CSEE 4840 Embedded System Design
Columbia Defense

Group members:
Zhefeng Xu(zx2152)
Chao Li (cl3169)
Yang Bai (yb2310)
Tianlei Zhou (tz2212)

Contents:

1. Overview

In this project, we plan to design and implement a tower defense game called “Columbia Defense”, which is based on Columbia University campus. The user can make the strategy, set up various defensers in the pathway in order to kill the enemies and protect our Columbia campus. The attackers are different kinds of monsters and the defenders are human being who has various features. All of the attackers and defenders have different kinds of features and attack techniques and furthermore in the bonus system, killing attackers will make some money for the user, which can be used to buy a more powerful defender. In the screen, we have 5 parellel pathways towards our campus and we should build our defenders on each path to accomplish the goal of protection. Once the campus is occupied by the attackers, our game will be over. As the user pass the elementary level of our game, next higher level will be available, where there are more powerful attackers.

There are some key points of this project with which we need to handle carefully. First is the VGA display, because there are lots of different attackers, defenders, special effects and campus background. We need to rewrite the driver and use different methods to solve the static and dynamic figures, such as sprite. Second key point is the algorithm issue. For both the attackers and the defenders, we need to set them walking along the designed pathways, the different behavior and movement when they attack or are attacked. For the defense facilities, what’s more, we need to design different characteristics according to upgrading. And how many times that our defenders can destory different attackers. The third point is controller, we use a PS2 controller and it has four direction button, four functional button by which the user will paly the game in a more interactive way. But the tradeoff is that we need to add the controller drive and handle the interrupt of controller.

2. Hardware Structure

The high-level hardware structure diagram is as blow:

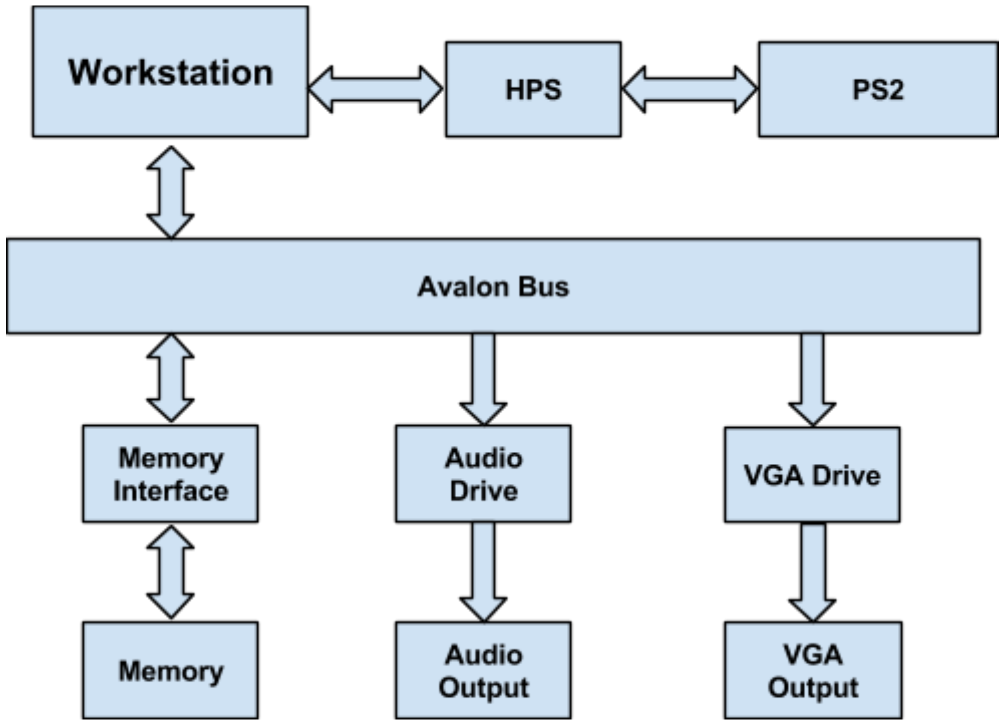

Figure.1

The major parts of our design: display module including sprite controller and VGA controller; data storage module including SDRAM for the system memory, SRAM for the image storage; PS2 controller. All of these parts will be discussed below.

3. Image Preparation

All the image resources are extracted from the internet including the background, the attackers, the defenders and the bullets. Since all the original images resources have different sizes and their background is not transparent, before we use them we need to do some changes on the images.

3.1 Image Category and Size

Background: Our background is grass lawn whose size is (75x55) and all the attackers and defenders will fight on it. Furthermore, we have a Columbia logo which symbolizes our theme and our protect goal. Since the limit of memory storage, we cannot display a image of that size on the screen. So we store only a small piece of grass lawn in the SRAM and repeat this small piece to a large size. Also through these small piece of image, we define the space of our game. Almost all the image resource from the internet are .jpg file or .png file but the images need to be transferred into .mif file so that they can be stored into the chip. The size of image is 75*55 and write some simple MatLab code to set the size of image and change the format of the file.

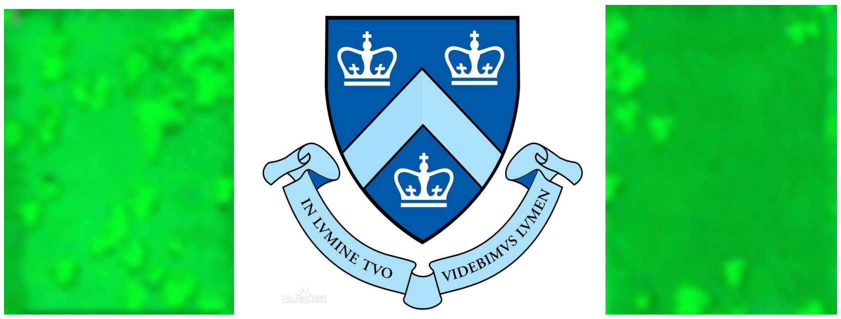

Figure.2 Image for grass land and columbia logo

Attackers: We have designed 3 different kinds of attackers and all of them are designed to be monsters. The size of each monster image is also 75x55, the same as the each piece of grass. One problem is that all the images resources has its own background, such as white or black, but obviouly if all the pixels of the images on the screen are displayed, it will looks ugly and unrealistic. Our method is that we write some Matlab code to process the image so that all the peripheral in the image are assigned to the same color and that means in the software design, we only need to neglect that color. The following is the image used for 3 monsters.

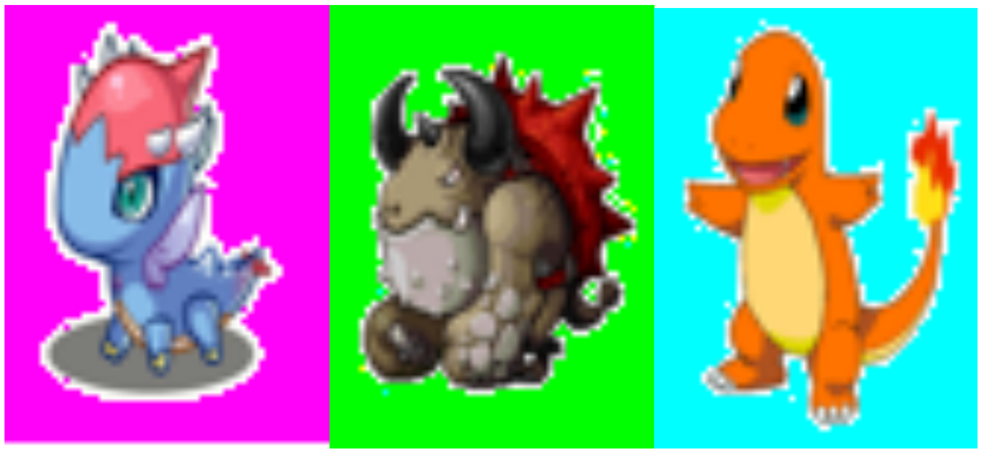
Figure.3 Attackers

Defenders: Images of defenders are similar with images of attackers as Figure.4 shows but they are all human beings and their own attacking technique. The size of each defenders are also 75*55 and peripherals need to be processed.

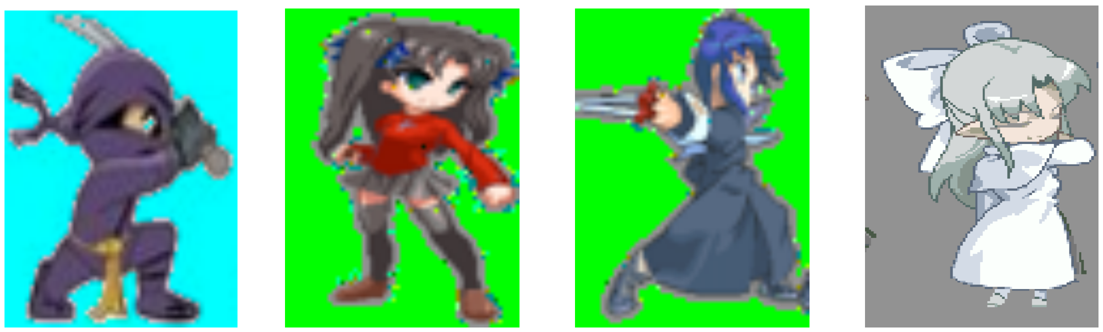
Figure.4 defenders

Bullets: Depending on the different kinds of defenders, four defenders have 3 different kinds of bullets, some of them are single bullet and some of them are a population of bullets, through which the monsters will be attacked and killed as the fourth image in Figure.5 shows. As for the Ninja defender in our game, his sword can only hit the enermy who is close to him and as a result, he doesn't have the effect of bullets. The size of bullets are different from each other and its peripherals still needs to processed.

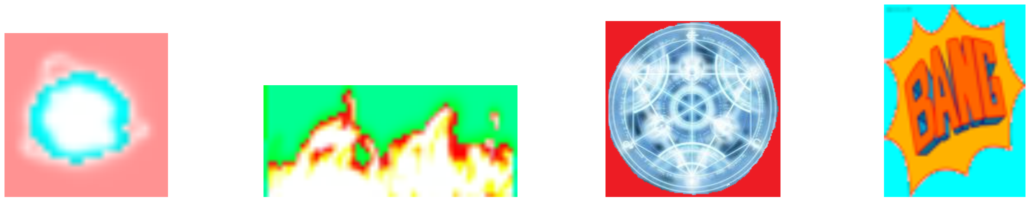

Figure.5 Bullets and Dead Effects

Image Category	File Name	Number of Images	Size
Background	test	1	75*55
	ROM_test3	1	75*55
Columbia logo	cu	1	75*55
Attacker1	mon1	1	75*55
	mon1mov	1	75*55
Attacker2	monster4	1	75*55
	monster4mov	1	75*55
Attacker3	monster5	1	75*55
	monster5	1	75*55
Defender1	chara1	1	75*55

	chara1mov	1	75*55
Defender2	chara3	1	75*55
	chara3mov	1	75*55
Defender3	chara4	1	75*55
	chara4mov	1	75*55
Defender4	chara5	1	75*55
	chara5mov	1	75*55
Bullet1	attack1	1	25*25
Bullet2	fireway	1	55*25
Bullet3	ice	1	25*25
Game Over	gameover	1	400*50
deadeffect	dead	1	55*75

Table.1 All used pictures information

3.2 Initial and End Screen demo

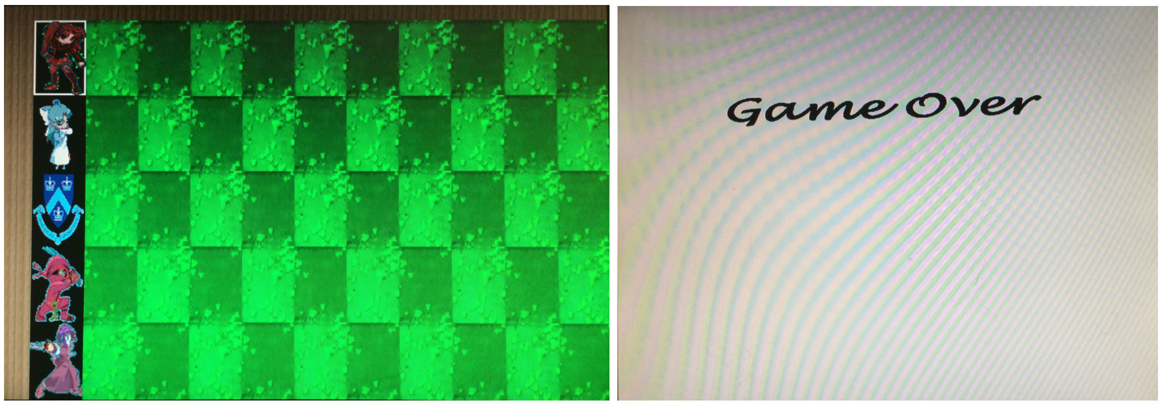

Figure.6 Static Screen demo

4. VGA Display and Sprite Control

In our project ,we are going to show the following elements:

1. Background including the background map of the game, the location we set our defender, the path towards our campus, etc.
2. Four different defenders and three different attackers and each has different appearance.
3. Different attacking, hitting and moving effect for attackers and defenders.
4. The death effect of the monsters and the game over screen.

To realize showing images we prepared on the screen, we will first use Matlab tool to convert general image documents such as jpg,bmp to mif document which contains RGB values for every pixel in the image. Then we use MegaWizard Function to store each RGB value in the ROM. When we are ready to display the image, we just need to read the RGB values from the ROM corresponding to its address.

As for Sprite Control for the images, considering the limited memory, we just use two different actions image for each monster and defender. We use software to transmit the signal to control the conversion of these two images thus we can realize the attack of defenders and the movement of monsters. The following image shows the two images with different actions for one sample defender and one sample monster.

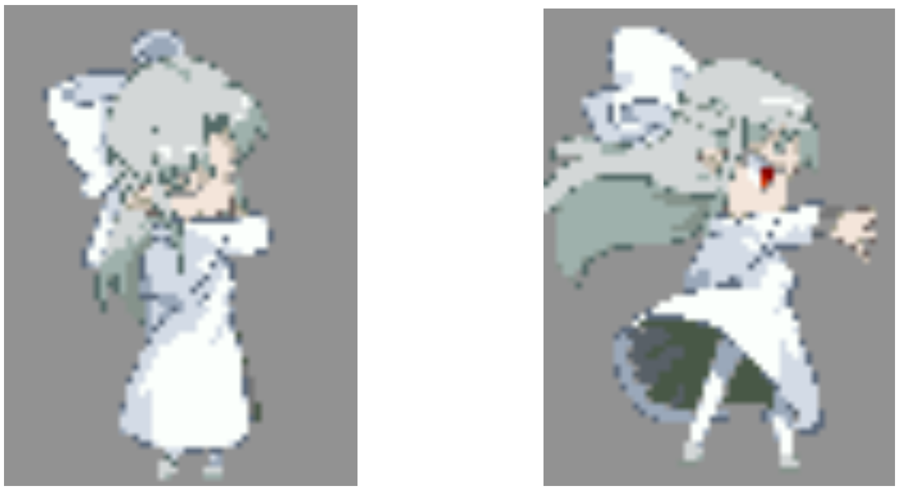

Figure.7 Defenders Sprites

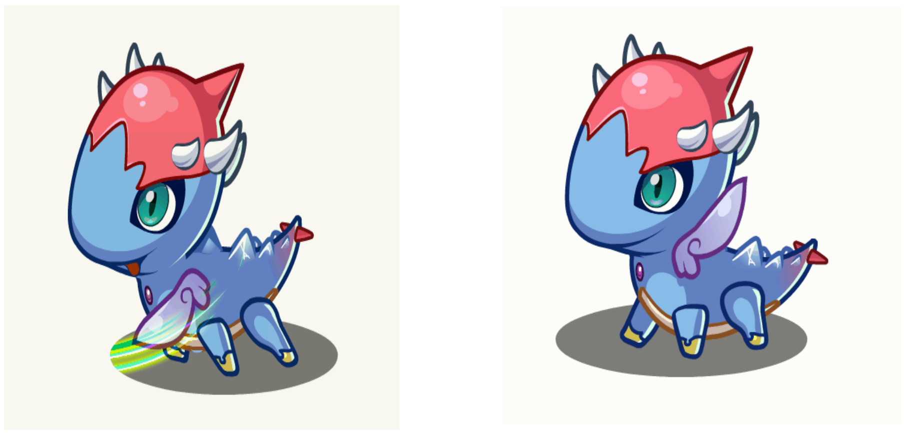
Figure.8 Monsters Sprites

5. AUD Display

The board provides high-quality 24-bit audio via the Analog Devices SSM2603 audio CODEC (Encoder/Decoder). This chip supports microphone-in, line-in, and line-out ports, with a sample rate adjustable from 8 kHz to 96 kHz. The SSM2603 is controlled via a serial I2C bus interface, which is connected to pins on the Cyclone V SoC FPGA. A schematic diagram of the audio circuitry is shown in the following figure.

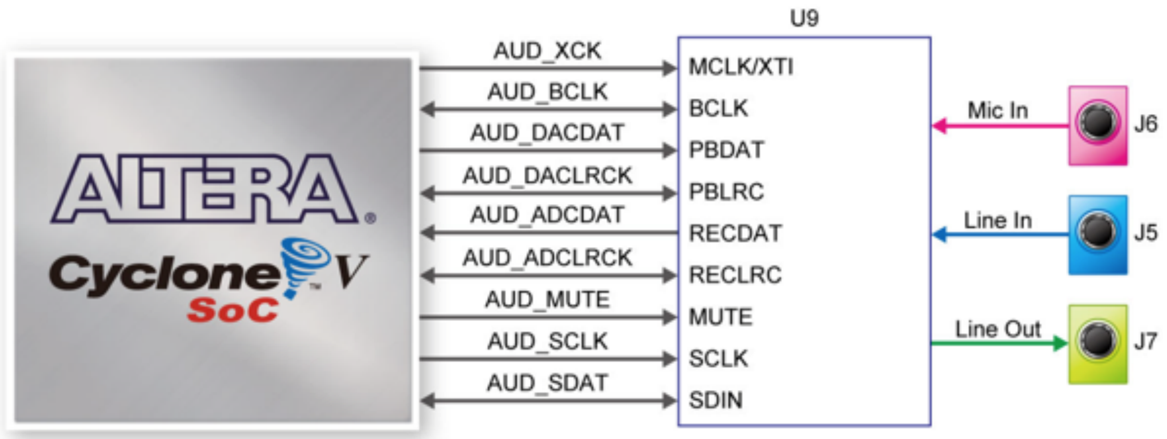

Figure.9 AUD Display

From the datasheet of this codec, we need to set the right configuration of Inter-Integrated Circuit(I^2C) interface at first and know how the codec is connected to the FPGA. The I^2C protocol has two wires which are labeled SDAR and SCLK for serial data and serial clock respectively. We need to keep the SCLK line high and then make SDAT line form high to low at beginning of transmission. After all data bits are tranmitted, the master should send an acknowledgement (ACK) by pulling SDAT low for the cycle. The codec takes 16-bit data words, so we will have 3 acks, which one is after address read or wirht, one after each tata byte.

From the datasheet, we know the maximum clock for SCLK is 526 kHz. So we know how fast I^2C can be toggled.

We also need to figure out how the data send through the configuration interface. The audio codec organized its configuration variable into 19 9-bit registers. And first seven bits are the address of register and last nine bits are the register contents. Each register controls different part of 16-bit data words. So we will set the right configuration controller as we showed in following verilog code called module i2c_av_config.

According to the datasheet, the clock we need to use is 11.2896 Mhz. But we cannot generate the that clock if we only simply dividing master clock. Fortunately, we can add a PLL clock form Megawizard to generte the exactly number of clock we need. We also need to consider how to generate bit clock, which is BCLK. For this clock we can simply divde it on the audio clock from the PLL.

The sound file we use in the project have 20000 btyes. Since we do not have enough ROM memory to store all the sound files. So the only audio file we will be triggered by attacking monsters.

