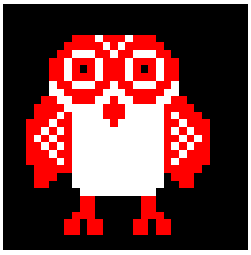
Rally-Owl



CSEE 4840: Embedded System Design

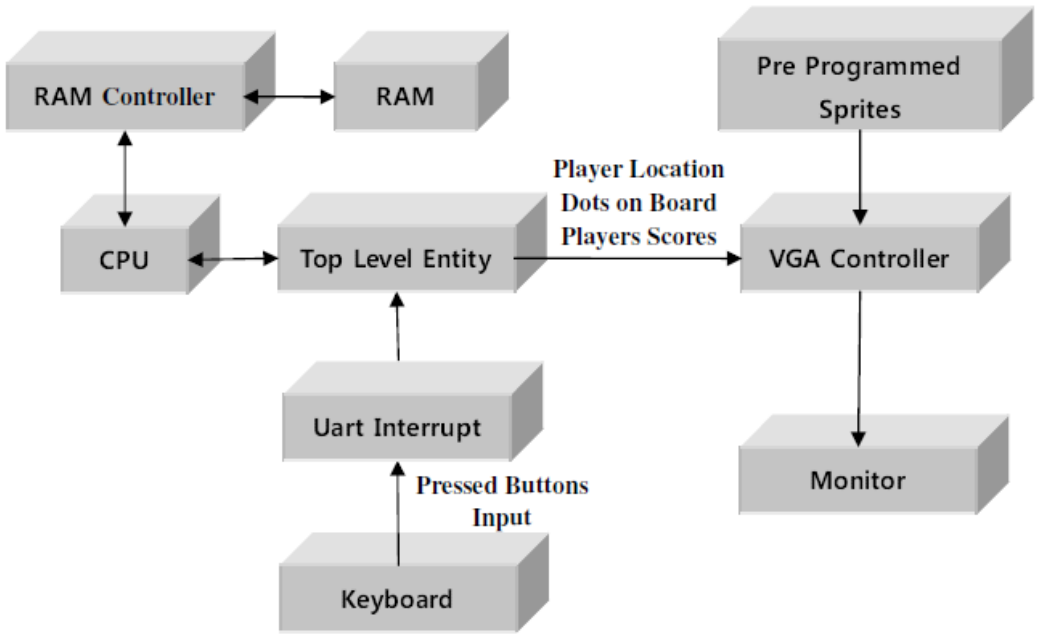
Spring 2011

Group Members:
David Jew (drj2109)
Shangche Peng (sp2922)
Edward Lai (el2564)
Yun Seong Nam (yn2213)

Table of Contents

I. INTRODUCTION

The goal of our final project was to implement a game similar to the popular 80's arcade game, Rally-X. We achieved this using a combination of C and VHDL. For the VHDL part, we used some of the base files from Lab #3 to create a VGA controller that is able to display all of our game content. This content includes the game map, player car, enemy car, flags, smoke, score, number of lives remaining, fuel and mini map. The VGA controller creates the mini map using a few calculations to rescale the original map. The VGA controller also partitions the screen into two main areas: the "main game area" and the "side bar" with game stats. The software is used to control the game logic and sends coordinates for the player (map - since it is scrolling), enemy cars, flags, smoke, lives, and fuel. The game is controlled by a PS/2 keyboard. We used the PS2 keyboard controller file from Lab # 3 to implement this. All of the sprites for our game are all stored in the on-block RAM in the VGA controller.



II. DESIGN

2.1 - Game Logic

Rally-Owl is a game with a goal of collecting flags, but in our case, collecting owls. The player has 3 lives when starting the game and the player can place a maximum of 3 smoke at once time to stop the enemy cars. the smoke will disappear after 15 seconds. This is done by design to increase difficultly and fuel efficiency of the car. Each time the player uses a smoke, it takes some fuels out of its fuel gauge. The player will die if the fuel is less than zero or get caught by the enemy cars. If the player collects 10 flags successfully, the player wins and exits the game. To reach the maximum possible score (550), the player must collect all 10 flags in one life.

2.2 - Hardware

2.2.a - VGA Controller

The de2_vga_raster.vhd file from Lab #3 was used as a base for the Rally-X VGA controller. To this file, we added the following ports: write, chipselect, address, and writedata. A process was created to convert the 50MHz clock to 25MHz for use with the VGA monitor. Separate processes were created for writing data to the VGA controller and for syncing the data so the the movement seen on the screen would be smooth. We used multiple processes to create rectangles (similar to that in the base file from Lab #3) to help partition the screen. These rectangles were also used to

decide layer priority (which sprite will appear on top of another if they are overlapping on the screen). These rectangles included:

-Side Bar (on the right side of the screen - 160x480 pixels)

-Rectangles for the words:

- "Score" (80x16 pixels)
- "Lives" (80x16 pixels)
- "Fuel" (64x16 pixels)

-Rectangles for the actual:

- Score (48x16 pixels)
- Lives (16x16 pixels)
- Fuel (158x16 pixels)

-Rectangles for the Mini-map (160x160 pixels)

-Main Map (480x480 pixels)

-Word "RallyX" (96x16 pixels)

-Word "gg" (32x16 pixels)

-Player Car (32x32 pixels) - acts as 24x24

-Enemy Cars (32x32 pixels) - acts as 24x24

-Flags (32x32 pixels)

-Smoke (32x32 pixels)

The map and enemy cars are all moved pixel by pixel. The flags and smoke are placed by tile by tile, however they also move pixel by pixel (with the map). We decided to create separate signals for each flag and smoke in the game instead of putting them directly into the game map array because we needed these signals anyways for the mini-map.

To display things on the mini-map, we took signals from the regular game map and scaled them down proportionally. In the game map, the drivable area is 31x31 tiles. The mini map has a size of 5x5 tiles (each tile is 32x32 pixels). The player car, enemy cars, flags, smoke, the words "Score", "Lives", and "Fuel" as well as the numbers for the player's score and number of lives are displayed using sprites. The fuel is displayed by an adjustable rectangle, depending the value passed in from the software. The game map is displayed as an array of tiles, with its starting point coordinates being shifted around the map by pixel.

For the main map area of the display, the layer priority is (from highest to lowest):

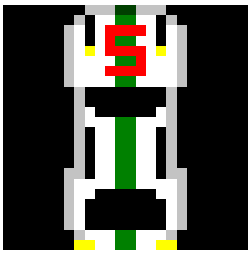
-Player Car

-Smoke

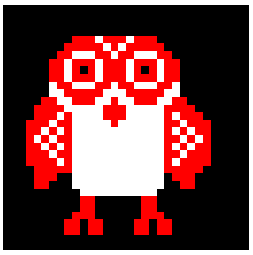
-Enemy Car # 1
-Enemy Car # 2
-Enemy Car # 3
-Game Map

Sprites:

We were able to store all of our sprites in on-block RAM within the VGA controller.



Player sprite



Owl sprite

Enemy car sprite

2.2.b - PS2 Keyboard Controller

Because our game only uses the 4 arrow keys and the space bar, we were able to use the PS2 keyboard controller file from Lab #3. We could have also used the more complex PS2 keyboard controller provided in Lab #2, but we did not need this much functionality

2.3 - Memory Mapping

Data is written from software directly to the VGA controller using IOWR_32Direct. The address port in our VGA controller is 5 bits.

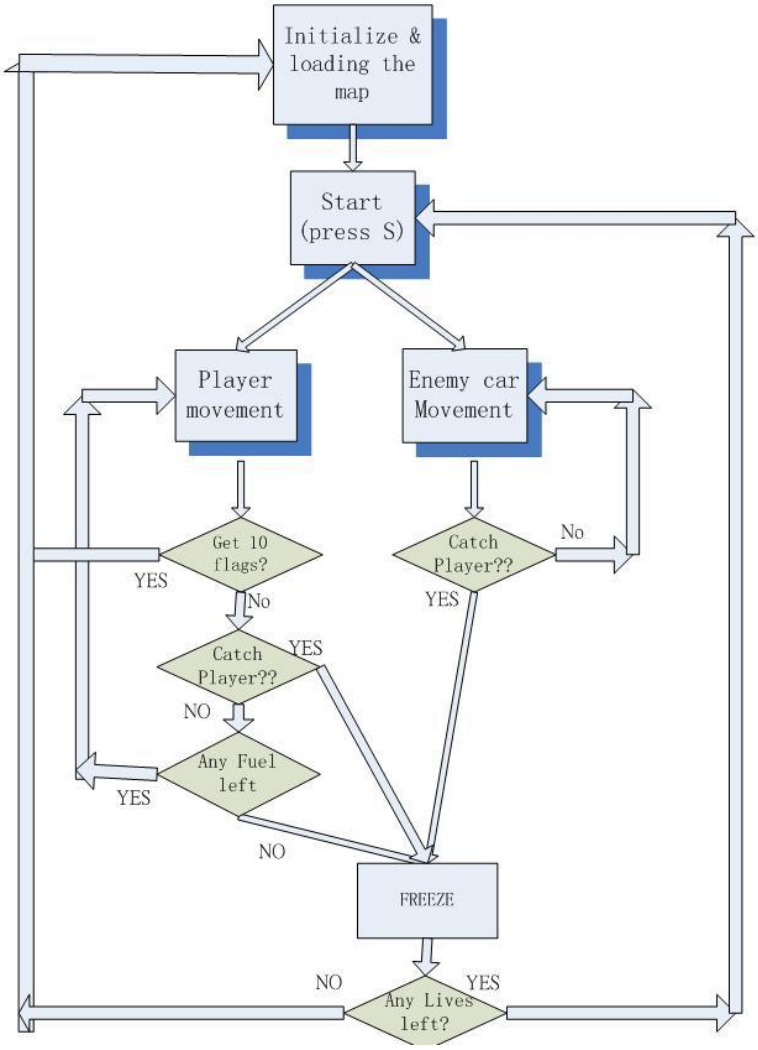
Adresss #	**Data (32 bits)**
0 - “00000”	Map X (bits 0-15), Map Y (bits 16-31) **[Map X,Y pixel shift coordinates]**
1 - “00001”	Player Direction (bits 0-2) **[Player’s movement direction]**
2 - “00010”	Fuel Status (bits 0-31) **[Fuel Gauge Amount]**
3 - “00011”	Ones Place (bits 0-3), Tens Place (bits 4-7), Hundreds Place (bits 8-11) **[Player’s Score]**
4 - “00100”	Flag Col (bits 0-11), Flag Row (bits 12-23), Flag On/Off (bit 24), Flag # (bit 25-28) **[Flag data]**
5 - “00101”	X (bits 0-11), Y (bits 12-23), Movement Direction (bits 24-26), Enemy On/Off (bit 27) **[Enemy 1 Data]**
6 - “00110”	X (bits 0-11), Y (bits 12-23), Movement Direction (bits 24-26), Enemy On/Off (bit 27) **[Enemy 2 Data]**
7 - “00111”	X (bits 0-11), Y (bits 12-23), Movement Direction (bits 24-26), Enemy On/Off (bit 27) **[Enemy 3 Data]**
8 - “01000”	Smoke Col (bits 0-11), Smoke Row (bits 12-23), Smoke On/Off (bit 24), Smoke # (bit 25-28) **[Smoke data]**
9 - “01001”	Lives (bits 0-31) **[Player’s Number of Lives Left]**
10 - “01010”	Game State (bits 0-31) **[0 - “Start”, 1 - “End”, 2 - “Play”]**

2.4 - Software

Software sends out the position of the player, 3 enemies, smoke, score, lives, score, and fuel amounts. Basically, the player car gets a direction from the user. Only thing the software should have to do is detection the wall. If the car hits the wall, software automatically change the direction of player car in counter clockwise. There are two possibilities. One thing it that the player car in the just one 32 by 32 bit block. In this case, the software can just check the front block. The other case is the car overlap two blocks. In this case, the software should check right or left block also. We made the player car move continuously. The algorithm to detecting the wall, we store the map in a array named ”control array”, if it’s a wall, a value of 0xf is initialized, 0x0 for road. we just check player’s next position with the control array’s to we know if it’s going to hit the wall or not.

The software also handles the enemy car AI. First time, we applied a simple AI. Enemy car calculate two distance, vertical distance between player car and enemy car and horizontal distance between player car and enemy car. And software just picks the longer one. It is not enough to handle the enemy car smartly. However, it is not bad for playing the game. So we decide to use this AI. Also enemy car check the wall in every times like the player car do.

We have a while loop to detect the S key, after we detected the S key, we break the loop and starts the game. At any point of the game, we can press the key “R” to reset the game and the space key is for placing the smoke behind the car.



III. CONCLUSION

3.1 Responsibilities

David Jew

- Hardware and software
- In charge of VGA controller and PS2 keyboard controller
- Setting up some basic software stuff (wall detection and memory mapping)

Shang-Che Peng

- Software
- Player controls (smoke, fuel, death, flag collection, lives)
- Fine tuned enemy player movement

Edward Lai

- Hardware and Artist
- Focused on VGA controller

- Drew all sprites

Yun Seung Nam
- Software
- Player and Enemy movement (movement, wall detection, auto turning)
- Enemy AI

This game was a success. Everything runs extremely smoothly, especially the tough hardware parts (vertical and horizontal map scrolling at high speed and mini map scaling). The game logic controlled by our software emulates the original game almost perfectly minus a few tweaks we made to make the game better (in our opinion). Although this was a simple game and we believe it was a success, it was definitely a lot harder to implement than we imagined.

If we learned anything from this project it's that we should work on the hardware and software parts at the same time, part by part, rather than completely finish the hardware before really starting the software. We found that even though we thought we finished the hardware, we had to keep going back to debug it as we implemented more software features.

The only advice we have for future projects is to consistently work on the project throughout the quarter and to spend a lot of time planning the overall implementation (both hardware and software) before actually starting.

IV. RELEVANT CODE

Software: hello_world.c

```
#include <io.h>
#include <system.h>
#include <stdio.h>
#include <math.h>

#define IOWR_VGA_DATA(base, offset, data) \
        IOWR_32DIRECT(base, (offset) * 32, data)
int main()
{
    int i;
    int score_ones = 0;
```

```
 int score_tens = 0;
 int score_hundy = 0;
 int temp = 0;
 int sOtemp = 0;
 int sTtemp = 0;
 int sHtemp = 0;
 int sdelay = 0;

 int flagon = 0;
 int flagdataout = 0;
 int flag_row = 0;
 int flag_col = 0;

 int fuel_amount = 158;
 int fuel_sendout = 0;
 int flag_address = 0;
 int pstatus = 0;
 int estatus[3]={0,0,0};
 int pstatus_data = 0;
 int lives=3;
 int flag_collected=0;
 int flag_for_score=0;
 int temp_s=0;
 int flag1=1;
 int flag2=1;
 int flag3=1;
 int flag4=1;
 int flag5=1;
 int flag6=1;
 int flag7=1;
 int flag8=1;
 int flag9=1;
 int flag10=1;
 printf("Hello Michael\n");



int x = 509, y =515;
```

```
int ex[3]={991+240,413+240, 35+240};
int ey[3]={983+240,985+240, 996+240};

int vx = 0, vy= 0;
int evx[3] = {0,0,0};
int evy[3] = {0,0,0};


   //int temp11 =0;
int delay=0;
int delay_smoke=0;

int w, ew1, ew2, ew3;
unsigned char code;
unsigned char previous;
int change=0;
int wallhit =0;

int convert_to_tile_r;
int convert_to_tile_c;
int convert_to_tile_r_e[3];
int convert_to_tile_c_e[3];


int player_x;
int player_y;
int enemy_x[3];

int enemy_y[3];

int rem_up =1;
int rem_down =1;
int rem_right =1;
int rem_left =1;

int rem_up_e[3] = {1,1,1};
int rem_down_e[3] ={1,1,1};
```

```
  int rem_right_e[3] ={1,1,1};
  int rem_left_e[3] ={1,1,1};

  ////////////////////

    int smoke_1_on=0;
    int smoke_2_on=0;
    int smoke_3_on=0;
    int smoke_1_timer=0;
    int smoke_2_timer=0;
    int smoke_3_timer=0;
    int smoke_1_xposition=0;
    int smoke_2_xposition=0;
    int smoke_3_xposition=0;
    int smoke_1_yposition=0;
    int smoke_2_yposition=0;
    int smoke_3_yposition=0;
    int fuel_counter=0;
    int smoke_cost=10;

    //int flagdataout;
  // brick : 1111 (0xf)
  // road   : 0000 (0x0)
  //                                0    1    2    3    4    5    6    7    8
9   10   11   12   13   14   15   16   17   18   19   20   21   22   23
24   25   26   27   28   29   30   31   32   33   34   35   36   37   38
39   40   41   42   43   44
  //                                |    |    |    |    |    |    |    |    |    |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|
  int control_array[47][47] = {0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 0
                                    0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
                                    0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
```

0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 1
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 2
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 3
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 4
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 5
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 6
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0x0, 0x0,
0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0,
0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
// 7
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0x0,
0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf,
0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
// 8
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0x0, 0x0,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0x0, 0x0, 0x0, 0xf, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
// 9
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0x0, 0x0,
0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0x0, 0xf, 0x0, 0x0, 0x0, 0xf, 0x0, 0x0, 0xf, 0xf, 0xf, 0x0,
0xf, 0xf, 0xf, 0x0, 0xf, 0xf, 0xf, 0xf, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, //
10
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0x0,
0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
// 11
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0x0,

0xf, 0x0, 0x0, 0x0, 0x0, 0xf, 0x0, 0xf, 0x0, 0x0, 0xf, 0xf, 0xf, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 12

0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0x0, 0xf, 0x0, 0xf, 0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0x0, 0x0, 0x0, 0xf, 0x0, 0x0, 0xf, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 13

0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0x0, 0xf, 0x0, 0xf, 0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0x0, 0xf, 0x0, 0xf, 0x0, 0x0, 0xf, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 14

0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0x0, 0x0, 0xf, 0x0, 0x0, 0x0, 0x0, 0xf, 0x0, 0xf, 0x0, 0x0, 0xf, 0x0, 0xf, 0x0, 0x0, 0xf, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 15

0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0x0, 0x0, 0xf, 0x0, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0x0, 0x0, 0xf, 0x0, 0xf, 0x0, 0x0, 0xf, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 16

0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0x0, 0x0, 0xf, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 17

0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0x0, 0xf, 0xf, 0xf, 0x0, 0x0, 0xf, 0xf, 0xf, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 18

0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 19

0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0xf, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 20

0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,

0xf, // 21

0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0xf, 0x0, 0x0, 0x0, 0xf, 0xf, 0x0, 0x0, 0xf, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 22

0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0xf, 0xf, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0xf, 0x0, 0x0, 0x0, 0xf, 0xf, 0x0, 0x0, 0xf, 0x0, 0xf, 0xf, 0xf, 0xf, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 23

0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0xf, 0xf, 0x0, 0x0, 0xf, 0xf, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0x0, 0x0, 0x0, 0xf, 0xf, 0x0, 0x0, 0xf, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 24

0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0xf, 0xf, 0x0, 0x0, 0xf, 0xf, 0x0, 0xf, 0x0, 0x0, 0xf, 0xf, 0x0, 0xf, 0x0, 0xf, 0xf, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 25

0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0x0, 0xf, 0x0, 0xf, 0xf, 0xf, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 26

0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0xf, 0xf, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0x0, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0xf, 0x0, 0xf, 0xf, 0x0, 0xf, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 27

0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0x0, 0xf, 0x0, 0x0, 0x0, 0x0, 0xf, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0x0, 0xf, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,// 28

0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0xf, 0xf, 0x0, 0xf, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0x0, 0x0, 0xf, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0x0, 0xf, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 29

0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0x0, 0x0, 0xf, 0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0xf, 0x0, 0x0, 0x0, 0x0, 0xf, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 30

