```c
/*
 * helloworld.c: simple test application
 * Currently used to test lab 3 for Space Invaders.
 * Taylor Cowley and Andrew Okazaki
 */

#include <stdio.h>
#include <stdint.h>
#include "platform.h"
#include "xparameters.h"
#include "xaxivdma.h"
#include "xio.h"
#include "time.h"
#include "unistd.h"
#include "bunkers.h"
#include "tank.h"
#include "interface.h"
#include "aliens.h"
#define DEBUG

#define SCREEN_RES_X 640    // Our screen resolution is 640 * 480
#define SCREEN_RES_Y 480    // Our screen resolution is 640 * 480
#define BLACK 0x00000000    // Hex value for black

void print(char *str);



#define FRAME_BUFFER_0_ADDR 0xC1000000  // Starting location in DDR where we will store the images that we display.

int main() {
    init_platform();                    // Necessary for all programs.
    int Status;                         // Keep track of success/failure of system function calls.
    XAxiVdma videoDMAController;
    // There are 3 steps to initializing the vdma driver and IP.
    // Step 1: lookup the memory structure that is used to access the vdma driver.
    XAxiVdma_Config * VideoDMAConfig = XAxiVdma_LookupConfig(XPAR_AXI_VDMA_0_DEVICE_ID);
    // Step 2: Initialize the memory structure and the hardware.
    if(XST_FAILURE == XAxiVdma_CfgInitialize(&videoDMAController, VideoDMAConfig, XPAR_AXI_VDMA_0_BASEADDR)) {
        xil_printf("VideoDMA Did not initialize.\r\n");
    }
    // Step 3: (optional) set the frame store number.
    if(XST_FAILURE ==  XAxiVdma_SetFrmStore(&videoDMAController, 2, XAXIVDMA_READ)) {
        xil_printf("Set Frame Store Failed.");
    }
    // Initialization is complete at this point.

    // Setup the frame counter. We want two read frames. We don't need any write frames but the
    // function generates an error if you set the write frame count to 0. We set it to 2
    // but ignore it because we don't need a write channel at all.
    XAxiVdma_FrameCounter myFrameConfig;
    myFrameConfig.ReadFrameCount = 2;
```

```
53     myFrameConfig.ReadDelayTimerCount = 10;
54     myFrameConfig.WriteFrameCount =2;
55     myFrameConfig.WriteDelayTimerCount = 10;
56     Status = XAxiVdma_SetFrameCounter(&videoDMAController, &myFrameConfig);
57     if (Status != XST_SUCCESS) {
58        xil_printf("Set frame counter failed %d\r\n", Status);
59        if(Status == XST_VDMA_MISMATCH_ERROR)
60            xil_printf("DMA Mismatch Error\r\n");
61     }
62     // Now we tell the driver about the geometry of our frame buffer and a few other
   things.
63     // Our image is 480 x 640.
64     XAxiVdma_DmaSetup myFrameBuffer;
65     myFrameBuffer.VertSizeInput = 480;    // 480 vertical pixels.
66     myFrameBuffer.HoriSizeInput = 640*4;  // 640 horizontal (32-bit pixels).
67     myFrameBuffer.Stride = 640*4;         // Dont' worry about the rest of the values.
68     myFrameBuffer.FrameDelay = 0;
69     myFrameBuffer.EnableCircularBuf=1;
70     myFrameBuffer.EnableSync = 0;
71     myFrameBuffer.PointNum = 0;
72     myFrameBuffer.EnableFrameCounter = 0;
73     myFrameBuffer.FixedFrameStoreAddr = 0;
74     if(XST_FAILURE == XAxiVdma_DmaConfig(&videoDMAController, XAXIVDMA_READ,
   &myFrameBuffer)) {
75         xil_printf("DMA Config Failed\r\n");
76      }
77     // We need to give the frame buffer pointers to the memory that it will use. This
   memory
78     // is where you will write your video data. The vdma IP/driver then streams it to
   the HDMI
79     // IP.
80      myFrameBuffer.FrameStoreStartAddr[0] = FRAME_BUFFER_0_ADDR;
81      myFrameBuffer.FrameStoreStartAddr[1] = FRAME_BUFFER_0_ADDR + 4*640*480;
82
83      if(XST_FAILURE == XAxiVdma_DmaSetBufferAddr(&videoDMAController, XAXIVDMA_READ,
84                            myFrameBuffer.FrameStoreStartAddr)) {
85          xil_printf("DMA Set Address Failed Failed\r\n");
86      }
87      // Print a sanity message if you get this far.
88      xil_printf("Woohoo! I made it through initialization.\n\r");
89      // Now, let's get ready to start displaying some stuff on the screen.
90      // The variables framePointer and framePointer1 are just pointers to the base
   address
91      // of frame 0 and frame 1.
92      uint32_t* framePointer0 = (uint32_t*) FRAME_BUFFER_0_ADDR;
93      // Just paint some large red, green, blue, and white squares in different
94      // positions of the image for each frame in the buffer (framePointer0 and
   framePointer1).
95      int row=0, col=0;
96           for( row=0; row<SCREEN_RES_Y; row++) {
97              for(col=0; col<SCREEN_RES_X; col++) {
98                 framePointer0[row*SCREEN_RES_X + col] = BLACK;
99              }
100          }
101
102     bunkers_init(framePointer0);       // initialize the bunkers
103     tank_init();                       // initialize the tank
104     tank_draw(framePointer0, false);   // draw the tank
```

```
105
106     interface_draw_line(framePointer0);            // draw the line at the bottom
107     interface_draw_tanks(framePointer0);           // draw the tanks at the top
108     aliens_init(framePointer0);            // initialize aliens
109
110
111
112     // This tells the HDMI controller the resolution of your display (there must be a
    better way to do this).
113     XIo_Out32(XPAR_AXI_HDMI_0_BASEADDR, 640*480);
114
115     // Start the DMA for the read channel only.
116     if(XST_FAILURE == XAxiVdma_DmaStart(&videoDMAController, XAXIVDMA_READ)){
117         xil_printf("DMA START FAILED\r\n");
118     }
119     int frameIndex = 0;
120     // We have two frames, let's park on frame 0. Use frameIndex to index them.
121     // Note that you have to start the DMA process before parking on a frame.
122
123     if (XST_FAILURE == XAxiVdma_StartParking(&videoDMAController, frameIndex,
    XAXIVDMA_READ)) {
124         xil_printf("vdma parking failed\n\r");
125     }
126     char input;
127     srand((unsigned)time( NULL ));
128     while(1){
129     input = getchar();
130     switch(input){
131     case '4':
132         tank_move_left(framePointer0);     // move the tank left
133         break;
134     case '6':
135         tank_move_right(framePointer0);        // move the tank right
136         break;
137     case '8':
138         aliens_move(framePointer0);    // move the aliens
139         break;
140     case '2':
141         aliens_kill(framePointer0);    // Kill an alien
142         break;
143     case '5':
144         tank_fire(framePointer0);      // Make the tank fire
145         break;
146     case '3':
147         alien_missle(framePointer0);   // Make the aliens fire
148         break;
149     case'9':
150         tank_update_bullet(framePointer0); // update all bullets
151         aliens_update_bullets(framePointer0);  // update all bullets
152         break;
153     case '7':
154         bunkers_hit_rand_bunker(framePointer0);    //Erode bunker
155         break;
156     }
157     }
158
159
160    cleanup_platform();
```

```
161
162     return 0;
163 }
164
```