6. PS2 Controller

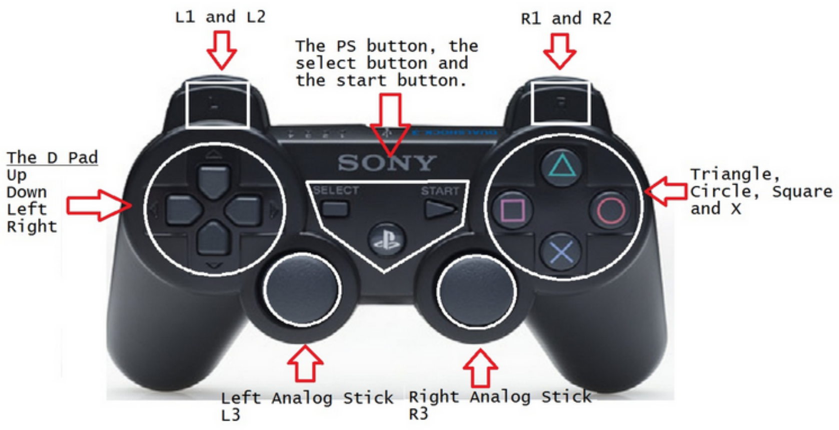

Figure.10 Play Station 2

Since the keyboard is not easy to control our game, the controller we used to play this game is a PlayStation2 as picture show abrove, through which we can get better user experience. We will use all direction parts of buttons on the left side and only ‘O’ button on the right. The ‘O’ button will be used to select defenders and put them on the place we like. By default, we set our choice box at the top-left of our game, one click of direction button means one step.

Once the controller is plugged in, the built-in library will help the board to identify the controller so we don’t need to write other controller driver. If any button, such as direction button or ‘O’ button is pressed and released, it would be captured and updated in time. We only need to use a ‘if’ function to judge whether the event occurs. Once the button event occurs and the signal is generated, just let the code do something in the ’if’ function according to different buttons. Through these buttons, we can move our choice box and choose.

7.Software and Logic Overview

When the game begins, the monster will step in towards the campus according to basic level designed in Software as the Figure.11 shows.
The first part of the Software is to recognize the button press action of the PS2 and realize its corresponding function.
The second part of the Software is to choose the defender and plant it. The select button is the “O” button of the PS2. If one type of defender has been selected, the choosing square will convert its initial white color to blue. For the next step when you move the choosing square to the grassland and press select button one more time. It will plant the defender on the path and let the choosing square back to initial states. Part 1 is showed in Figure.12.

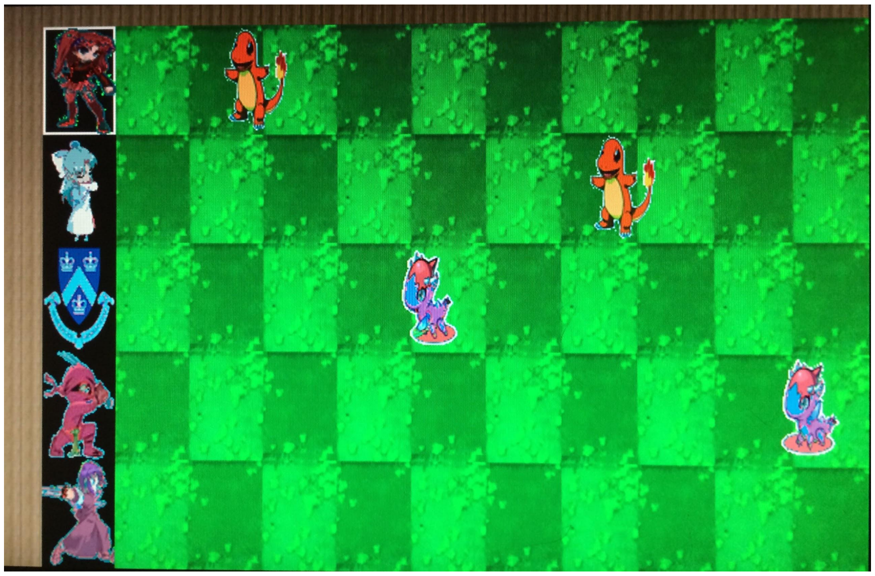
Figure.11 Monster Coming

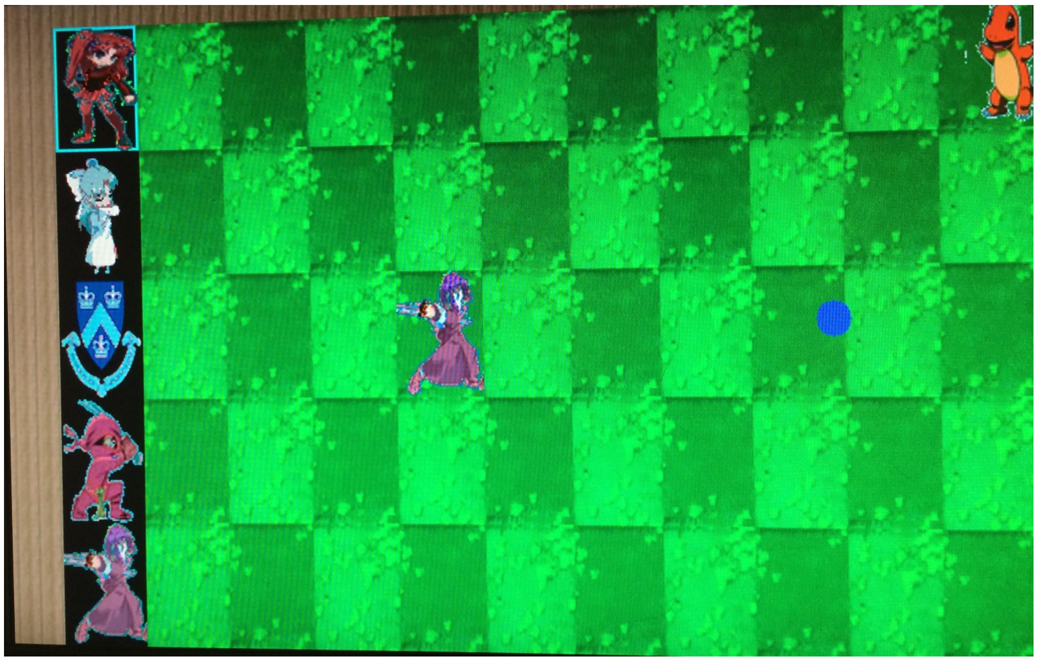
Figure.12 Plant defenders

For the logic of monster in the software, during the period of walking towards campus, when they are blocked by our defenders, they begin to attack and keep on moving after clearing the defenders ahead and all the attackers have the same behavior. In the meantime, there is a partof logic which to find out the most ahead monster in each path as the target of the defenders. However, when they are killed by our defenders, it will explode and explosion effect will last a few time, which can be implemented by setting a counter for an attacker. Then it will rejudge the most ahead monster once there is a monster dead. The dead effect is shown in Figure.13.

For the logic of defender in the software, every defender has its own attack range and attack effect. Once it is built and there is any monster in the same line. It begins to attack corresponding to its attack type. When the monster walk ahead of it, the defender will be attacked and its background become red as Figure.14. The defender has a fixed life amount and if it reduces to 0, it will be “eaten” by the attack and disappear.

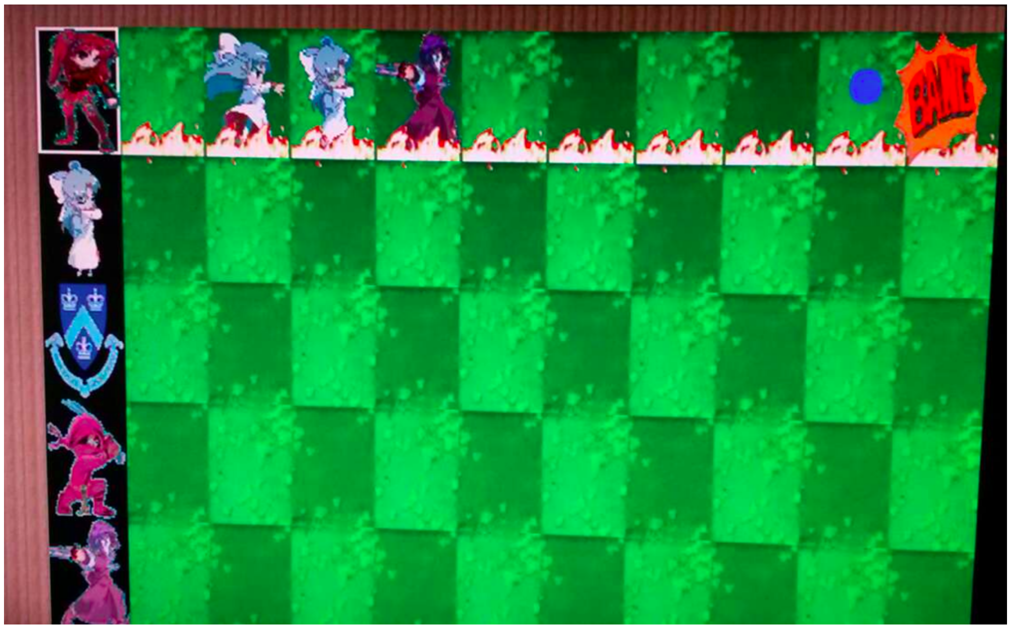

Figure.13. Dead Effect of Monster

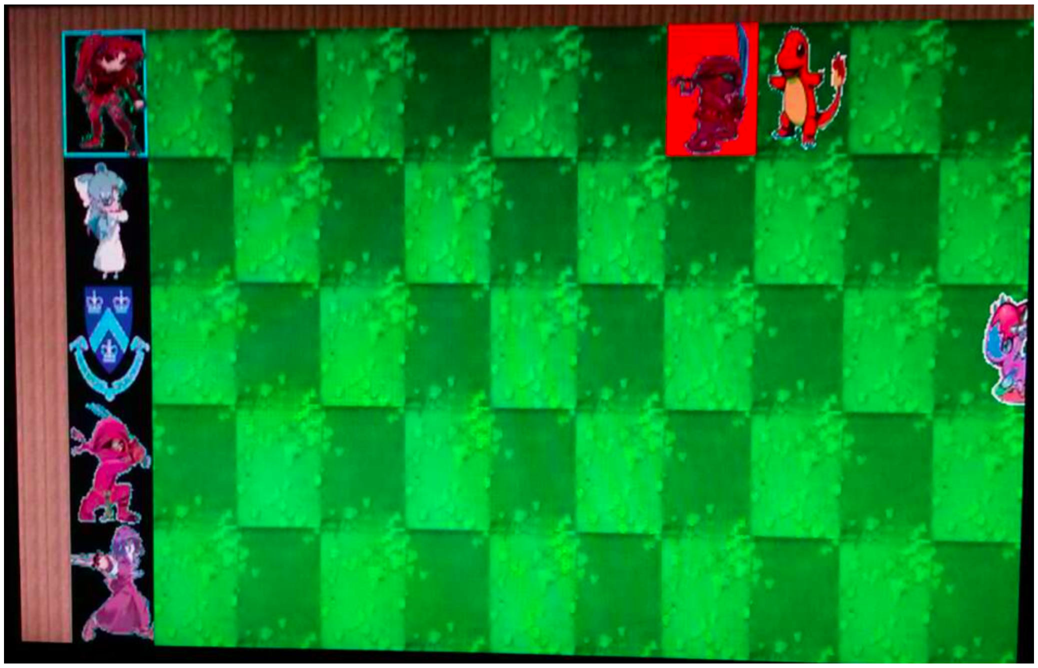

Figure.14. Being attacked

8. Class Learned

To tell the truth, we can hardly progress any a single step. Even though there was so much taught in our class and in Lab1~3, it was still a tough task to start up the huge project for us----how to display graphs on the screen, how to select the proper pictures, how to combine hardware and software, etc----such usually-regarded easy functions became such unattainable we are facing it. It was just when we came to this point did we realize that even a "simple", "tiny" game may cost the designer and the coding engineer much work. We have play so many games but that may never have considered how the games were implemented, never explored the "splendid" world inside the "covered package".

During the project process, we encountered so many challenges. Even though many were under the topics in professor's lectures but it was usually different to actually deal with the problems from learning lecture syllabus.

Specifically, after the project we have gained the abilities on below aspects,

1. Well design a game that may be achievable as well as in nice theme
2. How to write the images to the memory and read it from the memory and display the images from the memory on the screen through VGA
3. Use Sprite to control the dynamics of images
4. Use a USB PlayStation2 controller to control the game and handle the interrupt in the software
5. Add the audio file to the memory and play it through microphone
6. Well control the size and location of the 7-segment on the screen
7. When bugs come up, debug code problems based on the analysis on the problems' characteristics. (The specific techniques are different among various case)
8. Achieve highly effective group cooperation

9. Future Design Improvement

1. Through the DDR3, we want to add more kinds of attackers and defenders so that the user can choose their favourites.
2. Now we only use two static images to simulate the movement of each attacker and the attacking techniques of defenders. In the future, we want to use more images so that the movement of the attackers and defenders can be more vivid.
3. About the sound, we intend to add more sound effects of movement, killing and attacking nad also we wan to put some amazing background music. But since the limit of the memory, we need to put it in the next step.
4. We have some basic implementation of our game bonus system and in the next step, we hope to complete the bonus system.
5. We plan to develop more different levels in our game.
6. About the code, we want to optimize the code in order to get a good computation time.

10. Responsibility

- Zhefeng Xu: Combination of Project; VGA and Sprites control; PS2 controller part;
  Hardware and Software Coding.
- Chao Li: VGA and Sprite control; PS2 controller part; Image Preparation;
  Software Coding
- Tianlei Zhou: Audio Preparation and Audio Display, DDR3 study□Hardware Coding
- Yang Bai: Bonus System Coding, Audio Preparation and Audio Display, DDR3 study, .

Appendix: Project code

Matlab Code to generate mif document and process the peripheral background:

```
clear all; close all;

A = imread('words1.png');
figure(1)
imshow(A);
A1 = A(:,:,1);
A2 = A(:,:,2);
A3 = A(:,:,3);

B1 = fliplr(A1);
B2 = fliplr(A2);
B3 = fliplr(A3);

B(:,:,1) = B1;
B(:,:,2) = B2;
B(:,:,3) = B3;
figure(2)
imshow(B);

% C1 = imresize(A1, [75 55]);

% C2 = imresize(A2, [75 55]);
% C3 = imresize(A3, [75 55]);

C = imresize (A, [30,200]);
figure(3)
imshow(C);


C1 = C(:,:,1);
C2 = C(:,:,2);
C3 = C(:,:,3);


C1(C1 >= 240 & C2 == 252 & C3 >= 240) = 255;
C2(C1 >= 240 & C2 == 252 & C3 >= 240) = 255;
C3(C1 >= 240 & C2 == 252 & C3 >= 240) = 255;
```

```
D(:,:,1) = C1;
D(:,:,2) = C2;
D(:,:,3) = C3;
figure(4)
imshow(D);
imwrite(D, 'words11.png');
miffilegen('words11.png','words11.mif',30,200);
```

Verilog Code:

Section1:Top level code

```
//-----------------------------------------------SoCkit_Top.v----------------------------------------------//
module SoCKit_Top(

			///////////AUD////////////
			AUD_ADCDAT,
			AUD_ADCLRCK,
			AUD_BCLK,
			AUD_DACDAT,
			AUD_DACLRCK,
			AUD_I2C_SCLK,
			AUD_I2C_SDAT,
			AUD_MUTE,
			AUD_XCK,

`ifdef ENABLE_DDR3
			/////////DDR3/////////
			DDR3_A,
			DDR3_BA,
			DDR3_CAS_n,
			DDR3_CKE,
			DDR3_CK_n,
			DDR3_CK_p,
			DDR3_CS_n,
			DDR3_DM,
			DDR3_DQ,
			DDR3_DQS_n,
			DDR3_DQS_p,
			DDR3_ODT,
			DDR3_RAS_n,
			DDR3_RESET_n,
			DDR3_RZQ,
			DDR3_WE_n,
`endif /*ENABLE_DDR3*/

			/////////FAN/////////
			FAN_CTRL,

`ifdef ENABLE_HPS
			/////////HPS/////////
```

```
HPS_CLOCK_25,
HPS_CLOCK_50,
HPS_CONV_USB_n,
HPS_DDR3_A,
HPS_DDR3_BA,
HPS_DDR3_CAS_n,
HPS_DDR3_CKE,
HPS_DDR3_CK_n,
HPS_DDR3_CK_p,
HPS_DDR3_CS_n,
HPS_DDR3_DM,
HPS_DDR3_DQ,
HPS_DDR3_DQS_n,
HPS_DDR3_DQS_p,
HPS_DDR3_ODT,
HPS_DDR3_RAS_n,
HPS_DDR3_RESET_n,
HPS_DDR3_RZQ,
HPS_DDR3_WE_n,
HPS_ENET_GTX_CLK,
HPS_ENET_INT_n,
HPS_ENET_MDC,
HPS_ENET_MDIO,
HPS_ENET_RESET_n,
HPS_ENET_RX_CLK,
HPS_ENET_RX_DATA,
HPS_ENET_RX_DV,
HPS_ENET_TX_DATA,
HPS_ENET_TX_EN,
HPS_FLASH_DATA,
HPS_FLASH_DCLK,
HPS_FLASH_NCSO,
HPS_GSENSOR_INT,
HPS_I2C_CLK,
HPS_I2C_SDA,
HPS_KEY,
HPS_LCM_D_C,
HPS_LCM_RST_N,
HPS_LCM_SPIM_CLK,
HPS_LCM_SPIM_MISO,
                    HPS_LCM_SPIM_MOSI,
                    HPS_LCM_SPIM_SS,
                    HPS_LED,
                    HPS_LTC_GPIO,
                    HPS_RESET_n,
                    HPS_SD_CLK,
                    HPS_SD_CMD,
                    HPS_SD_DATA,
                    HPS_SPIM_CLK,
                    HPS_SPIM_MISO,
                    HPS_SPIM_MOSI,
                    HPS_SPIM_SS,
                    HPS_SW,
                    HPS_UART_RX,
                    HPS_UART_TX,
                    HPS_USB_CLKOUT,
                    HPS_USB_DATA,
                    HPS_USB_DIR,
                    HPS_USB_NXT,
                    HPS_USB_RESET_PHY,
                    HPS_USB_STP,
                    HPS_WARM_RST_n,
`endif /*ENABLE_HPS*/

                    //////////HSMC//////////
                    HSMC_CLKIN_n,
                    HSMC_CLKIN_p,
                    HSMC_CLKOUT_n,
                    HSMC_CLKOUT_p,
                    HSMC_CLK_IN0,
                    HSMC_CLK_OUT0,
                    HSMC_D,

`ifdef ENABLE_HSMC_XCVR

                    HSMC_GXB_RX_p,
                    HSMC_GXB_TX_p,
                    HSMC_REF_CLK_p,
`endif
                    HSMC_RX_n,
                    HSMC_RX_p,
HSMC_SCL,
HSMC_SDA,
HSMC_TX_n,
HSMC_TX_p,

/////////IRDA/////////
IRDA_RXD,

/////////KEY/////////
KEY,

/////////LED/////////
LED,

/////////OSC/////////
OSC_50_B3B,
OSC_50_B4A,
OSC_50_B5B,
OSC_50_B8A,

/////////PCIE/////////
PCIE_PERST_n,
PCIE_WAKE_n,

/////////RESET/////////
RESET_n,

/////////SI5338/////////
SI5338_SCL,
SI5338_SDA,

/////////SW/////////
SW,

/////////TEMP/////////
TEMP_CS_n,
TEMP_DIN,
TEMP_DOUT,
TEMP_SCLK,

/////////USB/////////
```

```
USB_B2_CLK,
USB_B2_DATA,
USB_EMPTY,
USB_FULL,
USB_OE_n,
USB_RD_n,
USB_RESET_n,
USB_SCL,
USB_SDA,
USB_WR_n,

//////////VGA/////////
VGA_B,
VGA_BLANK_n,
VGA_CLK,
VGA_G,
VGA_HS,
VGA_R,
VGA_SYNC_n,
VGA_VS,
///////////hps//////////
memory_mem_a,
memory_mem_ba,
memory_mem_ck,
memory_mem_ck_n,
memory_mem_cke,
memory_mem_cs_n,
memory_mem_ras_n,
memory_mem_cas_n,
memory_mem_we_n,
memory_mem_reset_n,
memory_mem_dq,
memory_mem_dqs,
memory_mem_dqs_n,
memory_mem_odt,
memory_mem_dm,
memory_oct_rzqin,
hps_io_hps_io_emac1_inst_TX_CLK,
hps_io_hps_io_emac1_inst_TXD0,
hps_io_hps_io_emac1_inst_TXD1,
hps_io_hps_io_emac1_inst_TXD2,
hps_io_hps_io_emac1_inst_TXD3,
hps_io_hps_io_emac1_inst_RXD0,
hps_io_hps_io_emac1_inst_MDIO,
hps_io_hps_io_emac1_inst_MDC,
hps_io_hps_io_emac1_inst_RX_CTL,
hps_io_hps_io_emac1_inst_TX_CTL,
hps_io_hps_io_emac1_inst_RX_CLK,
hps_io_hps_io_emac1_inst_RXD1,
hps_io_hps_io_emac1_inst_RXD2,
hps_io_hps_io_emac1_inst_RXD3,
hps_io_hps_io_qspi_inst_IO0,
hps_io_hps_io_qspi_inst_IO1,
hps_io_hps_io_qspi_inst_IO2,
hps_io_hps_io_qspi_inst_IO3,
hps_io_hps_io_qspi_inst_SS0,
hps_io_hps_io_qspi_inst_CLK,
hps_io_hps_io_sdio_inst_CMD,
hps_io_hps_io_sdio_inst_D0,
hps_io_hps_io_sdio_inst_D1,
hps_io_hps_io_sdio_inst_CLK,
hps_io_hps_io_sdio_inst_D2,
hps_io_hps_io_sdio_inst_D3,
hps_io_hps_io_usb1_inst_D0,
hps_io_hps_io_usb1_inst_D1,
hps_io_hps_io_usb1_inst_D2,
hps_io_hps_io_usb1_inst_D3,
hps_io_hps_io_usb1_inst_D4,
hps_io_hps_io_usb1_inst_D5,
hps_io_hps_io_usb1_inst_D6,
hps_io_hps_io_usb1_inst_D7,
hps_io_hps_io_usb1_inst_CLK,
hps_io_hps_io_usb1_inst_STP,
hps_io_hps_io_usb1_inst_DIR,
hps_io_hps_io_usb1_inst_NXT,
hps_io_hps_io_spim0_inst_CLK,
hps_io_hps_io_spim0_inst_MOSI,
hps_io_hps_io_spim0_inst_MISO,
hps_io_hps_io_spim0_inst_SS0,
hps_io_hps_io_spim1_inst_CLK,
hps_io_hps_io_spim1_inst_MOSI,
                hps_io_hps_io_spim1_inst_MISO,
                hps_io_hps_io_spim1_inst_SS0,
                hps_io_hps_io_uart0_inst_RX,
                hps_io_hps_io_uart0_inst_TX,
                hps_io_hps_io_i2c1_inst_SDA,
                hps_io_hps_io_i2c1_inst_SCL,
                hps_io_hps_io_gpio_inst_GPIO00
                );

  //=======================================================
  //  PORT declarations
  //=======================================================

  ///////// AUD /////////
  input                       AUD_ADCDAT;
  inout                       AUD_ADCLRCK;
  inout                       AUD_BCLK;
  output                      AUD_DACDAT;
  inout                       AUD_DACLRCK;
  output                      AUD_I2C_SCLK;
  inout                       AUD_I2C_SDAT;
  output                      AUD_MUTE;
  output                      AUD_XCK;

`ifdef ENABLE_DDR3
  ///////// DDR3 /////////
  output [14:0]                        DDR3_A;
  output [2:0]                    DDR3_BA;
  output                      DDR3_CAS_n;
  output                      DDR3_CKE;
  output                      DDR3_CK_n;
  output                      DDR3_CK_p;
  output                      DDR3_CS_n;
  output [3:0]                    DDR3_DM;
  inout [31:0]                    DDR3_DQ;
  inout [3:0]                          DDR3_DQS_n;
  inout [3:0]                          DDR3_DQS_p;
  output                      DDR3_ODT;
  output                      DDR3_RAS_n;
  output                      DDR3_RESET_n;
  input                       DDR3_RZQ;
  output                          DDR3_WE_n;
`endif /*ENABLE_DDR3*/

  ///////// FAN /////////
  output                          FAN_CTRL;

`ifdef ENABLE_HPS
  ///////// HPS /////////
  input                          HPS_CLOCK_25;
  input                          HPS_CLOCK_50;
  input                          HPS_CONV_USB_n;
  output [14:0]                             HPS_DDR3_A;
  output [2:0]                        HPS_DDR3_BA;
  output                          HPS_DDR3_CAS_n;
  output                          HPS_DDR3_CKE;
  output                          HPS_DDR3_CK_n;
  output                          HPS_DDR3_CK_p;
  output                          HPS_DDR3_CS_n;
  output [3:0]                        HPS_DDR3_DM;
  inout [31:0]                        HPS_DDR3_DQ;
  inout [3:0]                               HPS_DDR3_DQS_n;
  inout [3:0]                               HPS_DDR3_DQS_p;
  output                          HPS_DDR3_ODT;
  output                          HPS_DDR3_RAS_n;
  output                          HPS_DDR3_RESET_n;
  input                          HPS_DDR3_RZQ;
  output                          HPS_DDR3_WE_n;
  input                          HPS_ENET_GTX_CLK;
  input                          HPS_ENET_INT_n;
  output                          HPS_ENET_MDC;
  inout                          HPS_ENET_MDIO;
  output                          HPS_ENET_RESET_n;
  input                          HPS_ENET_RX_CLK;
  input [3:0]                               HPS_ENET_RX_DATA;
  input                          HPS_ENET_RX_DV;
  output [3:0]                        HPS_ENET_TX_DATA;
  output                          HPS_ENET_TX_EN;
  inout [3:0]                               HPS_FLASH_DATA;
  output                          HPS_FLASH_DCLK;
  output                          HPS_FLASH_NCSO;
  input                          HPS_GSENSOR_INT;
  inout                          HPS_I2C_CLK;
  inout                          HPS_I2C_SDA;
  inout [3:0]                              HPS_KEY;
  output                          HPS_LCM_D_C;
  output                          HPS_LCM_RST_N;
  input                          HPS_LCM_SPIM_CLK;
  inout                          HPS_LCM_SPIM_MISO;
  output                          HPS_LCM_SPIM_MOSI;
  output                          HPS_LCM_SPIM_SS;
  output [3:0]                        HPS_LED;
  inout                          HPS_LTC_GPIO;
  input                          HPS_RESET_n;
  output                          HPS_SD_CLK;
  inout                          HPS_SD_CMD;
  inout [3:0]                              HPS_SD_DATA;
  output                          HPS_SPIM_CLK;
  input                          HPS_SPIM_MISO;
  output                          HPS_SPIM_MOSI;
  output                          HPS_SPIM_SS;
  input [3:0]                              HPS_SW;
  input                          HPS_UART_RX;
  output                          HPS_UART_TX;
  input                          HPS_USB_CLKOUT;
  inout [7:0]                              HPS_USB_DATA;
  input                          HPS_USB_DIR;
  input                          HPS_USB_NXT;
  output                          HPS_USB_RESET_PHY;
  output                          HPS_USB_STP;
  input                          HPS_WARM_RST_n;
`endif /*ENABLE_HPS*/

  ///////// HSMC /////////
  input [2:1]                              HSMC_CLKIN_n;
  input [2:1]                              HSMC_CLKIN_p;
  output [2:1]                        HSMC_CLKOUT_n;
  output [2:1]                        HSMC_CLKOUT_p;
  input                          HSMC_CLK_IN0;
  output                          HSMC_CLK_OUT0;
  inout [3:0]                              HSMC_D;
`ifdef ENABLE_HSMC_XCVR
  input [7:0]                              HSMC_GXB_RX_p;
  output [7:0]                    HSMC_GXB_TX_p;
  input                       HSMC_REF_CLK_p;
`endif
  inout [16:0]                    HSMC_RX_n;
  inout [16:0]                    HSMC_RX_p;
  output                       HSMC_SCL;
  inout                       HSMC_SDA;
  inout [16:0]                    HSMC_TX_n;
  inout [16:0]                    HSMC_TX_p;

  ///////// IRDA /////////
  input                       IRDA_RXD;

  ///////// KEY /////////
  input [3:0]                           KEY;

  ///////// LED /////////
  output [3:0]                    LED;

  ///////// OSC /////////
  input                       OSC_50_B3B;
  input                       OSC_50_B4A;
  input                       OSC_50_B5B;
  input                       OSC_50_B8A;

  ///////// PCIE /////////
  input                       PCIE_PERST_n;
  input                       PCIE_WAKE_n;

  ///////// RESET /////////
  input                       RESET_n;

  ///////// SI5338 /////////
  inout                       SI5338_SCL;
  inout                       SI5338_SDA;

  ///////// SW /////////
  input [3:0]                           SW;

  ///////// TEMP /////////
  output                       TEMP_CS_n;
output                          TEMP_DIN;
input                          TEMP_DOUT;
output                          TEMP_SCLK;

///////// USB /////////
input                          USB_B2_CLK;
inout [7:0]                                USB_B2_DATA;
output                          USB_EMPTY;
output                          USB_FULL;
input                          USB_OE_n;
input                          USB_RD_n;
input                          USB_RESET_n;
inout                          USB_SCL;
inout                          USB_SDA;
input                          USB_WR_n;

///////// VGA /////////
output [7:0]                        VGA_B;
output                          VGA_BLANK_n;
output                          VGA_CLK;
output [7:0]                        VGA_G;
output                          VGA_HS;
output [7:0]                        VGA_R;
output                          VGA_SYNC_n;
output                          VGA_VS;

/////////hps pin///////
output wire [14:0]                        memory_mem_a;
output wire [2:0]                         memory_mem_ba;
output wire                               memory_mem_ck;
output wire                               memory_mem_ck_n;
output wire                               memory_mem_cke;
output wire                               memory_mem_cs_n;
output wire                               memory_mem_ras_n;
output wire                               memory_mem_cas_n;