0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0x0,

0xf, 0x0, 0xf, 0x0, 0x0, 0x0, 0xf, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0x0, 0xf, 0xf, 0xf, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, // 31
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0x0,
0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0x0, 0xf, 0xf, 0x0,
0xf, 0xf, 0x0, 0x0, 0xf, 0x0, 0xf, 0xf, 0xf, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, //
32
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0x0,
0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0x0, 0x0, 0x0, 0x0,
0xf, 0xf, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
// 33
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0x0,
0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0x0, 0xf, 0xf, 0x0,
0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, //
34
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0x0,
0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0x0, 0xf, 0xf, 0x0,
0xf, 0xf, 0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, //
35
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0xf, 0x0,
0xf, 0xf, 0xf, 0x0, 0xf, 0xf, 0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0xf, 0x0, 0x0, 0xf, 0xf, 0x0,
0xf, 0xf, 0x0, 0x0, 0xf, 0xf, 0x0, 0xf, 0xf, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, //
36
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0x0, 0x0, 0x0,
0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, // 37
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 38
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 39
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 40
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,

```
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 41
                                        0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 42
                                        0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
                                        0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, // 43
                                        0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf,
0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf, 0xf}; // 44
        code = IORD_8DIRECT(PS2_KEYBOARD_BASE,4);
        previous = code;
        sOtemp = (0 << 8) + (0 << 4) + 0;
        IOWR_32DIRECT(LEDS_BASE, 12, sOtemp);

 int flag1_pos_x=12;
 int flag1_pos_y=8;

 int flag2_pos_x=8;
 int flag2_pos_y=8;

 int flag3_pos_x=10;
 int flag3_pos_y=10;

 int flag4_pos_x=10;
 int flag4_pos_y=8;

 int flag5_pos_x=35;
 int flag5_pos_y=14;

 int flag6_pos_x=16;
 int flag6_pos_y=35;

 int flag7_pos_x=37;
 int flag7_pos_y=24;
```

```
 int flag8_pos_x=26;
 int flag8_pos_y=29;

 int flag9_pos_x=37;
 int flag9_pos_y=30;

 int flag10_pos_x=31;
 int flag10_pos_y=14;
int to_start=0;
 code='0';
                int smokedataout = 0;

// IOWR_32DIRECT(LEDS_BASE,40,to_start);

     IOWR_32DIRECT(LEDS_BASE, 40, 0);
            while (to_start==0)
{
     code = IORD_8DIRECT(PS2_KEYBOARD_BASE,4);

     to_start=0;
     switch (code)
            {
                 case 0x1B:

                      to_start=1;
                    break;

            default :
                    break;
            }
              lives=3;


                    IOWR_32DIRECT(LEDS_BASE, 36, lives);
                    flagdataout = (1 << 25) + (1 << 24) + (flag1_pos_y << 12) +
flag1_pos_x;
 IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
```

```
  flagdataout = (2 << 25) + (1 << 24) + (flag2_pos_y << 12) + flag2_pos_x;
IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
  flagdataout = (3 << 25) + (1 << 24) + (flag3_pos_y << 12) + flag3_pos_x;
IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
  flagdataout = (4 << 25) + (1 << 24) + (flag4_pos_y << 12) + flag4_pos_x;
IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
  flagdataout = (5 << 25) + (1 << 24) + (flag5_pos_y << 12) + flag5_pos_x;
IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
  flagdataout = (6 << 25) + (1 << 24) + (flag6_pos_y << 12) + flag6_pos_x;
IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
  flagdataout = (7 << 25) + (1 << 24) + (flag7_pos_y << 12) + flag7_pos_x;
IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
  flagdataout = (8 << 25) + (1 << 24) + (flag8_pos_y << 12) + flag8_pos_x;
IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
  flagdataout = (9 << 25) + (1 << 24) + (flag9_pos_y << 12) + flag9_pos_x;
IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
  flagdataout = (0 << 25) + (1 << 24) + (flag10_pos_y << 12) + flag10_pos_x;
IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
    }
    for (;;)
     {

     while (to_start==0)
{
     code = IORD_8DIRECT(PS2_KEYBOARD_BASE,4);


     switch (code)
              {
                   case 0x1B:

                        to_start=1;
                   break;

              default :
                   break;
              }
    }
```

```
if(code ==0x2D)
{

        sOtemp = (0 << 8) + (0 << 4) + 0;
        IOWR_32DIRECT(LEDS_BASE, 12, sOtemp);
                IOWR_32DIRECT(LEDS_BASE, 40, 0);
      to_start=0;
      while (to_start==0)
{
      code = IORD_8DIRECT(PS2_KEYBOARD_BASE,4);

      to_start=0;
      switch (code)
                {
                     case 0x1B:

                          to_start=1;
                        break;

                default :
                        break;
                }
    }
    smoke_1_timer=0;
    smoke_2_timer=0;
    smoke_3_timer=0;


    if ( smoke_1_on==1)
    {       smoke_1_on=0;
      smoke_1_xposition=0;
      smoke_1_yposition=0;
                    smokedataout = (1 << 25) + (0 << 24) + ( smoke_1_xposition <<
12) + smoke_1_yposition;
                     IOWR_32DIRECT(LEDS_BASE, 32, smokedataout);
    }
```

```
    if ( smoke_2_on==1)
    {smoke_2_on=0;
     smoke_2_xposition=0;
     smoke_2_yposition=0;
                    smokedataout = (2 << 25) + (0 << 24) + ( smoke_2_xposition <<
12) + smoke_2_yposition;
                     IOWR_32DIRECT(LEDS_BASE, 32, smokedataout);
    }
    if ( smoke_3_on==1)
    {smoke_3_on=0;
     smoke_3_xposition=0;
     smoke_3_yposition=0;
                    smokedataout = (3 << 25) + (0 << 24) + ( smoke_3_xposition <<
12) + smoke_3_yposition;
                     IOWR_32DIRECT(LEDS_BASE, 32, smokedataout);
    }
                     lives=3;


                    IOWR_32DIRECT(LEDS_BASE, 36, lives);
                          x = 509;
                          y =515;
                          ex[0]=991+240;
                          ey[0]=983+240;
                          ex[1]=413+240;
                          ey[1]=985+240;
                          ex[2]=35+240;
                          ey[2]=996+240;
                          flag_for_score=0;
                          if ( smoke_1_on==1)
    {
                    smokedataout = (1 << 25) + (0 << 24) + ( smoke_1_xposition <<
12) + smoke_1_yposition;
                     IOWR_32DIRECT(LEDS_BASE, 32, smokedataout);
    }
    if ( smoke_2_on==1)
    {
                    smokedataout = (2 << 25) + (0 << 24) + ( smoke_2_xposition <<
```

```
12) + smoke_2_yposition;
                    IOWR_32DIRECT(LEDS_BASE, 32, smokedataout);
    }
    if ( smoke_3_on==1)
    {
                    smokedataout = (3 << 25) + (0 << 24) + ( smoke_3_xposition <<
12) + smoke_3_yposition;
                    IOWR_32DIRECT(LEDS_BASE, 32, smokedataout);
    }

                        flag10=1;
                        flag1=1;
                        flag2=1;
                        flag3=1;
                        flag4=1;
                        flag5=1;
                        flag6=1;
                        flag7=1;
                        flag8=1;
                        flag9=1;
                          sOtemp = (0 << 8) + (0 << 4) + 0;
                          IOWR_32DIRECT(LEDS_BASE, 12, sOtemp);

                          vx = 0, vy= 0;
                          //evx[3] = {0,0,0};
                          //evy[3] = {0,0,0};

                          delay=0;
                          delay_smoke=0;

                          change=0;
                          wallhit =0;
                          rem_up =1;
                          rem_down =1;
                          rem_right =1;
                          rem_left =1;
                          pstatus =0;
                          fuel_amount=158;
```

```
                    score_tens=0;
                    score_hundy=0;

                flag_collected=0;
                flag_for_score=0;
                    flagdataout = (1 << 25) + (1 << 24) + (flag1_pos_y << 12)
+ flag1_pos_x;
  IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
    flagdataout = (2 << 25) + (1 << 24) + (flag2_pos_y << 12) + flag2_pos_x;
  IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
    flagdataout = (3 << 25) + (1 << 24) + (flag3_pos_y << 12) + flag3_pos_x;
  IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
    flagdataout = (4 << 25) + (1 << 24) + (flag4_pos_y << 12) + flag4_pos_x;
  IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
    flagdataout = (5 << 25) + (1 << 24) + (flag5_pos_y << 12) + flag5_pos_x;
  IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
    flagdataout = (6 << 25) + (1 << 24) + (flag6_pos_y << 12) + flag6_pos_x;
  IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
    flagdataout = (7 << 25) + (1 << 24) + (flag7_pos_y << 12) + flag7_pos_x;
  IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
    flagdataout = (8 << 25) + (1 << 24) + (flag8_pos_y << 12) + flag8_pos_x;
  IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
    flagdataout = (9 << 25) + (1 << 24) + (flag9_pos_y << 12) + flag9_pos_x;
  IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
    flagdataout = (0 << 25) + (1 << 24) + (flag10_pos_y << 12) + flag10_pos_x;
  IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
}
//    int i=0;
//    for (i=0;i<5;i++)
//    {
//    flagdataout = (i << 25) + (1 << 24) + (7 << 12) + i+10;
//            //flag # (0-9)
//            //flag ON or OFF
//            //Flag Row #
//            //Flag Column #
// IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
//    }
```

```
          // LEFT    ARROW: 224 107 224 240 107
         // RIGHT ARROW: 224 116 224 240 116
         // UP       ARROW: 224 117 224 240 117
         // DOWN    ARROW: 224 114 224 240 114

            code = IORD_8DIRECT(PS2_KEYBOARD_BASE,4);

            if(code != previous){
                wallhit=0;
                previous = code;
            }

     IOWR_32DIRECT(LEDS_BASE, 40, 2);
                int smoke_timer=0;
                smoke_timer++;
                int j = 0;
//                int smokex=7;
//                smokedataout = (1 << 25) + (1 << 24) + (convert_to_tile_r << 12) +
convert_to_tile_c;
//                 IOWR_32DIRECT(LEDS_BASE, 32, smokedataout);
//                if (smoke_timer==500)
//                 {
//                 smoke_timer=0;
//                 smokedataout = (1 << 25) + (0 << 24) + (convert_to_tile_r << 12) +
convert_to_tile_c;
//                 IOWR_32DIRECT(LEDS_BASE, 32, smokedataout);
//                 }

//                for (j = 1; j < 3; j++)
//                {
//
//                smokedataout = (j+1 << 25) + (1 << 24) + (8 << 12) + smokex;
//
//                IOWR_32DIRECT(LEDS_BASE, 32, smokedataout);
//                printf("%d\n", smokedataout);
//                smokex = smokex + 2;
//
```

```
//     }


            while(!IORD_8DIRECT(PS2_KEYBOARD_BASE, 0))
            {
//
//

                    //TEST FOR FUEL GAUGE

                if (fuel_counter == 120000){
                    if (fuel_amount > 0 && to_start==1)
                        fuel_amount--;
                    else

                        pstatus=4;    // player die....

                    fuel_counter=0;

                    //
                   //printf("%d\n", fuel_amount);
                   }
                    IOWR_32DIRECT(LEDS_BASE, 8, fuel_amount);



                    if (temp_s!=flag_collected)
                    {
                        temp_s=flag_collected;
                        score_tens=score_tens+flag_for_score;

                        if (score_tens>=10)
                        {
                            score_tens=score_tens-10;
                            score_hundy++;
                        }
                    IOWR_32DIRECT(LEDS_BASE, 12, sOtemp);
                    sOtemp = (score_hundy << 8) + (score_tens << 4) + 0;
```

```
                IOWR_32DIRECT(LEDS_BASE, 12, sOtemp);

               if (flag_collected==10)
                IOWR_32DIRECT(LEDS_BASE, 40, 1);
                }
//             }
//             else
//                 sdelay++;

            if (delay == 600){


                convert_to_tile_c = (player_x)/32;
                convert_to_tile_r = (player_y)/32;
                player_x = x + 240; // /32 = 7.5 -> round to eight
                player_y = y + 240;
                rem_up = (player_y - 12)%32;
                rem_down = 32 - (player_y + 12)%32;
                rem_right = 32 - (player_x + 12)%32;
                rem_left = (player_x - 12)%32;

                if(rem_right ==32)
                    rem_right=0;

                if(rem_down==32)
                    rem_down =0;



                for(i=0; i<3;i++){
                    convert_to_tile_c_e[i] = (enemy_x[i])/32;
                    convert_to_tile_r_e[i] = (enemy_y[i])/32;

                    enemy_x[i] = ex[i];
                    enemy_y[i] = ey[i];

                    rem_up_e[i] = (enemy_y[i] - 12)%32;
                    rem_down_e[i] = 32 - (enemy_y[i] + 12)%32;
```

```
                        rem_right_e[i] = 32 - (enemy_x[i] + 12)%32;
                        rem_left_e[i] = (enemy_x[i] - 12)%32;

                        if(rem_right_e[i] ==32)
                              rem_right_e[i]=0;

                        if(rem_down_e[i]==32)
                              rem_down_e[i] =0;

                        if((ex[i]-player_x)*(ex[i]-player_x) >
(ey[i]-player_y)*(ey[i]-player_y)){
                              if(ex[i]-player_x > 0){evx[i]=-1;}
                              else{evx[i]=1;}
                        }else{
                              if(ey[i]-player_y > 0){evy[i]=-1;}
                              else{evy[i]=1;}
                        }
//                        if((((ex[i]-player_x)*(ex[i]-player_x) ==
1)||((ex[i]-player_x)*(ex[i]-player_x)
==0))&&(((ey[i]-player_y)*(ey[i]-player_y)==1)||((ey[i]-player_y)*(ey[i]-player_y)=
=0)))
//                         {
//                               pstatus =4;
//                         }
                        //if(to_start==1){
                        if(   (player_x-ex[i])*(player_x-ex[i])<80 &&
(player_y-ey[i])*(player_y-ey[i])< 80 )
                        {

                              pstatus =4;
                              //to_start=0;
                        }
                        //}

                        //printf("%d %d\n", (ex[i]-player_x)*(ex[i]-player_x),
(ey[i]-player_y)*(ey[i]-player_y));
                  }
```

```
                    //printf("x y : %d %d,    convert : %d %d, playxy : %d %d, up:%d
down:%d    right:%d    left:%d\n",x, y, player_x, player_y, convert_to_tile_c,
convert_to_tile_r, rem_up, rem_down,rem_right, rem_left);
                    //printf("e1xy : %d %d,    e2xy : %d %d,    e3xy : %d %d\n",
ex[0],ey[0],ex[1],ey[1],ex[2],ey[2]);
                    //printf("%d %d     , %d %d    , up:%d    down:%d    right:%d
left:%d\n",convert_to_tile_c_e[0], convert_to_tile_r_e[0], enemy_x[0],
enemy_y[0], rem_up_e[0], rem_down_e[0], rem_right_e[0], rem_left_e[0]);


                    ////////////////////////////////////////////////////////////////////////
                    ////////////////////////////For Player////////////////////////////////
                    ////////////////////////////////////////////////////////////////////////
                    ////////////////////////////////////// left

                    if(pstatus!=4){

                    if (vx == - 1)
                    {
                        if(rem_up <=8 && rem_down <=8){
                            if ( x > 0 && (rem_left > 0 ||
control_array[convert_to_tile_r][convert_to_tile_c-1] != 0xf)){
                                x += vx;
                                pstatus = 1;
                            }
                            else{
                                vx = 0;
                                vy = 1;
                                pstatus = 2;
                                wallhit =1;
                            }
                        }

                        else{
                            if(rem_down > rem_up){
                                if ( x > 0 && (rem_left > 0 ||
(control_array[convert_to_tile_r][convert_to_tile_c-1] != 0xf    &&
```

```
control_array[convert_to_tile_r+1][convert_to_tile_c-1] != 0xf))){
                                        x += vx;
                                        pstatus = 1;
                                    }
                                    else{
                                        vx = 0;
                                        vy = 1;
                                        pstatus = 2;
                                        wallhit =1;
                                    }
                                }
                                else{
                                    if ( x > 0 && (rem_left > 0 ||
(control_array[convert_to_tile_r][convert_to_tile_c-1] != 0xf    &&
control_array[convert_to_tile_r-1][convert_to_tile_c-1] != 0xf))){
                                        x += vx;
                                        pstatus = 1;
                                    }
                                    else{
                                        vx = 0;
                                        vy = 1;
                                        pstatus = 2;
                                        wallhit =1;
                                    }
                                }

                            }


                        }

                        ////////////////////////////////// right
                        if (vx == 1){
                            if(rem_up <=8 && rem_down <=8){
                                if (rem_right > 0 ||
control_array[convert_to_tile_r][convert_to_tile_c+1] != 0xf){
                                    x += vx;
                                    pstatus = 0;
```

```
                    }
                    else{
                        vx = 0;
                        vy = -1;
                        pstatus = 3;
                        wallhit =1;
                    }
                }
                else{
                    if(rem_down > rem_up){
                        if ( rem_right > 0 ||
(control_array[convert_to_tile_r][convert_to_tile_c+1] != 0xf   &&
control_array[convert_to_tile_r+1][convert_to_tile_c+1] != 0xf)){
                            x += vx;
                            pstatus = 0;
                        }
                        else{
                            vx = 0;
                            vy =-1;
                            pstatus = 3;
                            wallhit =1;
                        }
                    }
                    else{
                        if ( rem_right > 0 ||
(control_array[convert_to_tile_r][convert_to_tile_c+1] != 0xf   &&
control_array[convert_to_tile_r-1][convert_to_tile_c+1] != 0xf)){
                            x += vx;
                            pstatus = 0;
                        }
                        else{
                            vx = 0;
                            vy =-1;
                            pstatus = 3;
                            wallhit =1;
                        }
                    }
```

```
                    }

                }

                ////////////////////////////////////////////UP
                if (vy == -1)
                {
                    if(rem_right <=8 && rem_left <=8 ){
                        if ( y > 0 &&(rem_up > 0 ||
control_array[convert_to_tile_r-1][convert_to_tile_c] != 0xf)){
                            y += vy;
                            pstatus =3;
                        }
                        else{//////////////
                            vy = 0;
                            vx = -1;
                            pstatus = 1;
                            wallhit =1;
                        }
                    }
                    else{
                        if(rem_right > rem_left){
                            if ( y > 0 && (rem_up > 0 ||
(control_array[convert_to_tile_r-1][convert_to_tile_c] != 0xf    &&
control_array[convert_to_tile_r-1][convert_to_tile_c+1] != 0xf))){
                                y += vy;
                                pstatus = 3;
                            }
                            else{
                                vy = 0;
                                vx = -1;
                                pstatus = 1;
                                wallhit =1;
                            }
                        }
                        else{
                            if ( y > 0 && (rem_up > 0 ||
(control_array[convert_to_tile_r-1][convert_to_tile_c] != 0xf    &&
```

```
control_array[convert_to_tile_r-1][convert_to_tile_c-1] != 0xf))){
                                        y += vy;
                                        pstatus = 3 ;
                                    }
                                    else{
                                        vy = 0;
                                        vx = -1;
                                        pstatus = 1;
                                        wallhit =1;
                                    }
                                }

                            }
                        }

                        ////////////////////////////////////////////////DOWN
                        if (vy == 1){
                            if(rem_right <=8 && rem_left<=8){
                                if ( rem_down > 0 ||
control_array[convert_to_tile_r+1][convert_to_tile_c] != 0xf){
                                    y += vy;
                                    pstatus = 2;
                                }
                                else{
                                        vy = 0;
                                        vx = 1;
                                        wallhit =1;
                                        pstatus = 0;
                                    }
                            }
                            else{
                                if(rem_right > rem_left){
                                    if ( rem_down > 0 ||
(control_array[convert_to_tile_r+1][convert_to_tile_c] != 0xf    &&
control_array[convert_to_tile_r+1][convert_to_tile_c+1] != 0xf)){
                                        y += vy;
                                        pstatus = 2;
                                    }
```

```
                    else{
                        vy = 0;
                        vx = 1;
                        pstatus = 0;
                        wallhit =1;
                    }
                }
                else{
                    if ( rem_down > 0 ||
(control_array[convert_to_tile_r+1][convert_to_tile_c] != 0xf    &&
control_array[convert_to_tile_r+1][convert_to_tile_c-1] != 0xf)){
                        y += vy;
                        pstatus = 2;
                    }
                    else{
                        vy = 0;
                        vx = 1;
                        pstatus = 0;
                        wallhit =1;
                    }
                }

            }
        }
        }
        ////////////////////////////////////////////////////////////////////////////
        ////////////////////////////For Enemey/////////////////////////////////////
        ////////////////////////////////////////////////////////////////////////////
        ////////////////////////////////////// left

        for(i=0; i<3; i++){
            if(to_start==1)
            {


            int conver_to_tile_em_x=ex[i]/32;
            int conver_to_tile_em_y=ey[i]/32;
    if (smoke_1_on==1 &&((smoke_1_xposition==conver_to_tile_em_x &&
```

```
smoke_1_yposition==conver_to_tile_em_y) ))
      //||   (smoke_1_xposition==conver_to_tile_em_x+1 &&
smoke_1_yposition==conver_to_tile_em_y) ||
//            ((smoke_1_xposition+1)==conver_to_tile_em_x &&
smoke_1_yposition==conver_to_tile_em_y) ||
//            (smoke_1_xposition==conver_to_tile_em_x &&
smoke_1_yposition==conver_to_tile_em_y+1) ||
//            (smoke_1_xposition==conver_to_tile_em_x &&
(smoke_1_yposition+1)==conver_to_tile_em_y)))
            {
                                    evx[i] = 0;
                                    evy[i] = 0;
            }
            else if (smoke_2_on==1
&&((smoke_2_xposition==conver_to_tile_em_x &&
smoke_2_yposition==conver_to_tile_em_y)))
//              ||(smoke_2_xposition==(conver_to_tile_em_x+1) &&
smoke_2_yposition==conver_to_tile_em_y) ||
//            ((smoke_2_xposition+1)==conver_to_tile_em_x&&
smoke_2_yposition==conver_to_tile_em_y) ||
//            (smoke_2_xposition==conver_to_tile_em_x &&
smoke_2_yposition==conver_to_tile_em_y+1) ||
//            (smoke_2_xposition==conver_to_tile_em_x &&
(smoke_2_yposition+1)==conver_to_tile_em_y)) )
            {
                                    evx[i] = 0;
                                    evy[i] = 0;
            }
            else if (smoke_3_on==1 &&
((smoke_3_xposition==conver_to_tile_em_x &&
smoke_3_yposition==conver_to_tile_em_y)))
            //||(smoke_3_xposition==(conver_to_tile_em_x+1) &&
smoke_3_yposition==conver_to_tile_em_y) ||
//            ((smoke_3_xposition+1)==conver_to_tile_em_x &&
smoke_3_yposition==conver_to_tile_em_y) ||
//            (smoke_3_xposition==conver_to_tile_em_x &&
smoke_3_yposition==(conver_to_tile_em_y+1)) ||
//            (smoke_3_xposition==conver_to_tile_em_x &&
```

```
(smoke_3_yposition+1)==conver_to_tile_em_y)) )
        {
                        evx[i] = 0;
                        evy[i] = 0;
        }

                if (evx[i] == - 1)
                {
                    if(rem_up_e[i] <=8 && rem_down_e[i] <=8){
                        if ( ex[i] > 0 && (rem_left_e[i] > 0 ||
control_array[convert_to_tile_r_e[i]][convert_to_tile_c_e[i]-1] != 0xf)){
                            ex[i] += evx[i];
                            estatus[i] = 1;
                        }
                        else{
                            evx[i] = 0;
                            evy[i] = 1;
                            estatus[i] = 2;
                        }
                    }

                    else{
                        if(rem_down_e[i] > rem_up_e[i]){
                            if ( ex[i] > 0 && (rem_left_e[i] > 0 ||
(control_array[convert_to_tile_r_e[i]][convert_to_tile_c_e[i]-1] != 0xf    &&
control_array[convert_to_tile_r_e[i]+1][convert_to_tile_c_e[i]-1] != 0xf))){
                                ex[i] += evx[i];
                                estatus[i] = 1;
                            }
                            else{
                                evx[i] = 0;
                                evy[i] = 1;
                                estatus[i] = 2;
                            }
                        }
                        else{
                            if ( ex[i] > 0 && (rem_left_e[i] > 0 ||
(control_array[convert_to_tile_r_e[i]][convert_to_tile_c_e[i]-1] != 0xf    &&
```

```
control_array[convert_to_tile_r_e[i]-1][convert_to_tile_c_e[i]-1] != 0xf))){
                                        ex[i] += evx[i];
                                        estatus[i] = 1;
                                    }
                                    else{
                                        evx[i] = 0;
                                        evy[i] = 1;
                                        estatus[i] = 2;
                                    }
                                }

                            }


                        }

                        ////////////////////////////////// right
                        if (evx[i] == 1){
                            if(rem_up_e[i] <=8 && rem_down_e[i] <=8){
                                if (rem_right_e[i] > 0 ||
control_array[convert_to_tile_r_e[i]][convert_to_tile_c_e[i]+1] != 0xf){
                                        ex[i] += evx[i];
                                        estatus[i] = 0;
                                }
                                else{
                                        evx[i] = 0;
                                        evy[i] = -1;
                                        estatus[i] = 3;
                                        //wallhit =1;
                                    }
                            }
                            else{
                                if(rem_down_e[i] > rem_up_e[i]){
                                    if ( rem_right_e[i] > 0 ||
(control_array[convert_to_tile_r_e[i]][convert_to_tile_c_e[i]+1] != 0xf    &&
control_array[convert_to_tile_r_e[i]+1][convert_to_tile_c_e[i]+1] != 0xf)){
                                        ex[i] += evx[i];
                                        estatus[i] = 0;
```

```
                    }
                    else{
                        evx[i] = 0;
                        evy[i] =-1;
                        estatus[i] = 3;
                        //wallhit =1;
                    }
                }
                else{
                    if ( rem_right_e[i] > 0 ||
(control_array[convert_to_tile_r_e[i]][convert_to_tile_c_e[i]+1] != 0xf    &&
control_array[convert_to_tile_r_e[i]-1][convert_to_tile_c_e[i]+1] != 0xf)){
                        ex[i] += evx[i];
                        estatus[i] = 0;
                    }
                    else{
                        evx[i] = 0;
                        evy[i] =-1;
                        estatus[i] = 3;
                        //wallhit =1;
                    }
                }

            }

        }

        ////////////////////////////////////////////UP
        if (evy[i] == -1)
        {
            if(rem_right_e[i] <=8 && rem_left_e[i] <=8 ){
                if ( ey[i] > 0 &&(rem_up_e[i] > 0 ||
control_array[convert_to_tile_r_e[i]-1][convert_to_tile_c_e[i]] != 0xf)){
                    ey[i] += evy[i];
                    estatus[i] = 3;
                }
                else{////////////
                    evy[i] = 0;
```

```
                    evx[i] = -1;
                    estatus[i] = 1;
                    //wallhit =1;
                }
            }
            else{
                if(rem_right_e[i] > rem_left_e[i]){
                    if ( ey[i] > 0 && (rem_up_e[i] > 0 ||
(control_array[convert_to_tile_r_e[i]-1][convert_to_tile_c_e[i]] != 0xf    &&
control_array[convert_to_tile_r_e[i]-1][convert_to_tile_c_e[i]+1] != 0xf))){
                        ey[i] += evy[i];
                        estatus[i] = 3;
                    }
                    else{
                        evy[i] = 0;
                        evx[i] = -1;
                        estatus[i] = 1;

                    }
                }
                else{
                    if ( ey[i] > 0 && (rem_up_e[i] > 0 ||
(control_array[convert_to_tile_r_e[i]-1][convert_to_tile_c_e[i]] != 0xf    &&
control_array[convert_to_tile_r_e[i]-1][convert_to_tile_c_e[i]-1] != 0xf))){
                        ey[i] += evy[i];
                        estatus[i] = 3;
                    }
                    else{
                        evy[i] = 0;
                        evx[i] = -1;
                        estatus[i] = 1;
                        //wallhit =1;
                    }
                }

            }
        }
```

```
                    ////////////////////////////////////////////DOWN
                    if (evy[i] == 1){
                        if(rem_right_e[i] <=8 && rem_left_e[i]<=8){
                            if ( rem_down_e[i] > 0 ||
control_array[convert_to_tile_r_e[i]+1][convert_to_tile_c_e[i]] != 0xf){
                                ey[i] += evy[i];
                                estatus[i] = 2;
                            }
                            else{
                                evy[i] = 0;
                                evx[i] = 1;
                                //wallhit =1;
                                estatus[i] = 0;
                            }
                        }
                        else{
                            if(rem_right_e[i] > rem_left_e[i]){
                                if ( rem_down_e[i] > 0 ||
(control_array[convert_to_tile_r_e[i]+1][convert_to_tile_c_e[i]] != 0xf    &&
control_array[convert_to_tile_r_e[i]+1][convert_to_tile_c_e[i]+1] != 0xf)){
                                    ey[i] += evy[i];
                                    estatus[i] = 2;
                                }
                                else{
                                    evy[i] = 0;
                                    evx[i] = 1;
                                    estatus[i] = 0;
                                    //wallhit =1;
                                }
                            }
                            else{
                                if ( rem_down_e[i] > 0 ||
(control_array[convert_to_tile_r_e[i]+1][convert_to_tile_c_e[i]] != 0xf    &&
control_array[convert_to_tile_r_e[i]+1][convert_to_tile_c_e[i]-1] != 0xf)){
                                    ey[i] += evy[i];
                                    estatus[i] = 2;
                                }
                                else{
```

```
                        evy[i] = 0;
                        evx[i] = 1;
                        estatus[i] = 0;
                        //wallhit =1;
                    }
                }

            }
        }

        }
        }// for loop (enemy)
        //w = y*pow(2,16)+x;
        //first 5 bits for address
        //next 12 bits for x
        //next 12 bits for y
        //bits 24-26 for player stats



        w = (y << 16) + x;
        ew1 = (1 << 27) +( estatus[0]<< 24 )+ ((ey[0]-10) << 12) +
(ex[0]-10);
        ew2 = (1 << 27) +( estatus[1]<< 24 )+ ((ey[1]-10) << 12) +
(ex[1]-10);
        ew3 = (1 << 27) +( estatus[2]<< 24 )+ ((ey[2]-10) << 12) +
(ex[2]-10);
        //printf("%d, %d\n", x, ex[0]);

        /////////// playerr

        convert_to_tile_c = (player_x)/32;
        convert_to_tile_r = (player_y)/32;
        if (convert_to_tile_r==flag1_pos_y &&
convert_to_tile_c==flag1_pos_x && flag1==1)
        {
            flag1=0;
            flag_collected++;
```

```
                    flag_for_score++;
                    flagdataout = (1 << 25) + (0 << 24) + (flag1_pos_y << 12) +
flag1_pos_x;
                    IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
                }
                else if (convert_to_tile_r==8 &&
convert_to_tile_c==flag2_pos_x && flag2==1)
                {
                    flag2=0;
                    flag_collected++;
                    flag_for_score++;
                    flagdataout = (2 << 25) + (0 << 24) + ( flag2_pos_y<< 12) +
flag2_pos_x;
                    IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
                }
                else if (convert_to_tile_r==flag3_pos_y &&
convert_to_tile_c==flag3_pos_x && flag3==1)
                {
                    flag3=0;
                    flag_collected++;
                    flag_for_score++;
                    flagdataout = (3 << 25) + (0 << 12) + (flag3_pos_y << 12) +
flag3_pos_x;
                    IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
                 }
                 else if (convert_to_tile_r==flag4_pos_y &&
convert_to_tile_c==flag4_pos_x && flag4==1)
                 {
                    flag4=0;
                    flag_collected++;
                    flag_for_score++;
                    flagdataout = (4 << 25) + (0 << 24) + (flag4_pos_y<< 12) +
flag4_pos_x;
                    IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
                 }
                 else if (convert_to_tile_r==flag5_pos_y &&
convert_to_tile_c==flag5_pos_x && flag5==1)
                 {
```

```
                    flag5=0;
                    flag_collected++;
                    flag_for_score++;
                    flagdataout = (5 << 25) + (0 << 24) + (flag5_pos_y << 12) +
flag5_pos_x;
                    IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
                }
                  else if (convert_to_tile_r==flag6_pos_y &&
convert_to_tile_c==flag6_pos_x && flag6==1)
                {
                    flag6=0;
                    flag_collected++;
                    flag_for_score++;
                    flagdataout = (6 << 25) + (0 << 24) + (flag6_pos_y << 12) +
flag6_pos_x;
                    IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
                }
                  else if (convert_to_tile_r==flag7_pos_y &&
convert_to_tile_c==flag7_pos_x && flag7==1)
                {
                    flag7=0;
                    flag_collected++;
                    flag_for_score++;
                    flagdataout = (7 << 25) + (0 << 24) + (flag7_pos_y << 12) +
flag7_pos_x;
                    IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
                }
                else if (convert_to_tile_r==flag8_pos_y    &&
convert_to_tile_c==flag8_pos_x && flag8==1)
                {
                    flag8=0;
                    flag_for_score++;
                    flag_collected++;
                    flagdataout = (8 << 25) + (0 << 24) + (flag8_pos_y << 12) +
flag8_pos_x;
                    IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
                }
                else if (convert_to_tile_r==flag9_pos_y &&
```

```
convert_to_tile_c==flag9_pos_x && flag9==1)
                    {
                        flag9=0;
                        flag_collected++;
                        flag_for_score++;
                        flagdataout = (9 << 25) + (0 << 24) + (flag9_pos_y << 12) +
flag9_pos_x;
                        IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
                    }
                     else if (convert_to_tile_r==flag10_pos_y    &&
convert_to_tile_c==flag10_pos_x && flag10==1)
                    {
                        flag10=0;
                        flag_collected++;
                        flag_for_score++;
                        flagdataout = (0 << 25) + (0 << 24) + (flag10_pos_y << 12) +
flag10_pos_x;
                        IOWR_32DIRECT(LEDS_BASE, 16, flagdataout);
                    }
                   //printf("%x\n", w);
                   IOWR_32DIRECT(LEDS_BASE, 0, w); // player

                   IOWR_32DIRECT(LEDS_BASE, 4, pstatus); // player sprite
(turning or dead)

                   //IOWR_32DIRECT(LEDS_BASE, 8, ew1);
                   IOWR_32DIRECT(LEDS_BASE, 20, ew1);
                   //IOWR_32DIRECT(LEDS_BASE, 12, estatue[0]);

                   //IOWR_32DIRECT(LEDS_BASE, 16, ew2);
                   IOWR_32DIRECT(LEDS_BASE, 24, ew2);
                   //IOWR_32DIRECT(LEDS_BASE, 20, estatue[1]);

                   //IOWR_32DIRECT(LEDS_BASE, 24, ew3);
                   IOWR_32DIRECT(LEDS_BASE, 28, ew3);
                   //IOWR_32DIRECT(LEDS_BASE, 28, estatue[2]);

                   //IOWR_32DIRECT(LEDS_BASE, 0, pstatus_data);
```

```
                    delay = 0;

                   //printf("code : %c, changed? : %d, \n", code, change);
     if (pstatus==4 && lives==0)
          {
               IOWR_32DIRECT(LEDS_BASE, 40, 1);
          }




        if(pstatus==4 && lives>0)
        {


        }
        if(pstatus==4 && lives>0)
                    {
                      lives--;
                      pstatus=0;
                    to_start=0;
    if ( smoke_1_on==1)
    {
      smoke_1_on=0;
      smoke_1_xposition=0;
      smoke_1_yposition=0;
                      smokedataout = (1 << 25) + (0 << 24) + ( smoke_1_xposition <<
12) + smoke_1_yposition;
                       IOWR_32DIRECT(LEDS_BASE, 32, smokedataout);
    }
    if ( smoke_2_on==1)
    {
      smoke_2_on=0;
      smoke_2_xposition=0;
      smoke_2_yposition=0;
                      smokedataout = (2 << 25) + (0 << 24) + ( smoke_2_xposition <<
12) + smoke_2_yposition;
                       IOWR_32DIRECT(LEDS_BASE, 32, smokedataout);
```

```
    }
    if ( smoke_3_on==1)
    {
      smoke_3_on=0;
      smoke_3_xposition=0;
      smoke_3_yposition=0;
                    smokedataout = (3 << 25) + (0 << 24) + ( smoke_3_xposition <<
12) + smoke_3_yposition;
                     IOWR_32DIRECT(LEDS_BASE, 32, smokedataout);
    }

                    IOWR_32DIRECT(LEDS_BASE, 36, lives);
                        x = 509;
                        y =515;
                        ex[0]=991+240;
                        ey[0]=983+240;
                        ex[1]=413+240;
                        ey[1]=985+240;
                        ex[2]=35+240;
                        ey[2]=996+240;
                        flag_for_score=0;


                      smoke_1_timer=0;
                      smoke_2_timer=0;
                      smoke_3_timer=0;
                        vx = 0, vy= 0;
                        //evx[3] = {0,0,0};
                        //evy[3] = {0,0,0};

                        delay=0;
                        delay_smoke=0;

                        change=0;
                        wallhit =0;
                        rem_up =1;
                        rem_down =1;
                        rem_right =1;
```

```
                    rem_left =1;
                    pstatus =0;
                    fuel_amount=158;
                    //rem_up_e[3] = {1,1,1};
                    //rem_down_e[3] ={1,1,1};
                    //rem_right_e[3] ={1,1,1};
                    //rem_left_e[3] ={1,1,1};

                }

            }
            else
            {
                 if(smoke_1_timer>=1000000 && smoke_1_on==1)
                {
                smokedataout = (1 << 25) + (0 << 24) + ( smoke_1_xposition <<
12) + smoke_1_yposition;
                   IOWR_32DIRECT(LEDS_BASE, 32, smokedataout);
                   smoke_1_on=0;
                   smoke_1_timer=0;
                }
                if(smoke_2_timer>=1000000 && smoke_2_on==1)
                {
                smokedataout = (2 << 25) + (0 << 24) + ( smoke_2_xposition <<
12) + smoke_2_yposition;
                   IOWR_32DIRECT(LEDS_BASE, 32, smokedataout);
                   smoke_2_on=0;
                   smoke_2_timer=0;
                }
                if(smoke_3_timer>=1000000 && smoke_3_on==1)
                {
                smokedataout = (3 << 25) + (0 << 24) + ( smoke_3_xposition <<
12) + smoke_3_yposition;
                   IOWR_32DIRECT(LEDS_BASE, 32, smokedataout);
                   smoke_3_timer=0;
                   smoke_3_on=0;
                }
```

```
            smoke_1_timer++;
            smoke_2_timer++;
            smoke_3_timer++;
            delay++;
            delay_smoke++;
            fuel_counter++;

        }
} // while



if(wallhit ==0){
        switch (code)
        {
        case 107: // LEFT
                    //if( rem_left !=0 ){
                    vx = -1;
                    vy = 0;
                    //}
                    break;

        case 116: // RIGHT
                    vx = 1;
                    vy = 0;
                    break;

        case 117: // UP


                    vy = -1;
                    vx =    0;
                    break;


        case 114:

                    vy = 1;
```

```
                vx = 0;
                break;



        case 0x29: //space bar = emergency break for now REMOVE
LATER

                ////////////////////


                smokedataout = (1+temp << 25) + (1 << 24) + (convert_to_tile_r
<< 12) + convert_to_tile_c;
                if(to_start==1)
                {
                if(temp==0 && smoke_1_on==0 && fuel_amount>30)
                {
                    smoke_1_on=1;

                  smoke_1_timer=0;
                    smoke_1_xposition=convert_to_tile_c;
                    smoke_1_yposition=convert_to_tile_r;
                temp=1;
                fuel_amount=fuel_amount-smoke_cost;
                IOWR_32DIRECT(LEDS_BASE, 32, smokedataout);
                }
                else if(temp==1 && smoke_2_on==0 && fuel_amount>30)
                {
                    smoke_2_on=1;

                  smoke_2_timer=0;
                    smoke_2_xposition=convert_to_tile_c;
                    smoke_2_yposition=convert_to_tile_r;
                temp=2;
                fuel_amount=fuel_amount-smoke_cost;
                IOWR_32DIRECT(LEDS_BASE, 32, smokedataout);

                }
```

```
                    else if(temp==2 && smoke_3_on==0&& fuel_amount>30)
                    {

                          fuel_amount=fuel_amount-smoke_cost;
                           smoke_3_timer=0;
                        smoke_3_on=1;
                        smoke_3_xposition=convert_to_tile_c;
                        smoke_3_yposition=convert_to_tile_r;
                        temp=0;
                        IOWR_32DIRECT(LEDS_BASE, 32, smokedataout);
                    }
                    }
                     break;

               default :
                        break;
          }
                  } // if not wallhit

   }

   return 0;
}
```

HardWare codes: de2_VGA_raster

```
--------------------------------------------------------------------------------
--
-- Simple VGA raster display
--
-- Stephen A. Edwards
-- sedwards@cs.columbia.edu
--
--------------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
```

```
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;

entity de2_vga_raster is

   port (
      reset : in std_logic;
      clk     : in std_logic;                          -- Should be 25.125 MHz
write : in std_logic;
chipselect : in std_logic;
address : in unsigned (4 downto 0);
writedata : in unsigned(31 downto 0);

      VGA_CLK,                                  -- Clock
      VGA_HS,                                   -- H_SYNC
      VGA_VS,                                   -- V_SYNC
      VGA_BLANK,                                  -- BLANK
      VGA_SYNC : out std_logic;                 -- SYNC
      VGA_R,                                    -- Red[9:0]
      VGA_G,                                    -- Green[9:0]
      VGA_B : out unsigned(9 downto 0) -- Blue[9:0]
      );

end de2_vga_raster;

architecture rtl of de2_vga_raster is

   -- Video parameters

   constant HTOTAL           : integer := 800;
   constant HSYNC            : integer := 96;
   constant HBACK_PORCH    : integer := 48;
   constant HACTIVE          : integer := 640;
   constant HFRONT_PORCH : integer := 16;

   constant VTOTAL           : integer := 525;
   constant VSYNC            : integer := 2;
   constant VBACK_PORCH    : integer := 33;
```

```
constant VACTIVE       : integer := 480;
constant VFRONT_PORCH : integer := 10;

-- Rectangle Constants

constant SIDEBAR_HSTART : integer := 480;
constant SIDEBAR_HEND   : integer := 640+1;
constant SIDEBAR_VSTART : integer := -1;
constant SIDEBAR_VEND   : integer := 480;

constant MINIMAP_HSTART : integer := 481;
constant MINIMAP_HEND   : integer := 640;
constant MINIMAP_VSTART : integer := 100;
constant MINIMAP_VEND   : integer := 260;

constant WORDSCOREBITS_HSTART : integer := 481;
constant WORDSCOREBITS_HEND   : integer := 481 + 5*16;
constant WORDSCOREBITS_VSTART : integer := 14;
constant WORDSCOREBITS_VEND   : integer := 14+16;

constant SCOREBITS_HSTART : integer := 480 + 2*16;
constant SCOREBITS_HEND   : integer := 480 + 5*16;
constant SCOREBITS_VSTART : integer := 31;
constant SCOREBITS_VEND   : integer := 31+16;

constant WORDLIVES_HSTART : integer := 481;
constant WORDLIVES_HEND   : integer := 481 + 5*16;
constant WORDLIVES_VSTART : integer := SCOREBITS_VEND+16;
constant WORDLIVES_VEND   : integer := WORDLIVES_VSTART+16;

constant NUMLIVES_HSTART : integer := 480+ 4*16;
constant NUMLIVES_HEND   : integer := NUMLIVES_HSTART + 16;
constant NUMLIVES_VSTART : integer := WORDLIVES_VEND + 3;
constant NUMLIVES_VEND   : integer := NUMLIVES_VSTART+16;

constant WORDFUEL_HSTART : integer := 481;
constant WORDFUEL_HEND   : integer := WORDFUEL_HSTART + 4*16;
constant WORDFUEL_VSTART : integer := MINIMAP_VEND+12;
```

```
   constant WORDFUEL_VEND     : integer := WORDFUEL_VSTART+16;

   constant FUEL_HSTART : integer := 481;
   constant FUEL_HEND     : integer := FUEL_HSTART + 159;
   constant FUEL_VSTART : integer := MINIMAP_VEND+32;
   constant FUEL_VEND     : integer := FUEL_VSTART+16;

--player-----------
--up&down--
   constant PLAYER_UD_HSTART : integer := 234;
   constant PLAYER_UD_HEND     : integer := PLAYER_UD_HSTART + 12;
   constant PLAYER_UD_VSTART : integer := 228;
   constant PLAYER_UD_VEND     : integer := PLAYER_UD_VSTART + 25;
--left&right--
   constant PLAYER_LR_HSTART : integer := 228;
   constant PLAYER_LR_HEND     : integer := PLAYER_LR_HSTART + 25;
   constant PLAYER_LR_VSTART : integer := 234;
   constant PLAYER_LR_VEND     : integer := PLAYER_LR_VSTART + 12;
-------------------

   constant MAINMAP_HSTART : integer := 0;
   constant MAINMAP_HEND     : integer := 480;
   constant MAINMAP_VSTART : integer := -1;
   constant MAINMAP_VEND     : integer := 480;

   constant START_HSTART : integer := 192;
   constant START_HEND     : integer := 192+6*16;
   constant START_VSTART : integer := 232;
   constant START_VEND     : integer := 232+16;

   constant OWL_HSTART : integer := 192+3*16;
   constant OWL_HEND     : integer := 192+3*16+32;
   constant OWL_VSTART : integer := 200;
   constant OWL_VEND     : integer := 232;

   constant END_HSTART : integer := 176;
   constant END_HEND     : integer := 176+8*32;
   constant END_VSTART : integer := 232;
```

```
    constant END_VEND     : integer := 232+16;

-- Rectangle Signals
    --signal rectangle_h, rectangle_v, rectangle : std_logic;    -- rectangle area
    signal game_state : unsigned(31 downto 0) := (others => '0');
    signal temp_game_state : unsigned(31 downto 0) := (others => '0');
    signal start_h, start_v, start_rect : std_logic;
    signal owl_h, owl_v, owl_rect : std_logic;
    signal end_h, end_v, end_rect : std_logic;
    signal owlpos : integer := 105;
    signal owlvel : std_logic := '1';
    signal owldelay : integer := 0;
    --side bar
    signal sidebar_h, sidebar_v, sidebar_rect : std_logic;
    signal minimap_h, minimap_v, minimap_rect : std_logic;
    signal wordscorebits_h, wordscorebits_v, wordscorebits_rect : std_logic;
    signal scorebits_h, scorebits_v, scorebits_rect : std_logic;
    signal wordlives_h, wordlives_v, wordlives_rect : std_logic;
    signal numlives_h, numlives_v, numlives_rect : std_logic;
    signal wordfuel_h, wordfuel_v, wordfuel_rect : std_logic;
    signal fuel_h, fuel_v, fuel_rect : std_logic;

    --main game area
    signal player_ud_h, player_ud_v, player_ud_rect : std_logic;
    signal player_lr_h, player_lr_v, player_lr_rect : std_logic;
    signal mainmap_h, mainmap_v, mainmap_rect : std_logic;

    --flag rectangles
    signal flag0_h, flag0_v, flag0_rect : std_logic;
    signal flag1_h, flag1_v, flag1_rect : std_logic;
    signal flag2_h, flag2_v, flag2_rect : std_logic;
    signal flag3_h, flag3_v, flag3_rect : std_logic;
    signal flag4_h, flag4_v, flag4_rect : std_logic;
    signal flag5_h, flag5_v, flag5_rect : std_logic;
    signal flag6_h, flag6_v, flag6_rect : std_logic;
    signal flag7_h, flag7_v, flag7_rect : std_logic;
    signal flag8_h, flag8_v, flag8_rect : std_logic;
    signal flag9_h, flag9_v, flag9_rect : std_logic;
```

```
    --enemy rectangles
    signal enemy0_h, enemy0_v, enemy0_rect : std_logic;
    signal enemy1_h, enemy1_v, enemy1_rect : std_logic;
    signal enemy2_h, enemy2_v, enemy2_rect : std_logic;

    --smoke? what's the max?
    signal smoke0_h, smoke0_v, smoke0_rect : std_logic;
    signal smoke1_h, smoke1_v, smoke1_rect : std_logic;
    signal smoke2_h, smoke2_v, smoke2_rect : std_logic;

-- Signals for the video controller
    signal Hcount : unsigned(15 downto 0);    -- Horizontal position (0-800)
    signal Vcount : unsigned(15 downto 0);    -- Vertical position (0-524)
    signal EndOfLine, EndOfField : std_logic;

    signal vga_hblank, vga_hsync,
            vga_vblank, vga_vsync : std_logic;    -- Sync. signals

--    signal rectangle_h, rectangle_v, rectangle : std_logic;    -- rectangle area
    signal clk_counter : std_logic := '0';
    signal clk25vga : std_logic := '0';
--for testing
    signal test_cnt : integer := 0;
    signal score_shift : integer := 15;

--// Score Signals
    signal score_ones : unsigned (3 downto 0) := "0001";
    signal score_tens : unsigned (3 downto 0) := "0010";
    signal score_hundy : unsigned (3 downto 0) := "0011";


--// Player Signals
    signal temp_inx, inx : unsigned(15 downto 0) := "0000000000000000";
    signal temp_iny, iny : unsigned(15 downto 0) := "0000000000000000";

    signal player_status : unsigned (2 downto 0) := "000";
    signal fuel_status : unsigned (31 downto 0) := (others => '0');
```

```
   signal num_lives : unsigned (31 downto 0) :=
"00000000000000000000000000000011";



   signal number_of_lives : unsigned (3 downto 0) := "0011";
--// Flags
   signal flag_0 : unsigned(31 downto 0) := (others => '0');
   signal flag_1 : unsigned(31 downto 0) := (others => '0');
   signal flag_2 : unsigned(31 downto 0) := (others => '0');
   signal flag_3 : unsigned(31 downto 0) := (others => '0');
   signal flag_4 : unsigned(31 downto 0) := (others => '0');
   signal flag_5 : unsigned(31 downto 0) := (others => '0');
   signal flag_6 : unsigned(31 downto 0) := (others => '0');
   signal flag_7 : unsigned(31 downto 0) := (others => '0');
   signal flag_8 : unsigned(31 downto 0) := (others => '0');
   signal flag_9 : unsigned(31 downto 0) := (others => '0');

--// Enemy cars
   signal enemy_0 : unsigned(31 downto 0) := (others => '0');
   signal enemy_1 : unsigned(31 downto 0) := (others => '0');
   signal enemy_2 : unsigned(31 downto 0) := (others => '0');

   signal temp_enemy_0 : unsigned(31 downto 0) := (others => '0');
   signal temp_enemy_1 : unsigned(31 downto 0) := (others => '0');
   signal temp_enemy_2 : unsigned(31 downto 0) := (others => '0');

--// Smoke
   signal smoke_0 : unsigned(31 downto 0) := (others => '0');
   signal smoke_1 : unsigned(31 downto 0) := (others => '0');
   signal smoke_2 : unsigned(31 downto 0) := (others => '0');

--// Minimap signals
   signal mmPlayer : std_logic := '0';



type array_type_32x32 is array (31 downto 0) of unsigned (31 downto 0);
   signal car_w : array_type_32x32;
```

```
    signal car_r : array_type_32x32;
    signal car_lg : array_type_32x32;
    signal car_b : array_type_32x32;
    signal car_y : array_type_32x32;
    signal car_g : array_type_32x32;

    signal enemy_up_y : array_type_32x32;
    signal enemy_up_o : array_type_32x32;
    signal enemy_up_r : array_type_32x32;
    signal enemy_up_b : array_type_32x32;
    signal enemy_down_y : array_type_32x32;
    signal enemy_down_o : array_type_32x32;
    signal enemy_down_r : array_type_32x32;
    signal enemy_down_b : array_type_32x32;
    signal enemy_left_y : array_type_32x32;
    signal enemy_left_o : array_type_32x32;
    signal enemy_left_r : array_type_32x32;
    signal enemy_left_b : array_type_32x32;
    signal enemy_right_y : array_type_32x32;
    signal enemy_right_o : array_type_32x32;
    signal enemy_right_r : array_type_32x32;
    signal enemy_right_b : array_type_32x32;

    signal owl_w   : array_type_32x32;
    signal owl_r   : array_type_32x32;
    signal owl_g   : array_type_32x32;
    signal brick_lg : array_type_32x32;
    signal brick_g : array_type_32x32;
    signal brick_dg : array_type_32x32;
    signal brick_z   : array_type_32x32;

    signal road_g : array_type_32x32;
    signal road_b : array_type_32x32;
    signal smoke_i : array_type_32x32;
    signal smoke_g : array_type_32x32;

type array_type_16x16 is array (15 downto 0) of unsigned (15 downto 0);
signal sprite_S : array_type_16x16;
```

```
signal sprite_C : array_type_16x16;
signal sprite_O : array_type_16x16;
signal sprite_R : array_type_16x16;
signal sprite_E : array_type_16x16;

signal sprite_A : array_type_16x16;
--signal sprite_L : array_type_16x16;
signal sprite_Y : array_type_16x16;
signal sprite_X : array_type_16x16;
signal sprite_G : array_type_16x16;
signal sprite_M : array_type_16x16;
signal sprite_F : array_type_16x16;
signal sprite_U : array_type_16x16;
signal sprite_L : array_type_16x16;
signal sprite_I : array_type_16x16;
signal sprite_V : array_type_16x16;

signal sprite_0 : array_type_16x16;
signal sprite_1 : array_type_16x16;
signal sprite_2 : array_type_16x16;
signal sprite_3 : array_type_16x16;
signal sprite_4 : array_type_16x16;
signal sprite_5 : array_type_16x16;
signal sprite_6 : array_type_16x16;
signal sprite_7 : array_type_16x16;
signal sprite_8 : array_type_16x16;
signal sprite_9 : array_type_16x16;

   type array_type_47x188 is array (46 downto 0, 187 downto 0) of std_logic;

   --  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34
35  36  37  38  39  40  41  42  43  44  45  46
   --  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |
   signal control_array : array_type_47x188 :=      (
```

