14   // Swaps 2 numbers using the XOR operation
15   void swap(int * x, int * y)
16   {
17       int temp = *x;
18       *x = *y;
19       *y = temp;
20   }
21
22   void draw_line(int x1, int y1, int x2, int y2, short int colour)
23   {
24       // Check steepness of the line, if it is steep, it's better
25       // to move along the y-axis when drawing
26       bool is_steep = abs(y2-y1) > abs(x2-x1);
27       // If it is steep switch the x and y values
28       // the drawing loop will decide how the drawing will occur
29       if(is_steep) {
30           swap(&x1,&y1);
31           swap(&x2,&y2);
32       }
33
34       // We are going to increment from x1 to x2 so
35       // swap the endpoints if x1 > x2
36       if(x1 > x2) {
37           swap(&x1,&x2);
38           swap(&y1,&y2);
39       }
40
41       int deltax = x2-x1;
42       int deltay = abs(y2-y1);
43       int error = -(deltax/2);
44       int x,y,y_step;
45
46       // Figure out how y will be incremented
47       if(y1<y2) y_step = 1;
48       else y_step = -1;
49
50       for(x=x1,y=y1; x<=x2; x++) {
51           // If the line is steep the x and y values are swapped
52           if(is_steep) plot_pixel(y,x,colour);
53           else plot_pixel(x,y,colour);
54
55           // Check margin of error
56           error += deltay;
57           if(error>=0) {
58               y += y_step; // Increment y val
59               error -= deltax; // Reset error
60           }
61       }
62   }
63
64   // Fill a rectangle with a chosen colour
65   void fill_rect(int x, int y, int width, int height, short int colour)
66   {
67       int dx,dy;
68       for(dx=0; dx<width; dx++) {
69           for(dy=0; dy<height; dy++) {
```

```c
70                  plot_pixel(x+dx,y+dy,colour);
71              }
72          }
73      }
74
75      // Draw black to every pixel on the screen
76      void clear_screen()
77      {
78          int x,y;
79          // The screen is 320x240
80          for(x=0; x<320; x++) {
81              for(y=0; y<240; y++) {
82                  plot_pixel(x,y,0x0000);
83              }
84          }
85      }
86
87      // Synchronizes the display with the VGA timing
88      void wait_for_vsync()
89      {
90          volatile int * pixel_ctrl_ptr = (int *)PIXEL_BUF_CTRL_BASE;
91          register int status;
92
93          *pixel_ctrl_ptr = 1; // Start synchronization process
94
95          // Keep waiting until the whole screen ahs been drawn
96          do {
97              status = *(pixel_ctrl_ptr + 3);
98          } while((status & 0x01) != 0);
99      }
100
101     int main(void)
102     {
103         volatile int * pixel_ctrl_ptr = (int *)PIXEL_BUF_CTRL_BASE;
104
105         srand(time(NULL)); // Set up for random number generation
106
107         short int colourBank[8] = {0x001F,0x07E0,0xF800,0xF81F,0x07FF,0xF81F,0xFFE,0xFFFF};
108         int numRects = 8; // Have 8 rectangles
109         int width[8],height[8],colour[8],x[8],y[8],x_step[8],y_step[8];
110
111         // Set up the rectangles, they will all 2x2 in size
112         int i;
113         for(i=0; i<numRects; i++) {
114             width[i] = 2;
115             height[i] = 2;
116             colour[i] = colourBank[rand()%8];
117             // Avoid spawning the rectangle out of bounds
118             x[i] = rand()%(320-width[i]);
119             y[i] = rand()%(240-height[i]);
120             // Set initial direction
121             x_step[i] = rand()%2 * 2 - 1; // +/- 1
122             y_step[i] = rand()%2 * 2 - 1; // +/- 1
123         }
124
125         /* set front pixel buffer to start of FPGA On-chip memory */
126         *(pixel_ctrl_ptr + 1) = FPGA_ONCHIP_BASE; // first store the address in the
127                                     // back buffer
128         /* now, swap the front/back buffers, to set the front buffer location */
129         wait_for_vsync();
130         /* initialize a pointer to the pixel buffer, used by drawing functions */
131         pixel_buffer_start = *pixel_ctrl_ptr;
132         clear_screen(); // pixel_buffer_start points to the pixel buffer
133         /* set back pixel buffer to start of SDRAM memory */
134         *(pixel_ctrl_ptr + 1) = DDR_BASE;
135         pixel_buffer_start = *(pixel_ctrl_ptr + 1); // we draw on the back buffer
136
137         while (1)
138         {
```

```c
139            // Erase any boxes and lines that were drawn in the last iteration
140            clear_screen();
141
142            // Draw boxes and lines and update locations
143            for(i=0; i<numRects; i++) {
144                // Draw the line connecting boxes
145                if(i==numRects-1) { // Wrap around
146                    draw_line(x[i],y[i],x[0],y[0],colour[i]);
147                } else {
148                    draw_line(x[i],y[i],x[i+1],y[i+1],colour[i]);
149                }
150
151                // Draw the box
152                fill_rect(x[i],y[i],width[i],height[i],colour[i]);
153
154                // Update location
155                x[i] += x_step[i];
156                y[i] += y_step[i];
157
158                // Horizontal bounce
159                if(x[i]+width[i]==319) x_step[i] = -1;
160                else if(x[i]==0) x_step[i] = 1;
161
162                // Vertical bounce
163                if(y[i]+height[i]==239) y_step[i] = -1;
164                else if(y[i]==0) y_step[i] = 1;
165            }
166
167            wait_for_vsync(); // swap front and back buffers on VGA vertical sync
168            pixel_buffer_start = *(pixel_ctrl_ptr + 1); // new back buffer
169        }
170
171        return 0;
172    }
173
```