output wire                               memory_mem_we_n;
output wire                               memory_mem_reset_n;
inout  wire [31:0]                        memory_mem_dq;
inout  wire [3:0]                         memory_mem_dqs;
inout  wire [3:0]                         memory_mem_dqs_n;
output wire                               memory_mem_odt;
output wire [3:0]                memory_mem_dm;
input  wire                      memory_oct_rzqin;
output wire                      hps_io_hps_io_emac1_inst_TX_CLK;
output wire                      hps_io_hps_io_emac1_inst_TXD0;
output wire                      hps_io_hps_io_emac1_inst_TXD1;
output wire                      hps_io_hps_io_emac1_inst_TXD2;
output wire                      hps_io_hps_io_emac1_inst_TXD3;
input  wire                      hps_io_hps_io_emac1_inst_RXD0;
inout  wire                      hps_io_hps_io_emac1_inst_MDIO;
output wire                      hps_io_hps_io_emac1_inst_MDC;
input  wire                      hps_io_hps_io_emac1_inst_RX_CTL;
output wire                      hps_io_hps_io_emac1_inst_TX_CTL;
input  wire                      hps_io_hps_io_emac1_inst_RX_CLK;
input  wire                      hps_io_hps_io_emac1_inst_RXD1;
input  wire                      hps_io_hps_io_emac1_inst_RXD2;
input  wire                      hps_io_hps_io_emac1_inst_RXD3;
inout  wire                      hps_io_hps_io_qspi_inst_IO0;
inout  wire                      hps_io_hps_io_qspi_inst_IO1;
inout  wire                      hps_io_hps_io_qspi_inst_IO2;
inout  wire                      hps_io_hps_io_qspi_inst_IO3;
output wire                      hps_io_hps_io_qspi_inst_SS0;
output wire                      hps_io_hps_io_qspi_inst_CLK;
inout  wire                      hps_io_hps_io_sdio_inst_CMD;
inout  wire                      hps_io_hps_io_sdio_inst_D0;
inout  wire                      hps_io_hps_io_sdio_inst_D1;
output wire                      hps_io_hps_io_sdio_inst_CLK;
inout  wire                      hps_io_hps_io_sdio_inst_D2;
inout  wire                      hps_io_hps_io_sdio_inst_D3;
inout  wire                      hps_io_hps_io_usb1_inst_D0;
inout  wire                      hps_io_hps_io_usb1_inst_D1;
inout  wire                      hps_io_hps_io_usb1_inst_D2;
inout  wire                      hps_io_hps_io_usb1_inst_D3;
inout  wire                      hps_io_hps_io_usb1_inst_D4;
inout  wire                      hps_io_hps_io_usb1_inst_D5;
inout  wire                      hps_io_hps_io_usb1_inst_D6;
inout  wire                      hps_io_hps_io_usb1_inst_D7;
input  wire                      hps_io_hps_io_usb1_inst_CLK;
output wire                      hps_io_hps_io_usb1_inst_STP;
input  wire                      hps_io_hps_io_usb1_inst_DIR;
input  wire                      hps_io_hps_io_usb1_inst_NXT;
```

```
  output wire                                   hps_io_hps_io_spim0_inst_CLK;
  output wire                                   hps_io_hps_io_spim0_inst_MOSI;
  input  wire                                   hps_io_hps_io_spim0_inst_MISO;
  output wire                                   hps_io_hps_io_spim0_inst_SS0;
  output wire                                   hps_io_hps_io_spim1_inst_CLK;
  output wire                                   hps_io_hps_io_spim1_inst_MOSI;
  input  wire                                   hps_io_hps_io_spim1_inst_MISO;
  output wire                                   hps_io_hps_io_spim1_inst_SS0;
  input  wire                                   hps_io_hps_io_uart0_inst_RX;
  output wire                                   hps_io_hps_io_uart0_inst_TX;
  inout  wire                                   hps_io_hps_io_i2c1_inst_SDA;
  inout  wire                                   hps_io_hps_io_i2c1_inst_SCL;
  inout  wire                                   hps_io_hps_io_gpio_inst_GPIO00;
  //=======================================================
  //  REG/WIRE declarations
  //=======================================================

        wire    [15:0] VGA_audio1;
        wire    [15:0] VGA_audio2;

 audio_top  Audio1 (
  .OSC_50_B8A  (OSC_50_B4A),
  .AUD_ADCLRCK (AUD_ADCLRCK),
  .AUD_ADCDAT (AUD_ADCDAT),
  .AUD_DACLRCK (AUD_DACLRCK),
  .AUD_DACDAT (AUD_DACDAT),
  .AUD_XCK (AUD_XCK),
  .AUD_BCLK (AUD_BCLK),
  .AUD_I2C_SCLK (AUD_I2C_SCLK),
  .AUD_I2C_SDAT (AUD_I2C_SDAT),
  .AUD_MUTE (AUD_MUTE),

  .KEY (KEY),
  .SW (SW),
        .flag_audio1(VGA_audio1),
        .flag_audio2(VGA_audio2)
);

        //Audio hex
  wire                              AUD_CTRL_CLK;       //        For Audio Controller
```

```
reg [31:0]                      Cont;
wire                        VGA_CTRL_CLK;
wire [9:0]                      mVGA_R;
wire [9:0]                      mVGA_G;
wire [9:0]                      mVGA_B;
wire [19:0]                     mVGA_ADDR;
wire                        DLY_RST;


//      For VGA Controller
wire                        mVGA_CLK;
wire [9:0]                      mRed;
wire [9:0]                      mGreen;
wire [9:0]                      mBlue;
wire                        VGA_Read;       //      VGA data request

wire [9:0]                      recon_VGA_R;
wire [9:0]                      recon_VGA_G;
wire [9:0]                      recon_VGA_B;

//      For Down Sample
wire [3:0]                      Remain;
wire [9:0]                      Quotient;

wire                        AUD_MUTE;

// Drive the LEDs with the switches
assign LED = SW;

// Make the FPGA reset cause an HPS reset
reg [19:0]                      hps_reset_counter = 20'h0;
reg                             hps_fpga_reset_n = 0;

always @(posedge OSC_50_B4A) begin
  if (hps_reset_counter == 20'h ffffff) hps_fpga_reset_n <= 1;
  hps_reset_counter <= hps_reset_counter + 1;
end

lab3 u0 (
      .clk_clk                  (OSC_50_B4A),           //          clk.clk
      .reset_reset_n              (hps_fpga_reset_n),       //          reset.reset_n
```

```
        .memory_mem_a                     (memory_mem_a),                  //
memory.mem_a
           .memory_mem_ba                    (memory_mem_ba),                //            .mem_ba
           .memory_mem_ck                    (memory_mem_ck),                //            .mem_ck
           .memory_mem_ck_n                  (memory_mem_ck_n),              //
.mem_ck_n
           .memory_mem_cke                   (memory_mem_cke),               //            .mem_cke
           .memory_mem_cs_n                  (memory_mem_cs_n),              //
.mem_cs_n
           .memory_mem_ras_n                 (memory_mem_ras_n),             //
.mem_ras_n
           .memory_mem_cas_n                 (memory_mem_cas_n),             //
.mem_cas_n
           .memory_mem_we_n                  (memory_mem_we_n),              //
.mem_we_n
           .memory_mem_reset_n               (memory_mem_reset_n),           //
.mem_reset_n
           .memory_mem_dq                    (memory_mem_dq),                //            .mem_dq
           .memory_mem_dqs                   (memory_mem_dqs),               //            .mem_dqs
           .memory_mem_dqs_n                 (memory_mem_dqs_n),             //
.mem_dqs_n
           .memory_mem_odt                   (memory_mem_odt),               //            .mem_odt
           .memory_mem_dm                    (memory_mem_dm),                //            .mem_dm
           .memory_oct_rzqin                 (memory_oct_rzqin),             //            .oct_rzqin
           .hps_io_hps_io_emac1_inst_TX_CLK
(hps_io_hps_io_emac1_inst_TX_CLK), //
.hps_0_hps_io.hps_io_emac1_inst_TX_CLK
           .hps_io_hps_io_emac1_inst_TXD0              (hps_io_hps_io_emac1_inst_TXD0),   //
.hps_io_emac1_inst_TXD0
           .hps_io_hps_io_emac1_inst_TXD1              (hps_io_hps_io_emac1_inst_TXD1),   //
.hps_io_emac1_inst_TXD1
           .hps_io_hps_io_emac1_inst_TXD2              (hps_io_hps_io_emac1_inst_TXD2),   //
.hps_io_emac1_inst_TXD2
           .hps_io_hps_io_emac1_inst_TXD3              (hps_io_hps_io_emac1_inst_TXD3),   //
.hps_io_emac1_inst_TXD3
           .hps_io_hps_io_emac1_inst_RXD0              (hps_io_hps_io_emac1_inst_RXD0),   //
.hps_io_emac1_inst_RXD0
           .hps_io_hps_io_emac1_inst_MDIO
(hps_io_hps_io_emac1_inst_MDIO),   //            .hps_io_emac1_inst_MDIO
```

```
        .hps_io_hps_io_emac1_inst_MDC           (hps_io_hps_io_emac1_inst_MDC),   //
.hps_io_emac1_inst_MDC
        .hps_io_hps_io_emac1_inst_RX_CTL
(hps_io_hps_io_emac1_inst_RX_CTL), //           .hps_io_emac1_inst_RX_CTL
        .hps_io_hps_io_emac1_inst_TX_CTL
(hps_io_hps_io_emac1_inst_TX_CTL), //           .hps_io_emac1_inst_TX_CTL
        .hps_io_hps_io_emac1_inst_RX_CLK
(hps_io_hps_io_emac1_inst_RX_CLK), //           .hps_io_emac1_inst_RX_CLK
        .hps_io_hps_io_emac1_inst_RXD1          (hps_io_hps_io_emac1_inst_RXD1),   //
.hps_io_emac1_inst_RXD1
        .hps_io_hps_io_emac1_inst_RXD2          (hps_io_hps_io_emac1_inst_RXD2),   //
.hps_io_emac1_inst_RXD2
        .hps_io_hps_io_emac1_inst_RXD3          (hps_io_hps_io_emac1_inst_RXD3),   //
.hps_io_emac1_inst_RXD3
        .hps_io_hps_io_qspi_inst_IO0            (hps_io_hps_io_qspi_inst_IO0),     //
.hps_io_qspi_inst_IO0
        .hps_io_hps_io_qspi_inst_IO1            (hps_io_hps_io_qspi_inst_IO1),     //
.hps_io_qspi_inst_IO1
        .hps_io_hps_io_qspi_inst_IO2            (hps_io_hps_io_qspi_inst_IO2),     //
.hps_io_qspi_inst_IO2
        .hps_io_hps_io_qspi_inst_IO3            (hps_io_hps_io_qspi_inst_IO3),     //
.hps_io_qspi_inst_IO3
        .hps_io_hps_io_qspi_inst_SS0            (hps_io_hps_io_qspi_inst_SS0),     //
.hps_io_qspi_inst_SS0
        .hps_io_hps_io_qspi_inst_CLK            (hps_io_hps_io_qspi_inst_CLK),     //
.hps_io_qspi_inst_CLK
        .hps_io_hps_io_sdio_inst_CMD            (hps_io_hps_io_sdio_inst_CMD),     //
.hps_io_sdio_inst_CMD
        .hps_io_hps_io_sdio_inst_D0             (hps_io_hps_io_sdio_inst_D0),      //
.hps_io_sdio_inst_D0
        .hps_io_hps_io_sdio_inst_D1             (hps_io_hps_io_sdio_inst_D1),      //
.hps_io_sdio_inst_D1
        .hps_io_hps_io_sdio_inst_CLK            (hps_io_hps_io_sdio_inst_CLK),     //
.hps_io_sdio_inst_CLK
        .hps_io_hps_io_sdio_inst_D2             (hps_io_hps_io_sdio_inst_D2),      //
.hps_io_sdio_inst_D2
        .hps_io_hps_io_sdio_inst_D3             (hps_io_hps_io_sdio_inst_D3),      //
.hps_io_sdio_inst_D3
        .hps_io_hps_io_usb1_inst_D0             (hps_io_hps_io_usb1_inst_D0),      //
.hps_io_usb1_inst_D0
```

```
		.hps_io_hps_io_usb1_inst_D1	(hps_io_hps_io_usb1_inst_D1),	//
.hps_io_usb1_inst_D1
		.hps_io_hps_io_usb1_inst_D2	(hps_io_hps_io_usb1_inst_D2),	//
.hps_io_usb1_inst_D2
		.hps_io_hps_io_usb1_inst_D3	(hps_io_hps_io_usb1_inst_D3),	//
.hps_io_usb1_inst_D3
		.hps_io_hps_io_usb1_inst_D4	(hps_io_hps_io_usb1_inst_D4),	//
.hps_io_usb1_inst_D4
		.hps_io_hps_io_usb1_inst_D5	(hps_io_hps_io_usb1_inst_D5),	//
.hps_io_usb1_inst_D5
		.hps_io_hps_io_usb1_inst_D6	(hps_io_hps_io_usb1_inst_D6),	//
.hps_io_usb1_inst_D6
		.hps_io_hps_io_usb1_inst_D7	(hps_io_hps_io_usb1_inst_D7),	//
.hps_io_usb1_inst_D7
		.hps_io_hps_io_usb1_inst_CLK	(hps_io_hps_io_usb1_inst_CLK),	//
.hps_io_usb1_inst_CLK
		.hps_io_hps_io_usb1_inst_STP	(hps_io_hps_io_usb1_inst_STP),	//
.hps_io_usb1_inst_STP
		.hps_io_hps_io_usb1_inst_DIR	(hps_io_hps_io_usb1_inst_DIR),	//
.hps_io_usb1_inst_DIR
		.hps_io_hps_io_usb1_inst_NXT	(hps_io_hps_io_usb1_inst_NXT),	//
.hps_io_usb1_inst_NXT
		.hps_io_hps_io_spim0_inst_CLK	(hps_io_hps_io_spim0_inst_CLK),	//
.hps_io_spim0_inst_CLK
		.hps_io_hps_io_spim0_inst_MOSI	(hps_io_hps_io_spim0_inst_MOSI),	//
.hps_io_spim0_inst_MOSI
		.hps_io_hps_io_spim0_inst_MISO	(hps_io_hps_io_spim0_inst_MISO),	//
.hps_io_spim0_inst_MISO
		.hps_io_hps_io_spim0_inst_SS0	(hps_io_hps_io_spim0_inst_SS0),	//
.hps_io_spim0_inst_SS0
		.hps_io_hps_io_spim1_inst_CLK	(hps_io_hps_io_spim1_inst_CLK),	//
.hps_io_spim1_inst_CLK
		.hps_io_hps_io_spim1_inst_MOSI	(hps_io_hps_io_spim1_inst_MOSI),	//
.hps_io_spim1_inst_MOSI
		.hps_io_hps_io_spim1_inst_MISO	(hps_io_hps_io_spim1_inst_MISO),	//
.hps_io_spim1_inst_MISO
		.hps_io_hps_io_spim1_inst_SS0	(hps_io_hps_io_spim1_inst_SS0),	//
.hps_io_spim1_inst_SS0
		.hps_io_hps_io_uart0_inst_RX	(hps_io_hps_io_uart0_inst_RX),	//
.hps_io_uart0_inst_RX
```

```
          .hps_io_hps_io_uart0_inst_TX                    (hps_io_hps_io_uart0_inst_TX),     //
.hps_io_uart0_inst_TX
          .hps_io_hps_io_i2c1_inst_SDA                    (hps_io_hps_io_i2c1_inst_SDA),     //
.hps_io_i2c1_inst_SDA
          .hps_io_hps_io_i2c1_inst_SCL                    (hps_io_hps_io_i2c1_inst_SCL) ,    //
.hps_io_i2c1_inst_SCL
.vga_R (VGA_R),
.vga_G (VGA_G),
.vga_B (VGA_B),
.vga_CLK (VGA_CLK),
.vga_HS (VGA_HS),
.vga_VS (VGA_VS),
.vga_BLANK_n (VGA_BLANK_n),
.vga_SYNC_n (VGA_SYNC_n),
.vga_audio1(VGA_audio1),
.vga_audio2(VGA_audio2)
            );
endmodule
```