"1111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111",

"1111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111",

"1111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111",

"1111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111",

"1111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111",

"1111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111",

"1111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111",

"1111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111",

"1111111111111111111111111111111100000000000000000000000000000000000
0000000000000000011111111111111111111111111110000000000000000000000
000000000000000000000011111111111111111111111111111111",

"1111111111111111111111111111111100001110000000000000000000000000000
0000000000000000000000000000000001111111111110000111111111111111111

11111111111110000000011111111111111111111111111111",

**"11111111111111111111111111111110000000000001111111111111111111111
10000111100000000000011110000000000000000000000000000000000000000000
00000000000000000000001111111111111111111111111111111",**

**"11111111111111111111111111111110000000000000000000000000000000111
10000111100000000000011110000000011111111111000011111111111000011
11111111111110000000011111111111111111111111111111111",**

**"11111111111111111111111111111111110000000011111111111111110000111
10000111100000000000000000000000000000000000000000000000000000000000
00000000000000000011111111111111111111111111111111111",**

**"11111111111111111111111111111111110000000011110000000000000000111
10000111100000000111111111111000000001111111111111111111111111111111
11111000000000000011111111111111111111111111111111111",**

**"11111111111111111111111111111111110000000011110000111111110000111
10000111100000000111100001111000000001111000000000000000000000000000
00111100000000000011111111111111111111111111111111111",**

**"11111111111111111111111111111111110000000011110000111111110000111
10000111100000000111100001111000000001111000000000000000000000000000
00111100000000000011111111111111111111111111111111111",**

**"11111111111111111111111111111111110000000000001111000011111111111111
10000111100000000111100001111000000001111000000000000000000000000000
00111100000000000000001111111111111111111111111111111",**

**"11111111111111111111111111111111110000111100001111000000000000000000
00000111100000000111100000000000000011111111111100000000111111111
11111000000000000000011111111111111111111111111111111",**

"11111111111111111111111111111110000111100001111000011111111111111
11111111100000000111111111111000000001111111111110000000111111111
11111100000000111111111111111111111111111111111111111",

"11111111111111111111111111111110000111100000000000000000000000000
00000000000001111111100000000000000000000000000000000000000000000
00000000000001111111111111111111111111111111111111111",

"11111111111111111111111111111110000111111111111111111111111111111
11111111000011111111000000000000000000000000000000000000000000011
11111111100000000111111111111111111111111111111111111",

"11111111111111111111111111111110000000000000000000000000000000000
00000000000000000000000000000001111111111111111111000000000000011
11111111100000000111111111111111111111111111111111111",

"11111111111111111111111111111110000000000000000000000011111111111
11111111000011111111000000000000111111110000000011110000000000000
00000000000000000001111111111111111111111111111111111",

"11111111111111111111111111111110000111111111111000000001111111111
11111111000011111111000000000000111111110000000011110000111111111
11111100000000000001111111111111111111111111111111111",

"11111111111111111111111111111110000111111111111000000001111111100
00000000000001111111100000000000001111111100000000111100000000000
00000000000000000000001111111111111111111111111111111",

"11111111111111111111111111111110000111111111111000000001111111100
01111000000001111111100001111000011111111000000000000000000000011
11111111111111000011111111111111111111111111111111111",

"11111111111111111111111111111110000000000000000000000000000000000
00000000000001111111100001111000011111111111000000000000000000011
11111100000000000011111111111111111111111111111111111",

"11111111111111111111111111111110000111111111111000011111111111111
11111000000000000000000001111000011111111111111100001111111000011

111111000011110000111111111111111111111111111111111111",

"111111111111111111111111111111110000000000000000000000000000000000
00000000000000000111100001111000000000000000000111100000000000000000011
111111000011110000111111111111111111111111111111111111",

"111111111111111111111111111111110000111111111111000011110000111111
111111111111100000000000011110000000000000000011110000111100000000011
111111000011110000111111111111111111111111111111111111",

"111111111111111111111111111111110000111100001111000011110000000000000
011110000000000000000000011111111111111111111111000011111111000000
000000000011110000000001111111111111111111111111111111",

"111111111111111111111111111111110000111100001111000011110000000000000
0111100000000000000000000000000000000000000000000000000000000000011
110000111111111111000011111111111111111111111111111111",

"111111111111111111111111111111110000111100001111000011110000111100
0111100001111000011110000111100000000111111100001111111000000000011
110000111111111111000011111111111111111111111111111111",

"111111111111111111111111111111110000111100001111000011110000111100
0111100001111000011110000111100000000000000000000011111110000000000000
000000000000000000000111111111111111111111111111111111",

"111111111111111111111111111111110000111100001111000011110000111100
0111100001111000011110000111100000000111111100000000000000000000011
111111111111111111000011111111111111111111111111111111",

"111111111111111111111111111111110000111100001111000011110000111100
0111100001111000011110000111100000000111111100001111111100000000000
000000000011111111000011111111111111111111111111111111",

"111111111111111111111111111111110000111100001111111111110000111111
111110000111100001111000011110000000011111110000111111110000000000011
111111000011111111000011111111111111111111111111111111",

```
"111111111111111111111111111111110000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000011
111111000000000000000011111111111111111111111111111111111111",

"111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111",

"111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111",

"111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111",

"111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111",

"111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111",

"111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111",

"111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111",

"111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111" );
  -- | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | |
| | | | |
```

```
  --  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34
35  36  37  38  39  40  41  42  43  44  45  46



begin
--CAR body--
--CAR --
--car_white--
--VGA_R <= "1111111111";
--VGA_G <= "1111111111";
--VGA_B <= "1111111111";
car_w(0)      <= "00000000000000000000000000000000";
car_w(1)      <= "00000000000000000000000000000000";
car_w(2)      <= "00000000000000000000000000000000";
car_w(3)      <= "00000000000000000000000000000000";
car_w(4)      <= "00000000000000000000000000000000";
car_w(5)      <= "00000000000001100110000000000000";
car_w(6)      <= "00000000000101000010100000000000";
car_w(7)      <= "00000000000101000110100000000000";
car_w(8)      <= "00000000000101000010100000000000";
car_w(9)      <= "00000000000111100011100000000000";
car_w(10)     <= "00000000000111000011100000000000";
car_w(11)     <= "00000000000111100111100000000000";
car_w(12)     <= "00000000000000000000000000000000";
car_w(13)     <= "00000000000000000000000000000000";
car_w(14)     <= "00000000000010000001000000000000";
car_w(15)     <= "00000000000011100111000000000000";
car_w(16)     <= "00000000000001100110000000000000";
car_w(17)     <= "00000000000001100110000000000000";
car_w(18)     <= "00000000000001100110000000000000";
car_w(19)     <= "00000000000001100110000000000000";
car_w(20)     <= "00000000000001100110000000000000";
car_w(21)     <= "00000000000111100111100000000000";
car_w(22)     <= "00000000000110000011000000000000";
car_w(23)     <= "00000000000100000000100000000000";
car_w(24)     <= "00000000000100000000100000000000";
```

```
car_w(25)    <= "00000000001000000010000000000";
car_w(26)    <= "00000000000111001110000000000";
car_w(27)    <= "00000000000011001100000000000";
car_w(28)    <= "00000000000000000000000000000";
car_w(29)    <= "00000000000000000000000000000";
car_w(30)    <= "00000000000000000000000000000";
car_w(31)    <= "00000000000000000000000000000";

--car_green--
--VGA_R <= "0000000000";
--VGA_G <= "1100000011";
--VGA_B <= "0000000000";
car_lg(0)     <= "00000000000000000000000000000";
car_lg(1)     <= "00000000000000000000000000000";
car_lg(2)     <= "00000000000000000000000000000";
car_lg(3)     <= "00000000000000000000000000000";
car_lg(4)     <= "00000000000000011000000000000";
car_lg(5)     <= "00000000000000011000000000000";
car_lg(6)     <= "00000000000000000000000000000";
car_lg(7)     <= "00000000000000011000000000000";
car_lg(8)     <= "00000000000000000000000000000";
car_lg(9)     <= "00000000000000011000000000000";
car_lg(10)    <= "00000000000000000000000000000";
car_lg(11)    <= "00000000000000011000000000000";
car_lg(12)    <= "00000000000000000000000000000";
car_lg(13)    <= "00000000000000000000000000000";
car_lg(14)    <= "00000000000000000000000000000";
car_lg(15)    <= "00000000000000011000000000000";
car_lg(16)    <= "00000000000000011000000000000";
car_lg(17)    <= "00000000000000011000000000000";
car_lg(18)    <= "00000000000000011000000000000";
car_lg(19)    <= "00000000000000011000000000000";
car_lg(20)    <= "00000000000000011000000000000";
car_lg(21)    <= "00000000000000011000000000000";
car_lg(22)    <= "00000000000000000000000000000";
car_lg(23)    <= "00000000000000000000000000000";
car_lg(24)    <= "00000000000000000000000000000";
car_lg(25)    <= "00000000000000000000000000000";
```

```
car_lg(26)    <= "00000000000000011000000000000000";
car_lg(27)    <= "00000000000000011000000000000000";
car_lg(28)    <= "00000000000000000000000000000000";
car_lg(29)    <= "00000000000000000000000000000000";
car_lg(30)    <= "00000000000000000000000000000000";
car_lg(31)    <= "00000000000000000000000000000000";


--car_red--
--VGA_R <= "1111111111";
--VGA_G <= "0000000000";
--VGA_B <= "0000000000";

car_r(0)      <= "00000000000000000000000000000000";
car_r(1)      <= "00000000000000000000000000000000";
car_r(2)      <= "00000000000000000000000000000000";
car_r(3)      <= "00000000000000000000000000000000";
car_r(4)      <= "00000000000000000000000000000000";
car_r(5)      <= "00000000000000000000000000000000";
car_r(6)      <= "00000000000000111100000000000000";
car_r(7)      <= "00000000000000100000000000000000";
car_r(8)      <= "00000000000000111100000000000000";
car_r(9)      <= "00000000000000000100000000000000";
car_r(10)     <= "00000000000000111100000000000000";
car_r(11)     <= "00000000000000000000000000000000";
car_r(12)     <= "00000000000000000000000000000000";
car_r(13)     <= "00000000000000000000000000000000";
car_r(14)     <= "00000000000000000000000000000000";
car_r(15)     <= "00000000000000000000000000000000";
car_r(16)     <= "00000000000000000000000000000000";
car_r(17)     <= "00000000000000000000000000000000";
car_r(18)     <= "00000000000000000000000000000000";
car_r(19)     <= "00000000000000000000000000000000";
car_r(20)     <= "00000000000000000000000000000000";
car_r(21)     <= "00000000000000000000000000000000";
car_r(22)     <= "00000000000000000000000000000000";
car_r(23)     <= "00000000000000000000000000000000";
car_r(24)     <= "00000000000000000000000000000000";
```

```
car_r(25)     <= "00000000000000000000000000000000";
car_r(26)     <= "00000000000000000000000000000000";
car_r(27)     <= "00000000000000000000000000000000";
car_r(28)     <= "00000000000000000000000000000000";
car_r(29)     <= "00000000000000000000000000000000";
car_r(30)     <= "00000000000000000000000000000000";
car_r(31)     <= "00000000000000000000000000000000";

--car_black--
--VGA_R <= "0000000000";
--VGA_G <= "0000000000";
--VGA_B <= "0000000000";
car_b(0)      <= "00000000000000000000000000000000";
car_b(1)      <= "00000000000000000000000000000000";
car_b(2)      <= "00000000000000000000000000000000";
car_b(3)      <= "00000000000000000000000000000000";
car_b(4)      <= "00000000000000000000000000000000";
car_b(5)      <= "00000000000010000001000000000000";
car_b(6)      <= "00000000000010000001000000000000";
car_b(7)      <= "00000000000010000001000000000000";
car_b(8)      <= "00000000000000000000000000000000";
car_b(9)      <= "00000000000000000000000000000000";
car_b(10)     <= "00000000000000000000000000000000";
car_b(11)     <= "00000000000000000000000000000000";
car_b(12)     <= "00000000000011111111000000000000";
car_b(13)     <= "00000000000011111111000000000000";
car_b(14)     <= "00000000000001111110000000000000";
car_b(15)     <= "00000000000000000000000000000000";
car_b(16)     <= "00000000000010000001000000000000";
car_b(17)     <= "00000000000010000001000000000000";
car_b(18)     <= "00000000000010000001000000000000";
car_b(19)     <= "00000000000010000001000000000000";
car_b(20)     <= "00000000000010000001000000000000";
car_b(21)     <= "00000000000000000000000000000000";
car_b(22)     <= "00000000000001111110000000000000";
car_b(23)     <= "00000000000011111111000000000000";
car_b(24)     <= "00000000000011111111000000000000";
car_b(25)     <= "00000000000011111111000000000000";
```

```
car_b(26)    <= "00000000000000000000000000000000";
car_b(27)    <= "00000000000000000000000000000000";
car_b(28)    <= "00000000000000000000000000000000";
car_b(29)    <= "00000000000000000000000000000000";
car_b(30)    <= "00000000000000000000000000000000";
car_b(31)    <= "00000000000000000000000000000000";

--car_yellow--
--VGA_R <= "1111111111";
--VGA_G <= "1111111111";
--VGA_B <= "0000000000";
car_y(0)     <= "00000000000000000000000000000000";
car_y(1)     <= "00000000000000000000000000000000";
car_y(2)     <= "00000000000000000000000000000000";
car_y(3)     <= "00000000000000000000000000000000";
car_y(4)     <= "00000000000000000000000000000000";
car_y(5)     <= "00000000000000000000000000000000";
car_y(6)     <= "00000000000000000000000000000000";
car_y(7)     <= "00000000000000000000000000000000";
car_y(8)     <= "00000000000010000001000000000000";
car_y(9)     <= "00000000000000000000000000000000";
car_y(10)    <= "00000000000000000000000000000000";
car_y(11)    <= "00000000000000000000000000000000";
car_y(12)    <= "00000000000000000000000000000000";
car_y(13)    <= "00000000000000000000000000000000";
car_y(14)    <= "00000000000000000000000000000000";
car_y(15)    <= "00000000000000000000000000000000";
car_y(16)    <= "00000000000000000000000000000000";
car_y(17)    <= "00000000000000000000000000000000";
car_y(18)    <= "00000000000000000000000000000000";
car_y(19)    <= "00000000000000000000000000000000";
car_y(20)    <= "00000000000000000000000000000000";
car_y(21)    <= "00000000000000000000000000000000";
car_y(22)    <= "00000000000000000000000000000000";
car_y(23)    <= "00000000000000000000000000000000";
car_y(24)    <= "00000000000000000000000000000000";
car_y(25)    <= "00000000000000000000000000000000";
car_y(26)    <= "00000000000000000000000000000000";
```

```
car_y(27)    <= "00000000000110000001100000000000";
car_y(28)    <= "00000000000000000000000000000000";
car_y(29)    <= "00000000000000000000000000000000";
car_y(30)    <= "00000000000000000000000000000000";
car_y(31)    <= "00000000000000000000000000000000";

--car_gray--
--VGA_R <= "1001100111";
--VGA_G <= "1001100111";
--VGA_B <= "1001100111";
car_g(0)     <= "00000000000000000000000000000000";
car_g(1)     <= "00000000000000000000000000000000";
car_g(2)     <= "00000000000000000000000000000000";
car_g(3)     <= "00000000000000000000000000000000";
car_g(4)     <= "00000000000011111111000000000000";
car_g(5)     <= "00000000001000000001000000000000";
car_g(6)     <= "00000000010000000010000000000000";
car_g(7)     <= "00000000010000000010000000000000";
car_g(8)     <= "00000000010000000010000000000000";
car_g(9)     <= "00000000010000000010000000000000";
car_g(10)    <= "00000000010000000010000000000000";
car_g(11)    <= "00000000010000000010000000000000";
car_g(12)    <= "00000000001000000001000000000000";
car_g(13)    <= "00000000001000000001000000000000";
car_g(14)    <= "00000000001000000001000000000000";
car_g(15)    <= "00000000001000000001000000000000";
car_g(16)    <= "00000000001000000001000000000000";
car_g(17)    <= "00000000001000000001000000000000";
car_g(18)    <= "00000000001000000001000000000000";
car_g(19)    <= "00000000001000000001000000000000";
car_g(20)    <= "00000000001000000000010000000000";
car_g(21)    <= "00000000001000000000010000000000";
car_g(22)    <= "00000000001000000000010000000000";
car_g(23)    <= "00000000001000000000010000000000";
car_g(24)    <= "00000000001000000000010000000000";
car_g(25)    <= "00000000001000000000010000000000";
car_g(26)    <= "00000000001000000001000000000000";
car_g(27)    <= "00000000000000000000000000000000";
```

```
car_g(28)    <= "00000000000000000000000000000000";
car_g(29)    <= "00000000000000000000000000000000";
car_g(30)    <= "00000000000000000000000000000000";
car_g(31)    <= "00000000000000000000000000000000";
?-------------------------------------------------
--enemyup-----------------------------------------
--enemy_up_yellow--
--VGA_R <= "1111111111";
--VGA_G <= "1111111111";
--VGA_B <= "0000000000";
enemy_up_y(0)      <= "00000000000000000000000000000000";
enemy_up_y(1)      <= "00000000000000000000000000000000";
enemy_up_y(2)      <= "00000000000000000000000000000000";
enemy_up_y(3)      <= "00000000000000000000000000000000";
enemy_up_y(4)      <= "00000000000011111111000000000000";
enemy_up_y(5)      <= "00000000000111111111100000000000";
enemy_up_y(6)      <= "00000000001111111111110000000000";
enemy_up_y(7)      <= "00000000001101011110110000000000";
enemy_up_y(8)      <= "00000000001101001010100000000000";
enemy_up_y(9)      <= "00000000000101001010100000000000";
enemy_up_y(10)     <= "00000000000110011001000000000000";
enemy_up_y(11)     <= "00000000000010010010000000000000";
enemy_up_y(12)     <= "00000000000000000000000000000000";
enemy_up_y(13)     <= "00000000000000000000000000000000";
enemy_up_y(14)     <= "00000000000010000010000000000000";
enemy_up_y(15)     <= "00000000000000010001000000000000";
enemy_up_y(16)     <= "00000000000000000000000000000000";
enemy_up_y(17)     <= "00000000000000000000000000000000";
enemy_up_y(18)     <= "00000000000000000000000000000000";
enemy_up_y(19)     <= "00000000000000000000000000000000";
enemy_up_y(20)     <= "00000000000000000000000000000000";
enemy_up_y(21)     <= "00000000000000000000000000000000";
enemy_up_y(22)     <= "00000000000000000000000000000000";
enemy_up_y(23)     <= "00000000000000000000000000000000";
enemy_up_y(24)     <= "00000000000000000000000000000000";
enemy_up_y(25)     <= "00000000000000000000000000000000";
enemy_up_y(26)     <= "00000000000000000000000000000000";
enemy_up_y(27)     <= "00000000000110000001100000000000";
```

```
enemy_up_y(28)    <= "00000000000000000000000000000000";
enemy_up_y(29)    <= "00000000000000000000000000000000";
enemy_up_y(30)    <= "00000000000000000000000000000000";
enemy_up_y(31)    <= "00000000000000000000000000000000";

--enemy_orange--
--VGA_R <= "1111111111";
--VGA_G <= "0110011011";
--VGA_B <= "0000000000";
enemy_up_o(0)     <= "00000000000000000000000000000000";
enemy_up_o(1)     <= "00000000000000000000000000000000";
enemy_up_o(2)     <= "00000000000000000000000000000000";
enemy_up_o(3)     <= "00000000000000000000000000000000";
enemy_up_o(4)     <= "00000000000000000000000000000000";
enemy_up_o(5)     <= "00000000000000000000000000000000";
enemy_up_o(6)     <= "00000000000000000000000000000000";
enemy_up_o(7)     <= "00000000000010100001000000000000";
enemy_up_o(8)     <= "00000000000010110101010000000000";
enemy_up_o(9)     <= "00000000001010110101010000000000";
enemy_up_o(10)    <= "00000000001001100110110000000000";
enemy_up_o(11)    <= "00000000011011011011110000000000";
enemy_up_o(12)    <= "00000000000100000000100000000000";
enemy_up_o(13)    <= "00000000000100000000100000000000";
enemy_up_o(14)    <= "00000000000100000000100000000000";
enemy_up_o(15)    <= "00000000000111101110100000000000";
enemy_up_o(16)    <= "00000000000101111110100000000000";
enemy_up_o(17)    <= "00000000000101010110100000000000";
enemy_up_o(18)    <= "00000000000100010100100000000000";
enemy_up_o(19)    <= "00000000000100100100100000000000";
enemy_up_o(20)    <= "00000000001000100110010000000000";
enemy_up_o(21)    <= "00000000000100010010010000000000";
enemy_up_o(22)    <= "00000000001000000000110000000000";
enemy_up_o(23)    <= "00000000001000000000100000000000";
enemy_up_o(24)    <= "00000000001000000000000000000000";
enemy_up_o(25)    <= "00000000000000000000000000000000";
enemy_up_o(26)    <= "00000000000000000000000000000000";
enemy_up_o(27)    <= "00000000000000000000000000000000";
enemy_up_o(28)    <= "00000000000000000000000000000000";
```

```
enemy_up_o(29)    <= "00000000000000000000000000000000";
enemy_up_o(30)    <= "00000000000000000000000000000000";
enemy_up_o(31)    <= "00000000000000000000000000000000";

--enemy_red--
--VGA_R <= "1111111111";
--VGA_G <= "0000000000";
--VGA_B <= "0000000000";
enemy_up_r(0)    <= "00000000000000000000000000000000";
enemy_up_r(1)    <= "00000000000000000000000000000000";
enemy_up_r(2)    <= "00000000000000000000000000000000";
enemy_up_r(3)    <= "00000000000000000000000000000000";
enemy_up_r(4)    <= "00000000000000000000000000000000";
enemy_up_r(5)    <= "00000000000000000000000000000000";
enemy_up_r(6)    <= "00000000000000000000000000000000";
enemy_up_r(7)    <= "00000000000000000000000000000000";
enemy_up_r(8)    <= "00000000000000000000000000000000";
enemy_up_r(9)    <= "00000000000000000000000000000000";
enemy_up_r(10)   <= "00000000000000000000000000000000";
enemy_up_r(11)   <= "00000000000000000000000000000000";
enemy_up_r(12)   <= "00000000000000000000000000000000";
enemy_up_r(13)   <= "00000000000000000000000000000000";
enemy_up_r(14)   <= "00000000000000000000000000000000";
enemy_up_r(15)   <= "00000000000000000000000000000000";
enemy_up_r(16)   <= "00000000000000000000000000000000";
enemy_up_r(17)   <= "00000000000000101000000000000000";
enemy_up_r(18)   <= "00000000000001101010100000000000";
enemy_up_r(19)   <= "00000000000001011010100000000000";
enemy_up_r(20)   <= "00000000000001011000000000000000";
enemy_up_r(21)   <= "00000000001011111101010000000000";
enemy_up_r(22)   <= "00000000011100000010100000000000";
enemy_up_r(23)   <= "00000000011000000001100000000000";
enemy_up_r(24)   <= "00000000011000000001100000000000";
enemy_up_r(25)   <= "00000000011000000001100000000000";
enemy_up_r(26)   <= "00000000001111111111000000000000";
enemy_up_r(27)   <= "00000000000001111110000000000000";
enemy_up_o(28)   <= "00000000000000000000000000000000";
enemy_up_o(29)   <= "00000000000000000000000000000000";
```

```
enemy_up_o(30)   <= "00000000000000000000000000000000";
enemy_up_o(31)   <= "00000000000000000000000000000000";
--enemy_black--
--VGA_R <= "0000000000";
--VGA_G <= "0000000000";
--VGA_B <= "0000000000";

enemy_up_b(0)   <= "00000000000000000000000000000000";
enemy_up_b(1)   <= "00000000000000000000000000000000";
enemy_up_b(2)   <= "00000000000000000000000000000000";
enemy_up_b(3)   <= "00000000000000000000000000000000";
enemy_up_b(4)   <= "00000000000000000000000000000000";
enemy_up_b(5)   <= "00000000000011000011000000000000";
enemy_up_b(6)   <= "00000000000010000001000000000000";
enemy_up_b(7)   <= "00000000000000000000000000000000";
enemy_up_b(8)   <= "00000000000000000000000000000000";
enemy_up_b(9)   <= "00000000000000000000000000000000";
enemy_up_b(10) <= "00000000000000000000000000000000";
enemy_up_b(11) <= "00000000000000000000000000000000";
enemy_up_b(12) <= "00000000000011111111000000000000";
enemy_up_b(13) <= "00000000000011111111000000000000";
enemy_up_b(14) <= "00000000000001111110000000000000";
enemy_up_b(15) <= "00000000000000000000000000000000";
enemy_up_b(16) <= "00000000000010000001000000000000";
enemy_up_b(17) <= "00000000000010000001000000000000";
enemy_up_b(18) <= "00000000000010000001000000000000";
enemy_up_b(19) <= "00000000000010000001000000000000";
enemy_up_b(20) <= "00000000000010000001000000000000";
enemy_up_b(21) <= "00000000000000000000000000000000";
enemy_up_b(22) <= "00000000000001111110000000000000";
enemy_up_b(23) <= "00000000000011111111000000000000";
enemy_up_b(24) <= "00000000000011111111000000000000";
enemy_up_b(25) <= "00000000000011111111000000000000";
enemy_up_b(26) <= "00000000000000000000000000000000";
enemy_up_b(27) <= "00000000000000000000000000000000";
enemy_up_b(28) <= "00000000000000000000000000000000";
enemy_up_b(29) <= "00000000000000000000000000000000";
enemy_up_b(30) <= "00000000000000000000000000000000";
```

```
enemy_up_b(31) <= "00000000000000000000000000000000";
--enemy down-----------------------------------------
--enemy_yellow--
--VGA_R <= "1111111111";
--VGA_G <= "1111111111";
--VGA_B <= "0000000000";
enemy_down_y(0)     <= "00000000000000000000000000000000";
enemy_down_y(1)     <= "00000000000000000000000000000000";
enemy_down_y(2)     <= "00000000000000000000000000000000";
enemy_down_y(3)     <= "00000000000000000000000000000000";
enemy_down_y(4)     <= "00000000000110000001100000000000";
enemy_down_y(5)     <= "00000000000000000000000000000000";
enemy_down_y(6)     <= "00000000000000000000000000000000";
enemy_down_y(7)     <= "00000000000000000000000000000000";
enemy_down_y(8)     <= "00000000000000000000000000000000";
enemy_down_y(9)     <= "00000000000000000000000000000000";