Section2: VGA part

2.1 Code of VGA_LED.sv

```
/**********************code of VGA_LED.sv********************************/

module VGA_LED(input logic        clk,
                input logic            reset,
                input logic [15:0]  writedata,
                input logic            write,
                input                  chipselect,
               input logic [7:0] address,
               output logic [15:0] VGA_audio1,VGA_audio2,
                output logic [7:0] VGA_R, VGA_G, VGA_B,
                output logic           VGA_CLK, VGA_HS, VGA_VS, VGA_BLANK_n,
                output logic           VGA_SYNC_n);

          logic [15:0] x,y;
          logic [15:0] a1,b1;
          logic [15:0] a2,b2,a3,b3;
          logic [15:0] run,hex1,hex2,hex3;
```

```
 logic [15:0] x11,y11,x12,y12,x13,y13,x14,y14;
 logic [15:0] z11,d11,z12,d12,z13,d13,z14,d14;
 logic [15:0] m1,m2,m3,m4;
 logic [15:0] x21,y21,x22,y22,x23,y23,x24,y24,x25,y25;
 logic [15:0] z21,d21,z22,d22,z23,d23,z24,d24,z25,d25;
 logic [15:0] m21,m22,m23,m24,m25;
 logic [15:0] x31,y31,x32,y32,x33,y33,x34,y34,x35,y35;
 logic [15:0] z31,d31,z32,d32,z33,d33,z34,d34,z35,d35;
 logic [15:0] m31,m32,m33,m34,m35;
 logic [15:0] x41,y41,x42,y42,x43,y43,x44,y44,x45,y45;
 logic [15:0] z41,d41,z42,d42,z43,d43,z44,d44,z45,d45;
 logic [15:0] m41,m42,m43,m44,m45;

logic [15:0] x51,y51,x52,y52,x53,y53,x54,y54,x55,y55;
 logic [15:0] z51,d51,z52,d52,z53,d53,z54,d54,z55,d55;
 logic [15:0] m51,m52,m53,m54,m55;
 logic [15:0] g1,g2,g3,g4,g5;
 logic [15:0] g11,g12,g13,g14;
 logic [15:0] g21,g22,g23,g24;
 logic [15:0] g31,g32,g33,g34;
 logic [15:0] g41,g42,g43,g44;
 logic [15:0] g51,g52,g53,g54;
 logic [15:0] p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13,p14,p15,p16,p17,p18,p19,p20;
 logic [15:0] q1,q2,q3,q4,q5,q6,q7,q8,q9,q10,q11,q12,q13,q14,q15,q16,q17,q18,q19,q20;
 logic [15:0] music1,music2;
 logic [15:0] start,over;

 //pixel coordinates
 wire [10:0] hcount;
 wire [9:0] vcount;
 wire [12:0] addr_bg;
 wire [23:0] M_bg;
 wire [23:0] M_1;
 wire [12:0] addr_1;
 wire [23:0] M_cu;
 wire [12:0] addr_cu;
 wire [23:0] M_ch1;
 wire [12:0] addr_ch1;
 wire [23:0] M_ch1mov;
 wire [12:0] addr_ch1mov;
 wire [23:0] M_mon1;
 wire [12:0] addr_mon1;
 wire [23:0] M_bg2;
 wire [7:0] addr_bg2;
 wire [23:0] M_col;
 wire [15:0] addr_col;
 wire [23:0] M_ch3;
 wire [12:0] addr_ch3;
 wire [23:0] M_ch3mov;
 wire [12:0] addr_ch3mov;
 wire [23:0] M_ch4;
 wire [12:0] addr_ch4;
 wire [23:0] M_ch4mov;
 wire [12:0] addr_ch4mov;
 wire [23:0] M_ch5;
 wire [12:0] addr_ch5;
 wire [23:0] M_ch5mov;
 wire [12:0] addr_ch5mov;
 wire [23:0] M_pause;
 wire [10:0] addr_pause;
 wire [23:0] M_mon1mov;
 wire [12:0] addr_mon1mov;
 wire [23:0] M_mon4;
 wire [12:0] addr_mon4;
 wire [23:0] M_mon5;
 wire [12:0] addr_mon5;
 wire [23:0] M_mon4mov;
 wire [12:0] addr_mon4mov;
 wire [23:0] M_mon5mov;
 wire [12:0] addr_mon5mov;
 wire [23:0] M_dead;
 wire [12:0] addr_dead;
 wire [23:0] M_att1;
 wire [9:0] addr_att1;
 wire [23:0] M_att2;
 wire [9:0] addr_att2;
 wire [23:0] M_att3;
 wire [9:0] addr_att3;
 wire [23:0] M_att4;
 wire [9:0] addr_att4;
 wire [23:0] M_att5;
```

```
	wire [9:0] addr_att5;
	wire [23:0] M_fire;
	wire [10:0] addr_fire;
	wire [23:0] M_gameover;
	wire [14:0] addr_gameover;

   VGA_LED_Emulator led_emulator(.clk50(clk),
                                 .reset(reset),
                                 .hcount(hcount),
                                 .vcount(vcount),
                                 .VGA_CLK (VGA_CLK),
                                 .VGA_HS (VGA_HS),
                                 .VGA_VS (VGA_VS),
                                 .VGA_BLANK_n (VGA_BLANK_n),
                                 .VGA_SYNC_n (VGA_SYNC_n));

ROM_test3  prom_emulator(.clock(VGA_CLK), .address(addr_bg), .q(M_bg));
test  bg_emulator(.clock(VGA_CLK), .address(addr_1), .q(M_1));
cu  cu_emulator(.clock(VGA_CLK), .address(addr_cu), .q(M_cu));
chara  chara_emulator(.clock(VGA_CLK), .address(addr_ch1), .q(M_ch1));
charamov  charamov_emulator(.clock(VGA_CLK), .address(addr_ch1mov), .q(M_ch1mov));
monster  monster_emulator(.clock(VGA_CLK), .address(addr_mon1), .q(M_mon1));
mon1mov          mon1mov_emulator(.clock(VGA_CLK),        .address(addr_mon1mov),
.q(M_mon1mov));
bg2  background_emulator(.clock(VGA_CLK), .address(addr_bg2), .q(M_bg2));
columbia  columbia_emulator(.clock(VGA_CLK), .address(addr_col), .q(M_col));
chara3  chara3_emulator(.clock(VGA_CLK), .address(addr_ch3), .q(M_ch3));
chara3mov  chara3mov_emulator(.clock(VGA_CLK), .address(addr_ch3mov), .q(M_ch3mov));
chara4  chara4_emulator(.clock(VGA_CLK), .address(addr_ch4), .q(M_ch4));
chara4mov  chara4mov_emulator(.clock(VGA_CLK), .address(addr_ch4mov), .q(M_ch4mov));
chara5  chara5_emulator(.clock(VGA_CLK), .address(addr_ch5), .q(M_ch5));
chara5mov  chara5mov_emulator(.clock(VGA_CLK), .address(addr_ch5mov), .q(M_ch5mov));
pause  pause_emulator(.clock(VGA_CLK), .address(addr_pause), .q(M_pause));
monster4  monster4_emulator(.clock(VGA_CLK), .address(addr_mon4), .q(M_mon4));
monster4mov       monster4mov_emulator(.clock(VGA_CLK),        .address(addr_mon4mov),
.q(M_mon4mov));
monster5  monster5_emulator(.clock(VGA_CLK), .address(addr_mon5), .q(M_mon5));
monster5mov         monster5mov_emulator(.clock(VGA_CLK),      .address(addr_mon5mov),
.q(M_mon5mov));
attack1  attack1_emulator(.clock(VGA_CLK), .address(addr_att1), .q(M_att1));
attack2  attack2_emulator(.clock(VGA_CLK), .address(addr_att2), .q(M_att2));
```

```
attack3  attack3_emulator(.clock(VGA_CLK), .address(addr_att3), .q(M_att3));
attack4  attack4_emulator(.clock(VGA_CLK), .address(addr_att4), .q(M_att4));
attack5  attack5_emulator(.clock(VGA_CLK), .address(addr_att5), .q(M_att5));
dead  dead_emulator(.clock(VGA_CLK), .address(addr_dead), .q(M_dead));
fireway  fireway_emulator(.clock(VGA_CLK), .address(addr_fire), .q(M_fire));
gameover                    gameover_emulator(.clock(VGA_CLK),          .address(addr_gameover),
.q(M_gameover));


RGB_controller controller_1(.hcount(hcount),
        .vcount(vcount), .x(x), .y(y),.a1(a1), .b1(b1),.a2(a2), .b2(b2),
        .clk(VGA_CLK),.a3(a3), .b3(b3),.hex1(hex1),.hex2(hex2),.hex3(hex3),
        .run(run), .x11(x11),.x12(x12),.x13(x13),.x14(x14),
        .y11(y11),.y12(y12),.y13(y13),.y14(y14),
        .z11(z11),.z12(z12),.z13(z13),.z14(z14),
        .d11(d11),.d12(d12),.d13(d13),.d14(d14),
         .x21(x21),.x22(x22),.x23(x23),.x24(x24),.x25(x25),
                .x31(x31),.x32(x32),.x33(x33),.x34(x34),.x35(x35),
                .x41(x41),.x42(x42),.x43(x43),.x44(x44),.x45(x45),
                .x51(x51),.x52(x52),.x53(x53),.x54(x54),.x55(x55),
                .y21(y21),.y22(y22),.y23(y23),.y24(y24),.y25(y25),
                .y31(y31),.y32(y32),.y33(y33),.y34(y34),.y35(y35),
                .y41(y41),.y42(y42),.y43(y43),.y44(y44),.y45(y45),
                .y51(y51),.y52(y52),.y53(y53),.y54(y54),.y55(y55),
                .z21(z21),.z22(z22),.z23(z23),.z24(z24),.z25(z25),
                .z31(z31),.z32(z32),.z33(z33),.z34(z34),.z35(z35),
                .z41(z41),.z42(z42),.z43(z43),.z44(z44),.z45(z45),
                .z51(z51),.z52(z52),.z53(z53),.z54(z54),.z55(z55),
                .d21(d21),.d22(d22),.d23(d23),.d24(d24),.d25(d25),
                .d31(d31),.d32(d32),.d33(d33),.d34(d34),.d35(d35),
                .d41(d41),.d42(d42),.d43(d43),.d44(d44),.d45(d45),
                .d51(d51),.d52(d52),.d53(d53),.d54(d54),.d55(d55),
                .m1(m1),.m2(m2),.m3(m3),.m4(m4),
                .m21(m21),.m22(m22),.m23(m23),.m24(m24),.m25(m25),
                .m31(m31),.m32(m32),.m33(m33),.m34(m34),.m35(m35),
                .m41(m41),.m42(m42),.m43(m43),.m44(m44),.m45(m45),
                .m51(m51),.m52(m52),.m53(m53),.m54(m54),.m55(m55),
                .g1(g1),.g2(g2),.g3(g3),.g4(g4),.g5(g5),
                .g11(g11),.g12(g12),.g13(g13),.g14(g14),
         .g21(g21),.g22(g22),.g23(g23),.g24(g24),
        .g31(g31),.g32(g32),.g33(g33),.g34(g34),
        .g41(g41),.g42(g42),.g43(g43),.g44(g44),
        .g51(g51),.g52(g52),.g53(g53),.g54(g54),
        .p1(p1),.p2(p2),.p3(p3),.p4(p4),.p5(p5),
        .p6(p6),.p7(p7),.p8(p8),.p9(p9),.p10(p10),
        .p11(p11),.p12(p12),.p13(p13),.p14(p14),.p15(p15),
        .p16(p16),.p17(p17),.p18(p18),.p19(p19),.p20(p20),
        .q1(q1),.q2(q2),.q3(q3),.q4(q4),.q5(q5),
        .q6(q6),.q7(q7),.q8(q8),.q9(q9),.q10(q10),
        .q11(q11),.q12(q12),.q13(q13),.q14(q14),.q15(q15),
        .q16(q16),.q17(q17),.q18(q18),.q19(q19),.q20(q20),.start(start),.over(over),
                .addr_bg(addr_bg),.M_bg(M_bg),.addr_1(addr_1),
                .M_cu(M_cu),.addr_cu(addr_cu),
                .M_ch1(M_ch1),.addr_ch1(addr_ch1),
                .M_ch1mov(M_ch1mov),.addr_ch1mov(addr_ch1mov),
                .M_mon1(M_mon1),.addr_mon1(addr_mon1),
                .M_bg2(M_bg2),.addr_bg2(addr_bg2),
                .M_col(M_col),.addr_col(addr_col),
                .M_ch3(M_ch3),.addr_ch3(addr_ch3),
                .M_ch3mov(M_ch3mov),.addr_ch3mov(addr_ch3mov),
                .M_ch4(M_ch4),.addr_ch4(addr_ch4),
                .M_ch4mov(M_ch4mov),.addr_ch4mov(addr_ch4mov),
                .M_ch5(M_ch5),.addr_ch5(addr_ch5),
                .M_ch5mov(M_ch5mov),.addr_ch5mov(addr_ch5mov),
                .M_mon4(M_mon4),.addr_mon4(addr_mon4),
                .M_mon5(M_mon5),.addr_mon5(addr_mon5),
                .M_mon1mov(M_mon1mov),.addr_mon1mov(addr_mon1mov),
                .M_mon4mov(M_mon4mov),.addr_mon4mov(addr_mon4mov),
                .M_mon5mov(M_mon5mov),.addr_mon5mov(addr_mon5mov),
                .M_pause(M_pause),.addr_pause(addr_pause),
                .M_att1(M_att1),.addr_att1(addr_att1),
                .M_att2(M_att2),.addr_att2(addr_att2),
                .M_att3(M_att3),.addr_att3(addr_att3),
                .M_att4(M_att4),.addr_att4(addr_att4),
                .M_att5(M_att5),.addr_att5(addr_att5),
                .M_dead(M_dead),.addr_dead(addr_dead),
                .M_fire(M_fire),.addr_fire(addr_fire),
                        .M_gameover(M_gameover),.addr_gameover(addr_gameover),
                        .M_1(M_1), .VGA_R(VGA_R),
                        .VGA_G(VGA_G), .VGA_B(VGA_B));

always_ff @(posedge clk)
```

```
if (reset) begin
      x <= 11'd0035; //
      y <= 11'd0105;
      a1 <= 11'd0584; //
      b1 <= 11'd0105;
      a2 <= 11'd0000; //
      b2 <= 11'd0000;
      a3 <= 11'd0000; //
       b3 <= 11'd0000;
      run <= 11'd0000;
      hex1 <= 11'd0000;
      hex2 <= 11'd0000;
      hex3 <= 11'd0000;
      x11 <= 11'd0000;
      y11 <= 11'd0000;
      x12 <= 11'd0000;
      y12 <= 11'd0000;
      x13 <= 11'd0000;
      y13 <= 11'd0000;
      x14 <= 11'd0000;
      y14 <= 11'd0000;
      z11 <= 11'd0000;
      d11 <= 11'd0000;
      z12 <= 11'd0000;
      d12 <= 11'd0000;
      z13 <= 11'd0000;
      d13 <= 11'd0000;
      z14 <= 11'd0000;
      d14 <= 11'd0000;
      m1 <= 11'd0000;
      m2 <= 11'd0000;
      m3 <= 11'd0000;
      m4 <= 11'd0000;
      x21 <= 11'd0000;      x31 <= 11'd0000;      x41 <= 11'd0000;      x51 <= 11'd0000;
      y21 <= 11'd0000;      y31 <= 11'd0000;      y41 <= 11'd0000;      y51 <= 11'd0000;
      x22 <= 11'd0000;      x32 <= 11'd0000;      x42 <= 11'd0000;      x52 <= 11'd0000;
      y22 <= 11'd0000;      y32 <= 11'd0000;      y42 <= 11'd0000;      y52 <= 11'd0000;
      x23 <= 11'd0000;      x33 <= 11'd0000;      x43 <= 11'd0000;      x53 <= 11'd0000;
      y23 <= 11'd0000;      y33 <= 11'd0000;      y43 <= 11'd0000;      y53 <= 11'd0000;
      x24 <= 11'd0000;      x34 <= 11'd0000;      x44 <= 11'd0000;      x54 <= 11'd0000;
```

```
        y24 <= 11'd0000;    y34 <= 11'd0000;    y44 <= 11'd0000;    y54 <= 11'd0000;
        x25 <= 11'd0000;    x35 <= 11'd0000;    x45 <= 11'd0000;    x55 <= 11'd0000;
        y25 <= 11'd0000;    y35 <= 11'd0000;    y45 <= 11'd0000;    y55 <= 11'd0000;
        z21 <= 11'd0000;    z31 <= 11'd0000;    z41 <= 11'd0000;    z51 <= 11'd0000;
        d21 <= 11'd0000;    d31 <= 11'd0000;    d41 <= 11'd0000;    d51 <= 11'd0000;
        z22 <= 11'd0000;    z32 <= 11'd0000;    z42 <= 11'd0000;    z52 <= 11'd0000;
        d22 <= 11'd0000;    d32 <= 11'd0000;    d42 <= 11'd0000;    d52 <= 11'd0000;
        z23 <= 11'd0000;    z33 <= 11'd0000;    z43 <= 11'd0000;    z53 <= 11'd0000;
        d23 <= 11'd0000;    d33 <= 11'd0000;    d43 <= 11'd0000;    d53 <= 11'd0000;
        z24 <= 11'd0000;    z34 <= 11'd0000;    z44 <= 11'd0000;    z54 <= 11'd0000;
        d24 <= 11'd0000;    d34 <= 11'd0000;    d44 <= 11'd0000;    d54 <= 11'd0000;
        z25 <= 11'd0000;    z35 <= 11'd0000;    z45 <= 11'd0000;    z55 <= 11'd0000;
        d25 <= 11'd0000;    d35 <= 11'd0000;    d45 <= 11'd0000;    d55 <= 11'd0000;
        m21 <= 11'd0000;    m31 <= 11'd0000;    m41 <= 11'd0000;    m51 <= 11'd0000;
        m22 <= 11'd0000;    m32 <= 11'd0000;    m42 <= 11'd0000;    m52 <= 11'd0000;
        m23 <= 11'd0000;    m33 <= 11'd0000;    m43 <= 11'd0000;    m53 <= 11'd0000;
        m24 <= 11'd0000;    m34 <= 11'd0000;    m44 <= 11'd0000;    m54 <= 11'd0000;
        m25 <= 11'd0000;    m35 <= 11'd0000;    m45 <= 11'd0000;    m55 <= 11'd0000;
        g11 <= 11'd0640;  g12 <= 11'd0640;  g13 <= 11'd0640;  g14 <= 11'd0640;
        g21 <= 11'd0640;  g22 <= 11'd0640;  g23 <= 11'd0640;  g24 <= 11'd0640;
        g31 <= 11'd0640;  g32 <= 11'd0640;  g33 <= 11'd0640;  g34 <= 11'd0640;
        g41 <= 11'd0640;  g42 <= 11'd0640;  g43 <= 11'd0640;  g44 <= 11'd0640;
        g51 <= 11'd0640;  g52 <= 11'd0640;  g53 <= 11'd0640;  g54 <= 11'd0640;
        g1 <= 11'd0000;  g2 <= 11'd0000;  g3 <= 11'd0000;  g4<= 11'd0000; g5<= 11'd0000;
        p1 <= 11'd0000;  p2 <= 11'd0000;  p3 <= 11'd0000;  p4<= 11'd0000; p5<= 11'd0000;
        p6 <= 11'd0000;  p7 <= 11'd0000;  p8 <= 11'd0000;  p9<= 11'd0000; p10<= 11'd0000;
        p11  <=  11'd0000;   p12  <=  11'd0000;   p13  <=  11'd0000;   p14<=  11'd0000;  p15<=
11'd0000;
        p16  <=  11'd0000;   p17  <=  11'd0000;   p18  <=  11'd0000;   p19<=  11'd0000;  p20<=
11'd0000;
        music1<=11'd0000; music2<=11'd0000;
        q1 <= 11'd0000;  q2 <= 11'd0000;  q3 <= 11'd0000;  q4<= 11'd0000; q5<= 11'd0000;
        q6 <= 11'd0000;  q7 <= 11'd0000;  q8 <= 11'd0000;  q9<= 11'd0000; q10<= 11'd0000;
        q11  <=  11'd0000;   q12  <=  11'd0000;   q13  <=  11'd0000;   q14<=  11'd0000;  q15<=
11'd0000;
        q16  <=  11'd0000;   q17  <=  11'd0000;   q18  <=  11'd0000;   q19<=  11'd0000;  q20<=
11'd0000;
        start <= 11'd0000;  over <= 11'd0000;
    end
        else if (chipselect && write)
```

```
begin

 case(address)
           8'd00 : x <= writedata;
           8'd01 : y <= writedata;
           8'd02 : a1 <= writedata;
           8'd03 : b1 <= writedata;
           8'd04 : a2 <= writedata;
           8'd05 : b2 <= writedata;
           8'd06 : a3 <= writedata;
           8'd07 : b3 <= writedata;
           8'd08 : run <= writedata;
           8'd09 : hex1 <= writedata;
           8'd10 : hex2 <= writedata;
           8'd11 : hex3 <= writedata;
           8'd12 : x11 <= writedata;
          8'd13 : y11 <= writedata;
           8'd14 : x12 <= writedata;
          8'd15 : y12 <= writedata;
          8'd16 : x13 <= writedata;
          8'd17 : y13 <= writedata;
          8'd18 : x14 <= writedata;
          8'd19 : y14 <= writedata;
           8'd20 : z11 <= writedata;
           8'd21 : d11 <= writedata;
           8'd22 : z12 <= writedata;
           8'd23 : d12 <= writedata;
           8'd24 : z13 <= writedata;
           8'd25 : d13 <= writedata;
           8'd26 : z14 <= writedata;
           8'd27 : d14 <= writedata;
           8'd28 : m1 <= writedata;
           8'd29 : m2 <= writedata;
           8'd30 : m3 <= writedata;
           8'd31 : m4 <= writedata;
8'd32 :x21 <= writedata;
8'd33 :y21 <= writedata;
8'd34 :x22 <= writedata;
8'd35 :y22 <= writedata;
8'd36 :x23 <= writedata;
8'd37 :y23 <= writedata;
```

```
8'd38 :x24 <= writedata;
8'd39 :y24 <= writedata;
8'd40 :x25 <= writedata;
8'd41 :y25 <= writedata;
8'd42 :z21 <= writedata;
8'd43 :d21 <= writedata;
8'd44 :z22 <= writedata;
8'd45 :d22 <= writedata;
8'd46 :z23 <= writedata;
8'd47 :d23 <= writedata;
8'd48 :z24 <= writedata;
8'd49 :d24 <= writedata;
8'd50 :z25 <= writedata;
8'd51 :d25 <= writedata;
8'd52 :m21 <= writedata;
8'd53 :m22 <= writedata;
8'd54 :m23 <= writedata;
8'd55 :m24 <= writedata;
8'd56 :m25 <= writedata;
8'd57 :x31 <= writedata;
8'd58 :y31 <= writedata;
8'd59 :x32 <= writedata;
8'd60 :y32 <= writedata;
8'd61 :x33 <= writedata;
8'd62 :y33 <= writedata;
8'd63 :x34 <= writedata;
8'd64 :y34 <= writedata;
8'd65 :x35 <= writedata;
8'd66 :y35 <= writedata;
8'd67 :z31 <= writedata;
8'd68 :d31 <= writedata;
8'd69 :z32 <= writedata;
8'd70 :d32 <= writedata;
8'd71 :z33 <= writedata;
8'd72 :d33 <= writedata;
8'd73 :z34 <= writedata;
8'd74 :d34 <= writedata;
8'd75 :z35 <= writedata;
8'd76 :d35 <= writedata;
8'd77 :m31 <= writedata;
```

```
8'd78 :m32 <= writedata;
8'd79 :m33 <= writedata;
8'd80 :m34 <= writedata;
8'd81 :m35 <= writedata;
8'd82 :x41 <= writedata;
8'd83 :y41 <= writedata;
8'd84 :x42 <= writedata;
8'd85 :y42 <= writedata;
8'd86 :x43 <= writedata;
8'd87 :y43 <= writedata;
8'd88 :x44 <= writedata;
8'd89 :y44 <= writedata;
8'd90 :x45 <= writedata;
8'd91 :y45 <= writedata;
8'd92 :z41 <= writedata;
8'd93 :d41 <= writedata;
8'd94 :z42 <= writedata;
8'd95 :d42 <= writedata;
8'd96 :z43 <= writedata;
8'd97 :d43 <= writedata;
8'd98 :z44 <= writedata;
8'd99 :d44 <= writedata;
8'd100 :z45 <= writedata;
8'd101 :d45 <= writedata;
8'd102 :m41 <= writedata;
8'd103 :m42 <= writedata;
8'd104 :m43 <= writedata;
8'd105 :m44 <= writedata;
8'd106 :m45 <= writedata;
8'd107 :x51 <= writedata;
8'd108 :y51 <= writedata;
8'd109 :x52 <= writedata;
8'd110 :y52 <= writedata;
8'd111 :x53 <= writedata;
8'd112 :y53 <= writedata;
8'd113 :x54 <= writedata;
8'd114 :y54 <= writedata;
8'd115 :x55 <= writedata;
8'd116 :y55 <= writedata;
8'd117 :z51 <= writedata;
```

```
8'd118 :d51 <= writedata;
8'd119 :z52 <= writedata;
8'd120 :d52 <= writedata;
8'd121 :z53 <= writedata;
8'd122 :d53 <= writedata;
8'd123 :z54 <= writedata;
8'd124 :d54 <= writedata;
8'd125 :z55 <= writedata;
8'd126 :d55 <= writedata;
8'd127 :m51 <= writedata;
8'd128 :m52 <= writedata;
8'd129 :m53 <= writedata;
8'd130 :m54 <= writedata;
8'd131 :m55 <= writedata;
8'd132 :g11 <= writedata;
8'd133 :g12 <= writedata;
8'd134 :g13 <= writedata;
8'd135 :g14 <= writedata;
8'd136 :g1 <= writedata;
8'd137 :g21 <= writedata;
8'd138 :g22 <= writedata;
8'd139 :g23 <= writedata;
8'd140 :g24 <= writedata;
8'd141 :g2 <= writedata;
8'd142 :g31 <= writedata;
8'd143 :g32 <= writedata;
8'd144 :g33 <= writedata;
8'd145 :g34 <= writedata;
8'd146 :g3 <= writedata;
8'd147 :g41 <= writedata;
8'd148 :g42 <= writedata;
8'd149 :g43 <= writedata;
8'd150 :g44 <= writedata;
8'd151 :g4 <= writedata;
8'd152 :g51 <= writedata;
8'd153 :g52 <= writedata;
8'd154 :g53 <= writedata;
8'd155 :g54 <= writedata;
8'd156 :g5 <= writedata;
8'd157 : p1 <= writedata;
```

```
8'd158 :p2 <= writedata;
8'd159 :p3 <= writedata;
8'd160 :p4 <= writedata;
8'd161 :p5 <= writedata;
8'd162 :p6 <= writedata;
8'd163 :p7 <= writedata;
8'd164 :p8 <= writedata;
8'd165 :p9 <= writedata;
8'd166 :p10 <= writedata;
8'd167 :p11 <= writedata;
8'd168 :p12 <= writedata;
8'd169 :p13 <= writedata;
8'd170 :p14 <= writedata;
8'd171 :p15 <= writedata;
8'd172 :p16 <= writedata;
8'd173 :p17 <= writedata;
8'd174 :p18 <= writedata;
8'd175 :p19 <= writedata;
8'd176 :p20 <= writedata;
8'd177 :music1 <= writedata;
8'd178 :music2 <= writedata;
8'd179 :q1 <= writedata;
8'd180 :q2 <= writedata;
8'd181 :q3 <= writedata;
8'd182 :q4 <= writedata;
8'd183 :q5 <= writedata;

8'd184 :q6 <= writedata;
8'd185 :q7 <= writedata;
8'd186 :q8 <= writedata;
8'd187 :q9 <= writedata;
8'd188 :q10 <= writedata;
8'd189 :q11 <= writedata;
8'd190 :q12 <= writedata;
8'd191 :q13 <= writedata;
8'd192 :q14 <= writedata;
8'd193 :q15 <= writedata;
8'd194 :q16 <= writedata;
8'd195 :q17 <= writedata;
8'd196 :q18 <= writedata;
8'd197 :q19 <= writedata;
```

```
          8'd198 :q20 <= writedata;
          8'd199 :start <= writedata;
          8'd200 :over <= writedata;
          endcase

          end

  always_ff @(posedge clk)
  begin
   VGA_audio1=music1;
  VGA_audio2=music2;
          end

endmodule
```

2.2 RGB_controller.sv code

```
//**************************RGB_controller.sv code*********************//

module RGB_controller(
                    x,y, a1,b1, a2,b2, a3,b3, hex1,hex2,hex3,run,  x11,y11,x12,y12,x13,y13,x14,y14,
z11,d11,z12,d12,z13,d13,z14,d14,
m1,m2,m3,m4,x21,y21,x22,y22,x23,y23,x24,y24,x25,y25,z21,d21,z22,d22,z23,d23,z24,d24,z25
,d25, m21,m22,m23,m24,m25, x31,y31,x32,y32,x33,y33,x34,y34,x35,y35,
z31,d31,z32,d32,z33,d33,z34,d34,z35,d35,                    m31,m32,m33,m34,m35,
,y41,x42,y42,x43,y43,x44,y44,x45,y45,          z41,d41,z42,d42,z43,d43,z44,d44,z45,d45,
m41,m42,m43,m44,m45,
x51,y51,x52,y52,x53,y53,x54,y54,x55,y55,
z51,d51,z52,d52,z53,d53,z54,d54,z55,d55,
m51,m52,m53,m54,m55,
g1,g2,g3,g4,g5,g11,g12,g13,g14, g21,g22,g23,g24, g31,g32,g33,g34, g41,g42,g43,g44,
g51,g52,g53,g54,
p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13,p14,p15,p16,p17,p18,p19,p20,q1,q2,q3,q4,q5,q6,q7
,q8,q9,q10,q11,q12,q13,q14,q15,q16,q17,q18,q19,q20,
start,over, hcount,vcount, clk, addr_bg, addr_1, M_bg, M_1, addr_cu, M_cu, addr_ch1,
M_ch1,addr_ch1mov,M_ch1mov,addr_mon1,M_mon1,addr_bg2,M_bg2,addr_col,
M_col,addr_ch3,M_ch3,addr_ch3mov,M_ch3mov,addr_ch4,M_ch4,addr_ch4mov,
M_ch4mov,addr_ch5,M_ch5,addr_ch5mov,M_ch5mov,addr_mon1mov,M_mon1mov,
addr_mon4,M_mon4,addr_mon4mov,M_mon4mov,
addr_mon5,M_mon5, addr_mon5mov, M_mon5mov, addr_pause, M_pause, addr_att1, M_att1,
addr_att2,addr_att3,addr_att4,addr_att5, M_att2,M_att3,M_att4,M_att5,
addr_dead, M_dead, addr_fire,M_fire, addr_gameover,M_gameover,
VGA_R, VGA_G, VGA_B
                                                            );

input  wire [10:0] hcount;
input wire [9:0]  vcount;
output wire [7:0] VGA_R, VGA_G, VGA_B;
input wire clk;
output wire [12:0] addr_bg;
output wire [12:0] addr_1;
output wire [12:0] addr_cu;
output wire [12:0] addr_ch1;
output wire [12:0] addr_ch1mov;
output wire [12:0] addr_mon1;
output wire [7:0] addr_bg2;
output wire [15:0] addr_col;
output wire [12:0] addr_ch3;
output wire [12:0] addr_ch3mov;
output wire [12:0] addr_ch4;
output wire [12:0] addr_ch4mov;
output wire [12:0] addr_ch5;
output wire [12:0] addr_ch5mov;
output wire [12:0] addr_mon4;
output wire [12:0] addr_mon5;
output wire [12:0] addr_mon1mov;
output wire [12:0] addr_mon4mov;
output wire [12:0] addr_mon5mov;
output wire [10:0] addr_pause;
output wire [12:0] addr_dead;
output wire [9:0] addr_att1;
output wire [9:0] addr_att2;
output wire [9:0] addr_att3;
output wire [9:0] addr_att4;
output wire [9:0] addr_att5;
output wire [10:0] addr_fire;
output wire [14:0] addr_gameover;

input wire [23:0] M_1;
input wire [15:0] x,y;
input wire [15:0] a1,b1;
input wire [15:0] a2,b2;
input wire [15:0] a3,b3;
input wire [15:0] run,hex1,hex2,hex3;
input wire [15:0] x11,y11,x12,y12,x13,y13,x14,y14;
input wire [15:0] z11,d11,z12,d12,z13,d13,z14,d14;
input wire [15:0] m1,m2,m3,m4;
input wire [15:0] x21,y21,x22,y22,x23,y23,x24,y24,x25,y25;
input wire [15:0] z21,d21,z22,d22,z23,d23,z24,d24,z25,d25;
input wire [15:0] m21,m22,m23,m24,m25;
input wire [15:0] x31,y31,x32,y32,x33,y33,x34,y34,x35,y35;
input wire [15:0] z31,d31,z32,d32,z33,d33,z34,d34,z35,d35;
input wire [15:0] m31,m32,m33,m34,m35;
input wire [15:0] x41,y41,x42,y42,x43,y43,x44,y44,x45,y45;
input wire [15:0] z41,d41,z42,d42,z43,d43,z44,d44,z45,d45;
input wire [15:0] m41,m42,m43,m44,m45;
input wire [15:0] x51,y51,x52,y52,x53,y53,x54,y54,x55,y55;
input wire [15:0] z51,d51,z52,d52,z53,d53,z54,d54,z55,d55;
input wire [15:0] m51,m52,m53,m54,m55;
input wire [15:0] g1,g2,g3,g4,g5;
input wire [15:0] g11,g12,g13,g14;
input wire [15:0] g21,g22,g23,g24;
input wire [15:0] g31,g32,g33,g34;
input wire [15:0] g41,g42,g43,g44;
input wire [15:0] g51,g52,g53,g54;
input wire [15:0] p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13,p14,p15,p16,p17,p18,p19,p20;
input wire [15:0] q1,q2,q3,q4,q5,q6,q7,q8,q9,q10,q11,q12,q13,q14,q15,q16,q17,q18,q19,q20;
input wire [15:0] start,over;
input wire [23:0] M_bg;
input wire [23:0] M_cu;
input wire [23:0] M_ch1;
input wire [23:0] M_ch1mov;
input wire [23:0] M_mon1;
input wire [23:0] M_bg2;
input wire [23:0] M_col;
input wire [23:0] M_ch3;
input wire [23:0] M_ch3mov;
input wire [23:0] M_ch4;
input wire [23:0] M_ch4mov;
input wire [23:0] M_ch5;
input wire [23:0] M_ch5mov;
input wire [23:0] M_mon4;
input wire [23:0] M_mon5;
input wire [23:0] M_mon1mov;
input wire [23:0] M_mon4mov;
input wire [23:0] M_mon5mov;
input wire [23:0] M_pause;
input wire [23:0] M_att1;
input wire [23:0] M_att2;
input wire [23:0] M_att3;
input wire [23:0] M_att4;
input wire [23:0] M_att5;
input wire [23:0] M_dead;
input wire [23:0] M_fire;
input wire [23:0] M_gameover;

assign addr_1 =(campusy%75)*55+campusx%55;
assign addr_bg = (regiony%75)*55+regionx%55;
assign addr_cu =(cuy%75)*55+cux;
assign addr_ch1 =(ch1y%75)*55+ch1x%55;
assign addr_ch1mov =(ch1movy%75)*55+ch1movx%55;

assign addr_bg2 =(bg2y%15)*15+bg2x%15;
assign addr_ch3 =(ch3y%75)*55+ch3x%55;
assign addr_ch3mov =(ch3movy%75)*55+ch3movx%55;
assign addr_ch4 =(ch4y%75)*55+ch4x%55;
assign addr_ch4mov =(ch4movy%75)*55+ch4movx%55;
assign addr_ch5 =(ch5y%75)*55+ch5x%55;
assign addr_ch5mov =(ch5movy%75)*55+ch5movx%55;
assign addr_col =coly*550+colx;
assign addr_pause =pausey*35+pausex;
assign addr_att1 =att1y*25+att1x;
assign addr_att2 =att2y*25+att2x;
assign addr_att3 =att3y*25+att3x;
assign addr_att4 =att4y*25+att4x;
assign addr_att5 =att5y*25+att5x;
assign addr_fire =(firey%25)*55+firex%55;
assign addr_gameover =gameovery*400+gameoverx;


logic [10:0] xrow;                              measuring graphs' sizes
logic [9:0] yrow;
logic [10:0] regionx, regiony;
logic [10:0] campusx, campusy;
logic [10:0] cux, cuy;
logic [10:0] ch1x, ch1y;
logic [10:0] ch1movx, ch1movy;
logic [10:0] mon11x, mon11y;
logic [10:0] mon12x, mon12y;
logic [10:0] mon13x, mon13y;
logic [10:0] mon14x, mon14y;
logic [10:0] mon15x, mon15y;
logic [10:0] mon16x, mon16y;
logic [10:0] mon17x, mon17y;
logic [10:0] bg2x, bg2y;
logic [10:0] colx, coly;
logic [10:0] ch3x, ch3y;
logic [10:0] ch3movx, ch3movy;
logic [10:0] ch4x, ch4y;
logic [10:0] ch4movx, ch4movy;
logic [10:0] ch5x, ch5y;
logic [10:0] ch5movx, ch5movy;
logic [10:0] mon41x, mon41y;
logic [10:0] mon42x, mon42y;
logic [10:0] mon43x, mon43y;
logic [10:0] mon44x, mon44y;
logic [10:0] mon45x, mon45y;
logic [10:0] mon46x, mon46y;
logic [10:0] mon51x, mon51y;
logic [10:0] mon52x, mon52y;
logic [10:0] mon53x, mon53y;
logic [10:0] mon54x, mon54y;
logic [10:0] mon55x, mon55y;
logic [10:0] mon56x, mon56y;
logic [10:0] mon57x, mon57y;
logic [10:0] pausex, pausey;
logic [10:0] gameoverx, gameovery;
logic [10:0] mon11movx, mon11movy;
logic [10:0] mon12movx, mon12movy;
logic [10:0] mon13movx, mon13movy;
logic [10:0] mon14movx, mon14movy;
logic [10:0] mon15movx, mon15movy;
logic [10:0] mon16movx, mon16movy;
logic [10:0] mon17movx, mon17movy;
logic [10:0] mon41movx, mon41movy;
logic [10:0] mon42movx, mon42movy;
logic [10:0] mon43movx, mon43movy;
logic [10:0] mon44movx, mon44movy;
logic [10:0] mon45movx, mon45movy;
logic [10:0] mon46movx, mon46movy;
logic [10:0] mon51movx, mon51movy;
logic [10:0] mon52movx, mon52movy;
logic [10:0] mon53movx, mon53movy;
logic [10:0] mon54movx, mon54movy;
logic [10:0] mon55movx, mon55movy;
logic [10:0] mon56movx, mon56movy;
logic [10:0] mon57movx, mon57movy;
logic [10:0] deadx, deady;
logic [10:0] att1x, att1y;
logic [10:0] att2x, att2y;
logic [10:0] att3x, att3y;
logic [10:0] att4x, att4y;
logic [10:0] att5x, att5y;
logic [10:0] firex, firey;
logic [10:0] ice1x, ice1y;
logic [10:0] ice2x, ice2y;
logic [10:0] ice3x, ice3y;
logic [10:0] ice4x, ice4y;
logic [10:0] ice5x, ice5y;
logic [10:0] p1x,p1y;
logic [10:0] p2x,p2y;
logic [10:0] p3x,p3y;
logic [10:0] p4x,p4y;
logic [10:0] p5x,p5y;
logic [10:0] p6x,p6y;
logic [10:0] p7x,p7y;
logic [10:0] p8x,p8y;
logic [10:0] p9x,p9y;
logic [10:0] p10x,p10y;
logic [10:0] p11x,p11y;
logic [10:0] p12x,p12y;
logic [10:0] p13x,p13y;
logic [10:0] p14x,p14y;
logic [10:0] p15x,p15y;
logic [10:0] p16x,p16y;
logic [10:0] p17x,p17y;
logic [10:0] p18x,p18y;


assign regionx=(xrow-11'd0090);
assign regiony = (yrow- 11'd0105);
assign campusx=(xrow-11'd0090);
assign campusy = (yrow- 11'd0105);
assign cux=(xrow-11'd0035);
assign cuy = (yrow- 11'd0105);
assign ch1x=(xrow-11'd0035);
assign ch1y = (yrow- 11'd0105);
assign ch1movx=(xrow-11'd0035);
assign ch1movy = (yrow- 11'd0105);
assign gameoverx=(xrow-11'd120);
assign gameovery=(yrow-11'd100);
assign mon41x=(xrow-g11);
assign mon41y = (yrow- 11'd0105);
assign mon41movx=(xrow-g11);
assign mon41movy = (yrow- 11'd0105);
assign mon51x=(xrow-g31);
assign mon51y = (yrow- 11'd0255);
assign mon51movx=(xrow-g31);
assign mon51movy = (yrow- 11'd0255);
assign mon42x=(xrow-g21);
assign mon42y = (yrow- 11'd0180);
assign mon42movx=(xrow-g21);
assign mon42movy = (yrow- 11'd0180);
assign mon52x=(xrow-g41);
assign mon52y = (yrow- 11'd0330);
assign mon52movx=(xrow-g41);
assign mon52movy = (yrow- 11'd0330);
assign mon11x=(xrow-g51);
assign mon11y = (yrow- 11'd0405);
assign mon11movx=(xrow-g51);
assign mon11movy = (yrow- 11'd0405);                    //   round 1

assign mon12x=(xrow-g12);
assign mon12y = (yrow- 11'd0105);
assign mon12movx=(xrow-g12);
assign mon12movy = (yrow- 11'd0105);
assign mon43x=(xrow-g22);
assign mon43y = (yrow- 11'd0180);
assign mon43movx=(xrow-g22);
assign mon43movy = (yrow- 11'd0180);
assign mon53x=(xrow-g32);
assign mon53y = (yrow- 11'd0255);
assign mon53movx=(xrow-g32);
assign mon53movy = (yrow- 11'd0255);
assign mon13x=(xrow-g42);
assign mon13y = (yrow- 11'd0330);
assign mon13movx=(xrow-g42);
assign mon13movy = (yrow- 11'd0330);
assign mon44x=(xrow-g52);
assign mon44y = (yrow- 11'd0405);
assign mon44movx=(xrow-g52);
assign mon44movy = (yrow- 11'd0405);
assign mon54x=(xrow-g53);
assign mon54y = (yrow- 11'd0405);
assign mon54movx=(xrow-g53);
assign mon54movy = (yrow- 11'd0405);   //   round 2 and 3

assign mon14x=(xrow-g13);
assign mon14y = (yrow- 11'd0105);
assign mon14movx=(xrow-g13);
assign mon14movy = (yrow- 11'd0105);
assign mon45x=(xrow-g23);
assign mon45y = (yrow- 11'd0180);
assign mon45movx=(xrow-g23);
assign mon45movy = (yrow- 11'd0180);
assign mon55x=(xrow-g33);
assign mon55y = (yrow- 11'd0255);
assign mon55movx=(xrow-g33);
assign mon55movy = (yrow- 11'd0255);

assign mon56x=(xrow-g14);
assign mon56y = (yrow- 11'd0105);
assign mon56movx=(xrow-g14);
assign mon56movy = (yrow- 11'd0105);
assign mon46x=(xrow-g24);
assign mon46y = (yrow- 11'd0180);
assign mon46movx=(xrow-g24);
assign mon46movy = (yrow- 11'd0180);
assign mon15x=(xrow-g34);
assign mon15y = (yrow- 11'd0255);
assign mon15movx=(xrow-g34);
assign mon15movy = (yrow- 11'd0255);
assign mon16x=(xrow-g43);
assign mon16y = (yrow- 11'd0330);
assign mon16movx=(xrow-g43);
assign mon16movy = (yrow- 11'd0330);  // round 4 and 5

assign bg2x=(xrow-11'd0000);
assign bg2y = (yrow- 11'd0090);
assign colx=(xrow-11'd0090);
assign coly = (yrow- 11'd0000);
assign ch3x=(xrow-11'd0035);
assign ch3y = (yrow- 11'd0105);
assign ch3movx=(xrow-11'd0035);
assign ch3movy = (yrow- 11'd0105);
assign ch4x=(xrow-11'd0035);
assign ch4y = (yrow- 11'd0105);
assign ch4movx=(xrow-11'd0035);
assign ch4movy = (yrow- 11'd0105);
assign ch5x=(xrow-11'd0035);
assign ch5y = (yrow- 11'd0105);
assign ch5movx=(xrow-11'd0035);
assign ch5movy = (yrow- 11'd0105);
assign pausex=(xrow-11'd0000);
assign pausey = (yrow- 11'd0055);
assign att1x=(xrow-hex2);
assign att1y=(yrow-hex3);
assign att2x=(xrow-z11);
assign att2y=(yrow-d11);
assign att3x=(xrow-z12);
assign att3y=(yrow-d12);
assign att4x=(xrow-z13);
assign att4y=(yrow-d13);
assign att5x=(xrow-z14);
assign att5y=(yrow-d14);
assign firex=(xrow-11'd0090);
assign firey=(yrow-11'd0155);
assign ice1x=(xrow-z51);
assign ice1y=(yrow-d51);
assign ice2x=(xrow-z52);
assign ice2y=(yrow-d52);
assign ice3x=(xrow-z53);
assign ice3y=(yrow-d53);
assign ice4x=(xrow-z54);
assign ice4y=(yrow-d54);
assign ice5x=(xrow-z55);
assign ice5y=(yrow-d55);
assign xrow = (hcount >> 1);
assign yrow = vcount;

logic spriteon;
logic campus;
logic cu;
logic [15:0] flag;
logic [15:0] flag2;
logic [1:0] ch1;
logic mon1;
logic bg2;
logic col;
logic [1:0] ch3;
logic [1:0] ch4;
logic [1:0] ch5;
logic mon4;
logic mon5;
logic pause;
logic mon1mov;
logic mon4mov;
logic mon5mov;
logic ch1mov;
logic ch3mov;
logic ch4mov;
logic ch5mov;
logic dead;
logic att1;
logic att2;
logic att3;
logic att4;
logic att5;
logic ice1,ice2,ice3,ice4,ice5;
logic fire;
logic [3:0] role1;
logic [3:0] role2;
logic [3:0] role3;
logic [3:0] role4;
logic [3:0] role5;
logic eat;
logic screen;
logic gameover;

        always@(posedge clk)
        if ((x11==0) && (x12==0)&&(x13==0)&&(x14==0)&&(a3!=0))
        begin
        role1<=4'd1;
        end
        else if((x11!=0)&&(x12==0)&&(x13==0)&&(x14==0)&&(a3!=0))
        begin
        role1<=4'd2;
        end
        else if((x11!=0)&&(x12!=0)&&(x13==0)&&(x14==0)&&(a3!=0))
        begin
        role1<=4'd3;
        end
        else if((x11!=0)&&(x12!=0)&&(x13!=0)&&(x14==0)&&(a3!=0))
        begin
        role1<=4'd4;
        end
        else if((x11!=0)&&(x12!=0)&&(x13!=0)&&(x14!=0)&&(a3!=0))
        begin
        role1<=4'd5;
        end
        else
        begin
        role1<=4'd0;
        end

         always@(posedge clk)
        if ((x35==0) && (x32==0)&&(x33==0)&&(x34==0)&&(x31!=0))
        begin
        role3<=4'd1;
```

```
end
else if((x35==0) && (x32!=0)&&(x33==0)&&(x34==0)&&(x31!=0))
begin
role3<=4'd2;
end
else if((x35==0) && (x32!=0)&&(x33!=0)&&(x34==0)&&(x31!=0))
begin
role3<=4'd3;
end
else if((x35==0) && (x32!=0)&&(x33!=0)&&(x34!=0)&&(x31!=0))
begin
role3<=4'd4;
end
else if((x35!=0) && (x32!=0)&&(x33!=0)&&(x34!=0)&&(x31!=0))
begin
role3<=4'd5;
end
else
begin
role3<=4'd0;
end

always@(posedge clk)
if ((x45==0) && (x42==0)&&(x43==0)&&(x44==0)&&(x41!=0))
begin
role4<=4'd1;
end
else if((x45==0) && (x42!=0)&&(x43==0)&&(x44==0)&&(x41!=0))
begin
role4<=4'd2;
end
else if((x45==0) && (x42!=0)&&(x43!=0)&&(x44==0)&&(x41!=0))
begin
role4<=4'd3;
end
else if((x45==0) && (x42!=0)&&(x43!=0)&&(x44!=0)&&(x41!=0))
begin
role4<=4'd4;
end
else if((x45!=0) && (x42!=0)&&(x43!=0)&&(x44!=0)&&(x41!=0))
begin
```

```
        role4<=4'd5;
        end
        else
        begin
        role4<=4'd0;
        end

        always@(posedge clk)
        if ((x55==0) && (x52==0)&&(x53==0)&&(x54==0)&&(x51!=0))
        begin
        role5<=4'd1;
        end
        else if((x55==0) && (x52!=0)&&(x53==0)&&(x54==0)&&(x51!=0))
        begin
        role5<=4'd2;
        end
        else if((x55==0) && (x52!=0)&&(x53!=0)&&(x54==0)&&(x51!=0))
        begin
        role5<=4'd3;
        end
        else if((x55==0) && (x52!=0)&&(x53!=0)&&(x54!=0)&&(x51!=0))
        begin
        role5<=4'd4;
        end
        else if((x55!=0) && (x52!=0)&&(x53!=0)&&(x54!=0)&&(x51!=0))
        begin
        role5<=4'd5;
        end
        else