enemy_down_y(10)    <= "00000000000000000000000000000000";
enemy_down_y(11)    <= "00000000000000000000000000000000";
enemy_down_y(12)    <= "00000000000000000000000000000000";
enemy_down_y(13)    <= "00000000000000000000000000000000";
enemy_down_y(14)    <= "00000000000000000000000000000000";
enemy_down_y(15)    <= "00000000000000000000000000000000";
enemy_down_y(16)    <= "00000000000010001000000000000000";
enemy_down_y(17)    <= "00000000000010000001000000000000";
enemy_down_y(18)    <= "00000000000000000000000000000000";
enemy_down_y(19)    <= "00000000000000000000000000000000";
enemy_down_y(20)    <= "00000000000001001001000000000000";
enemy_down_y(21)    <= "00000000000010011001100000000000";
enemy_down_y(22)    <= "00000000000101010010100000000000";
enemy_down_y(23)    <= "00000000000101010010110000000000";
enemy_down_y(24)    <= "00000000001101111010110000000000";
enemy_down_y(25)    <= "00000000001111111111110000000000";
enemy_down_y(26)    <= "00000000000111111111100000000000";
enemy_down_y(27)    <= "00000000000011111111000000000000";
enemy_down_y(28)    <= "00000000000000000000000000000000";
enemy_down_y(29)    <= "00000000000000000000000000000000";
enemy_down_y(30)    <= "00000000000000000000000000000000";
enemy_down_y(31)    <= "00000000000000000000000000000000";
```

```
--enemy_orange--
--VGA_R <= "1111111111";
--VGA_G <= "0110011011";
--VGA_B <= "0000000000";
enemy_down_o(0)     <= "00000000000000000000000000000000";
enemy_down_o(1)     <= "00000000000000000000000000000000";
enemy_down_o(2)     <= "00000000000000000000000000000000";
enemy_down_o(3)     <= "00000000000000000000000000000000";
enemy_down_o(4)     <= "00000000000000000000000000000000";
enemy_down_o(5)     <= "00000000000000000000000000000000";
enemy_down_o(6)     <= "00000000000000000000000000000000";
enemy_down_o(7)     <= "00000000000000000000010000000000";
enemy_down_o(8)     <= "00000000000100000000010000000000";
enemy_down_o(9)     <= "00000000001100000000010000000000";
enemy_down_o(10)    <= "00000000001001001000100000000000";
enemy_down_o(11)    <= "00000000001001100100010000000000";
enemy_down_o(12)    <= "00000000001001001001000000000000";
enemy_down_o(13)    <= "00000000001001010001000000000000";
enemy_down_o(14)    <= "00000000001011010101000000000000";
enemy_down_o(15)    <= "00000000001011111101000000000000";
enemy_down_o(16)    <= "00000000001011101111000000000000";
enemy_down_o(17)    <= "00000000001000000010000000000000";
enemy_down_o(18)    <= "00000000001000000010000000000000";
enemy_down_o(19)    <= "00000000001000000010000000000000";
enemy_down_o(20)    <= "00000000011101101101100000000000";
enemy_down_o(21)    <= "00000000011011001100100000000000";
enemy_down_o(22)    <= "00000000010101011010100000000000";
enemy_down_o(23)    <= "00000000010101011010000000000000";
enemy_down_o(24)    <= "00000000000010000101000000000000";
enemy_down_o(25)    <= "00000000000000000000000000000000";
enemy_down_o(26)    <= "00000000000000000000000000000000";
enemy_down_o(27)    <= "00000000000000000000000000000000";
enemy_down_o(28)    <= "00000000000000000000000000000000";
enemy_down_o(29)    <= "00000000000000000000000000000000";
enemy_down_o(30)    <= "00000000000000000000000000000000";
enemy_down_o(31)    <= "00000000000000000000000000000000";
--enemy_red--
```

```
--VGA_R <= "1111111111";
--VGA_G <= "0000000000";
--VGA_B <= "0000000000";
enemy_down_r(0)     <= "00000000000000000000000000000000";
enemy_down_r(1)     <= "00000000000000000000000000000000";
enemy_down_r(2)     <= "00000000000000000000000000000000";
enemy_down_r(3)     <= "00000000000000000000000000000000";
enemy_down_r(4)     <= "00000000000001111110000000000000";
enemy_down_r(5)     <= "00000000000111111111100000000000";
enemy_down_r(6)     <= "00000000001100000000110000000000";
enemy_down_r(7)     <= "00000000001100000000110000000000";
enemy_down_r(8)     <= "00000000001100000000110000000000";
enemy_down_r(9)     <= "00000000001010000001110000000000";
enemy_down_r(10)    <= "00000000001010111111010000000000";
enemy_down_r(11)    <= "00000000000000011010000000000000";
enemy_down_r(12)    <= "00000000000101011010000000000000";
enemy_down_r(13)    <= "00000000000101010110000000000000";
enemy_down_r(14)    <= "00000000000000010100000000000000";
enemy_down_r(15)    <= "00000000000000000000000000000000";
enemy_down_r(16)    <= "00000000000000000000000000000000";
enemy_down_r(17)    <= "00000000000000000000000000000000";
enemy_down_r(18)    <= "00000000000000000000000000000000";
enemy_down_r(19)    <= "00000000000000000000000000000000";
enemy_down_r(20)    <= "00000000000000000000000000000000";
enemy_down_r(21)    <= "00000000000000000000000000000000";
enemy_down_r(22)    <= "00000000000000000000000000000000";
enemy_down_r(23)    <= "00000000000000000000000000000000";
enemy_down_r(24)    <= "00000000000000000000000000000000";
enemy_down_r(25)    <= "00000000000000000000000000000000";
enemy_down_r(26)    <= "00000000000000000000000000000000";
enemy_down_r(27)    <= "00000000000000000000000000000000";
enemy_down_r(28)    <= "00000000000000000000000000000000";
enemy_down_r(29)    <= "00000000000000000000000000000000";
enemy_down_r(30)    <= "00000000000000000000000000000000";
enemy_down_r(31)    <= "00000000000000000000000000000000";

--enemy_black--
--VGA_R <= "0000000000";
```

```
--VGA_G <= "0000000000";
--VGA_B <= "0000000000";
enemy_down_b(0)      <= "00000000000000000000000000000000";
enemy_down_b(1)      <= "00000000000000000000000000000000";
enemy_down_b(2)      <= "00000000000000000000000000000000";
enemy_down_b(3)      <= "00000000000000000000000000000000";
enemy_down_b(4)      <= "00000000000000000000000000000000";
enemy_down_b(5)      <= "00000000000000000000000000000000";
enemy_down_b(6)      <= "00000000000011111111000000000000";
enemy_down_b(7)      <= "00000000000011111111000000000000";
enemy_down_b(8)      <= "00000000000011111111000000000000";
enemy_down_b(9)      <= "00000000000001111110000000000000";
enemy_down_b(10)     <= "00000000000000000000000000000000";
enemy_down_b(11)     <= "00000000000010000010000000000000";
enemy_down_b(12)     <= "00000000000010000010000000000000";
enemy_down_b(13)     <= "00000000000010000010000000000000";
enemy_down_b(14)     <= "00000000000010000010000000000000";
enemy_down_b(15)     <= "00000000000010000010000000000000";
enemy_down_b(16)     <= "00000000000000000000000000000000";
enemy_down_b(17)     <= "00000000000001111110000000000000";
enemy_down_b(18)     <= "00000000000011111111000000000000";
enemy_down_b(19)     <= "00000000000011111111000000000000";
enemy_down_b(20)     <= "00000000000000000000000000000000";
enemy_down_b(21)     <= "00000000000000000000000000000000";
enemy_down_b(22)     <= "00000000000000000000000000000000";
enemy_down_b(23)     <= "00000000000000000000000000000000";
enemy_down_b(24)     <= "00000000000000000000000000000000";
enemy_down_b(25)     <= "00000000000010000010000000000000";
enemy_down_b(26)     <= "00000000000011000011000000000000";
enemy_down_b(27)     <= "00000000000000000000000000000000";
enemy_down_r(28)     <= "00000000000000000000000000000000";
enemy_down_r(29)     <= "00000000000000000000000000000000";
enemy_down_r(30)     <= "00000000000000000000000000000000";
enemy_down_r(31)     <= "00000000000000000000000000000000";
--enemyleft-----------------------------------------
--enemy_yellow--
--VGA_R <= "1111111111";
--VGA_G <= "1111111111";
```

```
--VGA_B <= "0000000000";
enemy_left_y(0)      <= "00000000000000000000000000000000";
enemy_left_y(1)      <= "00000000000000000000000000000000";
enemy_left_y(2)      <= "00000000000000000000000000000000";
enemy_left_y(3)      <= "00000000000000000000000000000000";
enemy_left_y(4)      <= "00000000000000000000000000000000";
enemy_left_y(5)      <= "00000000000000000000000000000000";
enemy_left_y(6)      <= "00000000000000000000000000000000";
enemy_left_y(7)      <= "00000000000000000000000000000000";
enemy_left_y(8)      <= "00000000000000000000000000000000";
enemy_left_y(9)      <= "00000000000000000000000000000000";
enemy_left_y(10)     <= "00000011000000000000000000000000";
enemy_left_y(11)     <= "00000111110000000000000000010000";
enemy_left_y(12)     <= "00001110001000110000000000010000";
enemy_left_y(13)     <= "00001111110100000000000000000000";
enemy_left_y(14)     <= "00001111000000000000000000000000";
enemy_left_y(15)     <= "00001111111000000000000000000000";
enemy_left_y(16)     <= "00001111001100010000000000000000";
enemy_left_y(17)     <= "00001110000000000000000000000000";
enemy_left_y(18)     <= "00001111111000000000000000000000";
enemy_left_y(19)     <= "00001110001100100000000000010000";
enemy_left_y(20)     <= "00000111111000000000000000010000";
enemy_left_y(21)     <= "00000011100000000000000000000000";
enemy_left_y(22)     <= "00000000000000000000000000000000";
enemy_left_y(23)     <= "00000000000000000000000000000000";
enemy_left_y(24)     <= "00000000000000000000000000000000";
enemy_left_y(25)     <= "00000000000000000000000000000000";
enemy_left_y(26)     <= "00000000000000000000000000000000";
enemy_left_y(27)     <= "00000000000000000000000000000000";
enemy_left_y(28)     <= "00000000000000000000000000000000";
enemy_left_y(29)     <= "00000000000000000000000000000000";
enemy_left_y(30)     <= "00000000000000000000000000000000";
enemy_left_y(31)     <= "00000000000000000000000000000000";
--enemy_orange--
--VGA_R <= "1111111111";
--VGA_G <= "0110011011";
--VGA_B <= "0000000000";
enemy_left_o(0)      <= "00000000000000000000000000000000";
```

```
enemy_left_o(1)      <= "00000000000000000000000000000000";
enemy_left_o(2)      <= "00000000000000000000000000000000";
enemy_left_o(3)      <= "00000000000000000000000000000000";
enemy_left_o(4)      <= "00000000000000000000000000000000";
enemy_left_o(5)      <= "00000000000000000000000000000000";
enemy_left_o(6)      <= "00000000000000000000000000000000";
enemy_left_o(7)      <= "00000000000000000000000000000000";
enemy_left_o(8)      <= "00000000000000000000000000000000";
enemy_left_o(9)      <= "00000000000000000000000000000000";
enemy_left_o(10)     <= "00000000111100000000111000000000";
enemy_left_o(11)     <= "00000000001111111111001100000000";
enemy_left_o(12)     <= "00000001110100000000000000000000";
enemy_left_o(13)     <= "00000000001000011100110000000000";
enemy_left_o(14)     <= "00000000111100011111100000000000";
enemy_left_o(15)     <= "00000000000100011000000000000000";
enemy_left_o(16)     <= "00000000110000001110000000000000";
enemy_left_o(17)     <= "00000001111100011001100000000000";
enemy_left_o(18)     <= "00000000001100011100000000000000";
enemy_left_o(19)     <= "00000001110000010000000000000000";
enemy_left_o(20)     <= "00000000000111111111010000000000";
enemy_left_o(21)     <= "00000000011100000000101110000000";
enemy_left_o(22)     <= "00000000000000000000000000000000";
enemy_left_o(23)     <= "00000000000000000000000000000000";
enemy_left_o(24)     <= "00000000000000000000000000000000";
enemy_left_o(25)     <= "00000000000000000000000000000000";
enemy_left_o(26)     <= "00000000000000000000000000000000";
enemy_left_o(27)     <= "00000000000000000000000000000000";
enemy_left_o(28)     <= "00000000000000000000000000000000";
enemy_left_o(29)     <= "00000000000000000000000000000000";
enemy_left_o(30)     <= "00000000000000000000000000000000";
enemy_left_o(31)     <= "00000000000000000000000000000000";

--enemy_red--
--VGA_R <= "1111111111";
--VGA_G <= "0000000000";
--VGA_B <= "0000000000";
enemy_left_r(0)      <= "00000000000000000000000000000000";
enemy_left_r(1)      <= "00000000000000000000000000000000";
```

```
enemy_left_r(2)      <= "00000000000000000000000000000000";
enemy_left_r(3)      <= "00000000000000000000000000000000";
enemy_left_r(4)      <= "00000000000000000000000000000000";
enemy_left_r(5)      <= "00000000000000000000000000000000";
enemy_left_r(6)      <= "00000000000000000000000000000000";
enemy_left_r(7)      <= "00000000000000000000000000000000";
enemy_left_r(8)      <= "00000000000000000000000000000000";
enemy_left_r(9)      <= "00000000000000000000000000000000";
enemy_left_r(10)     <= "00000000000000000000011111000000";
enemy_left_r(11)     <= "00000000000000000011000111100000";
enemy_left_r(12)     <= "00000000000000000000011000100000";
enemy_left_r(13)     <= "00000000000000000011000000110000";
enemy_left_r(14)     <= "00000000000000000000010000110000";
enemy_left_r(15)     <= "00000000000000000111110000110000";
enemy_left_r(16)     <= "00000000000000000001110000110000";
enemy_left_r(17)     <= "00000000000000000110010000110000";
enemy_left_r(18)     <= "00000000000000000111110000110000";
enemy_left_r(19)     <= "00000000000000000000011000110000";
enemy_left_r(20)     <= "00000000000000000000001111100000";
enemy_left_r(21)     <= "00000000000000000000011111000000";
enemy_left_r(22)     <= "00000000000000000000000000000000";
enemy_left_r(23)     <= "00000000000000000000000000000000";
enemy_left_r(24)     <= "00000000000000000000000000000000";
enemy_left_r(25)     <= "00000000000000000000000000000000";
enemy_left_r(26)     <= "00000000000000000000000000000000";
enemy_left_r(27)     <= "00000000000000000000000000000000";
enemy_left_r(28)     <= "00000000000000000000000000000000";
enemy_left_r(29)     <= "00000000000000000000000000000000";
enemy_left_r(30)     <= "00000000000000000000000000000000";
enemy_left_r(31)     <= "00000000000000000000000000000000";

--enemy_black--
--VGA_R <= "0000000000";
--VGA_G <= "0000000000";
--VGA_B <= "0000000000";
enemy_left_b(0)      <= "00000000000000000000000000000000";
enemy_left_b(1)      <= "00000000000000000000000000000000";
enemy_left_b(2)      <= "00000000000000000000000000000000";
```

```
enemy_left_b(3)     <= "00000000000000000000000000000000";
enemy_left_b(4)     <= "00000000000000000000000000000000";
enemy_left_b(5)     <= "00000000000000000000000000000000";
enemy_left_b(6)     <= "00000000000000000000000000000000";
enemy_left_b(7)     <= "00000000000000000000000000000000";
enemy_left_b(8)     <= "00000000000000000000000000000000";
enemy_left_b(9)     <= "00000000000000000000000000000000";
enemy_left_b(10)    <= "00000000000000000000000000000000";
enemy_left_b(11)    <= "00000000000000000000000000000000";
enemy_left_b(12)    <= "00000000000000000000000000000000";
enemy_left_b(13)    <= "00000110000011001111100111000000";
enemy_left_b(14)    <= "00000100000011100000001111000000";
enemy_left_b(15)    <= "00000000000011100000001111000000";
enemy_left_b(16)    <= "00000000000011100000001111000000";
enemy_left_b(17)    <= "00000000000011100000001111000000";
enemy_left_b(18)    <= "00000000000011100000001111000000";
enemy_left_b(19)    <= "00000100000011100000001111000000";
enemy_left_b(20)    <= "00000110000011001111100111000000";
enemy_left_b(21)    <= "00000000000000000000000000000000";
enemy_left_b(22)    <= "00000000000000000000000000000000";
enemy_left_b(23)    <= "00000000000000000000000000000000";
enemy_left_b(24)    <= "00000000000000000000000000000000";
enemy_left_b(25)    <= "00000000000000000000000000000000";
enemy_left_b(26)    <= "00000000000000000000000000000000";
enemy_left_b(27)    <= "00000000000000000000000000000000";
enemy_left_b(28)    <= "00000000000000000000000000000000";
enemy_left_b(29)    <= "00000000000000000000000000000000";
enemy_left_b(30)    <= "00000000000000000000000000000000";
enemy_left_b(31)    <= "00000000000000000000000000000000";

--enemyright-------------------------------------------
--enemy_yellow--
--VGA_R <= "1111111111";
--VGA_G <= "1111111111";
--VGA_B <= "0000000000";
enemy_right_y(0)     <= "00000000000000000000000000000000";
enemy_right_y(1)     <= "00000000000000000000000000000000";
enemy_right_y(2)     <= "00000000000000000000000000000000";
```

```
enemy_right_y(3)     <= "00000000000000000000000000000000";
enemy_right_y(4)     <= "00000000000000000000000000000000";
enemy_right_y(5)     <= "00000000000000000000000000000000";
enemy_right_y(6)     <= "00000000000000000000000000000000";
enemy_right_y(7)     <= "00000000000000000000000000000000";
enemy_right_y(8)     <= "00000000000000000000000000000000";
enemy_right_y(9)     <= "00000000000000000000000000000000";
enemy_right_y(10)    <= "00000000000000000000000111000000";
enemy_right_y(11)    <= "00001000000000000000011111100000";
enemy_right_y(12)    <= "00001000000000000100110001110000";
enemy_right_y(13)    <= "00000000000000000000001111110000";
enemy_right_y(14)    <= "00000000000000000000000001110000";
enemy_right_y(15)    <= "00000000000000001000010011110000";
enemy_right_y(16)    <= "00000000000000000000011111110000";
enemy_right_y(17)    <= "00000000000000000000000011110000";
enemy_right_y(18)    <= "00000000000000000000001111110000";
enemy_right_y(19)    <= "00001000000000001100010001110000";
enemy_right_y(20)    <= "00001000000000000000001111100000";
enemy_right_y(21)    <= "00000000000000000000000011000000";
enemy_right_y(22)    <= "00000000000000000000000000000000";
enemy_right_y(23)    <= "00000000000000000000000000000000";
enemy_right_y(24)    <= "00000000000000000000000000000000";
enemy_right_y(25)    <= "00000000000000000000000000000000";
enemy_right_y(26)    <= "00000000000000000000000000000000";
enemy_right_y(27)    <= "00000000000000000000000000000000";
enemy_right_y(28)    <= "00000000000000000000000000000000";
enemy_right_y(29)    <= "00000000000000000000000000000000";
enemy_right_y(30)    <= "00000000000000000000000000000000";
enemy_right_y(31)    <= "00000000000000000000000000000000";
--enemy_orange--
--VGA_R <= "1111111111";
--VGA_G <= "0110011011";
--VGA_B <= "0000000000";
enemy_right_o(0)     <= "00000000000000000000000000000000";
enemy_right_o(1)     <= "00000000000000000000000000000000";
enemy_right_o(2)     <= "00000000000000000000000000000000";
enemy_right_o(3)     <= "00000000000000000000000000000000";
enemy_right_o(4)     <= "00000000000000000000000000000000";
```

```
enemy_right_o(5)      <= "00000000000000000000000000000000";
enemy_right_o(6)      <= "00000000000000000000000000000000";
enemy_right_o(7)      <= "00000000000000000000000000000000";
enemy_right_o(8)      <= "00000000000000000000000000000000";
enemy_right_o(9)      <= "00000000000000000000000000000000";
enemy_right_o(10)     <= "00000001110100000000111000000000";
enemy_right_o(11)     <= "00000000001011111111100000000000";
enemy_right_o(12)     <= "00000000000000001000001110000000";
enemy_right_o(13)     <= "00000000000000111000110000000000";
enemy_right_o(14)     <= "00000000000110011000111110000000";
enemy_right_o(15)     <= "00000000001001110000001100000000";
enemy_right_o(16)     <= "00000000000000011000100000000000";
enemy_right_o(17)     <= "00000000000111111000111100000000";
enemy_right_o(18)     <= "00000000001100111000010000000000";
enemy_right_o(19)     <= "00000000000000000000101110000000";
enemy_right_o(20)     <= "00000000110011111111110000000000";
enemy_right_o(21)     <= "00000000011100000000111100000000";
enemy_right_o(22)     <= "00000000000000000000000000000000";
enemy_right_o(23)     <= "00000000000000000000000000000000";
enemy_right_o(24)     <= "00000000000000000000000000000000";
enemy_right_o(25)     <= "00000000000000000000000000000000";
enemy_right_o(26)     <= "00000000000000000000000000000000";
enemy_right_o(27)     <= "00000000000000000000000000000000";
enemy_right_o(28)     <= "00000000000000000000000000000000";
enemy_right_o(29)     <= "00000000000000000000000000000000";
enemy_right_o(30)     <= "00000000000000000000000000000000";
enemy_right_o(31)     <= "00000000000000000000000000000000";

--enemy_red--
--VGA_R <= "1111111111";
--VGA_G <= "0000000000";
--VGA_B <= "0000000000";
enemy_right_r(0)      <= "00000000000000000000000000000000";
enemy_right_r(1)      <= "00000000000000000000000000000000";
enemy_right_r(2)      <= "00000000000000000000000000000000";
enemy_right_r(3)      <= "00000000000000000000000000000000";
enemy_right_r(4)      <= "00000000000000000000000000000000";
enemy_right_r(5)      <= "00000000000000000000000000000000";
```

```
enemy_right_r(6)      <= "00000000000000000000000000000000";
enemy_right_r(7)      <= "00000000000000000000000000000000";
enemy_right_r(8)      <= "00000000000000000000000000000000";
enemy_right_r(9)      <= "00000000000000000000000000000000";
enemy_right_r(10)     <= "00000011111000000000000000000000";
enemy_right_r(11)     <= "00000111110000000000000000000000";
enemy_right_r(12)     <= "00000100011000000000000000000000";
enemy_right_r(13)     <= "00001100001111000000000000000000";
enemy_right_r(14)     <= "00001100001001100000000000000000";
enemy_right_r(15)     <= "00001100001110000000000000000000";
enemy_right_r(16)     <= "00001100001111100000000000000000";
enemy_right_r(17)     <= "00001100001000000000000000000000";
enemy_right_r(18)     <= "00001100000011000000000000000000";
enemy_right_r(19)     <= "00000100011000000000000000000000";
enemy_right_r(20)     <= "00000111100011000000000000000000";
enemy_right_r(21)     <= "00000111111000000000000000000000";
enemy_right_r(22)     <= "00000000000000000000000000000000";
enemy_right_r(23)     <= "00000000000000000000000000000000";
enemy_right_r(24)     <= "00000000000000000000000000000000";
enemy_right_r(25)     <= "00000000000000000000000000000000";
enemy_right_r(26)     <= "00000000000000000000000000000000";
enemy_right_r(27)     <= "00000000000000000000000000000000";
enemy_right_r(28)     <= "00000000000000000000000000000000";
enemy_right_r(29)     <= "00000000000000000000000000000000";
enemy_right_r(30)     <= "00000000000000000000000000000000";
enemy_right_r(31)     <= "00000000000000000000000000000000";

--enemy_black--
--VGA_R <= "0000000000";
--VGA_G <= "0000000000";
--VGA_B <= "0000000000";

enemy_right_b(0)      <= "00000000000000000000000000000000";
enemy_right_b(1)      <= "00000000000000000000000000000000";
enemy_right_b(2)      <= "00000000000000000000000000000000";
enemy_right_b(3)      <= "00000000000000000000000000000000";
enemy_right_b(4)      <= "00000000000000000000000000000000";
enemy_right_b(5)      <= "00000000000000000000000000000000";
```

```
enemy_right_b(6)      <= "00000000000000000000000000000000";
enemy_right_b(7)      <= "00000000000000000000000000000000";
enemy_right_b(8)      <= "00000000000000000000000000000000";
enemy_right_b(9)      <= "00000000000000000000000000000000";
enemy_right_b(10)     <= "00000000000000000000000000000000";
enemy_right_b(11)     <= "00000000000000000000000000000000";
enemy_right_b(12)     <= "00000011100111110011000001100000";
enemy_right_b(13)     <= "00000011110000000111000000100000";
enemy_right_b(14)     <= "00000011110000000111000000000000";
enemy_right_b(15)     <= "00000011110000000111000000000000";
enemy_right_b(16)     <= "00000011110000000111000000000000";
enemy_right_b(17)     <= "00000011110000000111000000000000";
enemy_right_b(18)     <= "00000011110000000111000000100000";
enemy_right_b(19)     <= "00000011100111110011000001100000";
enemy_right_b(20)     <= "00000000000000000000000000000000";
enemy_right_b(21)     <= "00000000000000000000000000000000";
enemy_right_b(22)     <= "00000000000000000000000000000000";
enemy_right_b(23)     <= "00000000000000000000000000000000";
enemy_right_b(24)     <= "00000000000000000000000000000000";
enemy_right_b(25)     <= "00000000000000000000000000000000";
enemy_right_b(26)     <= "00000000000000000000000000000000";
enemy_right_b(27)     <= "00000000000000000000000000000000";
enemy_right_b(28)     <= "00000000000000000000000000000000";
enemy_right_b(29)     <= "00000000000000000000000000000000";
enemy_right_b(30)     <= "00000000000000000000000000000000";
enemy_right_b(31)     <= "00000000000000000000000000000000";
-------------------------------------------------
--road-----------------------------------------
--road_green
--G=1111111111
road_g(0)      <= "00000000000000000000000000000000";
road_g(1)      <= "01100110000000000000000001100110";
road_g(2)      <= "01100110000000000000000001100110";
road_g(3)      <= "00000000000000000000000000000000";
road_g(4)      <= "00000000000000000000000000000000";
road_g(5)      <= "01100110011000000000011001100110";
road_g(6)      <= "01100110011000000000011001100110";
road_g(7)      <= "00000000000000000000000000000000";
```

```
road_g(8)     <= "00000000000000000000000000000000";
road_g(9)     <= "00000110011001100110011001100000";
road_g(10)    <= "00000110011001100110011001100000";
road_g(11)    <= "00000000000000000000000000000000";
road_g(12)    <= "00000000000000000000000000000000";
road_g(13)    <= "00000000011001100110011000000000";
road_g(14)    <= "00000000011001100110011000000000";
road_g(15)    <= "00000000000000000000000000000000";
road_g(16)    <= "00000000000000000000000000000000";
road_g(17)    <= "00000000011001100110011000000000";
road_g(18)    <= "00000000011001100110011000000000";
road_g(19)    <= "00000000000000000000000000000000";
road_g(20)    <= "00000000000000000000000000000000";
road_g(21)    <= "00000110011001100110011001100000";
road_g(22)    <= "00000110011001100110011001100000";
road_g(23)    <= "00000000000000000000000000000000";
road_g(24)    <= "00000000000000000000000000000000";
road_g(25)    <= "01100110011000000000011001100110";
road_g(26)    <= "01100110011000000000011001100110";
road_g(27)    <= "00000000000000000000000000000000";
road_g(28)    <= "00000000000000000000000000000000";
road_g(29)    <= "01100110000000000000000001100110";
road_g(30)    <= "01100110000000000000000001100110";
road_g(31)    <= "00000000000000000000000000000000";

--road_black

road_b(0)     <= "11111111111111111111111111111111";
road_b(1)     <= "10011001111111111111111110011001";
road_b(2)     <= "10011001111111111111111110011001";
road_b(3)     <= "11111111111111111111111111111111";
road_b(4)     <= "11111111111111111111111111111111";
road_b(5)     <= "10011001100111111111100110011001";
road_b(6)     <= "10011001100111111111100110011001";
road_b(7)     <= "11111111111111111111111111111111";
road_b(8)     <= "11111111111111111111111111111111";
road_b(9)     <= "11111001100110011001100110011111";
road_b(10)    <= "11111001100110011001100110011111";
```

road_b(11)   <= "11111111111111111111111111111111";
road_b(12)   <= "11111111111111111111111111111111";
road_b(13)   <= "11111111100110011001100111111111";
road_b(14)   <= "11111111100110011001100111111111";
road_b(15)   <= "11111111111111111111111111111111";
road_b(16)   <= "11111111111111111111111111111111";
road_b(17)   <= "11111111100110011001100111111111";
road_b(18)   <= "11111111100110011001100111111111";
road_b(19)   <= "11111111111111111111111111111111";
road_b(20)   <= "11111111111111111111111111111111";
road_b(21)   <= "11111001100110011001100110011111";
road_b(22)   <= "11111001100110011001100110011111";
road_b(23)   <= "11111111111111111111111111111111";
road_b(24)   <= "11111111111111111111111111111111";
road_b(25)   <= "10011001100111111111100110011001";
road_b(26)   <= "10011001100111111111100110011001";
road_b(27)   <= "11111111111111111111111111111111";
road_b(28)   <= "11111111111111111111111111111111";
road_b(29)   <= "10011001111111111111111110011001";
road_b(30)   <= "10011001111111111111111110011001";
road_b(31)   <= "11111111111111111111111111111111";

--flag--
--flag--
--owl_green

--owl_red
--R=1111111111
owl_r(0)   <= "00000000000000000000000000000000";
owl_r(1)   <= "00000000000000000000000000000000";
owl_r(2)   <= "00000000000000000000000000000000";
owl_r(3)   <= "00000000000000000000000000000000";
owl_r(4)   <= "00000000011101110111000000000000";
owl_r(5)   <= "00000000111110101111100000000000";
owl_r(6)   <= "00000001100011011000110000000000";
owl_r(7)   <= "00000011011101110111011000000000";
owl_r(8)   <= "00000011010101110101011000000000";

```
owl_r(9)      <= "00000011011101110111011000000000";
owl_r(10)     <= "00000001100011011000110000000000";
owl_r(11)     <= "00000000111110001111100000000000";
owl_r(12)     <= "00000110011100100111001100000000";
owl_r(13)     <= "00001111000001110000011110000000";
owl_r(14)     <= "00001110100001110000010111000000";
owl_r(15)     <= "00011101100000100000011011100000";
owl_r(16)     <= "00011010100000000000010101100000";
owl_r(17)     <= "00010101100000000000011010100000";
owl_r(18)     <= "00011010100000000000010101100000";
owl_r(19)     <= "00011101100000000000011011100000";
owl_r(20)     <= "00011110100000000000010111100000";
owl_r(21)     <= "00001111100000000000011111000000";
owl_r(22)     <= "00001110000000000000001110000000";
owl_r(23)     <= "00001100000000000000000110000000";
owl_r(24)     <= "00000000000000000000000000000000";
owl_r(25)     <= "00000000001100000011000000000000";
owl_r(26)     <= "00000000001100000011000000000000";
owl_r(27)     <= "00000000011100000011100000000000";
owl_r(28)     <= "00000000110110000110110000000000";
owl_r(29)     <= "00000000110110000110110000000000";
owl_r(30)     <= "00000000000000000000000000000000";
owl_r(31)     <= "00000000000000000000000000000000";


--owl_white
--R=1111111111
--g=1111111111
--b=1111111111

owl_w(0)      <= "00000000000000000000000000000000";
owl_w(1)      <= "00000000000000000000000000000000";
owl_w(2)      <= "00000000000000000000000000000000";
owl_w(3)      <= "00000000000000000000000000000000";
owl_w(4)      <= "00000000000010001000000000000000";
owl_w(5)      <= "00000000000001010000000000000000";
owl_w(6)      <= "00000000011100100111000000000000";
owl_w(7)      <= "00000000100010001000100000000000";
```

```
owl_w(8)      <= "00000000100010001000100000000000";
owl_w(9)      <= "00000000100010001000100000000000";
owl_w(10)     <= "00000010011100100111001000000000";
owl_w(11)     <= "00000011000001110000011000000000";
owl_w(12)     <= "00000001100011011000110000000000";
owl_w(13)     <= "00000000111110001111100000000000";
owl_w(14)     <= "00000001011110001111101000000000";
owl_w(15)     <= "00000010011111011111100100000000";
owl_w(16)     <= "00000101011111111111101010000000";
owl_w(17)     <= "00001010011111111111100101000000";
owl_w(18)     <= "00000101011111111111101010000000";
owl_w(19)     <= "00000010011111111111100100000000";
owl_w(20)     <= "00000001011111111111101000000000";
owl_w(21)     <= "00000000011111111111100000000000";
owl_w(22)     <= "00000000011111111111100000000000";
owl_w(23)     <= "00000000011111111111100000000000";
owl_w(24)     <= "00000000001111111111000000000000";
owl_w(25)     <= "00000000000000000000000000000000";
owl_w(26)     <= "00000000000000000000000000000000";
owl_w(27)     <= "00000000000000000000000000000000";
owl_w(28)     <= "00000000000000000000000000000000";
owl_w(29)     <= "00000000000000000000000000000000";
owl_w(30)     <= "00000000000000000000000000000000";
owl_w(31)     <= "00000000000000000000000000000000";

--brick--
--brick_lightgreen
--G=1111111111
brick_lg(0)     <= "00000000000000011000000000000000";
brick_lg(1)     <= "00000000000001111110000000000000";
brick_lg(2)     <= "00000000000111111111100000000000";
brick_lg(3)     <= "00000000011111111111111000000000";
brick_lg(4)     <= "00000001111111111111111110000000";
brick_lg(5)     <= "00000111111111111111111111100000";
brick_lg(6)     <= "00011111111111111111111111111000";
brick_lg(7)     <= "00111111111111111111111111111100";
brick_lg(8)     <= "01111111111111111111111111111110";
brick_lg(9)     <= "11111111111111111111111111111111";
```

```
brick_lg(10)    <= "01111111111111111111111111111110";
brick_lg(11)    <= "00111111111111111111111111111100";
brick_lg(12)    <= "00001111111111111111111111110000";
brick_lg(13)    <= "00000011111111111111111111000000";
brick_lg(14)    <= "00000000111111111111111100000000";
brick_lg(15)    <= "00000000001111111111110000000000";
brick_lg(16)    <= "00000000000011111111000000000000";
brick_lg(17)    <= "00000000000000111100000000000000";
brick_lg(18)    <= "00000000000000011000000000000000";
brick_lg(19)    <= "00000000000000000000000000000000";
brick_lg(20)    <= "00000000000000000000000000000000";
brick_lg(21)    <= "00000000000000000000000000000000";
brick_lg(22)    <= "00000000000000000000000000000000";
brick_lg(23)    <= "00000000000000000000000000000000";
brick_lg(24)    <= "00000000000000000000000000000000";
brick_lg(25)    <= "00000000000000000000000000000000";
brick_lg(26)    <= "00000000000000000000000000000000";
brick_lg(27)    <= "00000000000000000000000000000000";
brick_lg(28)    <= "00000000000000000000000000000000";
brick_lg(29)    <= "00000000000000000000000000000000";
brick_lg(30)    <= "00000000000000000000000000000000";
brick_lg(31)    <= "00000000000000000000000000000000";

--brick_green
--G=1001100111
brick_g(0)      <= "00000000000000000000000000000000";
brick_g(1)      <= "00000000000000000000000000000000";
brick_g(2)      <= "00000000000000000000000000000000";
brick_g(3)      <= "00000000000000000000000000000000";
brick_g(4)      <= "00000000000000000000000000000000";
brick_g(5)      <= "00000000000000000000000000000000";
brick_g(6)      <= "00000000000000000000000000000000";
brick_g(7)      <= "00000000000000000000000000000000";
brick_g(8)      <= "00000000000000000000000000000000";
brick_g(9)      <= "00000000000000000000000000000000";
brick_g(10)     <= "00000000000000000000000000000001";
brick_g(11)     <= "00000000000000000000000000000011";
brick_g(12)     <= "00000000000000000000000000001111";
```

```
brick_g(13)    <= "00000000000000000000000000111111";
brick_g(14)    <= "00000000000000000000000011111111";
brick_g(15)    <= "00000000000000000000001111111111";
brick_g(16)    <= "00000000000000000000111111111111";
brick_g(17)    <= "00000000000000000011111111111111";
brick_g(18)    <= "00000000000000000111111111111111";
brick_g(19)    <= "00000000000000001111111111111111";
brick_g(20)    <= "00000000000000001111111111111111";
brick_g(21)    <= "00000000000000001111111111111111";
brick_g(22)    <= "00000000000000001111111111111111";
brick_g(23)    <= "00000000000000001111111111111110";
brick_g(24)    <= "00000000000000001111111111111100";
brick_g(25)    <= "00000000000000001111111111110000";
brick_g(26)    <= "00000000000000001111111111000000";
brick_g(27)    <= "00000000000000001111111100000000";
brick_g(28)    <= "00000000000000001111110000000000";
brick_g(29)    <= "00000000000000001111000000000000";
brick_g(30)    <= "00000000000000001100000000000000";
brick_g(31)    <= "00000000000000001000000000000000";

--brick_darkgreen
--G=0011001111
brick_dg(0)     <= "00000000000000000000000000000000";
brick_dg(1)     <= "00000000000000000000000000000000";
brick_dg(2)     <= "00000000000000000000000000000000";
brick_dg(3)     <= "00000000000000000000000000000000";
brick_dg(4)     <= "00000000000000000000000000000000";
brick_dg(5)     <= "00000000000000000000000000000000";
brick_dg(6)     <= "00000000000000000000000000000000";
brick_dg(7)     <= "00000000000000000000000000000000";
brick_dg(8)     <= "00000000000000000000000000000000";
brick_dg(9)     <= "00000000000000000000000000000000";
brick_dg(10)    <= "10000000000000000000000000000000";
brick_dg(11)    <= "11000000000000000000000000000000";
brick_dg(12)    <= "11110000000000000000000000000000";
brick_dg(13)    <= "11111100000000000000000000000000";
brick_dg(14)    <= "11111111000000000000000000000000";
brick_dg(15)    <= "11111111110000000000000000000000";
```

```
brick_dg(16)    <= "11111111111100000000000000000000";
brick_dg(17)    <= "11111111111110000000000000000000";
brick_dg(18)    <= "11111111111111000000000000000000";
brick_dg(19)    <= "11111111111111100000000000000000";
brick_dg(20)    <= "11111111111111110000000000000000";
brick_dg(21)    <= "11111111111111110000000000000000";
brick_dg(22)    <= "11111111111111110000000000000000";
brick_dg(23)    <= "01111111111111110000000000000000";
brick_dg(24)    <= "00111111111111110000000000000000";
brick_dg(25)    <= "00001111111111110000000000000000";
brick_dg(26)    <= "00000111111111110000000000000000";
brick_dg(27)    <= "00000001111111110000000000000000";
brick_dg(28)    <= "00000000011111110000000000000000";
brick_dg(29)    <= "00000000000111100000000000000000";
brick_dg(30)    <= "00000000000000110000000000000000";
brick_dg(31)    <= "00000000000000010000000000000000";


--smoke
--smoke_ivory--
--VGA_R <= "1111111111";
--VGA_G <= "1111111111";
--VGA_B <= "1011010011";

smoke_i(0)      <= "00000000000000000000000000000000";
smoke_i(1)      <= "00000000000111000000000000000000";
smoke_i(2)      <= "00000000000111100000000111110000";
smoke_i(3)      <= "00000111011111100000111111111000";
smoke_i(4)      <= "00001111011001100100111111111100";
smoke_i(5)      <= "00011111011111111100111111111100";
smoke_i(6)      <= "00011110011111111110011111111010";
smoke_i(7)      <= "00011110111110000011001111110010";
smoke_i(8)      <= "00001111111100111001100111000110";
smoke_i(9)      <= "00001111100000111101111111111110";
smoke_i(10)     <= "00000011101110111101111111111100";
smoke_i(11)     <= "01111011011100111101111000000000";
smoke_i(12)     <= "11111111011111111101110011110000";
smoke_i(13)     <= "11111111011111111111110111111000";
```

```
smoke_i(14)    <= "11111000000111100000011111111000";
smoke_i(15)    <= "11111011110011001111000111111100";
smoke_i(16)    <= "11110111111101011111110011111100";
smoke_i(17)    <= "11110111111110111111111011111100";
smoke_i(18)    <= "11100111111110111111111101111100";
smoke_i(19)    <= "11000011111110111111111001111000";
smoke_i(20)    <= "00011001000110111111111011110000";
smoke_i(21)    <= "00111100011110111111111111101000";
smoke_i(22)    <= "00111100111110011111110000011110";
smoke_i(23)    <= "00111101111110000111100110011110";
smoke_i(24)    <= "00011101111111111111001111101110";
smoke_i(25)    <= "00011111111111111111111111110000";
smoke_i(26)    <= "00001111110111111111111111110000";
smoke_i(27)    <= "00000111110011111001111111110000";
smoke_i(28)    <= "00000001111101110110111111100000";
smoke_i(29)    <= "00000000111100110111011110000000";
smoke_i(30)    <= "00000000001110111110000000000000";
smoke_i(31)    <= "00000000000000011100000000000000";

--smoke_green--
    --VGA_R <= "0000000000";
    --VGA_G <= "1111111111";
    --VGA_B <= "1100000000";

------------------------------------------------------------
--G
sprite_G(0)   <= "0000000000000000";
sprite_G(1)   <= "0000000000000000";
sprite_G(2)   <= "0000000000000000";
sprite_G(3)   <= "0001111111111000";
sprite_G(4)   <= "0001111111111000";
sprite_G(5)   <= "0001100000000000";
sprite_G(6)   <= "0001100000000000";
sprite_G(7)   <= "0001100000000000";
sprite_G(8)   <= "0001100011111000";
sprite_G(9)   <= "0001100011111000";
sprite_G(10) <= "0001100000011000";
sprite_G(11) <= "0001100000011000";
```

```
sprite_G(12) <= "0001111111111000";
sprite_G(13) <= "0000000000000000";
sprite_G(14) <= "0000000000000000";
sprite_G(15) <= "0000000000000000";

--M
sprite_M(0)   <= "0000000000000000";
sprite_M(1)   <= "0000000000000000";
sprite_M(2)   <= "0000000000000000";
sprite_M(3)   <= "0001100000011000";
sprite_M(4)   <= "0001110000111000";
sprite_M(5)   <= "0001111001111000";
sprite_M(6)   <= "0001100110011000";
sprite_M(7)   <= "0001100110011000";
sprite_M(8)   <= "0001100110011000";
sprite_M(9)   <= "0001100000011000";
sprite_M(10) <= "0001100000011000";
sprite_M(11) <= "0001100000011000";
sprite_M(12) <= "0001100000011000";
sprite_M(13) <= "0000000000000000";
sprite_M(14) <= "0000000000000000";
sprite_M(15) <= "0000000000000000";

--f
sprite_F(0)   <= "0000000000000000";
sprite_F(1)   <= "0000000000000000";
sprite_F(2)   <= "0000000000000000";
sprite_F(3)   <= "0001111111111000";
sprite_F(4)   <= "0001111111111000";
sprite_F(5)   <= "0001100000000000";
sprite_F(6)   <= "0001100000000000";
sprite_F(7)   <= "0001111111111000";
sprite_F(8)   <= "0001111111111000";
sprite_F(9)   <= "0001100000000000";
sprite_F(10) <= "0001100000000000";
sprite_F(11) <= "0001100000000000";
sprite_F(12) <= "0001100000000000";
sprite_F(13) <= "0000000000000000";
```

```
sprite_F(14) <= "0000000000000000";
sprite_F(15) <= "0000000000000000";
--u
sprite_U(0)   <= "0000000000000000";
sprite_U(1)   <= "0000000000000000";
sprite_U(2)   <= "0000000000000000";
sprite_U(3)   <= "0001100000011000";
sprite_U(4)   <= "0001100000011000";
sprite_U(5)   <= "0001100000011000";
sprite_U(6)   <= "0001100000011000";
sprite_U(7)   <= "0001100000011000";
sprite_U(8)   <= "0001100000011000";
sprite_U(9)   <= "0001100000011000";
sprite_U(10) <= "0001100000011000";
sprite_U(11) <= "0001111111111000";
sprite_U(12) <= "0001111111111000";
sprite_U(13) <= "0000000000000000";
sprite_U(14) <= "0000000000000000";
sprite_U(15) <= "0000000000000000";
--L
sprite_L(0)   <= "0000000000000000";
sprite_L(1)   <= "0000000000000000";
sprite_L(2)   <= "0000000000000000";
sprite_L(3)   <= "0001100000000000";
sprite_L(4)   <= "0001100000000000";
sprite_L(5)   <= "0001100000000000";
sprite_L(6)   <= "0001100000000000";
sprite_L(7)   <= "0001100000000000";
sprite_L(8)   <= "0001100000000000";
sprite_L(9)   <= "0001100000000000";
sprite_L(10) <= "0001100000000000";
sprite_L(11) <= "0001111111111000";
sprite_L(12) <= "0001111111111000";
sprite_L(13) <= "0000000000000000";
sprite_L(14) <= "0000000000000000";
sprite_L(15) <= "0000000000000000";
--I
sprite_I(0)   <= "0000000000000000";
```

```
sprite_I(1)   <= "0000000000000000";
sprite_I(2)   <= "0000000000000000";
sprite_I(3)   <= "0001111111111000";
sprite_I(4)   <= "0001111111111000";
sprite_I(5)   <= "0000000110000000";
sprite_I(6)   <= "0000000110000000";
sprite_I(7)   <= "0000000110000000";
sprite_I(8)   <= "0000000110000000";
sprite_I(9)   <= "0000000110000000";
sprite_I(10) <= "0000000110000000";
sprite_I(11) <= "0001111111111000";
sprite_I(12) <= "0001111111111000";
sprite_I(13) <= "0000000000000000";
sprite_I(14) <= "0000000000000000";
sprite_I(15) <= "0000000000000000";
--v
sprite_V(0)   <= "0000000000000000";
sprite_V(1)   <= "0000000000000000";
sprite_V(2)   <= "0000000000000000";
sprite_V(3)   <= "0001100000011000";
sprite_V(4)   <= "0001100000011000";
sprite_V(5)   <= "0001100000011000";
sprite_V(6)   <= "0001100000011000";
sprite_V(7)   <= "0001100000011000";
sprite_V(8)   <= "0001100000011000";
sprite_V(9)   <= "0000110000110000";
sprite_V(10) <= "0000011001100000";
sprite_V(11) <= "0000001111000000";
sprite_V(12) <= "0000000110000000";
sprite_V(13) <= "0000000000000000";
sprite_V(14) <= "0000000000000000";
sprite_V(15) <= "0000000000000000";

-- 'S' 'C' 'O' 'R' 'E'
------------------------------------------------------------
sprite_S(0)   <= "0000000000000000";
sprite_S(1)   <= "0000000000000000";
sprite_S(2)   <= "0000000000000000";
```

```
sprite_S(3)   <= "0001111111111000";
sprite_S(4)   <= "0001111111111000";
sprite_S(5)   <= "0001100000000000";
sprite_S(6)   <= "0001100000000000";
sprite_S(7)   <= "0001111111111000";
sprite_S(8)   <= "0001111111111000";
sprite_S(9)   <= "0000000000011000";
sprite_S(10) <= "0000000000011000";
sprite_S(11) <= "0001111111111000";
sprite_S(12) <= "0001111111111000";
sprite_S(13) <= "0000000000000000";
sprite_S(14) <= "0000000000000000";
sprite_S(15) <= "0000000000000000";

sprite_C( 0) <= "0000000000000000";
sprite_C( 1) <= "0000000000000000";
sprite_C( 2) <= "0000000000000000";
sprite_C( 3) <= "0001111111111000";
sprite_C( 4) <= "0001111111111000";
sprite_C( 5) <= "0001100000000000";
sprite_C( 6) <= "0001100000000000";
sprite_C( 7) <= "0001100000000000";
sprite_C( 8) <= "0001100000000000";
sprite_C( 9) <= "0001100000000000";
sprite_C(10) <= "0001100000000000";
sprite_C(11) <= "0001111111111000";
sprite_C(12) <= "0001111111111000";
sprite_C(13) <= "0000000000000000";
sprite_C(14) <= "0000000000000000";
sprite_C(15) <= "0000000000000000";

sprite_O( 0) <= "0000000000000000";
sprite_O( 1) <= "0000000000000000";
sprite_O( 2) <= "0000000000000000";
sprite_O( 3) <= "0001111111111000";
sprite_O( 4) <= "0001111111111000";
sprite_O( 5) <= "0001100000011000";
sprite_O( 6) <= "0001100000011000";
```

```
sprite_O( 7) <= "0001100000011000";
sprite_O( 8) <= "0001100000011000";
sprite_O( 9) <= "0001100000011000";
sprite_O(10) <= "0001100000011000";
sprite_O(11) <= "0001111111111000";
sprite_O(12) <= "0001111111111000";
sprite_O(13) <= "0000000000000000";
sprite_O(14) <= "0000000000000000";
sprite_O(15) <= "0000000000000000";

sprite_R( 0) <= "0000000000000000";
sprite_R( 1) <= "0000000000000000";
sprite_R( 2) <= "0000000000000000";
sprite_R( 3) <= "0001111111111000";
sprite_R( 4) <= "0001111111111000";
sprite_R( 5) <= "0001100000011000";
sprite_R( 6) <= "0001100000011000";
sprite_R( 7) <= "0001111111111000";
sprite_R( 8) <= "0001111111111000";
sprite_R( 9) <= "0001100111000000";
sprite_R(10) <= "0001100011100000";
sprite_R(11) <= "0001100001110000";
sprite_R(12) <= "0001100000111000";
sprite_R(13) <= "0000000000000000";
sprite_R(14) <= "0000000000000000";
sprite_R(15) <= "0000000000000000";

sprite_E( 0) <= "0000000000000000";
sprite_E( 1) <= "0000000000000000";
sprite_E( 2) <= "0000000000000000";
sprite_E( 3) <= "0001111111111000";
sprite_E( 4) <= "0001111111111000";
sprite_E( 5) <= "0001100000000000";
sprite_E( 6) <= "0001100000000000";
sprite_E( 7) <= "0001111111111000";
sprite_E( 8) <= "0001111111111000";
sprite_E( 9) <= "0001100000000000";
sprite_E(10) <= "0001100000000000";
```

```
sprite_E(11) <= "0001111111111000";
sprite_E(12) <= "0001111111111000";
sprite_E(13) <= "0000000000000000";
sprite_E(14) <= "0000000000000000";
sprite_E(15) <= "0000000000000000";

--A
sprite_A(0)  <= "0000000000000000";
sprite_A(1)  <= "0000000000000000";
sprite_A(2)  <= "0000000000000000";
sprite_A(3)  <= "0001111111111000";
sprite_A(4)  <= "0001111111111000";
sprite_A(5)  <= "0001100000011000";
sprite_A(6)  <= "0001100000011000";
sprite_A(7)  <= "0001111111111000";
sprite_A(8)  <= "0001111111111000";
sprite_A(9)  <= "0001100000011000";
sprite_A(10) <= "0001100000011000";
sprite_A(11) <= "0001100000011000";
sprite_A(12) <= "0001100000011000";
sprite_A(13) <= "0000000000000000";
sprite_A(14) <= "0000000000000000";
sprite_A(15) <= "0000000000000000";


--L
sprite_L(0)  <= "0000000000000000";
sprite_L(1)  <= "0000000000000000";
sprite_L(2)  <= "0000000000000000";
sprite_L(3)  <= "0001100000000000";
sprite_L(4)  <= "0001100000000000";
sprite_L(5)  <= "0001100000000000";
sprite_L(6)  <= "0001100000000000";
sprite_L(7)  <= "0001100000000000";
sprite_L(8)  <= "0001100000000000";
sprite_L(9)  <= "0001100000000000";
sprite_L(10) <= "0001100000000000";
sprite_L(11) <= "0001111111111000";
```

```
sprite_L(12) <= "0001111111111000";
sprite_L(13) <= "0000000000000000";
sprite_L(14) <= "0000000000000000";
sprite_L(15) <= "0000000000000000";


--Y
sprite_Y(0)  <= "0000000000000000";
sprite_Y(1)  <= "0000000000000000";
sprite_Y(2)  <= "0000000000000000";
sprite_Y(3)  <= "0001100000011000";
sprite_Y(4)  <= "0000110000110000";
sprite_Y(5)  <= "0000011001100000";
sprite_Y(6)  <= "0000001111000000";
sprite_Y(7)  <= "0000000110000000";
sprite_Y(8)  <= "0000000110000000";
sprite_Y(9)  <= "0000000110000000";
sprite_Y(10) <= "0000000110000000";
sprite_Y(11) <= "0000000110000000";
sprite_Y(12) <= "0000000110000000";
sprite_Y(13) <= "0000000000000000";
sprite_Y(14) <= "0000000000000000";
sprite_Y(15) <= "0000000000000000";

--X
sprite_X(0)  <= "0000000000000000";
sprite_X(1)  <= "0000000000000000";
sprite_X(2)  <= "0000000000000000";
sprite_X(3)  <= "0001100000011000";
sprite_X(4)  <= "0000110000110000";
sprite_X(5)  <= "0000011001100000";
sprite_X(6)  <= "0000001111000000";
sprite_X(7)  <= "0000000110000000";
sprite_X(8)  <= "0000000110000000";
sprite_X(9)  <= "0000001111000000";
sprite_X(10) <= "0000011001100000";
sprite_X(11) <= "0000110000110000";
sprite_X(12) <= "0001100000011000";
```

```
sprite_X(13) <= "0000000000000000";
sprite_X(14) <= "0000000000000000";
sprite_X(15) <= "0000000000000000";

------------------------------------------------------------
--Digits for score '0' '1' '2' '3' '4' '5' '6' '7' '8' '9'
------------------------------------------------------------
sprite_0( 0) <= "0000000000000000";
sprite_0( 1) <= "0000000000000000";
sprite_0( 2) <= "0000000000000000";
sprite_0( 3) <= "0000111111110000";
sprite_0( 4) <= "0001111111111000";
sprite_0( 5) <= "0001100000011000";
sprite_0( 6) <= "0001100000011000";
sprite_0( 7) <= "0001100000011000";
sprite_0( 8) <= "0001100000011000";
sprite_0( 9) <= "0001100000011000";
sprite_0(10) <= "0001100000011000";
sprite_0(11) <= "0001111111111000";
sprite_0(12) <= "0000111111110000";
sprite_0(13) <= "0000000000000000";
sprite_0(14) <= "0000000000000000";
sprite_0(15) <= "0000000000000000";

sprite_1( 0) <= "0000000000000000";
sprite_1( 1) <= "0000000000000000";
sprite_1( 2) <= "0000000000000000";
sprite_1( 3) <= "0000001111000000";
sprite_1( 4) <= "0000011111000000";
sprite_1( 5) <= "0000111111000000";
sprite_1( 6) <= "0000000111000000";
sprite_1( 7) <= "0000000111000000";
sprite_1( 8) <= "0000000111000000";
sprite_1( 9) <= "0000000111000000";
sprite_1(10) <= "0000000111000000";
sprite_1(11) <= "0000000111000000";
sprite_1(12) <= "0000000111000000";
sprite_1(13) <= "0000000000000000";
```

```
sprite_1(14) <= "0000000000000000";
sprite_1(15) <= "0000000000000000";

sprite_2( 0) <= "0000000000000000";
sprite_2( 1) <= "0000000000000000";
sprite_2( 2) <= "0000000000000000";
sprite_2( 3) <= "0000111111110000";
sprite_2( 4) <= "0001111111111000";
sprite_2( 5) <= "0001110000111000";
sprite_2( 6) <= "0000000000111000";
sprite_2( 7) <= "0000000001110000";
sprite_2( 8) <= "0000000011100000";
sprite_2( 9) <= "0000000111000000";
sprite_2(10) <= "0000001110000000";
sprite_2(11) <= "0000111111111000";
sprite_2(12) <= "0001111111111000";
sprite_2(13) <= "0000000000000000";
sprite_2(14) <= "0000000000000000";
sprite_2(15) <= "0000000000000000";

sprite_3( 0) <= "0000000000000000";
sprite_3( 1) <= "0000000000000000";
sprite_3( 2) <= "0000000000000000";
sprite_3( 3) <= "0000111111110000";
sprite_3( 4) <= "0001111111111000";
sprite_3( 5) <= "0001100000011000";
sprite_3( 6) <= "0000000000011000";
sprite_3( 7) <= "0000000111110000";
sprite_3( 8) <= "0000000111110000";
sprite_3( 9) <= "0000000000011000";
sprite_3(10) <= "0001100000011000";
sprite_3(11) <= "0001111111111000";
sprite_3(12) <= "0000111111110000";
sprite_3(13) <= "0000000000000000";
sprite_3(14) <= "0000000000000000";
sprite_3(15) <= "0000000000000000";

sprite_4( 0) <= "0000000000000000";
```

```
sprite_4( 1) <= "0000000000000000";
sprite_4( 2) <= "0000000000000000";
sprite_4( 3) <= "0000000011111000";
sprite_4( 4) <= "0000000111111000";
sprite_4( 5) <= "0000001110011000";
sprite_4( 6) <= "0000011100011000";
sprite_4( 7) <= "0000111000011000";
sprite_4( 8) <= "0001111111111100";
sprite_4( 9) <= "0001111111111100";
sprite_4(10) <= "0000000000011000";
sprite_4(11) <= "0000000000011000";
sprite_4(12) <= "0000000000011000";
sprite_4(13) <= "0000000000000000";
sprite_4(14) <= "0000000000000000";
sprite_4(15) <= "0000000000000000";

sprite_5( 0) <= "0000000000000000";
sprite_5( 1) <= "0000000000000000";
sprite_5( 2) <= "0000000000000000";
sprite_5( 3) <= "0001111111111000";
sprite_5( 4) <= "0001111111111000";
sprite_5( 5) <= "0001100000000000";
sprite_5( 6) <= "0001100000000000";
sprite_5( 7) <= "0001111111110000";
sprite_5( 8) <= "0001111111111000";
sprite_5( 9) <= "0000000000011000";
sprite_5(10) <= "0001100000011000";
sprite_5(11) <= "0001111111111000";
sprite_5(12) <= "0000111111110000";
sprite_5(13) <= "0000000000000000";
sprite_5(14) <= "0000000000000000";
sprite_5(15) <= "0000000000000000";

sprite_6( 0) <= "0000000000000000";
sprite_6( 1) <= "0000000000000000";
sprite_6( 2) <= "0000000000000000";
sprite_6( 3) <= "0000111111110000";
sprite_6( 4) <= "0001111111111000";
```

```
sprite_6( 5) <= "0001100000011000";
sprite_6( 6) <= "0001100000000000";
sprite_6( 7) <= "0001111111110000";
sprite_6( 8) <= "0001111111111000";
sprite_6( 9) <= "0001100000011000";
sprite_6(10) <= "0001100000011000";
sprite_6(11) <= "0001111111111000";
sprite_6(12) <= "0000111111110000";
sprite_6(13) <= "0000000000000000";
sprite_6(14) <= "0000000000000000";
sprite_6(15) <= "0000000000000000";

sprite_7( 0) <= "0000000000000000";
sprite_7( 1) <= "0000000000000000";
sprite_7( 2) <= "0000000000000000";
sprite_7( 3) <= "0001111111111000";
sprite_7( 4) <= "0001111111111000";
sprite_7( 5) <= "0000000001110000";
sprite_7( 6) <= "0000000001110000";
sprite_7( 7) <= "0000000011100000";
sprite_7( 8) <= "0000000011100000";
sprite_7( 9) <= "0000000111000000";
sprite_7(10) <= "0000000111000000";
sprite_7(11) <= "0000001110000000";
sprite_7(12) <= "0000001110000000";
sprite_7(13) <= "0000000000000000";
sprite_7(14) <= "0000000000000000";
sprite_7(15) <= "0000000000000000";

sprite_8( 0) <= "0000000000000000";
sprite_8( 1) <= "0000000000000000";
sprite_8( 2) <= "0000000000000000";
sprite_8( 3) <= "0000111111110000";
sprite_8( 4) <= "0001111111111000";
sprite_8( 5) <= "0001100000011000";
sprite_8( 6) <= "0001100000011000";
sprite_8( 7) <= "0000111111110000";
sprite_8( 8) <= "0000111111110000";
```

```
sprite_8( 9) <= "0001100000011000";
sprite_8(10) <= "0001100000011000";
sprite_8(11) <= "0001111111111000";
sprite_8(12) <= "0000111111110000";
sprite_8(13) <= "0000000000000000";
sprite_8(14) <= "0000000000000000";
sprite_8(15) <= "0000000000000000";

sprite_9( 0) <= "0000000000000000";
sprite_9( 1) <= "0000000000000000";
sprite_9( 2) <= "0000000000000000";
sprite_9( 3) <= "0000111111110000";
sprite_9( 4) <= "0001111111111000";
sprite_9( 5) <= "0001100000011000";
sprite_9( 6) <= "0001100000011000";
sprite_9( 7) <= "0001111111111000";
sprite_9( 8) <= "0000111111111000";
sprite_9( 9) <= "0000000000011000";
sprite_9(10) <= "0001100000011000";
sprite_9(11) <= "0001111111111000";
sprite_9(12) <= "0000111111110000";
sprite_9(13) <= "0000000000000000";
sprite_9(14) <= "0000000000000000";
sprite_9(15) <= "0000000000000000";
------------------------------------------------------------
   -- Horizontal and vertical counters

   HCounter : process (clk25vga)
   begin
      if rising_edge(clk25vga) then
         if reset = '0' then
            Hcount <= (others => '0');
         elsif EndOfLine = '1' then
            Hcount <= (others => '0');
         else
            Hcount <= Hcount + 1;
         end if;
      end if;
```

```
end process HCounter;

EndOfLine <= '1' when Hcount = HTOTAL - 1 else '0';

VCounter: process (clk25vga)
begin
   if rising_edge(clk25vga) then
      if reset = '0' then
         Vcount <= (others => '0');
      elsif EndOfLine = '1' then
         if EndOfField = '1' then
            Vcount <= (others => '0');
         else
            Vcount <= Vcount + 1;
         end if;
      end if;
   end if;
end process VCounter;

EndOfField <= '1' when Vcount = VTOTAL - 1 else '0';

-- State machines to generate HSYNC, VSYNC, HBLANK, and VBLANK

HSyncGen : process (clk25vga)
begin
   if rising_edge(clk25vga) then
      if reset = '0' or EndOfLine = '1' then
         vga_hsync <= '1';
      elsif Hcount = HSYNC - 1 then
         vga_hsync <= '0';
      end if;
   end if;
end process HSyncGen;

HBlankGen : process (clk25vga)
begin
   if rising_edge(clk25vga) then
      if reset = '0' then
```

```
        vga_hblank <= '1';
      elsif Hcount = HSYNC + HBACK_PORCH then
        vga_hblank <= '0';
      elsif Hcount = HSYNC + HBACK_PORCH + HACTIVE then
        vga_hblank <= '1';
      end if;
   end if;
end process HBlankGen;

VSyncGen : process (clk25vga)
begin
   if rising_edge(clk25vga) then
      if reset = '0' then
        vga_vsync <= '1';
      elsif EndOfLine ='1' then
        if EndOfField = '1' then
          vga_vsync <= '1';
        elsif Vcount = VSYNC - 1 then
          vga_vsync <= '0';
        end if;
      end if;
   end if;
end process VSyncGen;

VBlankGen : process (clk25vga)
begin
   if rising_edge(clk25vga) then
      if reset = '0' then
        vga_vblank <= '1';
      elsif EndOfLine = '1' then
        if Vcount = VSYNC + VBACK_PORCH - 1 then
          vga_vblank <= '0';
        elsif Vcount = VSYNC + VBACK_PORCH + VACTIVE - 1 then
          vga_vblank <= '1';
        end if;
      end if;
   end if;
end process VBlankGen;
```

```
-- Rectangle GeNeRaToRs
--// game start game end

--// SCORE BITS
   STARTHGen : process (clk)
   begin
      if rising_edge(clk) then
         if reset = '0' or Hcount = HSYNC + HBACK_PORCH + START_HSTART
then
            start_h <= '1';
         elsif Hcount = HSYNC + HBACK_PORCH + START_HEND then
            start_h <= '0';
         end if;
      end if;
   end process STARTHGen;

   STARTVGen : process (clk)
   begin
      if rising_edge(clk) then
         if reset = '0' then
            start_v <= '0';
         elsif EndOfLine = '1' then
            if Vcount = VSYNC + VBACK_PORCH - 1 + START_VSTART then
               start_v <= '1';
            elsif Vcount = VSYNC + VBACK_PORCH - 1 + START_VEND then
               start_v <= '0';
            end if;
         end if;
      end if;
   end process STARTVGen;
   start_rect <= start_h and start_v;

   OWLMOVEGen : process (clk)
   begin
      if rising_edge(clk) then
         if owldelay = 1000000 then
               if owlpos > 380 then
```

```
            owlvel <= '0';
        end if;

        if owlpos < 100 then
            owlvel <= '1';
        end if;

        if owlvel = '1' then
            owlpos <= owlpos + 1;
        else
           owlpos <= owlpos -1;
        end if;

        owldelay <= 0;
    else
        owldelay <= owldelay + 1;
    end if;
  end if;
end process OWLMOVEGen;

OWLHGen : process (clk)
begin
  if rising_edge(clk) then
    if reset = '0' or Hcount = HSYNC + HBACK_PORCH + owlpos then
      owl_h <= '1';
    elsif Hcount = HSYNC + HBACK_PORCH + owlpos + 32 then
      owl_h <= '0';
    end if;
  end if;
end process OWLHGen;

OWLVGen : process (clk)
begin
  if rising_edge(clk) then
    if reset = '0' then
      owl_v <= '0';
    elsif EndOfLine = '1' then
      if Vcount = VSYNC + VBACK_PORCH - 1 + OWL_VSTART then
```

```
            owl_v <= '1';
          elsif Vcount = VSYNC + VBACK_PORCH - 1 + OWL_VEND then
            owl_v <= '0';
          end if;
        end if;
      end if;
    end process OWLVGen;
    owl_rect <= owl_h and owl_v;


    ENDHGen : process (clk)
    begin
      if rising_edge(clk) then
        if reset = '0' or Hcount = HSYNC + HBACK_PORCH + END_HSTART then
          end_h <= '1';
        elsif Hcount = HSYNC + HBACK_PORCH + END_HEND then
          end_h <= '0';
        end if;
      end if;
    end process ENDHGen;

    ENDVGen : process (clk)
    begin
      if rising_edge(clk) then
        if reset = '0' then
          end_v <= '0';
        elsif EndOfLine = '1' then
          if Vcount = VSYNC + VBACK_PORCH - 1 + END_VSTART then
            end_v <= '1';
          elsif Vcount = VSYNC + VBACK_PORCH - 1 + END_VEND then