        begin
        role5<=4'd0;
        end

        always@(posedge clk)                              //the grass piece 1
        if (((regionx>= 0 )&& (regionx <= 54) && (regiony>= 0) && (regiony <= 74)) ||
((regionx>= 110 )&& (regionx <= 164) && (regiony>= 0) && (regiony <= 74))||
((regionx>= 220 )&& (regionx <= 274) && (regiony>= 0) && (regiony <= 74))||
((regionx>= 330 )&& (regionx <= 384) && (regiony>= 0) && (regiony <= 74))||
((regionx>= 440 )&& (regionx <= 494) && (regiony>= 0) && (regiony <= 74))||
((regionx>= 0 )&& (regionx <= 54) && (regiony>= 150) && (regiony <= 224)) ||
((regionx>= 110 )&& (regionx <= 164) && (regiony>= 150) && (regiony <= 224))||
((regionx>= 220 )&& (regionx <= 274) && (regiony>= 150) && (regiony <= 224))||
((regionx>= 330 )&& (regionx <= 384) && (regiony>= 150) && (regiony <= 224))||
((regionx>= 440 )&& (regionx <= 494) && (regiony>= 150) && (regiony <= 224))||
((regionx>= 0 )&& (regionx <= 54) && (regiony>= 300) && (regiony <= 374)) ||
((regionx>= 110 )&& (regionx <= 164) && (regiony>= 300) && (regiony <= 374))||
((regionx>= 220 )&& (regionx <= 274) && (regiony>= 300) && (regiony <= 374))||
((regionx>= 330 )&& (regionx <= 384) && (regiony>= 300) && (regiony <= 374))||
((regionx>= 440 )&& (regionx <= 494) && (regiony>= 300) && (regiony <= 374))||
((regionx>= 55 )&& (regionx <= 109) && (regiony>= 75) && (regiony <= 149)) ||
((regionx>= 165 )&& (regionx <= 219) && (regiony>= 75) && (regiony <= 149))||
((regionx>= 275 )&& (regionx <= 329) && (regiony>= 75) && (regiony <= 149))||
((regionx>= 385 )&& (regionx <= 439) && (regiony>= 75) && (regiony <= 149))||
((regionx>= 495 )&& (regionx <= 549) && (regiony>= 75) && (regiony <= 149))||
((regionx>= 55 )&& (regionx <= 109) && (regiony>= 225) && (regiony <= 299)) ||
((regionx>= 165 )&& (regionx <= 219) && (regiony>= 225) && (regiony <= 299))||
((regionx>= 275 )&& (regionx <= 329) && (regiony>= 225) && (regiony <= 299))||
((regionx>= 385 )&& (regionx <= 439) && (regiony>= 225) && (regiony <= 299))||
((regionx>= 495 )&& (regionx <= 549) && (regiony>= 225) && (regiony <= 299)))       begin
        spriteon <= 1;
        end
        else begin
        spriteon <= 0;
        end

        always@(posedge clk)                                        // the grass piece 2
        if (((campusx>= 55 )&& (campusx<=109 ) && (campusy>= 0) && (campusy <= 74)) ||
        ((campusx>= 165 )&& (campusx<=219 ) && (campusy>= 0) && (campusy <= 74)) ||
        ((campusx>= 275 )&& (campusx<=329 ) && (campusy>= 0) && (campusy <= 74)) ||
        ((campusx>= 385 )&& (campusx<=439 ) && (campusy>= 0) && (campusy <= 74)) ||
        ((campusx>= 495 )&& (campusx<=549 ) && (campusy>= 0) && (campusy <= 74)) ||
        ((campusx>= 55 )&& (campusx<=109 ) && (campusy>= 150) && (campusy <= 224)) ||
        ((campusx>= 165 )&& (campusx<=219 ) && (campusy>= 150) && (campusy <= 224))
||
        ((campusx>= 275 )&& (campusx<=329 ) && (campusy>= 150) && (campusy <= 224))
||
        ((campusx>= 385 )&& (campusx<=439 ) && (campusy>= 150) && (campusy <= 224))
||
        ((campusx>= 495 )&& (campusx<=549 ) && (campusy>=150) && (campusy <= 224)) ||
        ((campusx>= 55 )&& (campusx<=109 ) && (campusy>= 300) && (campusy <= 374)) ||
        ((campusx>= 165 )&& (campusx<=219 ) && (campusy>= 300) && (campusy <= 374))
||
	((campusx>= 275 )&& (campusx<=329 ) && (campusy>= 300) && (campusy <= 374))
||
	((campusx>= 385 )&& (campusx<=439 ) && (campusy>= 300) && (campusy <= 374))
||
	((campusx>= 495 )&& (campusx<=549 ) && (campusy>= 300) && (campusy <= 374))
||
	((campusx>= 0 )&& (campusx<=54 ) && (campusy>= 75) && (campusy <= 149)) ||
	((campusx>= 110 )&& (campusx<=164 ) && (campusy>= 75) && (campusy <= 149)) ||
	((campusx>= 220 )&& (campusx<=274 ) && (campusy>= 75) && (campusy <= 149)) ||
	((campusx>= 330 )&& (campusx<=384 ) && (campusy>= 75) && (campusy <= 149)) ||
	((campusx>= 440 )&& (campusx<=494 ) && (campusy>= 75) && (campusy <= 149)) ||
	((campusx>= 0 )&& (campusx<=54 ) && (campusy>= 225) && (campusy <= 299)) ||
	((campusx>= 110 )&& (campusx<=164 ) && (campusy>= 225) && (campusy <= 299))
||
	((campusx>= 220 )&& (campusx<=274 ) && (campusy>= 225) && (campusy <= 299))
||
	((campusx>= 330 )&& (campusx<=384 ) && (campusy>= 225) && (campusy <= 299))
||
	((campusx>= 440 )&& (campusx<=494 ) && (campusy>= 225) && (campusy <= 299))
) begin
	campus <= 1;
	end
	else begin
	campus <= 0;
	end

		always@(posedge clk)				// school logo
	if ((cux>= 0 )&& (cux<=54 ) && (cuy>= 150) && (cuy <= 224)) begin
	cu <= 1;
	end
	else begin
	cu <= 0;
	end

	always@(posedge clk)
	if (((((xrow-x)>= 0 )&& ((xrow-x)<= 54) && ((yrow-y)>=0) && ((yrow-y)<= 1))
	||(((xrow-x)>= 0 )&& ((xrow-x)<= 54) && ((yrow-y)>= 73) && ((yrow-y)<= 74))
	||(((xrow-x)>= 0 )&& ((xrow-x)<= 1) && ((yrow-y)>= 0) && ((yrow-y)<= 74))
	||(((xrow-x)>= 53 )&& ((xrow-x)<= 54) && ((yrow-y)>= 0) && ((yrow-y)<=
74)))&&(a2==0)) begin
	flag <= 1;
```

```
        end
        else if (((((xrow-x)>= 0 )&& ((xrow-x)<= 54) && ((yrow-y)>=0) && ((yrow-y)<= 1))
        ||(((xrow-x)>= 0 )&& ((xrow-x)<= 54) && ((yrow-y)>= 73) && ((yrow-y)<= 74))
        ||(((xrow-x)>= 0 )&& ((xrow-x)<= 1) && ((yrow-y)>= 0) && ((yrow-y)<= 74))
        ||(((xrow-x)>= 53 )&& ((xrow-x)<= 54) && ((yrow-y)>= 0) && ((yrow-y)<=
74)))&&(a2==1))  begin
        flag <= 2;
        end
        else begin
        flag <= 0;
        end


        always@(posedge clk)                    // charater 1
        if ((xrow>= 35 )&& (xrow<=89 ) && (yrow>= 105) && (yrow <= 179)) begin
        ch1 <= 1;
        end
        else if ((((xrow-a3)>= 0 )&& ((xrow-a3)<= 54) && ((yrow-b3)>=0) && ((yrow-b3)<=
74)
        &&((run%40<=10)||(run%40>=20))&&(run!=0)&&(role1>=4'd1)&&(q1!=2))
        || (((xrow-x11)>= 0 )&& ((xrow-x11)<= 54) && ((yrow-y11)>=0) && ((yrow-y11)<=
74)
        &&((m1%40<=10)||(m1%40>=20))&&(m1!=0)&&(role1>=4'd2)&&(q2!=2))
        ||(((xrow-x12)>= 0 )&& ((xrow-x12)<= 54) && ((yrow-y12)>=0) && ((yrow-y12)<=
74)
        &&((m2%40<=10)||(m2%40>=20))&&(m2!=0)&&(role1>=4'd3)&&(q3!=2))
        ||(((xrow-x13)>= 0 )&& ((xrow-x13)<= 54) && ((yrow-y13)>=0) && ((yrow-y13)<=
74)
        &&((m3%40<=10)||(m3%40>=20))&&(m3!=0)&&(role1>=4'd4)&&(q4!=2))
        ||(((xrow-x14)>= 0 )&& ((xrow-x14)<= 54) && ((yrow-y14)>=0) && ((yrow-y14)<=
74)
        &&((m4%40<=10)||(m4%40>=20))&&(m4!=0)&&(role1==4'd5)&&(q5!=2)))
        begin
        ch1 <= 1;
        end
        else begin
        ch1 <= 0;
        end

        always@(posedge clk)                    // charater 1 move
        if ((((xrow-a3)>= 0 )&& ((xrow-a3)<= 54) && ((yrow-b3)>=0) && ((yrow-b3)<= 74)
        &&(run%40>10)&&(run%40<20)&&(run!=0)&&(role1>=4'd1)&&(q1!=2))
        || (((xrow-x11)>= 0 )&& ((xrow-x11)<= 54) && ((yrow-y11)>=0) && ((yrow-y11)<=
74)
        &&(m1%40>10)&&(m1%40<20)&&(m1!=0)&&(role1>=4'd2)&&(q2!=2))
        ||(((xrow-x12)>= 0 )&& ((xrow-x12)<= 54) && ((yrow-y12)>=0) && ((yrow-y12)<=
74)
        &&(m2%40>10)&&(m2%40<20)&&(m2!=0)&&(role1>=4'd3)&&(q3!=2))
        ||(((xrow-x13)>= 0 )&& ((xrow-x13)<= 54) && ((yrow-y13)>=0) && ((yrow-y13)<=
74)
        &&(m3%40>10)&&(m3%40<20)&&(m3!=0)&&(role1>=4'd4)&&(q4!=2))
        ||(((xrow-x14)>= 0 )&& ((xrow-x14)<= 54) && ((yrow-y14)>=0) && ((yrow-y14)<=
74)
        &&(m4%40>10)&&(m4%40<20)&&(m4!=0)&&(role1>=4'd5)&&(q5!=2)))
        begin
        ch1mov <= 1;
        end
        else begin
        ch1mov <= 0;
        end

        always@(posedge clk)                    // charater 2
        if ((xrow>= 35 )&& (xrow<=89 ) && (yrow>= 330) && (yrow <= 404)) begin
        ch4 <= 1;
        end
        else if ((((xrow-x41)>= 0 )&& ((xrow-x41)<= 54) && ((yrow-y41)>=0) && ((yrow-
y41)<= 74)
        &&((m41%30<=10)||(m41%30>=20))&&(m41!=0)&&(role4>=4'd1)&&(q11!=2))
        || (((xrow-x42)>= 0 )&& ((xrow-x42)<= 54) && ((yrow-y42)>=0) && ((yrow-y42)<=
74)
        &&((m42%30<=10)||(m42%30>=20))&&(m42!=0)&&(role4>=4'd2)&&(q12!=2))
        ||(((xrow-x43)>= 0 )&& ((xrow-x43)<= 54) && ((yrow-y43)>=0) && ((yrow-y43)<=
74)
        &&((m43%30<=10)||(m43%30>=20))&&(m43!=0)&&(role4>=4'd3)&&(q13!=2))
        ||(((xrow-x44)>= 0 )&& ((xrow-x44)<= 54) && ((yrow-y44)>=0) && ((yrow-y44)<=
74)
        &&((m44%30<=10)||(m44%30>=20))&&(m44!=0)&&(role4>=4'd4)&&(q14!=2))
        ||(((xrow-x45)>= 0 )&& ((xrow-x45)<= 54) && ((yrow-y45)>=0) && ((yrow-y45)<=
74)
        &&((m45%30<=10)||(m45%30>=20))&&(m45!=0)&&(role4==4'd5)&&(q15!=2)))
        begin
        ch4 <= 1;
        end
        else begin
        ch4 <= 0;
        end

        always@(posedge clk)                    // charater 2 move
        if ((((xrow-x41)>= 0 )&& ((xrow-x41)<= 54) && ((yrow-y41)>=0) && ((yrow-y41)<=
74)
        &&(m41%30>10)&&(m41%30<20)&&(m41!=0)&&(role4>=4'd1)&&(q11!=2))
        || (((xrow-x42)>= 0 )&& ((xrow-x42)<= 54) && ((yrow-y42)>=0) && ((yrow-y42)<=
74)
        &&(m42%30>10)&&(m42%30<20)&&(m42!=0)&&(role4>=4'd2)&&(q12!=2))
        ||(((xrow-x43)>= 0 )&& ((xrow-x43)<= 54) && ((yrow-y43)>=0) && ((yrow-y43)<=
74)
        &&(m43%30>10)&&(m43%30<20)&&(m43!=0)&&(role4>=4'd3)&&(q13!=2))
        ||(((xrow-x44)>= 0 )&& ((xrow-x44)<= 54) && ((yrow-y44)>=0) && ((yrow-y44)<=
74)
        &&(m44%30>10)&&(m44%30<20)&&(m44!=0)&&(role4>=4'd4)&&(q14!=2))
        ||(((xrow-x45)>= 0 )&& ((xrow-x45)<= 54) && ((yrow-y45)>=0) && ((yrow-y45)<=
74)
        &&(m45%30>10)&&(m45%30<20)&&(m45!=0)&&(role4>=4'd5)&&(q15!=2)))
        begin
        ch4mov <= 1;
        end
        else begin
        ch4mov <= 0;
        end


        always@(posedge clk)                    // charater 3
        if ((xrow>= 35 )&& (xrow<=89 ) && (yrow>= 180) && (yrow <= 254)) begin
        ch3 <= 1;
        end
        else if ((((xrow-x31)>= 0 )&& ((xrow-x31)<= 54) && ((yrow-y31)>=0) && ((yrow-
y31)<= 74)
        &&((m31%100<=20)||(m31%100>=40))&&(m31!=0)&&(role3>=4'd1)&&(q6!=2))
        || (((xrow-x32)>= 0 )&& ((xrow-x32)<= 54) && ((yrow-y32)>=0) && ((yrow-y32)<=
74)
        &&((m32%100<=20)||(m32%100>=40))&&(m32!=0)&&(role3>=4'd2)&&(q7!=2))
        ||(((xrow-x33)>= 0 )&& ((xrow-x33)<= 54) && ((yrow-y33)>=0) && ((yrow-y33)<=
74)
```

```
        &&((m33%100<=20)||(m33%100>=40))&&(m33!=0)&&(role3>=4'd3)&&(q8!=2))
        ||(((xrow-x34)>= 0 )&& ((xrow-x34)<= 54) && ((yrow-y34)>=0) && ((yrow-y34)<=
74)
        &&((m34%100<=20)||(m34%100>=40))&&(m34!=0)&&(role3>=4'd4)&&(q9!=2))
        ||(((xrow-x35)>= 0 )&& ((xrow-x35)<= 54) && ((yrow-y35)>=0) && ((yrow-y35)<=
74)
        &&((m35%100<=20)||(m35%100>=40))&&(m35!=0)&&(role3==4'd5)&&(q10!=2)))
        begin
        ch3 <= 1;
        end
        else begin
        ch3 <= 0;
        end

        always@(posedge clk)                    // charater 3 move
        if  ((((xrow-x31)>= 0 )&& ((xrow-x31)<= 54) && ((yrow-y31)>=0) && ((yrow-y31)<=
74)
        &&(m31%100>20)&&(m31%100<40)&&(m31!=0)&&(role3>=4'd1)&&(q6!=2))
        || (((xrow-x32)>= 0 )&& ((xrow-x32)<= 54) && ((yrow-y32)>=0) && ((yrow-y32)<=
74)
        &&(m32%100>20)&&(m32%100<40)&&(m32!=0)&&(role3>=4'd2)&&(q7!=2))
        ||(((xrow-x33)>= 0 )&& ((xrow-x33)<= 54) && ((yrow-y33)>=0) && ((yrow-y33)<=
74)
        &&(m33%100>20)&&(m33%100<40)&&(m33!=0)&&(role3>=4'd3)&&(q8!=2))
        ||(((xrow-x34)>= 0 )&& ((xrow-x34)<= 54) && ((yrow-y34)>=0) && ((yrow-y34)<=
74)
        &&(m34%100>20)&&(m34%100<40)&&(m34!=0)&&(role3>=4'd4)&&(q9!=2))
        ||(((xrow-x35)>= 0 )&& ((xrow-x35)<= 54) && ((yrow-y35)>=0) && ((yrow-y35)<=
74)
        &&(m35%100>20)&&(m35%100<40)&&(m35!=0)&&(role3>=4'd5)&&(q10!=2)))
        begin
        ch3mov <= 1;
        end
        else begin
        ch3mov <= 0;
        end

         always@(posedge clk)                  // charater 4
        if ((xrow>= 35 )&& (xrow<=89 ) && (yrow>= 405) && (yrow <= 479)) begin
        ch5 <= 1;
        end
```

```
else if  ((((xrow-x51)>= 0 )&& ((xrow-x51)<= 54) && ((yrow-y51)>=0) && ((yrow-y51)<= 74)
&&((m51%30<=10)||(m51%30>=20))&&(m51!=0)&&(role5>=4'd1)&&(q16!=2))
|| (((xrow-x52)>= 0 )&& ((xrow-x52)<= 54) && ((yrow-y52)>=0) && ((yrow-y52)<= 74)
&&((m52%30<=10)||(m52%30>=20))&&(m52!=0)&&(role5>=4'd2)&&(q17!=2))
||(((xrow-x53)>= 0 )&& ((xrow-x53)<= 54) && ((yrow-y53)>=0) && ((yrow-y53)<= 74)
&&((m53%30<=10)||(m53%30>=20))&&(m53!=0)&&(role5>=4'd3)&&(q18!=2))
||(((xrow-x54)>= 0 )&& ((xrow-x54)<= 54) && ((yrow-y54)>=0) && ((yrow-y54)<= 74)
&&((m54%30<=10)||(m54%30>=20))&&(m54!=0)&&(role5>=4'd4)&&(q19!=2))
||(((xrow-x55)>= 0 )&& ((xrow-x55)<= 54) && ((yrow-y55)>=0) && ((yrow-y55)<= 74)
&&((m55%30<=10)||(m55%30>=20))&&(m55!=0)&&(role5==4'd5)&&(q20!=2)))
begin
ch5 <= 1;
end
else begin
ch5 <= 0;
end


always@(posedge clk)                    // charater 4move
if  ((((xrow-x51)>= 0 )&& ((xrow-x51)<= 54) && ((yrow-y51)>=0) && ((yrow-y51)<= 74)
&&(m51%30>10)&&(m51%30<20)&&(m51!=0)&&(role5>=4'd1)&&(q16!=2))
|| (((xrow-x52)>= 0 )&& ((xrow-x52)<= 54) && ((yrow-y52)>=0) && ((yrow-y52)<= 74)
&&(m52%30>10)&&(m52%30<20)&&(m52!=0)&&(role5>=4'd2)&&(q17!=2))
||(((xrow-x53)>= 0 )&& ((xrow-x53)<= 54) && ((yrow-y53)>=0) && ((yrow-y53)<= 74)
&&(m53%30>10)&&(m53%30<20)&&(m53!=0)&&(role5>=4'd3)&&(q18!=2))
||(((xrow-x54)>= 0 )&& ((xrow-x54)<= 54) && ((yrow-y54)>=0) && ((yrow-y54)<= 74)
&&(m54%30>10)&&(m54%30<20)&&(m54!=0)&&(role5>=4'd4)&&(q19!=2))
||(((xrow-x55)>= 0 )&& ((xrow-x55)<= 54) && ((yrow-y55)>=0) && ((yrow-y55)<= 74)
&&(m55%30>10)&&(m55%30<20)&&(m55!=0)&&(role5>=4'd5)&&(q20!=2)))
begin
ch5mov <= 1;
```

```
        end
        else begin
        ch5mov <= 0;
        end

        always@(posedge clk)                    // dead of plant
        if  ((((xrow-x51)>= 0 )&& ((xrow-x51)<= 54) && ((yrow-y51)>=0) && ((yrow-y51)<=
74)
        &&(m51!=0)&&(role5>=4'd1)&&(q16==1))
        || (((xrow-x52)>= 0 )&& ((xrow-x52)<= 54) && ((yrow-y52)>=0) && ((yrow-y52)<=
74)
        &&(m52!=0)&&(role5>=4'd2)&&(q17==1))
        ||(((xrow-x53)>= 0 )&& ((xrow-x53)<= 54) && ((yrow-y53)>=0) && ((yrow-y53)<=
74)
        &&(m53!=0)&&(role5>=4'd2)&&(q18==1))
        ||(((xrow-x54)>= 0 )&& ((xrow-x54)<= 54) && ((yrow-y54)>=0) && ((yrow-y54)<=
74)
        &&(m54!=0)&&(role5>=4'd2)&&(q19==1))
        ||(((xrow-x55)>= 0 )&& ((xrow-x55)<= 54) && ((yrow-y55)>=0) && ((yrow-y55)<=
74)
        &&(m55!=0)&&(role5>=4'd2)&&(q20==1)))
        begin
        eat <= 1;
        end
        else if((((xrow-x31)>= 0 )&& ((xrow-x31)<= 54) && ((yrow-y31)>=0) && ((yrow-
y31)<= 74)
        &&(m31!=0)&&(role3>=4'd1)&&(q6==1))
        || (((xrow-x32)>= 0 )&& ((xrow-x32)<= 54) && ((yrow-y32)>=0) && ((yrow-y32)<=
74)
        &&(m32!=0)&&(role3>=4'd2)&&(q7==1))
        ||(((xrow-x33)>= 0 )&& ((xrow-x33)<= 54) && ((yrow-y33)>=0) && ((yrow-y33)<=
74)
        &&(m33!=0)&&(role3>=4'd3)&&(q8==1))
        ||(((xrow-x34)>= 0 )&& ((xrow-x34)<= 54) && ((yrow-y34)>=0) && ((yrow-y34)<=
74)
        &&(m34!=0)&&(role3>=4'd4)&&(q9==1))
        ||(((xrow-x35)>= 0 )&& ((xrow-x35)<= 54) && ((yrow-y35)>=0) && ((yrow-y35)<=