            end_v <= '0';
          end if;
        end if;
      end if;
    end process ENDVGen;
    end_rect <= end_h and end_v;

--// MINIMAP
```

```
  SideBarHGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset = '0' or Hcount = HSYNC + HBACK_PORCH + SIDEBAR_HSTART
then
        sidebar_h <= '1';
      elsif Hcount = HSYNC + HBACK_PORCH + SIDEBAR_HEND then
        sidebar_h <= '0';
      end if;
    end if;
  end process SideBarHGen;

  SideBarVGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset = '0' then
        sidebar_v <= '0';
      elsif EndOfLine = '1' then
        if Vcount = VSYNC + VBACK_PORCH - 1 + SIDEBAR_VSTART then
          sidebar_v <= '1';
        elsif Vcount = VSYNC + VBACK_PORCH - 1 + SIDEBAR_VEND then
          sidebar_v <= '0';
        end if;
      end if;
    end if;
  end process SideBarVGen;

  sidebar_rect <= sidebar_h and sidebar_v;

  MiniMapHGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset = '0' or Hcount = HSYNC + HBACK_PORCH + MINIMAP_HSTART
then
        minimap_h <= '1';
      elsif Hcount = HSYNC + HBACK_PORCH + MINIMAP_HEND then
        minimap_h <= '0';
      end if;
```

```
    end if;
  end process MiniMapHGen;

  MiniMapVGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset = '0' then
        minimap_v <= '0';
      elsif EndOfLine = '1' then
        if Vcount = VSYNC + VBACK_PORCH - 1 + MINIMAP_VSTART then
          minimap_v <= '1';
        elsif Vcount = VSYNC + VBACK_PORCH - 1 + MINIMAP_VEND then
          minimap_v <= '0';
        end if;
      end if;
    end if;
  end process MiniMapVGen;

  minimap_rect <= minimap_h and minimap_v;
--// Word "SCORE"
  WordScoreHGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset = '0' or Hcount = HSYNC + HBACK_PORCH +
WORDSCOREBITS_HSTART then
        wordscorebits_h <= '1';
      elsif Hcount = HSYNC + HBACK_PORCH + WORDSCOREBITS_HEND
then
        wordscorebits_h <= '0';
      end if;
    end if;
  end process WordScoreHGen;

  WordScoreVGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset = '0' then
        wordscorebits_v <= '0';
```

```
        elsif EndOfLine = '1' then
          if Vcount = VSYNC + VBACK_PORCH - 1 + WORDSCOREBITS_VSTART
then
            wordscorebits_v <= '1';
          elsif Vcount = VSYNC + VBACK_PORCH - 1 +
WORDSCOREBITS_VEND then
            wordscorebits_v <= '0';
          end if;
        end if;
      end if;
  end process WordScoreVGen;

  wordscorebits_rect <= wordscorebits_h and wordscorebits_v;

--// SCORE BITS
  ScoreBitsHGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset = '0' or Hcount = HSYNC + HBACK_PORCH +
SCOREBITS_HSTART then
        scorebits_h <= '1';
      elsif Hcount = HSYNC + HBACK_PORCH + SCOREBITS_HEND then
        scorebits_h <= '0';
      end if;
    end if;
  end process ScoreBitsHGen;

  ScoreBitsVGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset = '0' then
        scorebits_v <= '0';
      elsif EndOfLine = '1' then
        if Vcount = VSYNC + VBACK_PORCH - 1 + SCOREBITS_VSTART then
          scorebits_v <= '1';
        elsif Vcount = VSYNC + VBACK_PORCH - 1 + SCOREBITS_VEND then
          scorebits_v <= '0';
        end if;
```

```
        end if;
      end if;
   end process ScoreBitsVGen;

   scorebits_rect <= scorebits_h and scorebits_v;

--// Word "FUEL"
   WordFuelHGen : process (clk)
   begin
      if rising_edge(clk) then
         if reset = '0' or Hcount = HSYNC + HBACK_PORCH +
WORDFUEL_HSTART then
            wordfuel_h <= '1';
         elsif Hcount = HSYNC + HBACK_PORCH + WORDFUEL_HEND then
            wordfuel_h <= '0';
         end if;
      end if;
   end process WordFuelHGen;

   WordFuelVGen : process (clk)
   begin
      if rising_edge(clk) then
         if reset = '0' then
            wordfuel_v <= '0';
         elsif EndOfLine = '1' then
            if Vcount = VSYNC + VBACK_PORCH - 1 + WORDFUEL_VSTART then
               wordfuel_v <= '1';
            elsif Vcount = VSYNC + VBACK_PORCH - 1 + WORDFUEL_VEND then
               wordfuel_v <= '0';
            end if;
         end if;
      end if;
   end process WordFuelVGen;

   wordfuel_rect <= wordfuel_h and wordfuel_v;

--// FUEL GAUGE
   FuelHGen : process (clk)
```

```
    begin
      if rising_edge(clk) then
        if reset = '0' or Hcount = HSYNC + HBACK_PORCH + FUEL_HSTART then
          fuel_h <= '1';
        elsif Hcount = HSYNC + HBACK_PORCH + FUEL_HEND then
          fuel_h <= '0';
        end if;
      end if;
    end process FuelHGen;

    FuelVGen : process (clk)
    begin
      if rising_edge(clk) then
        if reset = '0' then
          fuel_v <= '0';
        elsif EndOfLine = '1' then
          if Vcount = VSYNC + VBACK_PORCH - 1 + FUEL_VSTART then
            fuel_v <= '1';
          elsif Vcount = VSYNC + VBACK_PORCH - 1 + FUEL_VEND then
            fuel_v <= '0';
          end if;
        end if;
      end if;
    end process FuelVGen;

    fuel_rect <= fuel_h and fuel_v;

  --// Word "LIVES"
    WordLivesHGen : process (clk)
    begin
      if rising_edge(clk) then
        if reset = '0' or Hcount = HSYNC + HBACK_PORCH +
  WORDLIVES_HSTART then
          wordlives_h <= '1';
        elsif Hcount = HSYNC + HBACK_PORCH + WORDLIVES_HEND then
          wordlives_h <= '0';
        end if;
      end if;
```

```
    end process WordLivesHGen;

    WordLivesVGen : process (clk)
    begin
      if rising_edge(clk) then
        if reset = '0' then
          wordlives_v <= '0';
        elsif EndOfLine = '1' then
          if Vcount = VSYNC + VBACK_PORCH - 1 + WORDLIVES_VSTART then
            wordlives_v <= '1';
          elsif Vcount = VSYNC + VBACK_PORCH - 1 + WORDLIVES_VEND then
            wordlives_v <= '0';
          end if;
        end if;
      end if;
    end process WordLivesVGen;
    wordlives_rect <= wordlives_h and wordlives_v;

--// Number of LIVES
    NumLivesHGen : process (clk)
    begin
      if rising_edge(clk) then
        if reset = '0' or Hcount = HSYNC + HBACK_PORCH + NUMLIVES_HSTART
then
          numlives_h <= '1';
        elsif Hcount = HSYNC + HBACK_PORCH + NUMLIVES_HEND then
          numlives_h <= '0';
        end if;
      end if;
    end process NumLivesHGen;

    NumLivesVGen : process (clk)
    begin
      if rising_edge(clk) then
        if reset = '0' then
          numlives_v <= '0';
        elsif EndOfLine = '1' then
          if Vcount = VSYNC + VBACK_PORCH - 1 + NUMLIVES_VSTART then
```

```
            numlives_v <= '1';
          elsif Vcount = VSYNC + VBACK_PORCH - 1 + NUMLIVES_VEND then
            numlives_v <= '0';
          end if;
        end if;
      end if;
    end process NumLivesVGen;

    numlives_rect <= numlives_h and numlives_v;
--// PLAYER
--UD---
    UD_PlayerHGen : process (clk)
    begin
      if rising_edge(clk) then
        if reset = '0' or Hcount = HSYNC + HBACK_PORCH +
PLAYER_UD_HSTART then
          player_ud_h <= '1';
        elsif Hcount = HSYNC + HBACK_PORCH + PLAYER_UD_HEND then
          player_ud_h <= '0';
        end if;
      end if;
    end process UD_PlayerHGen;

    UD_PlayerVGen : process (clk)
    begin
      if rising_edge(clk) then
        if reset = '0' then
          player_ud_v <= '0';
        elsif EndOfLine = '1' then
          if Vcount = VSYNC + VBACK_PORCH - 1 + PLAYER_UD_VSTART then
            player_ud_v <= '1';
          elsif Vcount = VSYNC + VBACK_PORCH - 1 + PLAYER_UD_VEND then
            player_ud_v <= '0';
          end if;
        end if;
      end if;
    end process UD_PlayerVGen;
```

```
   player_ud_rect <= player_ud_h and player_ud_v;
--LR--
LR_PlayerHGen : process (clk)
   begin
      if rising_edge(clk) then
         if reset = '0' or Hcount = HSYNC + HBACK_PORCH +
PLAYER_LR_HSTART then
            player_lr_h <= '1';
         elsif Hcount = HSYNC + HBACK_PORCH + PLAYER_LR_HEND then
            player_lr_h <= '0';
         end if;
      end if;
   end process LR_PlayerHGen;

   LR_PlayerVGen : process (clk)
   begin
      if rising_edge(clk) then
         if reset = '0' then
            player_lr_v <= '0';
         elsif EndOfLine = '1' then
            if Vcount = VSYNC + VBACK_PORCH - 1 + PLAYER_LR_VSTART then
               player_lr_v <= '1';
            elsif Vcount = VSYNC + VBACK_PORCH - 1 + PLAYER_LR_VEND then
               player_lr_v <= '0';
            end if;
         end if;
      end if;
   end process LR_PlayerVGen;

   player_lr_rect <= player_lr_h and player_lr_v;
--// MAIN MAP
   MainMapHGen : process (clk)
   begin
      if rising_edge(clk) then
         if reset = '0' or Hcount = HSYNC + HBACK_PORCH + MAINMAP_HSTART
then
            mainmap_h <= '1';
         elsif Hcount = HSYNC + HBACK_PORCH + MAINMAP_HEND then
```

```
            mainmap_h <= '0';
          end if;
        end if;
    end process MainMapHGen;

    MainMapVGen : process (clk)
    begin
      if rising_edge(clk) then
        if reset = '0' then
          mainmap_v <= '0';
        elsif EndOfLine = '1' then
          if Vcount = VSYNC + VBACK_PORCH - 1 + MAINMAP_VSTART then
            mainmap_v <= '1';
          elsif Vcount = VSYNC + VBACK_PORCH - 1 + MAINMAP_VEND then
            mainmap_v <= '0';
          end if;
        end if;
      end if;
    end process MainMapVGen;

    mainmap_rect <= mainmap_h and mainmap_v;

    Flag0HGen : process (clk)
    begin
      if rising_edge(clk) then
        if reset = '0' or (Hcount < HTOTAL - 1 and Hcount = HSYNC +
HBACK_PORCH + (to_integer(flag_0(11 downto 0) sll 5) - to_integer(inx))) then
          flag0_h <= '1';
        elsif Hcount = HTOTAL - 1 or Hcount = HSYNC + HBACK_PORCH +
(to_integer(flag_0(11 downto 0) sll 5)+32 - to_integer(inx)) then
          flag0_h <= '0';
        end if;
      end if;
    end process Flag0HGen;

    Flag0VGen : process (clk)
    begin
      if rising_edge(clk) then
```

```
        if reset = '0' then
          flag0_v <= '0';
        elsif EndOfLine = '1' then
          if Vcount < VTOTAL - 1 and Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(flag_0(23 downto 12) sll 5) - 1 - to_integer(iny)) then
            flag0_v <= '1';
          elsif Vcount = VTOTAL - 1 or Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(flag_0(23 downto 12) sll 5)+32 - 1 - to_integer(iny)) then
            flag0_v <= '0';
          end if;
        end if;
      end if;
    end process Flag0VGen;
    flag0_rect <= flag0_h and flag0_v;

    Flag1HGen : process (clk)
    begin
      if rising_edge(clk) then
        if reset = '0' or (Hcount < HTOTAL - 1 and Hcount = HSYNC +
HBACK_PORCH + (to_integer(flag_1(11 downto 0) sll 5) - to_integer(inx))) then
          flag1_h <= '1';
        elsif Hcount = HTOTAL - 1 or Hcount = HSYNC + HBACK_PORCH +
(to_integer(flag_1(11 downto 0) sll 5)+32 - to_integer(inx)) then
          flag1_h <= '0';
        end if;
      end if;
    end process Flag1HGen;

    Flag1VGen : process (clk)
    begin
      if rising_edge(clk) then
        if reset = '0' then
          flag1_v <= '0';
        elsif EndOfLine = '1' then
          if Vcount < VTOTAL - 1 and Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(flag_1(23 downto 12) sll 5) - 1 - to_integer(iny)) then
            flag1_v <= '1';
          elsif Vcount = VTOTAL - 1 or Vcount = VSYNC + VBACK_PORCH - 1 +
```

```
(to_integer(flag_1(23 downto 12) sll 5)+32 - 1 - to_integer(iny)) then
                flag1_v <= '0';
            end if;
        end if;
    end if;
  end process Flag1VGen;
  flag1_rect <= flag1_h and flag1_v;


  Flag2HGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset = '0' or (Hcount < HTOTAL - 1 and Hcount = HSYNC +
HBACK_PORCH + (to_integer(flag_2(11 downto 0) sll 5) - to_integer(inx))) then
        flag2_h <= '1';
      elsif Hcount = HTOTAL - 1 or Hcount = HSYNC + HBACK_PORCH +
(to_integer(flag_2(11 downto 0) sll 5)+32 - to_integer(inx)) then
        flag2_h <= '0';
      end if;
    end if;
  end process Flag2HGen;

  Flag2VGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset = '0' then
        flag2_v <= '0';
      elsif EndOfLine = '1' then
        if Vcount < VTOTAL - 1 and Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(flag_2(23 downto 12) sll 5) - 1 - to_integer(iny)) then
          flag2_v <= '1';
        elsif Vcount = VTOTAL - 1 or Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(flag_2(23 downto 12) sll 5)-1+32 - to_integer(iny)) then
          flag2_v <= '0';
        end if;
      end if;
    end if;
  end process Flag2VGen;
```

```
  flag2_rect <= flag2_h and flag2_v;

  Flag3HGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset = '0' or (Hcount < HTOTAL - 1 and Hcount = HSYNC +
HBACK_PORCH + (to_integer(flag_3(11 downto 0) sll 5) - to_integer(inx))) then
        flag3_h <= '1';
      elsif Hcount = HTOTAL - 1 or Hcount = HSYNC + HBACK_PORCH +
(to_integer(flag_3(11 downto 0) sll 5)+32 - to_integer(inx)) then
        flag3_h <= '0';
      end if;
    end if;
  end process Flag3HGen;

  Flag3VGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset = '0' then
        flag3_v <= '0';
      elsif EndOfLine = '1' then
        if Vcount < VTOTAL - 1 and Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(flag_3(23 downto 12) sll 5) - 1 - to_integer(iny)) then
          flag3_v <= '1';
        elsif Vcount = VTOTAL - 1 or Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(flag_3(23 downto 12) sll 5)+32 - 1 - to_integer(iny)) then
          flag3_v <= '0';
        end if;
      end if;
    end if;
  end process Flag3VGen;
 flag3_rect <= flag3_h and flag3_v;

  Flag4HGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset = '0' or (Hcount < HTOTAL - 1 and Hcount = HSYNC +
HBACK_PORCH + (to_integer(flag_4(11 downto 0) sll 5) - to_integer(inx))) then
```

```
            flag4_h <= '1';
          elsif Hcount = HTOTAL - 1 or Hcount = HSYNC + HBACK_PORCH +
(to_integer(flag_4(11 downto 0) sll 5)+32 - to_integer(inx)) then
            flag4_h <= '0';
          end if;
        end if;
    end process Flag4HGen;

    Flag4VGen : process (clk)
    begin
      if rising_edge(clk) then
        if reset = '0' then
          flag4_v <= '0';
        elsif EndOfLine = '1' then
          if Vcount < VTOTAL - 1 and Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(flag_4(23 downto 12) sll 5) - 1 - to_integer(iny)) then
            flag4_v <= '1';
          elsif Vcount = VTOTAL - 1 or Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(flag_4(23 downto 12) sll 5)+32 - 1 - to_integer(iny)) then
            flag4_v <= '0';
          end if;
        end if;
      end if;
    end process Flag4VGen;
  flag4_rect <= flag4_h and flag4_v;

    Flag5HGen : process (clk)
    begin
      if rising_edge(clk) then
        if reset = '0' or (Hcount < HTOTAL - 1 and Hcount = HSYNC +
HBACK_PORCH + (to_integer(flag_5(11 downto 0) sll 5) - to_integer(inx))) then
          flag5_h <= '1';
        elsif Hcount = HTOTAL - 1 or Hcount = HSYNC + HBACK_PORCH +
(to_integer(flag_5(11 downto 0) sll 5)+32 - to_integer(inx)) then
          flag5_h <= '0';
        end if;
      end if;
    end process Flag5HGen;
```

```
  Flag5VGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset = '0' then
        flag5_v <= '0';
      elsif EndOfLine = '1' then
        if Vcount < VTOTAL - 1 and Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(flag_5(23 downto 12) sll 5) - 1 - to_integer(iny)) then
          flag5_v <= '1';
        elsif Vcount = VTOTAL - 1 or Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(flag_5(23 downto 12) sll 5)+32 - 1 - to_integer(iny)) then
          flag5_v <= '0';
        end if;
      end if;
    end if;
  end process Flag5VGen;
 flag5_rect <= flag5_h and flag5_v;

  Flag6HGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset = '0' or (Hcount < HTOTAL - 1 and Hcount = HSYNC +
HBACK_PORCH + (to_integer(flag_6(11 downto 0) sll 5) - to_integer(inx))) then
        flag6_h <= '1';
      elsif Hcount = HTOTAL - 1 or Hcount = HSYNC + HBACK_PORCH +
(to_integer(flag_6(11 downto 0) sll 5)+32 - to_integer(inx)) then
        flag6_h <= '0';
      end if;
    end if;
  end process Flag6HGen;

  Flag6VGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset = '0' then
        flag6_v <= '0';
      elsif EndOfLine = '1' then
```

```
            if Vcount < VTOTAL - 1 and Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(flag_6(23 downto 12) sll 5) - 1 - to_integer(iny)) then
               flag6_v <= '1';
            elsif Vcount = VTOTAL - 1 or Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(flag_6(23 downto 12) sll 5)+32 - 1 - to_integer(iny)) then
               flag6_v <= '0';
            end if;
         end if;
      end if;
   end process Flag6VGen;
  flag6_rect <= flag6_h and flag6_v;

   Flag7HGen : process (clk)
   begin
      if rising_edge(clk) then
         if reset = '0' or (Hcount < HTOTAL - 1 and Hcount = HSYNC +
HBACK_PORCH + (to_integer(flag_7(11 downto 0) sll 5) - to_integer(inx))) then
            flag7_h <= '1';
         elsif Hcount = HTOTAL - 1 or Hcount = HSYNC + HBACK_PORCH +
(to_integer(flag_7(11 downto 0) sll 5)+32 - to_integer(inx)) then
            flag7_h <= '0';
         end if;
      end if;
   end process Flag7HGen;

   Flag7VGen : process (clk)
   begin
      if rising_edge(clk) then
         if reset = '0' then
            flag7_v <= '0';
         elsif EndOfLine = '1' then
            if Vcount < VTOTAL - 1 and Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(flag_7(23 downto 12) sll 5) - 1 - to_integer(iny)) then
               flag7_v <= '1';
            elsif Vcount = VTOTAL - 1 or Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(flag_7(23 downto 12) sll 5)+32 - 1 - to_integer(iny)) then
               flag7_v <= '0';
            end if;
```

```
        end if;
      end if;
    end process Flag7VGen;
  flag7_rect <= flag7_h and flag7_v;

    Flag8HGen : process (clk)
    begin
      if rising_edge(clk) then
        if reset = '0' or (Hcount < HTOTAL - 1 and Hcount = HSYNC +
HBACK_PORCH + (to_integer(flag_8(11 downto 0) sll 5) - to_integer(inx))) then
          flag8_h <= '1';
        elsif Hcount = HTOTAL - 1 or Hcount = HSYNC + HBACK_PORCH +
(to_integer(flag_8(11 downto 0) sll 5)+32 - to_integer(inx)) then
          flag8_h <= '0';
        end if;
      end if;
    end process Flag8HGen;

    Flag8VGen : process (clk)
    begin
      if rising_edge(clk) then
        if reset = '0' then
          flag8_v <= '0';
        elsif EndOfLine = '1' then
          if Vcount < VTOTAL - 1 and Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(flag_8(23 downto 12) sll 5) - 1 - to_integer(iny)) then
            flag8_v <= '1';
          elsif Vcount = VTOTAL - 1 or Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(flag_8(23 downto 12) sll 5)+32 - 1 - to_integer(iny)) then
            flag8_v <= '0';
          end if;
        end if;
      end if;
    end process Flag8VGen;
  flag8_rect <= flag8_h and flag8_v;

    Flag9HGen : process (clk)
    begin
```

```
      if rising_edge(clk) then
        if reset = '0' or (Hcount < HTOTAL - 1 and Hcount = HSYNC +
HBACK_PORCH + (to_integer(flag_9(11 downto 0) sll 5) - to_integer(inx))) then
          flag9_h <= '1';
        elsif Hcount = HTOTAL - 1 or Hcount = HSYNC + HBACK_PORCH +
(to_integer(flag_9(11 downto 0) sll 5)+32 - to_integer(inx)) then
          flag9_h <= '0';
        end if;
      end if;
  end process Flag9HGen;

  Flag9VGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset = '0' then
        flag9_v <= '0';
      elsif EndOfLine = '1' then
        if Vcount < VTOTAL - 1 and Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(flag_9(23 downto 12) sll 5) - 1 - to_integer(iny)) then
          flag9_v <= '1';
        elsif Vcount = VTOTAL - 1 or Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(flag_9(23 downto 12) sll 5)+32 - 1 - to_integer(iny)) then
          flag9_v <= '0';
        end if;
      end if;
    end if;
  end process Flag9VGen;
 flag9_rect <= flag9_h and flag9_v;

  Enemy0HGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset = '0' or (Hcount < HTOTAL - 1 and Hcount = HSYNC +
HBACK_PORCH - 1 +
         (to_integer(enemy_0(11 downto 0)) - 1 - to_integer(inx))) then
        enemy0_h <= '1';
      elsif Hcount = HTOTAL - 1 or Hcount = HSYNC + HBACK_PORCH - 1 +
(to_integer(enemy_0(11 downto 0))
```

```
                +32 - 1 - to_integer(inx)) then
            enemy0_h <= '0';
          end if;
        end if;
    end process Enemy0HGen;

    Enemy0VGen : process (clk)
    begin
      if rising_edge(clk) then
        if reset = '0' then
          enemy0_v <= '0';
        elsif EndOfLine = '1' then
          if Vcount < VTOTAL - 1 and Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(enemy_0(23 downto 12))
              - 1 - to_integer(iny)) then
            enemy0_v <= '1';
          elsif Vcount = VTOTAL - 1 or Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(enemy_0(23 downto 12))
                  +32 - 1 - to_integer(iny)) then
            enemy0_v <= '0';
          end if;
        end if;
      end if;
    end process Enemy0VGen;
  enemy0_rect <= enemy0_h and enemy0_v;

    Enemy1HGen : process (clk)
    begin
      if rising_edge(clk) then
        if reset = '0' or (Hcount < HTOTAL - 1 and Hcount = HSYNC +
HBACK_PORCH - 1 +
            (to_integer(enemy_1(11 downto 0)) - 1 - to_integer(inx))) then
          enemy1_h <= '1';
        elsif Hcount = HTOTAL - 1 or Hcount = HSYNC + HBACK_PORCH - 1 +
(to_integer(enemy_1(11 downto 0))
                +32 - 1 - to_integer(inx)) then
          enemy1_h <= '0';
        end if;
```

```
    end if;
  end process Enemy1HGen;

  Enemy1VGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset = '0' then
        enemy1_v <= '0';
      elsif EndOfLine = '1' then
        if Vcount < VTOTAL - 1 and Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(enemy_1(23 downto 12))
            - 1 - to_integer(iny)) then
          enemy1_v <= '1';
        elsif Vcount = VTOTAL - 1 or Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(enemy_1(23 downto 12))
              +32 - 1 - to_integer(iny)) then
          enemy1_v <= '0';
        end if;
      end if;
    end if;
  end process Enemy1VGen;
 enemy1_rect <= enemy1_h and enemy1_v;

  Enemy2HGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset = '0' or (Hcount < HTOTAL - 1 and Hcount = HSYNC +
HBACK_PORCH - 1 +
          (to_integer(enemy_2(11 downto 0)) - 1 - to_integer(inx))) then
        enemy2_h <= '1';
      elsif Hcount = HTOTAL - 1 or Hcount = HSYNC + HBACK_PORCH - 1 +
(to_integer(enemy_2(11 downto 0))
            +32 - 1 - to_integer(inx)) then
        enemy2_h <= '0';
      end if;
    end if;
  end process Enemy2HGen;
```

```
   Enemy2VGen : process (clk)
   begin
      if rising_edge(clk) then
         if reset = '0' then
            enemy2_v <= '0';
         elsif EndOfLine = '1' then
            if Vcount < VTOTAL - 1 and Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(enemy_2(23 downto 12))
                - 1 - to_integer(iny)) then
               enemy2_v <= '1';
            elsif Vcount = VTOTAL - 1 or Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(enemy_2(23 downto 12))
                     +32 - 1 - to_integer(iny)) then
               enemy2_v <= '0';
            end if;
         end if;
      end if;
   end process Enemy2VGen;
  enemy2_rect <= enemy2_h and enemy2_v;

   Smoke0HGen : process (clk)
   begin
      if rising_edge(clk) then
         if reset = '0' or (Hcount < HTOTAL - 1 and Hcount = HSYNC +
HBACK_PORCH + (to_integer(smoke_0(11 downto 0) sll 5) - to_integer(inx)))
then
            smoke0_h <= '1';
         elsif Hcount = HTOTAL - 1 or Hcount = HSYNC + HBACK_PORCH +
(to_integer(smoke_0(11 downto 0) sll 5)+32 - to_integer(inx)) then
            smoke0_h <= '0';
         end if;
      end if;
   end process Smoke0HGen;

   Smoke0VGen : process (clk)
   begin
      if rising_edge(clk) then
         if reset = '0' then
```

```
            smoke0_v <= '0';
          elsif EndOfLine = '1' then
            if Vcount < VTOTAL - 1 and Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(smoke_0(23 downto 12) sll 5) - 1 - to_integer(iny)) then
              smoke0_v <= '1';
            elsif Vcount = VTOTAL - 1 or Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(smoke_0(23 downto 12) sll 5)+32 - 1 - to_integer(iny)) then
              smoke0_v <= '0';
            end if;
          end if;
        end if;
    end process Smoke0VGen;
   smoke0_rect <= smoke0_h and smoke0_v;

    Smoke1HGen : process (clk)
    begin
      if rising_edge(clk) then
        if reset = '0' or (Hcount < HTOTAL - 1 and Hcount = HSYNC +
HBACK_PORCH + (to_integer(smoke_1(11 downto 0) sll 5) - to_integer(inx)))
then
            smoke1_h <= '1';
          elsif Hcount = HTOTAL - 1 or Hcount = HSYNC + HBACK_PORCH +
(to_integer(smoke_1(11 downto 0) sll 5)+32 - to_integer(inx)) then
            smoke1_h <= '0';
          end if;
        end if;
    end process Smoke1HGen;

    Smoke1VGen : process (clk)
    begin
      if rising_edge(clk) then
        if reset = '0' then
          smoke1_v <= '0';
        elsif EndOfLine = '1' then
          if Vcount < VTOTAL - 1 and Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(smoke_1(23 downto 12) sll 5) - 1 - to_integer(iny)) then
            smoke1_v <= '1';
          elsif Vcount = VTOTAL - 1 or Vcount = VSYNC + VBACK_PORCH - 1 +
```

```
(to_integer(smoke_1(23 downto 12) sll 5)+32 - 1 - to_integer(iny)) then
              smoke1_v <= '0';
            end if;
          end if;
        end if;
    end process Smoke1VGen;
   smoke1_rect <= smoke1_h and smoke1_v;

    Smoke2HGen : process (clk)
    begin
      if rising_edge(clk) then
        if reset = '0' or (Hcount < HTOTAL - 1 and Hcount = HSYNC +
HBACK_PORCH + (to_integer(smoke_2(11 downto 0) sll 5) - to_integer(inx)))
then
          smoke2_h <= '1';
        elsif Hcount = HTOTAL - 1 or Hcount = HSYNC + HBACK_PORCH +
(to_integer(smoke_2(11 downto 0) sll 5)+32 - to_integer(inx)) then
          smoke2_h <= '0';
        end if;
      end if;
    end process Smoke2HGen;

    Smoke2VGen : process (clk)
    begin
      if rising_edge(clk) then
        if reset = '0' then
          smoke2_v <= '0';
        elsif EndOfLine = '1' then
          if Vcount < VTOTAL - 1 and Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(smoke_2(23 downto 12) sll 5) - 1 - to_integer(iny)) then
            smoke2_v <= '1';
          elsif Vcount = VTOTAL - 1 or Vcount = VSYNC + VBACK_PORCH - 1 +
(to_integer(smoke_2(23 downto 12) sll 5)+32 - 1 - to_integer(iny)) then
            smoke2_v <= '0';
          end if;
        end if;
      end if;
    end process Smoke2VGen;
```

```
 smoke2_rect <= smoke2_h and smoke2_v;

-- TESTING
WriteGen : process (clk)
begin
if rising_edge(clk) then

--position
	if (chipselect = '1' and write = '1' and address = "00000") then
		temp_inx <= writedata(15 downto 0);
		temp_iny <= writedata(31 downto 16);
	end if;
--COMBINE SPRITE STATUS AND FUEL AMOUNT LATER??

--sprite status
	if (chipselect = '1' and write = '1' and address = "00001") then
		player_status <= writedata(2 downto 0);
	end if;

--	--fuel amount
	if (chipselect = '1' and write = '1' and address = "00010") then
		fuel_status <= writedata(31 downto 0);
	end if;
--
--	--score data
	if (chipselect = '1' and write = '1' and address = "00011") then
		score_ones <= writedata(3 downto 0);
		score_tens <= writedata(7 downto 4);
		score_hundy <= writedata(11 downto 8);
	end if;

--write or delete a flag
	if (chipselect = '1' and write = '1' and address = "00100") then
		if (writedata(28 downto 25) = "0000") then
			flag_0 <= writedata(31 downto 0);
		--flag 1
		elsif (writedata(28 downto 25) = "0001") then
			flag_1 <= writedata(31 downto 0);
```

```
        --flag 2
        elsif (writedata(28 downto 25) = "0010") then
            flag_2 <= writedata(31 downto 0);
        --flag 3
        elsif (writedata(28 downto 25) = "0011") then
            flag_3 <= writedata(31 downto 0);
        --flag 4
        elsif (writedata(28 downto 25) = "0100") then
            flag_4 <= writedata(31 downto 0);
        --flag 5
        elsif (writedata(28 downto 25) = "0101") then
            flag_5 <= writedata(31 downto 0);
        --flag 6
        elsif (writedata(28 downto 25) = "0110") then
            flag_6 <= writedata(31 downto 0);
        --flag 7
        elsif (writedata(28 downto 25) = "0111") then
            flag_7 <= writedata(31 downto 0);
        --flag 8
        elsif (writedata(28 downto 25) = "1000") then
            flag_8 <= writedata(31 downto 0);
        --flag 9
        elsif (writedata(28 downto 25) = "1001") then
            flag_9 <= writedata(31 downto 0);
        end if;
    end if;

    -- enemy 0
    if (chipselect = '1' and write = '1' and address = "00101") then
        temp_enemy_0 <= writedata(31 downto 0);
    end if;
    -- enemy 1
    if (chipselect = '1' and write = '1' and address = "00110") then
        temp_enemy_1 <= writedata(31 downto 0);
    end if;
    -- enemy 2
    if (chipselect = '1' and write = '1' and address = "00111") then
        temp_enemy_2 <= writedata(31 downto 0);
```

```
        end if;

--smoke
        if (chipselect = '1' and write = '1' and address = "01000") then
            if (writedata(28 downto 25) = "0001") then          --smoke 0
                smoke_0 <= writedata(31 downto 0);
            elsif (writedata(28 downto 25) = "0010") then   --smoke 1
                smoke_1 <= writedata(31 downto 0);
            elsif (writedata(28 downto 25) = "0011") then   --smoke 2
                smoke_2 <= writedata(31 downto 0);
            end if;
        end if;

        if (chipselect = '1' and write = '1' and address = "01001") then
            num_lives <= writedata(31 downto 0);
        end if;

        if (chipselect = '1' and write = '1' and address = "01010") then
            temp_game_state <= writedata(31 downto 0);

        end if;
       end if;
    end process WriteGen;

-- TESTING END

PlayerCoordinateSync : process(clk)
begin
        if (rising_edge(clk) and vga_vsync = '1' and vga_hsync = '1') then
            inx <= temp_inx;
            iny <= temp_iny;
            enemy_0 <= temp_enemy_0;
            enemy_1 <= temp_enemy_1;
            enemy_2 <= temp_enemy_2;
            game_state <= temp_game_state;
        end if;
end process PlayerCoordinateSync;
```

```
-- Registered video signals going to the video DAC

VideoOut: process (clk25vga, reset)
   variable control_array_row : unsigned (187 downto 0);
   variable brick_lg_row   : unsigned (31 downto 0);
   variable brick_g_row    : unsigned (31 downto 0);
   variable brick_dg_row   : unsigned (31 downto 0);
   variable brick_z_row    : unsigned (31 downto 0);
   variable owl_r_row    : unsigned (31 downto 0);
   variable owl_g_row    : unsigned (31 downto 0);
   variable owl_w_row    : unsigned (31 downto 0);
   variable road_g_row     : unsigned (31 downto 0);
   variable road_b_row     : unsigned (31 downto 0);
   variable smoke_i_row     : unsigned (31 downto 0);
   variable smoke_g_row     : unsigned (31 downto 0);

   variable enemy1_up_y_row    : unsigned (31 downto 0);
   variable enemy1_up_o_row    : unsigned (31 downto 0);
   variable enemy1_up_r_row    : unsigned (31 downto 0);
   variable enemy1_up_b_row    : unsigned (31 downto 0);

   variable enemy1_down_y_row    : unsigned (31 downto 0);
   variable enemy1_down_o_row    : unsigned (31 downto 0);
   variable enemy1_down_r_row    : unsigned (31 downto 0);
   variable enemy1_down_b_row    : unsigned (31 downto 0);

   variable enemy1_left_y_row    : unsigned (31 downto 0);
   variable enemy1_left_o_row    : unsigned (31 downto 0);
   variable enemy1_left_r_row    : unsigned (31 downto 0);
   variable enemy1_left_b_row    : unsigned (31 downto 0);

   variable enemy1_right_y_row    : unsigned (31 downto 0);
   variable enemy1_right_o_row    : unsigned (31 downto 0);
   variable enemy1_right_r_row    : unsigned (31 downto 0);
   variable enemy1_right_b_row    : unsigned (31 downto 0);

   variable enemy2_up_y_row    : unsigned (31 downto 0);
   variable enemy2_up_o_row    : unsigned (31 downto 0);
```

```
variable enemy2_up_r_row   : unsigned (31 downto 0);
variable enemy2_up_b_row   : unsigned (31 downto 0);

variable enemy2_down_y_row   : unsigned (31 downto 0);
variable enemy2_down_o_row   : unsigned (31 downto 0);
variable enemy2_down_r_row   : unsigned (31 downto 0);
variable enemy2_down_b_row   : unsigned (31 downto 0);

variable enemy2_left_y_row   : unsigned (31 downto 0);
variable enemy2_left_o_row   : unsigned (31 downto 0);
variable enemy2_left_r_row   : unsigned (31 downto 0);
variable enemy2_left_b_row   : unsigned (31 downto 0);

variable enemy2_right_y_row   : unsigned (31 downto 0);
variable enemy2_right_o_row   : unsigned (31 downto 0);
variable enemy2_right_r_row   : unsigned (31 downto 0);
variable enemy2_right_b_row   : unsigned (31 downto 0);

variable enemy3_up_y_row   : unsigned (31 downto 0);
variable enemy3_up_o_row   : unsigned (31 downto 0);
variable enemy3_up_r_row   : unsigned (31 downto 0);
variable enemy3_up_b_row   : unsigned (31 downto 0);

variable enemy3_down_y_row   : unsigned (31 downto 0);
variable enemy3_down_o_row   : unsigned (31 downto 0);
variable enemy3_down_r_row   : unsigned (31 downto 0);
variable enemy3_down_b_row   : unsigned (31 downto 0);

variable enemy3_left_y_row   : unsigned (31 downto 0);
variable enemy3_left_o_row   : unsigned (31 downto 0);
variable enemy3_left_r_row   : unsigned (31 downto 0);
variable enemy3_left_b_row   : unsigned (31 downto 0);

variable enemy3_right_y_row   : unsigned (31 downto 0);
variable enemy3_right_o_row   : unsigned (31 downto 0);
variable enemy3_right_r_row   : unsigned (31 downto 0);
variable enemy3_right_b_row   : unsigned (31 downto 0);
```

```
--PLAYER CAR
variable up_car_w_row: unsigned (31 downto 0);
variable up_car_r_row: unsigned (31 downto 0);
variable up_car_lg_row: unsigned (31 downto 0);
variable up_car_b_row: unsigned (31 downto 0);
variable up_car_y_row: unsigned (31 downto 0);
variable up_car_g_row: unsigned (31 downto 0);

variable down_car_w_row: unsigned (31 downto 0);
variable down_car_r_row: unsigned (31 downto 0);
variable down_car_lg_row: unsigned (31 downto 0);
variable down_car_b_row: unsigned (31 downto 0);
variable down_car_y_row: unsigned (31 downto 0);
variable down_car_g_row: unsigned (31 downto 0);

variable left_car_w_row: unsigned (31 downto 0);
variable left_car_r_row: unsigned (31 downto 0);
variable left_car_lg_row: unsigned (31 downto 0);
variable left_car_b_row: unsigned (31 downto 0);
variable left_car_y_row: unsigned (31 downto 0);
variable left_car_g_row: unsigned (31 downto 0);

variable right_car_w_row: unsigned (31 downto 0);
variable right_car_r_row: unsigned (31 downto 0);
variable right_car_lg_row: unsigned (31 downto 0);
variable right_car_b_row: unsigned (31 downto 0);
variable right_car_y_row: unsigned (31 downto 0);
variable right_car_g_row: unsigned (31 downto 0);
--ENEMY CAR

--SCORE
variable WSB_sprite_S_row: unsigned (15 downto 0);
variable WSB_sprite_C_row: unsigned (15 downto 0);
variable WSB_sprite_O_row: unsigned (15 downto 0);
variable WSB_sprite_R_row: unsigned (15 downto 0);
variable WSB_sprite_E_row: unsigned (15 downto 0);

variable score_ones_row: unsigned (15 downto 0);
```

```
variable score_tens_row: unsigned (15 downto 0);
variable score_hundy_row: unsigned (15 downto 0);

--LIVES
variable WL_sprite_L_row: unsigned (15 downto 0);
variable WL_sprite_I_row: unsigned (15 downto 0);
variable WL_sprite_V_row: unsigned (15 downto 0);
variable WL_sprite_E_row: unsigned (15 downto 0);
variable WL_sprite_S_row: unsigned (15 downto 0);

--FUEL
variable WF_sprite_F_row: unsigned (15 downto 0);
variable WF_sprite_U_row: unsigned (15 downto 0);
variable WF_sprite_E_row: unsigned (15 downto 0);
variable WF_sprite_L_row: unsigned (15 downto 0);

--START AND END GAME
variable GE_sprite1_row: unsigned (15 downto 0);
variable GE_sprite2_row: unsigned (15 downto 0);
variable GE_sprite3_row: unsigned (15 downto 0);
variable GE_sprite4_row: unsigned (15 downto 0);
variable GE_sprite5_row: unsigned (15 downto 0);
variable GE_sprite6_row: unsigned (15 downto 0);
variable GE_sprite7_row: unsigned (15 downto 0);
variable GE_sprite8_row: unsigned (15 downto 0);
variable GS_sprite1_row: unsigned (15 downto 0);
variable GS_sprite2_row: unsigned (15 downto 0);
variable GS_sprite3_row: unsigned (15 downto 0);
variable GS_sprite4_row: unsigned (15 downto 0);
variable GS_sprite5_row: unsigned (15 downto 0);
variable GS_sprite6_row: unsigned (15 downto 0);
variable GS_OWL_w_row       : unsigned (31 downto 0);
variable GS_OWL_r_row       : unsigned (31 downto 0);
--GGGGG start game
--GG end game
variable GE_sprite_array_V16: unsigned (15 downto 0);
variable GE_sprite_array_H16: unsigned (15 downto 0);
```

```
variable GS_sprite_array_V16: unsigned (15 downto 0);
variable GS_sprite_array_H16: unsigned (15 downto 0);

variable GSO_sprite_array_V32: unsigned (15 downto 0);
variable GSO_sprite_array_H32: unsigned (15 downto 0);

variable GS_array_H : unsigned (15 downto 0);
variable GS_array_V : unsigned (15 downto 0);

variable GSO_array_H : unsigned (15 downto 0);
variable GSO_array_V : unsigned (15 downto 0);

variable GE_array_H : unsigned (15 downto 0);
variable GE_array_V : unsigned (15 downto 0);

--MAP AND PLAYER
variable control_array_V : unsigned (15 downto 0);
variable control_array_H : unsigned (15 downto 0);

variable sprite_array_V : unsigned (15 downto 0);
variable sprite_array_H : unsigned (15 downto 0);

variable sprite_array_V16: unsigned (15 downto 0);
variable sprite_array_H16: unsigned (15 downto 0);

variable car_array_V : unsigned (15 downto 0);
variable car_array_H : unsigned (15 downto 0);

variable car_count_V : unsigned (15 downto 0);
variable car_count_H : unsigned (15 downto 0);

variable player_radar_x : unsigned (15 downto 0);
variable player_radar_y : unsigned (15 downto 0);

variable wordscorebits_array_H : unsigned (15 downto 0);
variable wordscorebits_array_V : unsigned (15 downto 0);

variable scorebits_array_H : unsigned (15 downto 0);
```

```
   variable scorebits_array_V : unsigned (15 downto 0);

   variable wordlives_array_H : unsigned (15 downto 0);
   variable wordlives_array_V : unsigned (15 downto 0);

   variable wordfuel_array_H : unsigned (15 downto 0);
   variable wordfuel_array_V : unsigned (15 downto 0);

   variable WSB_sprite_array_V16 : unsigned (15 downto 0);
   variable WSB_sprite_array_H16 : unsigned (15 downto 0);

   variable SB_sprite_array_V16 : unsigned (15 downto 0);
   variable SB_sprite_array_H16 : unsigned (15 downto 0);

   variable WL_sprite_array_V16 : unsigned (15 downto 0);
   variable WL_sprite_array_H16 : unsigned (15 downto 0);

   variable NL_sprite_array_V16 : unsigned (15 downto 0);
   variable NL_sprite_array_H16 : unsigned (15 downto 0);

   variable WF_sprite_array_V16 : unsigned (15 downto 0);
   variable WF_sprite_array_H16 : unsigned (15 downto 0);

   variable sprite_array_Ve1 : unsigned (15 downto 0);
   variable sprite_array_Ve2 : unsigned (15 downto 0);
   variable sprite_array_Ve3 : unsigned (15 downto 0);

   variable sprite_array_He1 : unsigned (15 downto 0);
   variable sprite_array_He2 : unsigned (15 downto 0);
   variable sprite_array_He3 : unsigned (15 downto 0);

   variable lives_row: unsigned (15 downto 0);

   begin

control_array_V := (Vcount - (VSYNC + VBACK_PORCH) + to_integer(iny)+1) srl
5;
control_array_H := (Hcount - (HSYNC + HBACK_PORCH) + to_integer(inx)) srl 5;
```

```
sprite_array_V := (Vcount - (VSYNC + VBACK_PORCH) + to_integer(iny)+1)
AND "0000000000011111";
sprite_array_H := (Hcount - (HSYNC + HBACK_PORCH) + to_integer(inx)) AND
"0000000000011111";

--//test
WSB_sprite_array_V16 := (Vcount - (VSYNC + VBACK_PORCH) -
WORDSCOREBITS_VSTART) AND "0000000000001111";
WSB_sprite_array_H16 := (Hcount - (HSYNC + HBACK_PORCH) -
WORDSCOREBITS_HSTART) AND "0000000000001111";

SB_sprite_array_V16 := (Vcount - (VSYNC + VBACK_PORCH) -
SCOREBITS_VSTART) AND "0000000000001111";
SB_sprite_array_H16 := (Hcount - (HSYNC + HBACK_PORCH) -
SCOREBITS_HSTART) AND "0000000000001111";

WL_sprite_array_V16 := (Vcount - (VSYNC + VBACK_PORCH) -
WORDLIVES_VSTART) AND "0000000000001111";
WL_sprite_array_H16 := (Hcount - (HSYNC + HBACK_PORCH) -
WORDLIVES_HSTART) AND "0000000000001111";

NL_sprite_array_V16 := (Vcount - (VSYNC + VBACK_PORCH) -
NUMLIVES_VSTART) AND "0000000000001111";
NL_sprite_array_H16 := (Hcount - (HSYNC + HBACK_PORCH) -
NUMLIVES_HSTART) AND "0000000000001111";

WF_sprite_array_V16 := (Vcount - (VSYNC + VBACK_PORCH) -
WORDFUEL_VSTART) AND "0000000000001111";
WF_sprite_array_H16 := (Hcount - (HSYNC + HBACK_PORCH) -
WORDFUEL_HSTART) AND "0000000000001111";

GE_sprite_array_V16 := (Vcount - (VSYNC + VBACK_PORCH) - END_VSTART)
AND "0000000000001111";
GE_sprite_array_H16 := (Hcount - (HSYNC + HBACK_PORCH) - END_HSTART)
AND "0000000000001111";

GS_sprite_array_V16 := (Vcount - (VSYNC + VBACK_PORCH) - START_VSTART)
```

```
AND "0000000000001111";
GS_sprite_array_H16 := (Hcount - (HSYNC + HBACK_PORCH) -
START_HSTART) AND "0000000000001111";

GSO_sprite_array_V32 := (Vcount - (VSYNC + VBACK_PORCH) - OWL_VSTART )
AND "0000000000011111";
GSO_sprite_array_H32 := (Hcount - (HSYNC + HBACK_PORCH) - owlpos ) AND
"0000000000011111";




--//end test

sprite_array_V16 := (Vcount - (VSYNC + VBACK_PORCH)) AND
"0000000000001111";
sprite_array_H16 := (Hcount - (HSYNC + HBACK_PORCH)) AND
"0000000000001111";

car_array_V := (Vcount - (VSYNC + VBACK_PORCH)) AND
"0000000000011111";
car_array_H := (Hcount - (HSYNC + HBACK_PORCH)) AND
"0000000000011111";

car_count_V    := (Vcount - (VSYNC + VBACK_PORCH)) ;
car_count_H    := (Hcount - (HSYNC + HBACK_PORCH)) ;

sprite_array_Ve1 := (Vcount - (VSYNC + VBACK_PORCH) - enemy_0(23 downto
12)+ to_integer(iny)+1) AND "0000000000011111";
sprite_array_Ve2 := (Vcount - (VSYNC + VBACK_PORCH) - enemy_1(23 downto
12)+ to_integer(iny)+1) AND "0000000000011111";
sprite_array_Ve3 := (Vcount - (VSYNC + VBACK_PORCH) - enemy_2(23 downto
12)+ to_integer(iny)+1) AND "0000000000011111";

sprite_array_He1 := (Hcount - (HSYNC + HBACK_PORCH) - enemy_0(11 downto
0)+ to_integer(inx)) AND "0000000000011111";
sprite_array_He2 := (Hcount - (HSYNC + HBACK_PORCH) - enemy_1(11 downto
0)+ to_integer(inx)) AND "0000000000011111";
```

```
sprite_array_He3 := (Hcount - (HSYNC + HBACK_PORCH) - enemy_2(11 downto
0)+ to_integer(inx)) AND "0000000000011111";

wordscorebits_array_H := (Hcount - (HSYNC + HBACK_PORCH) -
WORDSCOREBITS_HSTART) srl 4;

scorebits_array_V := (Vcount - (VSYNC + VBACK_PORCH) -
SCOREBITS_VSTART) srl 4;
scorebits_array_H := (Hcount - (HSYNC + HBACK_PORCH) -
SCOREBITS_HSTART) srl 4;

wordlives_array_V := (Vcount - (VSYNC + VBACK_PORCH) -
WORDLIVES_VSTART) srl 4;
wordlives_array_H := (Hcount - (HSYNC + HBACK_PORCH) -
WORDLIVES_HSTART) srl 4;

wordfuel_array_V := (Vcount - (VSYNC + VBACK_PORCH) -
WORDFUEL_VSTART) srl 4;
wordfuel_array_H := (Hcount - (HSYNC + HBACK_PORCH) -
WORDFUEL_HSTART) srl 4;

GS_array_V := (Vcount - (VSYNC + VBACK_PORCH) - START_VSTART) srl 4;
GS_array_H := (Hcount - (HSYNC + HBACK_PORCH) - START_HSTART) srl 4;

GSO_array_V := (Vcount - (VSYNC + VBACK_PORCH) - OWL_VSTART + 1) srl 5;
GSO_array_H := (Hcount - (HSYNC + HBACK_PORCH) - owlpos) srl 5;

GE_array_V := (Vcount - (VSYNC + VBACK_PORCH) - END_VSTART) srl 4;
GE_array_H := (Hcount - (HSYNC + HBACK_PORCH) - END_HSTART) srl 4;

if reset = '0' then
VGA_R <= "0000000000";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
elsif clk25vga'event and clk25vga = '1' then
--// DAVID BEGIN //-----------------------------------------------------------------------

        up_car_w_row := car_w(to_integer(unsigned(car_array_V)));
```

```
            up_car_r_row := car_r(to_integer(unsigned(car_array_V)));
            up_car_lg_row := car_lg(to_integer(unsigned(car_array_V)));
            up_car_b_row := car_b(to_integer(unsigned(car_array_V)));
            up_car_y_row := car_y(to_integer(unsigned(car_array_V)));
            up_car_g_row := car_g(to_integer(unsigned(car_array_V)));

            down_car_w_row := car_w(to_integer(unsigned(32-car_array_V)));
            down_car_r_row := car_r(to_integer(unsigned(32-car_array_V)));
            down_car_lg_row := car_lg(to_integer(unsigned(32-car_array_V)));
            down_car_b_row := car_b(to_integer(unsigned(32-car_array_V)));
            down_car_y_row := car_y(to_integer(unsigned(32-car_array_V)));
            down_car_g_row := car_g(to_integer(unsigned(32-car_array_V)));

            left_car_w_row := car_w(to_integer(unsigned(car_array_H)));
            left_car_r_row := car_r(to_integer(unsigned(car_array_H)));
            left_car_lg_row := car_lg(to_integer(unsigned(car_array_H)));
            left_car_b_row := car_b(to_integer(unsigned(car_array_H)));
            left_car_y_row := car_y(to_integer(unsigned(car_array_H)));
            left_car_g_row := car_g(to_integer(unsigned(car_array_H)));

            right_car_w_row := car_w(to_integer(unsigned(32-car_array_H)));
            right_car_r_row := car_r(to_integer(unsigned(32-car_array_H)));
            right_car_lg_row := car_lg(to_integer(unsigned(32-car_array_H)));
            right_car_b_row := car_b(to_integer(unsigned(32-car_array_H)));
            right_car_y_row := car_y(to_integer(unsigned(32-car_array_H)));
            right_car_g_row := car_g(to_integer(unsigned(32-car_array_H)));


--          sprite_S_row := sprite_S(to_integer(unsigned(sprite_array_V16)));
--          sprite_C_row := sprite_C(to_integer(unsigned(sprite_array_V16)));
--          sprite_O_row := sprite_O(to_integer(unsigned(sprite_array_V16)));
--          sprite_R_row := sprite_R(to_integer(unsigned(sprite_array_V16)));
--          sprite_E_row := sprite_E(to_integer(unsigned(sprite_array_V16)));

            GS_sprite1_row :=
sprite_X(to_integer(unsigned(GS_sprite_array_V16)));
            GS_sprite2_row :=
sprite_Y(to_integer(unsigned(GS_sprite_array_V16)));
```

```
            GS_sprite3_row :=
sprite_L(to_integer(unsigned(GS_sprite_array_V16)));
            GS_sprite4_row :=
sprite_L(to_integer(unsigned(GS_sprite_array_V16)));
            GS_sprite5_row :=
sprite_A(to_integer(unsigned(GS_sprite_array_V16)));
            GS_sprite6_row :=
sprite_R(to_integer(unsigned(GS_sprite_array_V16)));
            GS_OWL_w_row     :=
owl_w(to_integer(unsigned(GSO_sprite_array_V32)));
            GS_OWL_r_row     :=
owl_r(to_integer(unsigned(GSO_sprite_array_V32)));

            GE_sprite1_row :=
sprite_R(to_integer(unsigned(GS_sprite_array_V16)));
            GE_sprite2_row :=
sprite_E(to_integer(unsigned(GS_sprite_array_V16)));
            GE_sprite3_row :=
sprite_V(to_integer(unsigned(GS_sprite_array_V16)));
            GE_sprite4_row :=
sprite_O(to_integer(unsigned(GS_sprite_array_V16)));
            GE_sprite5_row :=
sprite_E(to_integer(unsigned(GS_sprite_array_V16)));
            GE_sprite6_row :=
sprite_M(to_integer(unsigned(GS_sprite_array_V16)));
            GE_sprite7_row :=
sprite_A(to_integer(unsigned(GS_sprite_array_V16)));
            GE_sprite8_row :=
sprite_G(to_integer(unsigned(GS_sprite_array_V16)));

WSB_sprite_S_row := sprite_S(to_integer(unsigned(WSB_sprite_array_V16)));
WSB_sprite_C_row := sprite_C(to_integer(unsigned(WSB_sprite_array_V16)));
WSB_sprite_O_row := sprite_O(to_integer(unsigned(WSB_sprite_array_V16)));
WSB_sprite_R_row := sprite_R(to_integer(unsigned(WSB_sprite_array_V16)));
WSB_sprite_E_row := sprite_E(to_integer(unsigned(WSB_sprite_array_V16)));

WL_sprite_L_row := sprite_L(to_integer(unsigned(WL_sprite_array_V16)));
WL_sprite_I_row := sprite_I(to_integer(unsigned(WL_sprite_array_V16)));
```

```
WL_sprite_V_row := sprite_V(to_integer(unsigned(WL_sprite_array_V16)));
WL_sprite_E_row := sprite_E(to_integer(unsigned(WL_sprite_array_V16)));
WL_sprite_S_row := sprite_S(to_integer(unsigned(WL_sprite_array_V16)));

WF_sprite_F_row := sprite_F(to_integer(unsigned(WF_sprite_array_V16)));
WF_sprite_U_row := sprite_U(to_integer(unsigned(WF_sprite_array_V16)));
WF_sprite_E_row := sprite_E(to_integer(unsigned(WF_sprite_array_V16)));
WF_sprite_L_row := sprite_L(to_integer(unsigned(WF_sprite_array_V16)));

smoke_i_row := smoke_i(to_integer(unsigned(sprite_array_V)));
smoke_g_row := smoke_g(to_integer(unsigned(sprite_array_V)));

enemy1_up_y_row := enemy_up_y(to_integer(unsigned(sprite_array_Ve1)));
enemy1_up_o_row := enemy_up_o(to_integer(unsigned(sprite_array_Ve1)));
enemy1_up_r_row := enemy_up_r(to_integer(unsigned(sprite_array_Ve1)));
enemy1_up_b_row := enemy_up_b(to_integer(unsigned(sprite_array_Ve1)));

enemy1_down_y_row :=
enemy_down_y(to_integer(unsigned(sprite_array_Ve1)));
enemy1_down_o_row :=
enemy_down_o(to_integer(unsigned(sprite_array_Ve1)));
enemy1_down_r_row :=
enemy_down_r(to_integer(unsigned(sprite_array_Ve1)));
enemy1_down_b_row :=
enemy_down_b(to_integer(unsigned(sprite_array_Ve1)));

enemy1_left_y_row := enemy_left_y(to_integer(unsigned(sprite_array_Ve1)));
enemy1_left_o_row := enemy_left_o(to_integer(unsigned(sprite_array_Ve1)));
enemy1_left_r_row := enemy_left_r(to_integer(unsigned(sprite_array_Ve1)));
enemy1_left_b_row := enemy_left_b(to_integer(unsigned(sprite_array_Ve1)));

enemy1_right_y_row :=
enemy_right_y(to_integer(unsigned(sprite_array_Ve1)));
enemy1_right_o_row :=
enemy_right_o(to_integer(unsigned(sprite_array_Ve1)));
enemy1_right_r_row :=
enemy_right_r(to_integer(unsigned(sprite_array_Ve1)));
enemy1_right_b_row :=
```

```
enemy_right_b(to_integer(unsigned(sprite_array_Ve1)));

enemy2_up_y_row := enemy_up_y(to_integer(unsigned(sprite_array_Ve2)));
enemy2_up_o_row := enemy_up_o(to_integer(unsigned(sprite_array_Ve2)));
enemy2_up_r_row := enemy_up_r(to_integer(unsigned(sprite_array_Ve2)));
enemy2_up_b_row := enemy_up_b(to_integer(unsigned(sprite_array_Ve2)));

enemy2_down_y_row :=
enemy_down_y(to_integer(unsigned(sprite_array_Ve2)));
enemy2_down_o_row :=
enemy_down_o(to_integer(unsigned(sprite_array_Ve2)));
enemy2_down_r_row :=
enemy_down_r(to_integer(unsigned(sprite_array_Ve2)));
enemy2_down_b_row :=
enemy_down_b(to_integer(unsigned(sprite_array_Ve2)));

enemy2_left_y_row := enemy_left_y(to_integer(unsigned(sprite_array_Ve2)));
enemy2_left_o_row := enemy_left_o(to_integer(unsigned(sprite_array_Ve2)));
enemy2_left_r_row := enemy_left_r(to_integer(unsigned(sprite_array_Ve2)));
enemy2_left_b_row := enemy_left_b(to_integer(unsigned(sprite_array_Ve2)));

enemy2_right_y_row :=
enemy_right_y(to_integer(unsigned(sprite_array_Ve2)));
enemy2_right_o_row :=
enemy_right_o(to_integer(unsigned(sprite_array_Ve2)));
enemy2_right_r_row :=
enemy_right_r(to_integer(unsigned(sprite_array_Ve2)));
enemy2_right_b_row :=
enemy_right_b(to_integer(unsigned(sprite_array_Ve2)));


enemy3_up_y_row := enemy_up_y(to_integer(unsigned(sprite_array_Ve3)));
enemy3_up_o_row := enemy_up_o(to_integer(unsigned(sprite_array_Ve3)));
enemy3_up_r_row := enemy_up_r(to_integer(unsigned(sprite_array_Ve3)));
enemy3_up_b_row := enemy_up_b(to_integer(unsigned(sprite_array_Ve3)));

enemy3_down_y_row :=
enemy_down_y(to_integer(unsigned(sprite_array_Ve3)));
```

```
enemy3_down_o_row :=
enemy_down_o(to_integer(unsigned(sprite_array_Ve3)));
enemy3_down_r_row :=
enemy_down_r(to_integer(unsigned(sprite_array_Ve3)));
enemy3_down_b_row :=
enemy_down_b(to_integer(unsigned(sprite_array_Ve3)));

enemy3_left_y_row := enemy_left_y(to_integer(unsigned(sprite_array_Ve3)));
enemy3_left_o_row := enemy_left_o(to_integer(unsigned(sprite_array_Ve3)));
enemy3_left_r_row := enemy_left_r(to_integer(unsigned(sprite_array_Ve3)));
enemy3_left_b_row := enemy_left_b(to_integer(unsigned(sprite_array_Ve3)));

enemy3_right_y_row :=
enemy_right_y(to_integer(unsigned(sprite_array_Ve3)));
enemy3_right_o_row :=
enemy_right_o(to_integer(unsigned(sprite_array_Ve3)));
enemy3_right_r_row :=
enemy_right_r(to_integer(unsigned(sprite_array_Ve3)));
enemy3_right_b_row :=
enemy_right_b(to_integer(unsigned(sprite_array_Ve3)));

owl_w_row := owl_w(to_integer(unsigned(sprite_array_V)));
owl_r_row := owl_r(to_integer(unsigned(sprite_array_V)));
--owl_g_row := owl_g(to_integer(unsigned(sprite_array_V)));

brick_g_row := brick_g(to_integer(unsigned(sprite_array_V)));
brick_lg_row := brick_lg(to_integer(unsigned(sprite_array_V)));
brick_dg_row := brick_dg(to_integer(unsigned(sprite_array_V)));
--brick_z_row := brick_z(to_integer(unsigned(sprite_array_V)));
--// CALCULATING LIVES //--
if      (to_integer(num_lives) = 1) then --1
lives_row := sprite_1(to_integer(unsigned(NL_sprite_array_V16)));
elsif (to_integer(num_lives) = 2) then --2
lives_row := sprite_2(to_integer(unsigned(NL_sprite_array_V16)));
elsif (to_integer(num_lives) = 3) then --3
lives_row := sprite_3(to_integer(unsigned(NL_sprite_array_V16)));
else -- 0
      lives_row := sprite_0(to_integer(unsigned(NL_sprite_array_V16)));
```

```
end if;

--// CALCULATING SCORE //--
--// ONES SPOT //--
if      (to_integer(score_ones) = 0) then
score_ones_row := sprite_0(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_ones) = 1) then
score_ones_row := sprite_1(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_ones) = 2) then
score_ones_row := sprite_2(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_ones) = 3) then
score_ones_row := sprite_3(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_ones) = 4) then
score_ones_row := sprite_4(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_ones) = 5) then
score_ones_row := sprite_5(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_ones) = 6) then
score_ones_row := sprite_6(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_ones) = 7) then
score_ones_row := sprite_7(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_ones) = 8) then
score_ones_row := sprite_8(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_ones) = 9) then
score_ones_row := sprite_9(to_integer(unsigned(SB_sprite_array_V16)));
end if;
--// TENS SPOT //--
if      (to_integer(score_tens) = 0) then
score_tens_row := sprite_0(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_tens) = 1) then
score_tens_row := sprite_1(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_tens) = 2) then
score_tens_row := sprite_2(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_tens) = 3) then
score_tens_row := sprite_3(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_tens) = 4) then
score_tens_row := sprite_4(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_tens) = 5) then
score_tens_row := sprite_5(to_integer(unsigned(SB_sprite_array_V16)));
```

```
elsif (to_integer(score_tens) = 6) then
score_tens_row := sprite_6(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_tens) = 7) then
score_tens_row := sprite_7(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_tens) = 8) then
score_tens_row := sprite_8(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_tens) = 9) then
score_tens_row := sprite_9(to_integer(unsigned(SB_sprite_array_V16)));
end if;
--// HUNDY SPOT //--
if      (to_integer(score_hundy) = 0) then
score_hundy_row := sprite_0(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_hundy) = 1) then
score_hundy_row := sprite_1(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_hundy) = 2) then
score_hundy_row := sprite_2(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_hundy) = 3) then
score_hundy_row := sprite_3(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_hundy) = 4) then
score_hundy_row := sprite_4(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_hundy) = 5) then
score_hundy_row := sprite_5(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_hundy) = 6) then
score_hundy_row := sprite_6(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_hundy) = 7) then
score_hundy_row := sprite_7(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_hundy) = 8) then
score_hundy_row := sprite_8(to_integer(unsigned(SB_sprite_array_V16)));
elsif (to_integer(score_hundy) = 9) then
score_hundy_row := sprite_9(to_integer(unsigned(SB_sprite_array_V16)));
end if;
--// DISPLAYING THINGS

--// SIDE BAR - "SCORE", SCORE BITS, "LIVES", NUM OF LIVES, "FUEL", FUEL
GAUGE, MINI-MAP
      if (sidebar_rect = '1') then
      --// MINI MAP
            if (minimap_rect = '1') then
```

```
        --player
            if (car_count_H - (480 + 5*(to_integer(inx) + 240 -
8*32)/31)+1)*(car_count_H - (480 + 5*(to_integer(inx) + 240 - 8*32)/31)+1) +
                    (car_count_V - (100 + 5*(to_integer(iny) + 240 -
8*32)/31)+1)*(car_count_V - (100 + 5*(to_integer(iny) + 240 - 8*32)/31)+1) < 4 then
                            VGA_R <= "0000000000";
                            VGA_G <= "1111111111";
                            VGA_B <= "0000000000";
        --enemy0
            elsif((car_count_H - (480 + 5*(to_integer(enemy_0(11 downto
0))-8*32)/31)-1)*(car_count_H - (480 + 5*(to_integer(enemy_0(11 downto
0))-8*32)/31)-1) +
                    (car_count_V - (100 + 5*(to_integer(enemy_0(23 downto
12))-8*32)/31)-1)*(car_count_V - (100 +5*(to_integer(enemy_0(23 downto
12))-8*32)/31)-1) < 4 and enemy_0(27) = '1') or
        --enemy1
                ((car_count_H - (480 + 5*(to_integer(enemy_1(11 downto
0))-8*32)/31)-1)*(car_count_H - (480 + 5*(to_integer(enemy_1(11 downto
0))-8*32)/31)-1) +
                    (car_count_V - (100 + 5*(to_integer(enemy_1(23 downto
12))-8*32)/31)-1)*(car_count_V - (100 +5*(to_integer(enemy_1(23 downto
12))-8*32)/31)-1) < 4 and enemy_1(27) = '1') or
        --enemy2
                ((car_count_H - (480 + 5*(to_integer(enemy_2(11 downto
0))-8*32)/31)-1)*(car_count_H - (480 + 5*(to_integer(enemy_2(11 downto
0))-8*32)/31)-1) +
                    (car_count_V - (100 + 5*(to_integer(enemy_2(23 downto
12))-8*32)/31)-1)*(car_count_V - (100 +5*(to_integer(enemy_2(23 downto
12))-8*32)/31)-1) < 4 and enemy_2(27) = '1') then
                        VGA_R <= "1111111111";
                        VGA_G <= "0000000000";
                        VGA_B <= "0000000000";
        --flag0
            elsif((car_count_H - (480 + 5*(to_integer(flag_0(11 downto 0) sll
5)-8*32)/31)-1)*(car_count_H - (480 + 5*(to_integer(flag_0(11 downto 0)sll
5)-8*32)/31)-1) +
                    (car_count_V - (100 + 5*(to_integer(flag_0(23 downto 12)sll
5)-8*32)/31)-1)*(car_count_V - (100 +5*(to_integer(flag_0(23 downto 12)sll
```

```
5)-8*32)/31)-1) < 4 and flag_0(24) = '1') or
            --flag1
                        ((car_count_H - (480 + 5*(to_integer(flag_1(11 downto 0) sll
5)-8*32)/31)-1)*(car_count_H - (480 + 5*(to_integer(flag_1(11 downto 0)sll
5)-8*32)/31)-1) +
                         (car_count_V - (100 + 5*(to_integer(flag_1(23 downto 12)sll
5)-8*32)/31)-1)*(car_count_V - (100 +5*(to_integer(flag_1(23 downto 12)sll
5)-8*32)/31)-1) < 4 and flag_1(24) = '1') or
            --flag2
                        ((car_count_H - (480 + 5*(to_integer(flag_2(11 downto 0) sll
5)-8*32)/31)-1)*(car_count_H - (480 + 5*(to_integer(flag_2(11 downto 0)sll
5)-8*32)/31)-1) +
                         (car_count_V - (100 + 5*(to_integer(flag_2(23 downto 12)sll
5)-8*32)/31)-1)*(car_count_V - (100 +5*(to_integer(flag_2(23 downto 12)sll
5)-8*32)/31)-1) < 4 and flag_2(24) = '1') or
            --flag3
                        ((car_count_H - (480 + 5*(to_integer(flag_3(11 downto 0) sll
5)-8*32)/31)-1)*(car_count_H - (480 + 5*(to_integer(flag_3(11 downto 0)sll
5)-8*32)/31)-1) +
                         (car_count_V - (100 + 5*(to_integer(flag_3(23 downto 12)sll
5)-8*32)/31)-1)*(car_count_V - (100 +5*(to_integer(flag_3(23 downto 12)sll
5)-8*32)/31)-1) < 4 and flag_3(24) = '1') or
            --flag4
                        ((car_count_H - (480 + 5*(to_integer(flag_4(11 downto 0) sll
5)-8*32)/31)-1)*(car_count_H - (480 + 5*(to_integer(flag_4(11 downto 0)sll
5)-8*32)/31)-1) +
                         (car_count_V - (100 + 5*(to_integer(flag_4(23 downto 12)sll
5)-8*32)/31)-1)*(car_count_V - (100 +5*(to_integer(flag_4(23 downto 12)sll
5)-8*32)/31)-1) < 4 and flag_4(24) = '1') or
            --flag5
                        ((car_count_H - (480 + 5*(to_integer(flag_5(11 downto 0) sll
5)-8*32)/31)-1)*(car_count_H - (480 + 5*(to_integer(flag_5(11 downto 0)sll
5)-8*32)/31)-1) +
                         (car_count_V - (100 + 5*(to_integer(flag_5(23 downto 12)sll
5)-8*32)/31)-1)*(car_count_V - (100 +5*(to_integer(flag_5(23 downto 12)sll
5)-8*32)/31)-1) < 4 and flag_5(24) = '1') or
            --flag6
                        ((car_count_H - (480 + 5*(to_integer(flag_6(11 downto 0) sll
```

```
5)-8*32)/31)-1)*(car_count_H - (480 + 5*(to_integer(flag_6(11 downto 0)sll
5)-8*32)/31)-1) +
                    (car_count_V - (100 + 5*(to_integer(flag_6(23 downto 12)sll
5)-8*32)/31)-1)*(car_count_V - (100 +5*(to_integer(flag_6(23 downto 12)sll
5)-8*32)/31)-1) < 4 and flag_6(24) = '1') or
        --flag7
                ((car_count_H - (480 + 5*(to_integer(flag_7(11 downto 0) sll
5)-8*32)/31)-1)*(car_count_H - (480 + 5*(to_integer(flag_7(11 downto 0)sll
5)-8*32)/31)-1) +
                    (car_count_V - (100 + 5*(to_integer(flag_7(23 downto 12)sll
5)-8*32)/31)-1)*(car_count_V - (100 +5*(to_integer(flag_7(23 downto 12)sll
5)-8*32)/31)-1) < 4 and flag_7(24) = '1') or
        --flag8
                ((car_count_H - (480 + 5*(to_integer(flag_8(11 downto 0) sll
5)-8*32)/31)-1)*(car_count_H - (480 + 5*(to_integer(flag_8(11 downto 0)sll
5)-8*32)/31)-1) +
                    (car_count_V - (100 + 5*(to_integer(flag_8(23 downto 12)sll
5)-8*32)/31)-1)*(car_count_V - (100 +5*(to_integer(flag_8(23 downto 12)sll
5)-8*32)/31)-1) < 4 and flag_8(24) = '1') or
        --flag9
                ((car_count_H - (480 + 5*(to_integer(flag_9(11 downto 0) sll
5)-8*32)/31)-1)*(car_count_H - (480 + 5*(to_integer(flag_9(11 downto 0)sll
5)-8*32)/31)-1) +
                    (car_count_V - (100 + 5*(to_integer(flag_9(23 downto 12)sll
5)-8*32)/31)-1)*(car_count_V - (100 +5*(to_integer(flag_9(23 downto 12)sll
5)-8*32)/31)-1) < 4 and flag_9(24) = '1') then
                        VGA_R <= "0000000000";
                        VGA_G <= "0000000000";
                        VGA_B <= "0000000000";
            else
                VGA_R <= "1111111111";
                VGA_G <= "1111111111";
                VGA_B <= "1111111111";
            end if;
--// WORD "SCORE"
        elsif (wordscorebits_rect = '1') then
            if(to_integer(wordscorebits_array_H) = 4 and
```

```
WSB_sprite_E_row(to_integer(unsigned(15-WSB_sprite_array_H16))) = '1') then
                        VGA_R <= "1111111111";
                        VGA_G <= "0000000000";
                        VGA_B <= "0000000000";
                    elsif(to_integer(wordscorebits_array_H) = 3 and

WSB_sprite_R_row(to_integer(unsigned(15-WSB_sprite_array_H16))) = '1')
then
                        VGA_R <= "1111111111";
                        VGA_G <= "0000000000";
                        VGA_B <= "0000000000";
                    elsif(to_integer(wordscorebits_array_H) = 2 and

WSB_sprite_O_row(to_integer(unsigned(15-WSB_sprite_array_H16))) = '1')
then
                        VGA_R <= "1111111111";
                        VGA_G <= "0000000000";
                        VGA_B <= "0000000000";
                    elsif(to_integer(wordscorebits_array_H) = 1 and

WSB_sprite_C_row(to_integer(unsigned(15-WSB_sprite_array_H16))) = '1')
then
                        VGA_R <= "1111111111";
                        VGA_G <= "0000000000";
                        VGA_B <= "0000000000";
                    elsif(to_integer(wordscorebits_array_H) = 0 and

WSB_sprite_S_row(to_integer(unsigned(15-WSB_sprite_array_H16))) = '1') then
                        VGA_R <= "1111111111";
                        VGA_G <= "0000000000";
                        VGA_B <= "0000000000";
                    else
                        VGA_R <= "0000000000";
                        VGA_G <= "1111111111";
                        VGA_B <= "1111111111";
                    end if;
--// WORD "LIVES"
            elsif (wordlives_rect = '1') then
```

```
                if(to_integer(wordlives_array_H) = 4 and

WL_sprite_S_row(to_integer(unsigned(15-WL_sprite_array_H16))) = '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                elsif(to_integer(wordlives_array_H) = 3 and

WL_sprite_E_row(to_integer(unsigned(15-WL_sprite_array_H16))) = '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                elsif(to_integer(wordlives_array_H) = 2 and

WL_sprite_V_row(to_integer(unsigned(15-WL_sprite_array_H16))) = '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                elsif(to_integer(wordlives_array_H) = 1 and

WL_sprite_I_row(to_integer(unsigned(15-WL_sprite_array_H16))) = '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                elsif(to_integer(wordlives_array_H) = 0 and

WL_sprite_L_row(to_integer(unsigned(15-WL_sprite_array_H16))) = '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                else
                    VGA_R <= "0000000000";
                    VGA_G <= "1111111111";
                    VGA_B <= "1111111111";
                end if;
--// Number of Lives Remaining
            elsif (numlives_rect = '1') then
                if(lives_row(to_integer(unsigned(15-NL_sprite_array_H16))) = '1')
```

```
then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                else
                    VGA_R <= "0000000000";
                    VGA_G <= "1111111111";
                    VGA_B <= "1111111111";
                end if;
--// WORD "FUEL"
            elsif (wordfuel_rect = '1') then
                if(to_integer(wordfuel_array_H) = 3 and

WF_sprite_L_row(to_integer(unsigned(15-WF_sprite_array_H16))) = '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                elsif(to_integer(wordfuel_array_H) = 2 and

WF_sprite_E_row(to_integer(unsigned(15-WF_sprite_array_H16))) = '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                elsif(to_integer(wordfuel_array_H) = 1 and

WF_sprite_U_row(to_integer(unsigned(15-WF_sprite_array_H16))) = '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                elsif(to_integer(wordfuel_array_H) = 0 and

WF_sprite_F_row(to_integer(unsigned(15-WF_sprite_array_H16))) = '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                else
                    VGA_R <= "0000000000";
                    VGA_G <= "1111111111";
```

```
                    VGA_B <= "1111111111";
                end if;
--// SCORE BITS
            elsif (scorebits_rect = '1') then
                if(to_integer(scorebits_array_H) = 2 and

score_ones_row(to_integer(unsigned(15-SB_sprite_array_H16))) = '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                elsif(to_integer(scorebits_array_H) = 1 and
                  score_tens_row(to_integer(unsigned(15-SB_sprite_array_H16)))
= '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                elsif(to_integer(scorebits_array_H) = 0 and

score_hundy_row(to_integer(unsigned(15-SB_sprite_array_H16))) = '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                else
                    VGA_R <= "0000000000";
                    VGA_G <= "1111111111";
                    VGA_B <= "1111111111";
                end if;
--// FUEL GAUGE
            elsif (fuel_rect = '1') then
                if (to_integer(car_count_H) < FUEL_HSTART +
to_integer(fuel_status)) then
                    VGA_R <= "0000000000";
                    VGA_G <= "1111000111";
                    VGA_B <= "0000000000";
                else
                    VGA_R <= "0000000000";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
```

```
                end if;
            else
                VGA_R <= "1111111111";
                VGA_G <= "1111111111";
                VGA_B <= "0000000000";
            end if;
--// MAIN GAME MAP AREA - PLAYER, ENEMY CARS, FLAGS, SMOKE, MAP
--// PLAYER
        elsif (mainmap_rect = '1') then
            if (game_state = 0) then
                if(start_rect = '1' and to_integer(GS_array_H) = 5 and
                  GS_sprite1_row(to_integer(unsigned(15-GS_sprite_array_H16)))
= '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                elsif(start_rect = '1' and to_integer(GS_array_H) = 4 and
                  GS_sprite2_row(to_integer(unsigned(15-GS_sprite_array_H16)))
= '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                elsif(start_rect = '1' and to_integer(GS_array_H) = 3 and
                  GS_sprite3_row(to_integer(unsigned(15-GS_sprite_array_H16)))
= '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                elsif(start_rect = '1' and to_integer(GS_array_H) = 2 and
                  GS_sprite4_row(to_integer(unsigned(15-GS_sprite_array_H16)))
= '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                elsif(start_rect = '1' and to_integer(GS_array_H) = 1 and
                  GS_sprite5_row(to_integer(unsigned(15-GS_sprite_array_H16)))
= '1') then
                    VGA_R <= "1111111111";
```

```
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                elsif(start_rect = '1' and to_integer(GS_array_H) = 0 and
                  GS_sprite6_row(to_integer(unsigned(15-GS_sprite_array_H16)))
= '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                elsif(owl_rect = '1'and

GS_OWL_r_row(to_integer(unsigned(31-GSO_sprite_array_H32))) = '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                elsif(owl_rect = '1' and

GS_OWL_w_row(to_integer(unsigned(31-GSO_sprite_array_H32))) = '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "1111111111";
                    VGA_B <= "1111111111";
                else
                    VGA_R <= "0000000000";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                end if;
            elsif (game_state = 1) then
                GE_sprite1_row :=
sprite_R(to_integer(unsigned(GE_sprite_array_V16)));
                GE_sprite2_row :=
sprite_E(to_integer(unsigned(GE_sprite_array_V16)));
                GE_sprite3_row :=
sprite_V(to_integer(unsigned(GE_sprite_array_V16)));
                GE_sprite4_row :=
sprite_O(to_integer(unsigned(GE_sprite_array_V16)));
                GE_sprite5_row :=
sprite_E(to_integer(unsigned(GE_sprite_array_V16)));
                GE_sprite6_row :=
sprite_M(to_integer(unsigned(GE_sprite_array_V16)));
```

```
                GE_sprite7_row :=
sprite_A(to_integer(unsigned(GE_sprite_array_V16)));
                GE_sprite8_row :=
sprite_G(to_integer(unsigned(GE_sprite_array_V16)));
--              if (end_rect = '1' and
GE_sprite_row(to_integer(unsigned(15-GE_sprite_array_H16))) = '1') then
--                  VGA_R <= "1111111111";
--                  VGA_G <= "0000000000";
--                  VGA_B <= "0000000000";
----------------------------------------
                if(end_rect = '1' and to_integer(GE_array_H) = 7 and
                  GE_sprite1_row(to_integer(unsigned(15-GE_sprite_array_H16)))
= '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                elsif(end_rect = '1' and to_integer(GE_array_H) = 6 and
                  GE_sprite2_row(to_integer(unsigned(15-GE_sprite_array_H16)))
= '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                elsif(end_rect = '1' and to_integer(GE_array_H) = 5 and
                  GE_sprite3_row(to_integer(unsigned(15-GE_sprite_array_H16)))
= '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                elsif(end_rect = '1' and to_integer(GE_array_H) = 4 and
                  GE_sprite4_row(to_integer(unsigned(15-GE_sprite_array_H16)))
= '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                elsif(end_rect = '1' and to_integer(GE_array_H) = 3 and
                  GE_sprite5_row(to_integer(unsigned(15-GE_sprite_array_H16)))
= '1') then
                    VGA_R <= "1111111111";
```

```
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                elsif(end_rect = '1' and to_integer(GE_array_H) = 2 and
                  GE_sprite6_row(to_integer(unsigned(15-GE_sprite_array_H16)))
= '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                elsif(end_rect = '1' and to_integer(GE_array_H) = 1 and
                  GE_sprite7_row(to_integer(unsigned(15-GE_sprite_array_H16)))
= '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                elsif(end_rect = '1' and to_integer(GE_array_H) = 0 and
                  GE_sprite8_row(to_integer(unsigned(15-GE_sprite_array_H16)))
= '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";


                else
                    VGA_R <= "0000000000";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";
                end if;
            else
      --player turn 1
        if (player_lr_rect = '1' and player_status = 0 and
right_car_w_row(to_integer(unsigned(31-car_array_V))) = '1' ) then
              VGA_R <= "1111111111";
              VGA_G <= "1111111111";
              VGA_B <= "1111111111";
        elsif (player_lr_rect = '1' and player_status = 0 and
right_car_r_row(to_integer(unsigned(31-car_array_V))) = '1' ) then
              VGA_R <= "1111111111";
              VGA_G <= "0000000000";
```

```
                VGA_B <= "0000000000";
            elsif (player_lr_rect = '1' and player_status = 0 and
right_car_lg_row(to_integer(unsigned(31-car_array_V))) = '1' ) then
                VGA_R <= "0000000000";
                VGA_G <= "1100000011";
                VGA_B <= "0000000000";
            elsif (player_lr_rect = '1' and player_status = 0 and
right_car_b_row(to_integer(unsigned(31-car_array_V))) = '1' ) then
                VGA_R <= "0000000000";
                VGA_G <= "0011001111";
                VGA_B <= "0000000000";
            elsif (player_lr_rect = '1' and player_status = 0 and
right_car_y_row(to_integer(unsigned(31-car_array_V))) = '1') then
                VGA_R <= "1111111111";
                VGA_G <= "1111111111";
                VGA_B <= "0000000000";
            elsif (player_lr_rect = '1' and player_status = 0 and
right_car_g_row(to_integer(unsigned(31-car_array_V))) = '1' ) then
                VGA_R <= "1001100111";
                VGA_G <= "1001100111";
                VGA_B <= "1001100111";
        --player turn 2
            elsif (player_lr_rect = '1' and player_status = 1 and
left_car_w_row(to_integer(unsigned(car_array_V))) = '1') then
                VGA_R <= "1111111111";
                VGA_G <= "1111111111";
                VGA_B <= "1111111111";
            elsif (player_lr_rect = '1' and player_status = 1 and
left_car_r_row(to_integer(unsigned(31-car_array_V))) = '1' ) then
                VGA_R <= "1111111111";
                VGA_G <= "0000000000";
                VGA_B <= "0000000000";
            elsif (player_lr_rect = '1' and player_status = 1 and
left_car_lg_row(to_integer(unsigned(31-car_array_V))) = '1') then
                VGA_R <= "0000000000";
                VGA_G <= "1100000011";
                VGA_B <= "0000000000";
            elsif (player_lr_rect = '1' and player_status = 1 and
```

```
left_car_b_row(to_integer(unsigned(31-car_array_V))) = '1' ) then
                VGA_R <= "0000000000";
                VGA_G <= "0011001111";
                VGA_B <= "0000000000";
        elsif (player_lr_rect = '1' and player_status = 1 and
left_car_y_row(to_integer(unsigned(31-car_array_V))) = '1' ) then
                VGA_R <= "1111111111";
                VGA_G <= "1111111111";
                VGA_B <= "0000000000";
        elsif (player_lr_rect = '1' and player_status = 1 and
left_car_g_row(to_integer(unsigned(31-car_array_V))) = '1' ) then
                VGA_R <= "1001100111";
                VGA_G <= "1001100111";
                VGA_B <= "1001100111";
      --player turn 3

        elsif (player_ud_rect = '1' and player_status = 2 and
down_car_w_row(to_integer(unsigned(31-car_array_H))) = '1' ) then
                VGA_R <= "1111111111";
                VGA_G <= "1111111111";
                VGA_B <= "1111111111";
        elsif (player_ud_rect = '1' and player_status = 2 and
down_car_r_row(to_integer(unsigned(31-car_array_H))) = '1') then
                VGA_R <= "1111111111";
                VGA_G <= "0000000000";
                VGA_B <= "0000000000";
        elsif (player_ud_rect = '1' and player_status = 2 and
down_car_lg_row(to_integer(unsigned(31-car_array_H))) = '1') then
                VGA_R <= "0000000000";
                VGA_G <= "1100000011";
                VGA_B <= "0000000000";
        elsif (player_ud_rect = '1' and player_status = 2 and
down_car_b_row(to_integer(unsigned(31-car_array_H))) = '1' ) then
                VGA_R <= "0000000000";
                VGA_G <= "0011001111";
                VGA_B <= "0000000000";
        elsif (player_ud_rect = '1' and player_status = 2 and
down_car_y_row(to_integer(unsigned(31-car_array_H))) = '1' ) then
```

```
                VGA_R <= "1111111111";
                VGA_G <= "1111111111";
                VGA_B <= "0000000000";
            elsif (player_ud_rect = '1' and player_status = 2 and
down_car_g_row(to_integer(unsigned(31-car_array_H))) = '1') then
                VGA_R <= "1001100111";
                VGA_G <= "1001100111";
                VGA_B <= "1001100111";
        --player turn 4
            elsif (player_ud_rect = '1' and player_status = 3 and
up_car_w_row(to_integer(unsigned(car_array_H))) = '1' ) then
                VGA_R <= "1111111111";
                VGA_G <= "1111111111";
                VGA_B <= "1111111111";
            elsif (player_ud_rect = '1' and player_status = 3 and
up_car_r_row(to_integer(unsigned(31-car_array_H))) = '1' ) then
                VGA_R <= "1111111111";
                VGA_G <= "0000000000";
                VGA_B <= "0000000000";
            elsif (player_ud_rect = '1' and player_status = 3 and
up_car_lg_row(to_integer(unsigned(31-car_array_H))) = '1') then
                VGA_R <= "0000000000";
                VGA_G <= "1100000011";
                VGA_B <= "0000000000";
            elsif (player_ud_rect = '1' and player_status = 3 and
up_car_b_row(to_integer(unsigned(31-car_array_H))) = '1') then
                VGA_R <= "0000000000";
                VGA_G <= "0011001111";
                VGA_B <= "0000000000";
            elsif (player_ud_rect = '1' and player_status = 3 and
up_car_y_row(to_integer(unsigned(31-car_array_H))) = '1' ) then
                VGA_R <= "1111111111";
                VGA_G <= "1111111111";
                VGA_B <= "0000000000";
            elsif (player_ud_rect = '1' and player_status = 3 and
up_car_g_row(to_integer(unsigned(car_array_H))) = '1' ) then
                VGA_R <= "1001100111";
                VGA_G <= "1001100111";
```

```
                VGA_B <= "1001100111";

        --smoke

           elsif (((smoke0_rect = '1' and smoke_0(24) = '1') or
             (smoke1_rect = '1' and smoke_1(24) = '1') or
             (smoke2_rect = '1' and smoke_2(24) = '1')) and
smoke_i_row(to_integer(unsigned(31 - sprite_array_H))) = '1') then
                VGA_R <= "1111111111";
                VGA_G <= "1111111111";
                VGA_B <= "1011010011";
           elsif (((smoke0_rect = '1' and smoke_0(24) = '1') or
             (smoke1_rect = '1' and smoke_1(24) = '1') or
             (smoke2_rect = '1' and smoke_2(24) = '1')) and
smoke_g_row(to_integer(unsigned(31 - sprite_array_H))) = '1') then
                VGA_R <= "0000000000";
                VGA_G <= "1111111111";
                VGA_B <= "1100000000";

        --enemy cars
        -- RIGHT

        elsif (enemy0_rect = '1' and enemy_0(27) = '1' and enemy_0(26 downto 24)
= "000" and enemy1_right_y_row(to_integer(unsigned(31 - sprite_array_He1)))
= '1') then
             VGA_R <= "1111111111";
             VGA_G <= "1111111111";
             VGA_B <= "0000000000";
        elsif (enemy0_rect = '1' and enemy_0(27) = '1' and enemy_0(26 downto 24)
= "000" and enemy1_right_o_row(to_integer(unsigned(31 - sprite_array_He1)))
= '1') then
             VGA_R <= "1111111111";
             VGA_G <= "0110011011";
             VGA_B <= "0000000000";
        elsif (enemy0_rect = '1' and enemy_0(27) = '1' and enemy_0(26 downto 24)
= "000" and enemy1_right_r_row(to_integer(unsigned(31 - sprite_array_He1))) =
'1') then
             VGA_R <= "1111111111";
```

```
            VGA_G <= "0000000000";
            VGA_B <= "0000000000";
      elsif (enemy0_rect = '1' and enemy_0(27) = '1' and enemy_0(26 downto 24)
= "000" and enemy1_right_b_row(to_integer(unsigned(31 - sprite_array_He1)))
= '1') then
            VGA_R <= "0000000000";
            VGA_G <= "0000000000";
            VGA_B <= "0000000000";

       -- LEFT
      elsif (enemy0_rect = '1' and enemy_0(27) = '1' and enemy_0(26 downto 24)
= "001" and enemy1_left_y_row(to_integer(unsigned(31 - sprite_array_He1))) =
'1') then
            VGA_R <= "1111111111";
            VGA_G <= "1111111111";
            VGA_B <= "0000000000";
      elsif (enemy0_rect = '1' and enemy_0(27) = '1' and enemy_0(26 downto 24)
= "001" and enemy1_left_o_row(to_integer(unsigned(31 - sprite_array_He1))) =
'1') then
            VGA_R <= "1111111111";
            VGA_G <= "0110011011";
            VGA_B <= "0000000000";
      elsif (enemy0_rect = '1' and enemy_0(27) = '1' and enemy_0(26 downto 24)
= "001" and enemy1_left_r_row(to_integer(unsigned(31 - sprite_array_He1))) =
'1') then
            VGA_R <= "1111111111";
            VGA_G <= "0000000000";
            VGA_B <= "0000000000";
      elsif (enemy0_rect = '1' and enemy_0(27) = '1' and enemy_0(26 downto 24)
= "001" and enemy1_left_b_row(to_integer(unsigned(31 - sprite_array_He1))) =
'1') then
            VGA_R <= "0000000000";
            VGA_G <= "0000000000";
            VGA_B <= "0000000000";

       -- DOWN
      elsif (enemy0_rect = '1' and enemy_0(27) = '1' and enemy_0(26 downto 24)
= "010" and enemy1_down_y_row(to_integer(unsigned(31 - sprite_array_He1)))
```

```
= '1') then
            VGA_R <= "1111111111";
            VGA_G <= "1111111111";
            VGA_B <= "0000000000";
        elsif (enemy0_rect = '1' and enemy_0(27) = '1' and enemy_0(26 downto 24)
= "010" and enemy1_down_o_row(to_integer(unsigned(31 - sprite_array_He1)))
= '1') then
            VGA_R <= "1111111111";
            VGA_G <= "0110011011";
            VGA_B <= "0000000000";
        elsif (enemy0_rect = '1' and enemy_0(27) = '1' and enemy_0(26 downto 24)
= "010" and enemy1_down_r_row(to_integer(unsigned(31 - sprite_array_He1)))
= '1') then
            VGA_R <= "1111111111";
            VGA_G <= "0000000000";
            VGA_B <= "0000000000";
        elsif (enemy0_rect = '1' and enemy_0(27) = '1' and enemy_0(26 downto 24)
= "010" and enemy1_down_b_row(to_integer(unsigned(31 - sprite_array_He1)))
= '1') then
            VGA_R <= "0000000000";
            VGA_G <= "0000000000";
            VGA_B <= "0000000000";

        -- UP
        elsif (enemy0_rect = '1' and enemy_0(27) = '1' and enemy_0(26 downto 24)
= "011" and enemy1_up_y_row(to_integer(unsigned(31 - sprite_array_He1))) =
'1') then
            VGA_R <= "1111111111";
            VGA_G <= "1111111111";
            VGA_B <= "0000000000";
        elsif (enemy0_rect = '1' and enemy_0(27) = '1' and enemy_0(26 downto 24)
= "011" and enemy1_up_o_row(to_integer(unsigned(31 - sprite_array_He1))) =
'1') then
            VGA_R <= "1111111111";
            VGA_G <= "0110011011";
            VGA_B <= "0000000000";
        elsif (enemy0_rect = '1' and enemy_0(27) = '1' and enemy_0(26 downto 24)
= "011" and enemy1_up_r_row(to_integer(unsigned(31 - sprite_array_He1))) =
```

```
'1') then
            VGA_R <= "1111111111";
            VGA_G <= "0000000000";
            VGA_B <= "0000000000";
      elsif (enemy0_rect = '1' and enemy_0(27) = '1' and enemy_0(26 downto 24)
= "011" and enemy1_up_b_row(to_integer(unsigned(31 - sprite_array_He1))) =
'1') then
            VGA_R <= "0000000000";
            VGA_G <= "0000000000";
            VGA_B <= "0000000000";

      -- RIGHT
      elsif (enemy1_rect = '1' and enemy_1(27) = '1' and enemy_1(26 downto 24)
= "000" and enemy2_right_y_row(to_integer(unsigned(31 - sprite_array_He2)))
= '1') then
            VGA_R <= "1111111111";
            VGA_G <= "1111111111";
            VGA_B <= "0000000000";
      elsif (enemy1_rect = '1' and enemy_1(27) = '1' and enemy_1(26 downto 24)
= "000" and enemy2_right_o_row(to_integer(unsigned(31 - sprite_array_He2)))
= '1') then
            VGA_R <= "1111111111";
            VGA_G <= "0110011011";
            VGA_B <= "0000000000";
      elsif (enemy1_rect = '1' and enemy_1(27) = '1' and enemy_1(26 downto 24)
= "000" and enemy2_right_r_row(to_integer(unsigned(31 - sprite_array_He2))) =
'1') then
            VGA_R <= "1111111111";
            VGA_G <= "0000000000";
            VGA_B <= "0000000000";
      elsif (enemy1_rect = '1' and enemy_1(27) = '1' and enemy_1(26 downto 24)
= "000" and enemy2_right_b_row(to_integer(unsigned(31 - sprite_array_He2)))
= '1') then
            VGA_R <= "0000000000";
            VGA_G <= "0000000000";
            VGA_B <= "0000000000";

       -- LEFT
```

```
      elsif (enemy1_rect = '1' and enemy_1(27) = '1' and enemy_1(26 downto 24)
= "001" and enemy2_left_y_row(to_integer(unsigned(31 - sprite_array_He2))) =
'1') then
            VGA_R <= "1111111111";
            VGA_G <= "1111111111";
            VGA_B <= "0000000000";
      elsif (enemy1_rect = '1' and enemy_1(27) = '1' and enemy_1(26 downto 24)
= "001" and enemy2_left_o_row(to_integer(unsigned(31 - sprite_array_He2))) =
'1') then
            VGA_R <= "1111111111";
            VGA_G <= "0110011011";
            VGA_B <= "0000000000";
      elsif (enemy1_rect = '1' and enemy_1(27) = '1' and enemy_1(26 downto 24)
= "001" and enemy2_left_r_row(to_integer(unsigned(31 - sprite_array_He2))) =
'1') then
            VGA_R <= "1111111111";
            VGA_G <= "0000000000";
            VGA_B <= "0000000000";
      elsif (enemy1_rect = '1' and enemy_1(27) = '1' and enemy_1(26 downto 24)
= "001" and enemy2_left_b_row(to_integer(unsigned(31 - sprite_array_He2))) =
'1') then
            VGA_R <= "0000000000";
            VGA_G <= "0000000000";
            VGA_B <= "0000000000";

       -- DOWN
      elsif (enemy1_rect = '1' and enemy_1(27) = '1' and enemy_1(26 downto 24)
= "010" and enemy2_down_y_row(to_integer(unsigned(31 - sprite_array_He2)))
= '1') then
            VGA_R <= "1111111111";
            VGA_G <= "1111111111";
            VGA_B <= "0000000000";
      elsif (enemy1_rect = '1' and enemy_1(27) = '1' and enemy_1(26 downto 24)
= "010" and enemy2_down_o_row(to_integer(unsigned(31 - sprite_array_He2)))
= '1') then
            VGA_R <= "1111111111";
            VGA_G <= "0110011011";
            VGA_B <= "0000000000";
```

```
      elsif (enemy1_rect = '1' and enemy_1(27) = '1' and enemy_1(26 downto 24)
= "010" and enemy2_down_r_row(to_integer(unsigned(31 - sprite_array_He2)))
= '1') then
            VGA_R <= "1111111111";
            VGA_G <= "0000000000";
            VGA_B <= "0000000000";
      elsif (enemy1_rect = '1' and enemy_1(27) = '1' and enemy_1(26 downto 24)
= "010" and enemy2_down_b_row(to_integer(unsigned(31 - sprite_array_He2)))
= '1') then
            VGA_R <= "0000000000";
            VGA_G <= "0000000000";
            VGA_B <= "0000000000";

      -- UP
      elsif (enemy1_rect = '1' and enemy_1(27) = '1' and enemy_1(26 downto 24)
= "011" and enemy2_up_y_row(to_integer(unsigned(31 - sprite_array_He2))) =
'1') then
            VGA_R <= "1111111111";
            VGA_G <= "1111111111";
            VGA_B <= "0000000000";
      elsif (enemy1_rect = '1' and enemy_1(27) = '1' and enemy_1(26 downto 24)
= "011" and enemy2_up_o_row(to_integer(unsigned(31 - sprite_array_He2))) =
'1') then
            VGA_R <= "1111111111";
            VGA_G <= "0110011011";
            VGA_B <= "0000000000";
      elsif (enemy1_rect = '1' and enemy_1(27) = '1' and enemy_1(26 downto 24)
= "011" and enemy2_up_r_row(to_integer(unsigned(31 - sprite_array_He2))) =
'1') then
            VGA_R <= "1111111111";
            VGA_G <= "0000000000";
            VGA_B <= "0000000000";
      elsif (enemy1_rect = '1' and enemy_1(27) = '1' and enemy_1(26 downto 24)
= "011" and enemy2_up_b_row(to_integer(unsigned(31 - sprite_array_He2))) =
'1') then
            VGA_R <= "0000000000";
            VGA_G <= "0000000000";
            VGA_B <= "0000000000";
```

```
        -- RIGHT
        elsif (enemy2_rect = '1' and enemy_2(27) = '1' and enemy_2(26 downto 24)
= "000" and enemy3_right_y_row(to_integer(unsigned(31 - sprite_array_He3)))
= '1') then
                VGA_R <= "1111111111";
                VGA_G <= "1111111111";
                VGA_B <= "0000000000";
        elsif (enemy2_rect = '1' and enemy_2(27) = '1' and enemy_2(26 downto 24)
= "000" and enemy3_right_o_row(to_integer(unsigned(31 - sprite_array_He3)))
= '1') then
                VGA_R <= "1111111111";
                VGA_G <= "0110011011";
                VGA_B <= "0000000000";
        elsif (enemy2_rect = '1' and enemy_2(27) = '1' and enemy_2(26 downto 24)
= "000" and enemy3_right_r_row(to_integer(unsigned(31 - sprite_array_He3))) =
'1') then
                VGA_R <= "1111111111";
                VGA_G <= "0000000000";
                VGA_B <= "0000000000";
        elsif (enemy2_rect = '1' and enemy_2(27) = '1' and enemy_2(26 downto 24)
= "000" and enemy3_right_b_row(to_integer(unsigned(31 - sprite_array_He3)))
= '1') then
                VGA_R <= "0000000000";
                VGA_G <= "0000000000";
                VGA_B <= "0000000000";

         -- LEFT
        elsif (enemy2_rect = '1' and enemy_2(27) = '1' and enemy_2(26 downto 24)
= "001" and enemy3_left_y_row(to_integer(unsigned(31 - sprite_array_He3))) =
'1') then
                VGA_R <= "1111111111";
                VGA_G <= "1111111111";
                VGA_B <= "0000000000";
        elsif (enemy2_rect = '1' and enemy_2(27) = '1' and enemy_2(26 downto 24)
= "001" and enemy3_left_o_row(to_integer(unsigned(31 - sprite_array_He3))) =
'1') then
                VGA_R <= "1111111111";
                VGA_G <= "0110011011";
```

```
            VGA_B <= "0000000000";
        elsif (enemy2_rect = '1' and enemy_2(27) = '1' and enemy_2(26 downto 24)
= "001" and enemy3_left_r_row(to_integer(unsigned(31 - sprite_array_He3))) =
'1') then
            VGA_R <= "1111111111";
            VGA_G <= "0000000000";
            VGA_B <= "0000000000";
        elsif (enemy2_rect = '1' and enemy_2(27) = '1' and enemy_2(26 downto 24)
= "001" and enemy3_left_b_row(to_integer(unsigned(31 - sprite_array_He3))) =
'1') then
            VGA_R <= "0000000000";
            VGA_G <= "0000000000";
            VGA_B <= "0000000000";

         -- DOWN
        elsif (enemy2_rect = '1' and enemy_2(27) = '1' and enemy_2(26 downto 24)
= "010" and enemy3_down_y_row(to_integer(unsigned(31 - sprite_array_He3)))
= '1') then
            VGA_R <= "1111111111";
            VGA_G <= "1111111111";
            VGA_B <= "0000000000";
        elsif (enemy2_rect = '1' and enemy_2(27) = '1' and enemy_2(26 downto 24)
= "010" and enemy3_down_o_row(to_integer(unsigned(31 - sprite_array_He3)))
= '1') then
            VGA_R <= "1111111111";
            VGA_G <= "0110011011";
            VGA_B <= "0000000000";
        elsif (enemy2_rect = '1' and enemy_2(27) = '1' and enemy_2(26 downto 24)
= "010" and enemy3_down_r_row(to_integer(unsigned(31 - sprite_array_He3)))
= '1') then
            VGA_R <= "1111111111";
            VGA_G <= "0000000000";
            VGA_B <= "0000000000";
        elsif (enemy2_rect = '1' and enemy_2(27) = '1' and enemy_2(26 downto 24)
= "010" and enemy3_down_b_row(to_integer(unsigned(31 - sprite_array_He3)))
= '1') then
            VGA_R <= "0000000000";
            VGA_G <= "0000000000";
```

```
            VGA_B <= "0000000000";

        -- UP
        elsif (enemy2_rect = '1' and enemy_2(27) = '1' and enemy_2(26 downto 24)
= "011" and enemy3_up_y_row(to_integer(unsigned(31 - sprite_array_He3))) =
'1') then
            VGA_R <= "1111111111";
            VGA_G <= "1111111111";
            VGA_B <= "0000000000";
        elsif (enemy2_rect = '1' and enemy_2(27) = '1' and enemy_2(26 downto 24)
= "011" and enemy3_up_o_row(to_integer(unsigned(31 - sprite_array_He3))) =
'1') then
            VGA_R <= "1111111111";
            VGA_G <= "0110011011";
            VGA_B <= "0000000000";
        elsif (enemy2_rect = '1' and enemy_2(27) = '1' and enemy_2(26 downto 24)
= "011" and enemy3_up_r_row(to_integer(unsigned(31 - sprite_array_He3))) =
'1') then
            VGA_R <= "1111111111";
            VGA_G <= "0000000000";
            VGA_B <= "0000000000";
        elsif (enemy2_rect = '1' and enemy_2(27) = '1' and enemy_2(26 downto 24)
= "011" and enemy3_up_b_row(to_integer(unsigned(31 - sprite_array_He3))) =
'1') then
            VGA_R <= "0000000000";
            VGA_G <= "0000000000";
            VGA_B <= "0000000000";

        --displaying flags

        elsif (((flag0_rect = '1' and flag_0(24) = '1') or
            (flag1_rect = '1' and flag_1(24) = '1') or
            (flag2_rect = '1' and flag_2(24) = '1') or
            (flag3_rect = '1' and flag_3(24) = '1') or
            (flag4_rect = '1' and flag_4(24) = '1') or
            (flag5_rect = '1' and flag_5(24) = '1') or
            (flag6_rect = '1' and flag_6(24) = '1') or
            (flag7_rect = '1' and flag_7(24) = '1') or
```

```
            (flag8_rect = '1' and flag_8(24) = '1') or
            (flag9_rect = '1' and flag_9(24) = '1')) and
owl_w_row(to_integer(unsigned(31 - sprite_array_H))) = '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "1111111111";
                    VGA_B <= "1111111111";
        elsif (((flag0_rect = '1' and flag_0(24) = '1') or
            (flag1_rect = '1' and flag_1(24) = '1') or
            (flag2_rect = '1' and flag_2(24) = '1') or
            (flag3_rect = '1' and flag_3(24) = '1') or
            (flag4_rect = '1' and flag_4(24) = '1') or
            (flag5_rect = '1' and flag_5(24) = '1') or
            (flag6_rect = '1' and flag_6(24) = '1') or
            (flag7_rect = '1' and flag_7(24) = '1') or
            (flag8_rect = '1' and flag_8(24) = '1') or
            (flag9_rect = '1' and flag_9(24) = '1')) and
owl_r_row(to_integer(unsigned(31 - sprite_array_H))) = '1') then
                    VGA_R <= "1111111111";
                    VGA_G <= "0000000000";
                    VGA_B <= "0000000000";

        elsif (control_array(to_integer(unsigned(46 -
control_array_V)),to_integer(unsigned(187 - (control_array_H sll 2) - 0))) = '1'
and
            control_array(to_integer(unsigned(46 -
control_array_V)),to_integer(unsigned(187 - (control_array_H sll 2) - 1))) = '1'
and
            control_array(to_integer(unsigned(46 -
control_array_V)),to_integer(unsigned(187 - (control_array_H sll 2) - 2))) = '1'
and
            control_array(to_integer(unsigned(46 -
control_array_V)),to_integer(unsigned(187 - (control_array_H sll 2) - 3))) = '1'
and brick_g_row(to_integer(unsigned(31 - sprite_array_H))) = '1') then
                    VGA_R <= "0000000000";
                    VGA_G <= "1001100111";
                    VGA_B <= "0000000000";
        elsif (control_array(to_integer(unsigned(46 -
control_array_V)),to_integer(unsigned(187 - (control_array_H sll 2) - 0))) = '1'
```

```
and
        control_array(to_integer(unsigned(46 -
control_array_V)),to_integer(unsigned(187 - (control_array_H sll 2) - 1))) = '1'
and
        control_array(to_integer(unsigned(46 -
control_array_V)),to_integer(unsigned(187 - (control_array_H sll 2) - 2))) = '1'
and
        control_array(to_integer(unsigned(46 -
control_array_V)),to_integer(unsigned(187 - (control_array_H sll 2) - 3))) = '1'
and brick_lg_row(to_integer(unsigned(31 - sprite_array_H))) = '1') then
                VGA_R <= "0000000000";
                VGA_G <= "1111111111";
                VGA_B <= "0000000000";
    elsif (control_array(to_integer(unsigned(46 -
control_array_V)),to_integer(unsigned(187 - (control_array_H sll 2) - 0))) = '1'
and
        control_array(to_integer(unsigned(46 -
control_array_V)),to_integer(unsigned(187 - (control_array_H sll 2) - 1))) = '1'
and
        control_array(to_integer(unsigned(46 -
control_array_V)),to_integer(unsigned(187 - (control_array_H sll 2) - 2))) = '1'
and
        control_array(to_integer(unsigned(46 -
control_array_V)),to_integer(unsigned(187 - (control_array_H sll 2) - 3))) = '1'
and brick_dg_row(to_integer(unsigned(31 - sprite_array_H))) = '1') then
                VGA_R <= "0000000000";
                VGA_G <= "0011001111";
                VGA_B <= "0000000000";

    else
        VGA_R <= "0000000000";
        VGA_G <= "0000000000";
        VGA_B <= "0000000000";
    end if;
end if;
elsif (vga_hblank = '0' and vga_vblank ='0') then
    VGA_R <= "0000000000";
    VGA_G <= "0000000000";
```

```
        VGA_B <= "1111111111";
else
        VGA_R <= "0000000000";
        VGA_G <= "0000000000";
        VGA_B <= "0000000000";
end if;
end if;
end process VideoOut;

VGACLKGEN : process (clk)
begin
        if rising_edge(clk) then
           clk25vga <= not clk25vga;
        end if;
end process VGACLKGEN;

   VGA_CLK <= clk25vga;
   VGA_HS <= not vga_hsync;
   VGA_VS <= not vga_vsync;
   VGA_SYNC <= '0';
   VGA_BLANK <= not (vga_hsync or vga_vsync);

end rtl;
-------------------------------------DAVID END
```