74)
        &&(m35!=0)&&(role3>=4'd5)&&(q10==1)))
        begin
        eat <= 1;
        end
        else if((((xrow-x41)>= 0 )&& ((xrow-x41)<= 54) && ((yrow-y41)>=0) && ((yrow-
y41)<= 74)
        &&(m41!=0)&&(role4>=4'd1)&&(q11==1))
        || (((xrow-x42)>= 0 )&& ((xrow-x42)<= 54) && ((yrow-y42)>=0) && ((yrow-y42)<=
74)
        &&(m42!=0)&&(role4>=4'd2)&&(q12==1))
        ||(((xrow-x43)>= 0 )&& ((xrow-x43)<= 54) && ((yrow-y43)>=0) && ((yrow-y43)<=
74)
        &&(m43!=0)&&(role4>=4'd3)&&(q13==1))
        ||(((xrow-x44)>= 0 )&& ((xrow-x44)<= 54) && ((yrow-y44)>=0) && ((yrow-y44)<=
74)
        &&(m44!=0)&&(role4>=4'd4)&&(q14==1))
        ||(((xrow-x45)>= 0 )&& ((xrow-x45)<= 54) && ((yrow-y45)>=0) && ((yrow-y45)<=
74)
        &&(m45!=0)&&(role4>=4'd5)&&(q15==1)))
        begin
        eat <= 1;
        end
        else if((((xrow-a3)>= 0 )&& ((xrow-a3)<= 54) && ((yrow-b3)>=0) && ((yrow-b3)<=
74)
        &&(run!=0)&&(role1>=4'd1)&&(q1==1))
        || (((xrow-x11)>= 0 )&& ((xrow-x11)<= 54) && ((yrow-y11)>=0) && ((yrow-y11)<=
74)
        &&(m1!=0)&&(role1>=4'd2)&&(q2==1))
        ||(((xrow-x12)>= 0 )&& ((xrow-x12)<= 54) && ((yrow-y12)>=0) && ((yrow-y12)<=
74)
        &&(m2!=0)&&(role1>=4'd3)&&(q3==1))
        ||(((xrow-x13)>= 0 )&& ((xrow-x13)<= 54) && ((yrow-y13)>=0) && ((yrow-y13)<=
74)
        &&(m3!=0)&&(role1>=4'd4)&&(q4==1))
        ||(((xrow-x14)>= 0 )&& ((xrow-x14)<= 54) && ((yrow-y14)>=0) && ((yrow-y14)<=
74)
        &&(m4!=0)&&(role1>=4'd5)&&(q5==1)))
        begin
        eat <= 1;
        end
        else begin
        eat <= 0;
        end
        always@(posedge clk)                                    // monster 1
        if  ((mon11x>=  0  )&&  (mon11x<=54  )  &&  (mon11y>=  0)  &&  (mon11y  <=
74)&&(g51%2==0)&&(p17==0)) begin
        addr_mon1<=mon11y*55+mon11x;
        mon1 <=1;
        end
        else  if  ((mon12x>=  0  )&&  (mon12x<=54  )  &&  (mon12y>=  0)  &&  (mon12y  <=
74)&&(g12%2==0)&&(p2==0)) begin
        addr_mon1<=mon12y*55+mon12x;
        mon1 <=1;
        end
        else  if  ((mon13x>=  0  )&&  (mon13x<=54  )  &&  (mon13y>=  0)  &&  (mon13y  <=
74)&&(g42%2==0)&&(p14==0)) begin
        addr_mon1<=mon13y*55+mon13x;
        mon1 <=1;
        end
        else  if  ((mon14x>=  0  )&&  (mon14x<=54  )  &&  (mon14y>=  0)  &&  (mon14y  <=
74)&&(g13%2==0)&&(p3==0)) begin
        addr_mon1<=mon14y*55+mon14x;
        mon1 <=1;
        end
        else  if  ((mon15x>=  0  )&&  (mon15x<=54  )  &&  (mon15y>=  0)  &&  (mon15y  <=
74)&&(g34%2==0)&&(p12==0)) begin
        addr_mon1<=mon15y*55+mon15x;
        mon1 <=1;
        end
        else  if  ((mon16x>=  0  )&&  (mon16x<=54  )  &&  (mon16y>=  0)  &&  (mon16y  <=
74)&&(g43%2==0)&&(p15==0)) begin
        addr_mon1<=mon16y*55+mon16x;
        mon1 <=1;
        end
        else begin
        mon1 <= 0;
        end

        always@(posedge clk)                                    // monster 1move
        if  ((mon11movx>=  0  )&&  (mon11movx<=54  )  &&  (mon11movy>=  0)  &&
(mon11movy <= 74)&&(g51%2==1)&&(p17==0)) begin
        addr_mon1mov<=mon11movy*55+mon11movx;
        mon1mov <=1;
        end
        else if ((mon12movx>= 0 )&& (mon12movx<=54 ) && (mon12movy>= 0) &&
(mon12movy <= 74)&&(g12%2==1)&&(p2==0)) begin
        addr_mon1mov<=mon12movy*55+mon12movx;
        mon1mov <=1;
        end
        else if ((mon13movx>= 0 )&& (mon13movx<=54 ) && (mon13movy>= 0) &&
(mon13movy <= 74)&&(g42%2==1)&&(p14==0)) begin
        addr_mon1mov<=mon13movy*55+mon13movx;
        mon1mov <=1;
        end
        else if ((mon14movx>= 0 )&& (mon14movx<=54 ) && (mon14movy>= 0) &&
(mon14movy <= 74)&&(g13%2==1)&&(p3==0)) begin
        addr_mon1mov<=mon14movy*55+mon14movx;
        mon1mov <=1;
        end
        else if ((mon15movx>= 0 )&& (mon15movx<=54 ) && (mon15movy>= 0) &&
(mon15movy <= 74)&&(g34%2==1)&&(p12==0)) begin
        addr_mon1mov<=mon15movy*55+mon15movx;
        mon1mov <=1;
        end
        else if ((mon16movx>= 0 )&& (mon16movx<=54 ) && (mon16movy>= 0) &&
(mon16movy <= 74)&&(g43%2==1)&&(p15==0)) begin
        addr_mon1mov<=mon16movy*55+mon16movx;
        mon1mov <=1;
        end
        else begin
        mon1mov <= 0;
        end

        always@(posedge clk)                        // monster 2
        if ((mon41x>= 0 )&& (mon41x<=54 ) && (mon41y>= 0) && (mon41y <=
74)&&(g11%2==0)&&(p1==0)) begin
        addr_mon4<=mon41y*55+mon41x;
        mon4 <=1;
        end
        else if ((mon42x>= 0 )&& (mon42x<=54 ) && (mon42y>= 0) && (mon42y <=
74)&&(g21%2==0)&&(p5==0)) begin
        addr_mon4<=mon42y*55+mon42x;
        mon4 <=1;
        end
        else if ((mon43x>= 0 )&& (mon43x<=54 ) && (mon43y>= 0) && (mon43y <=
74)&&(g22%2==0)&&(p6==0)) begin
        addr_mon4<=mon43y*55+mon43x;
        mon4 <=1;
        end
        else if ((mon44x>= 0 )&& (mon44x<=54 ) && (mon44y>= 0) && (mon44y <=
74)&&(g52%2==0)&&(p18==0)) begin
        addr_mon4<=mon44y*55+mon44x;
        mon4 <=1;
        end
        else if ((mon45x>= 0 )&& (mon45x<=54 ) && (mon45y>= 0) && (mon45y <=
74)&&(g23%2==0)&&(p7==0)) begin
        addr_mon4<=mon45y*55+mon45x;
        mon4 <=1;
        end
        else if ((mon46x>= 0 )&& (mon46x<=54 ) && (mon46y>= 0) && (mon46y <=
74)&&(g24%2==0)&&(p8==0)) begin
        addr_mon4<=mon46y*55+mon46x;
        mon4 <=1;
        end
        else begin
        mon4 <= 0;
        end

        always@(posedge clk)                    // monster 2 move
        if ((mon41movx>= 0 )&& (mon41movx<=54 ) && (mon41movy>= 0) &&
(mon41movy <= 74)&&(g11%2==1)&&(p1==0)) begin
        addr_mon4mov<=mon41movy*55+mon41movx;
        mon4mov <=1;
        end
        else if ((mon42movx>= 0 )&& (mon42movx<=54 ) && (mon42movy>= 0) &&
(mon42movy <= 74)&&(g21%2==1)&&(p5==0)) begin
   addr_mon4mov<=mon42movy*55+mon42movx;
        mon4mov <=1;
        end
        else if ((mon43movx>= 0 )&& (mon43movx<=54 ) && (mon43movy>= 0) &&
(mon43movy <= 74)&&(g22%2==1)&&(p6==0)) begin
        addr_mon4mov<=mon43movy*55+mon43movx;
        mon4mov <=1;
        end
        else if ((mon44movx>= 0 )&& (mon44movx<=54 ) && (mon44movy>= 0) &&
(mon44movy <= 74)&&(g52%2==1)&&(p18==0)) begin
        addr_mon4mov<=mon44movy*55+mon44movx;
        mon4mov <=1;
        end
        else if ((mon45movx>= 0 )&& (mon45movx<=54 ) && (mon45movy>= 0) &&
(mon45movy <= 74)&&(g23%2==1)&&(p7==0)) begin
        addr_mon4mov<=mon45movy*55+mon45movx;
        mon4mov <=1;
        end
        else if ((mon46movx>= 0 )&& (mon46movx<=54 ) && (mon46movy>= 0) &&
(mon46movy <= 74)&&(g24%2==1)&&(p8==0)) begin
        addr_mon4mov<=mon46movy*55+mon46movx;
        mon4mov <=1;
        end
        else begin
        mon4mov <= 0;
        end

        always@(posedge clk)                              // monster 3
        if ((mon51x>= 0 )&& (mon51x<=54 ) && (mon51y>= 0) && (mon51y <=
74)&&(g31%2==0)&&(p9==0)) begin
        addr_mon5<=mon51y*55+mon51x;
        mon5 <=1;
        end
        else if ((mon52x>= 0 )&& (mon52x<=54 ) && (mon52y>= 0) && (mon52y <=
74)&&(g41%2==0)&&(p13==0)) begin
        addr_mon5<=mon52y*55+mon52x;
        mon5 <=1;
        end
        else if ((mon53x>= 0 )&& (mon53x<=54 ) && (mon53y>= 0) && (mon53y <=
74)&&(g32%2==0)&&(p10==0)) begin
        addr_mon5<=mon53y*55+mon53x;
        mon5 <=1;
        end
        else if ((mon54x>= 0 )&& (mon54x<=54 ) && (mon54y>= 0) && (mon54y <=
74)&&(g53%2==0)&&(p19==0)) begin
        addr_mon5<=mon54y*55+mon54x;
        mon5 <=1;
        end
        else if ((mon55x>= 0 )&& (mon55x<=54 ) && (mon55y>= 0) && (mon55y <=
74)&&(g33%2==0)&&(p11==0)) begin
        addr_mon5<=mon55y*55+mon55x;
        mon5 <=1;
        end
        else if ((mon56x>= 0 )&& (mon56x<=54 ) && (mon56y>= 0) && (mon56y <=
74)&&(g14%2==0)&&(p4==0)) begin
        addr_mon5<=mon56y*55+mon56x;
        mon5 <=1;
        end
        else begin
        mon5 <= 0;
        end

        always@(posedge clk)                                // monster 3move
        if ((mon51movx>= 0 )&& (mon51movx<=54 ) && (mon51movy>= 0) &&
(mon51movy <= 74)&&(g31%2==1)&&(p9==0)) begin
        addr_mon5mov<=mon51movy*55+mon51movx;
        mon5mov <=1;
        end
        else if ((mon52movx>= 0 )&& (mon52movx<=54 ) && (mon52movy>= 0) &&
(mon52movy <= 74)&&(g41%2==1)&&(p13==0)) begin
        addr_mon5mov<=mon52movy*55+mon52movx;
        mon5mov <=1;
        end
        else if ((mon53movx>= 0 )&& (mon53movx<=54 ) && (mon53movy>= 0) &&
(mon53movy <= 74)&&(g32%2==1)&&(p10==0)) begin
        addr_mon5mov<=mon53movy*55+mon53movx;
        mon5mov <=1;
        end
        else if ((mon54movx>= 0 )&& (mon54movx<=54 ) && (mon54movy>= 0) &&
(mon54movy <= 74)&&(g53%2==1)&&(p19==0)) begin
        addr_mon5mov<=mon54movy*55+mon54movx;
        mon5mov <=1;
        end
        else if ((mon55movx>= 0 )&& (mon55movx<=54 ) && (mon55movy>= 0) &&
(mon55movy <= 74)&&(g33%2==1)&&(p11==0)) begin
        addr_mon5mov<=mon55movy*55+mon55movx;
        mon5mov <=1;
        end
```

```
	else if ((mon56movx>= 0 )&& (mon56movx<=54 ) && (mon56movy>= 0) &&
(mon56movy <= 74)&&(g14%2==1)&&(p4==0)) begin
	addr_mon5mov<=mon56movy*55+mon56movx;
	mon5mov <=1;
	end
	else begin
	mon5mov <= 0;
	end



	always@(posedge clk)					// dead of monster
	if ((mon11x>= 0 )&& (mon11x<=54 ) && (mon11y>= 0) && (mon11y <=
74)&&(p17==1)) begin
	addr_dead<=mon11y*55+mon11x;
	dead <=1;
	end
	else if ((mon12x>= 0 )&& (mon12x<=54 ) && (mon12y>= 0) && (mon12y <=
74)&&(p2==1)) begin
	addr_dead<=mon12y*55+mon12x;
	dead <=1;
	end
	else if ((mon13x>= 0 )&& (mon13x<=54 ) && (mon13y>= 0) && (mon13y <=
74)&&(p14==1)) begin
	addr_dead<=mon13y*55+mon13x;
	dead <=1;
	end
	else if ((mon14x>= 0 )&& (mon14x<=54 ) && (mon14y>= 0) && (mon14y <=
74)&&(p3==1)) begin
	addr_dead<=mon14y*55+mon14x;
	dead <=1;
	end
	else if ((mon15x>= 0 )&& (mon15x<=54 ) && (mon15y>= 0) && (mon15y <=
74)&&(p12==1)) begin
	addr_dead<=mon15y*55+mon15x;
	dead <=1;
	end
	else if ((mon16x>= 0 )&& (mon16x<=54 ) && (mon16y>= 0) && (mon16y <=
74)&&(p15==1)) begin
	addr_dead<=mon16y*55+mon16x;
	dead <=1;
	end
	else if ((mon41x>= 0 )&& (mon41x<=54 ) && (mon41y>= 0) && (mon41y <=
74)&&(p1==1)) begin
	addr_dead<=mon41y*55+mon41x;
	dead <=1;
	end
	else if ((mon42x>= 0 )&& (mon42x<=54 ) && (mon42y>= 0) && (mon42y <=
74)&&(p5==1)) begin
	addr_dead<=mon42y*55+mon42x;
	dead <=1;
	end
	else if ((mon43x>= 0 )&& (mon43x<=54 ) && (mon43y>= 0) && (mon43y <=
74)&&(p6==1)) begin
	addr_dead<=mon43y*55+mon43x;
	dead <=1;
	end
	else if ((mon44x>= 0 )&& (mon44x<=54 ) && (mon44y>= 0) && (mon44y <=
74)&&(p18==1)) begin
	addr_dead<=mon44y*55+mon44x;
	dead <=1;
	end
	else if ((mon45x>= 0 )&& (mon45x<=54 ) && (mon45y>= 0) && (mon45y <=
74)&&(p7==1)) begin
	addr_dead<=mon45y*55+mon45x;
	dead <=1;
	end
	else if ((mon46x>= 0 )&& (mon46x<=54 ) && (mon46y>= 0) && (mon46y <=
74)&&(p8==1)) begin
	addr_dead<=mon46y*55+mon46x;
	dead <=1;
	end
	else if ((mon51x>= 0 )&& (mon51x<=54 ) && (mon51y>= 0) && (mon51y <=
74)&&(p9==1)) begin
	addr_dead<=mon51y*55+mon51x;
	dead <=1;
	end
	else if ((mon52x>= 0 )&& (mon52x<=54 ) && (mon52y>= 0) && (mon52y <=
74)&&(p13==1)) begin
	addr_dead <=mon52y*55+mon52x;
	dead <=1;
	end
	else if ((mon53x>= 0 )&& (mon53x<=54 ) && (mon53y>= 0) && (mon53y <=
74)&&(p10==1)) begin
	addr_dead<=mon53y*55+mon53x;
	dead <=1;
	end
	else if ((mon54x>= 0 )&& (mon54x<=54 ) && (mon54y>= 0) && (mon54y <=
74)&&(p19==1)) begin
	addr_dead<=mon54y*55+mon54x;
	dead <=1;
	end
	else if ((mon55x>= 0 )&& (mon55x<=54 ) && (mon55y>= 0) && (mon55y <=
74)&&(p11==1)) begin
	addr_dead<=mon55y*55+mon55x;
	dead <=1;
	end
	else if ((mon56x>= 0 )&& (mon56x<=54 ) && (mon56y>= 0) && (mon56y <=
74)&&(p4==1)) begin
	addr_dead<=mon56y*55+mon56x;
	dead <=1;
	end
	else begin
	dead <= 0;
	end

	always@(posedge clk)                          // the brown background
	if (((bg2x>= 0 )&& (bg2x<=639 ) && (bg2y>= 0) && (bg2y <= 14))||
	((bg2x>= 0 )&& (bg2x<=34 ) && (bg2y>= 15) && (bg2y <= 389)))
	begin
	bg2 <= 1;
	end
	else begin
	bg2 <= 0;
	end


	always@(posedge clk)                   // bullet
	if (((xrow-hex2)>= 0 )&& ((xrow-hex2)<= 24 ) && ((yrow-hex3)>= 0) && ((yrow-
hex3) <= 24)&&(role1>=4'd1)
```

```
        &&(run>=11)&&(run!=0)&&(((hex2<g1+20)&&(b3==105)&&(g1!=0))||((hex2<g2+20)
&&(b3==180)&&(g2!=0))

        ||((hex2<g3+20)&&(b3==255)&&(g3!=0))||((hex2<g4+20)&&(b3==330)&&(g4!=0))||((
hex2<g5+20)&&(b3==405)&&(g5!=0))))
        begin
        att1 <= 1;
        end
        else if(((xrow-hex2)>= 0 )&& ((xrow-hex2)<= 24 ) && ((yrow-hex3)>= 0) && ((yrow-
hex3) <= 24)&&(role1>=4'd1)

        &&(run>=11)&&(run!=0)&&(((g1==0)&&(b3==105))||((g2==0)&&(b3==180))||((g3==
0)&&(b3==255))||((g4==0)&&(b3==330))||
        ((g5==0)&&(b3==405))))
        begin
        att1 <= 1;
        end
        else begin
        att1 <= 0;
        end

         always@(posedge clk)
        if (((xrow-z11)>= 0 )&& ((xrow-z11)<= 24 ) && ((yrow-d11)>= 0) && ((yrow-d11) <=
24)&&(role1>=4'd2)
        &&(m1>=11)&&(m1!=0)&&(((z11<g1+20)&&(y11==105)&&(g1!=0))||

        ((z11<g2+20)&&(y11==180)&&(g2!=0))||((z11<g3+20)&&(y11==255)&&(g3!=0))||((z
11<g4+20)&&(y11==330)&&(g4!=0))||
        ((z11<g5+20)&&(y11==405)&&(g5!=0))))
        begin
        att2 <= 1;
        end
        else if (((xrow-z11)>= 0 )&& ((xrow-z11)<= 24 ) && ((yrow-d11)>= 0) && ((yrow-d11)
<= 24)&&(role1>=4'd2)

        &&(m1>=11)&&(m1!=0)&&(((g1==0)&&(y11==105))||((g2==0)&&(y11==180))||((g3
==0)&&(y11==255))||((g4==0)&&(y11==330))||
        ((g5==0)&&(y11==405))))
        begin
        att2 <= 1;
        end
        else begin
        att2 <= 0;
        end

        always@(posedge clk)
        if (((xrow-z12)>= 0 )&& ((xrow-z12)<= 24 ) && ((yrow-d12)>= 0) && ((yrow-d12) <=
24)&&(role1>=4'd3)
        &&(m2>=11)&&(m2!=0)&&(((z12<g1+20)&&(y12==105)&&(g1!=0))||

        ((z12<g2+20)&&(y12==180)&&(g2!=0))||((z12<g3+20)&&(y12==255)&&(g3!=0))||((z
12<g4+20)&&(y12==330)&&(g4!=0))||
        ((z12<g5+20)&&(y12==405)&&(g5!=0))))
        begin
        att3 <= 1;
        end
        else if (((xrow-z12)>= 0 )&& ((xrow-z12)<= 24 ) && ((yrow-d12)>= 0) && ((yrow-d12)
<= 24)&&(role1>=4'd3)

        &&(m2>=11)&&(m2!=0)&&(((g1==0)&&(y12==105))||((g2==0)&&(y12==180))||((g3
==0)&&(y12==255))||((g4==0)&&(y12==330))||
        ((g5==0)&&(y12==405))))
        begin
        att3 <= 1;
        end
        else begin
        att3 <= 0;
        end

        always@(posedge clk)
        if (((xrow-z13)>= 0 )&& ((xrow-z13)<= 24 ) && ((yrow-d13)>= 0) && ((yrow-d13) <=
24)&&(role1>=4'd4)
        &&(m3>=11)&&(m3!=0)&&(((z13<g1+20)&&(y13==105)&&(g1!=0))||

        ((z13<g2+20)&&(y13==180)&&(g2!=0))||((z13<g3+20)&&(y13==255)&&(g3!=0))||((z
13<g4+20)&&(y13==330)&&(g4!=0))||
        ((z13<g5+20)&&(y13==405)&&(g5!=0))))
        begin
        att4 <= 1;
        end
        else if (((xrow-z13)>= 0 )&& ((xrow-z13)<= 24 ) && ((yrow-d13)>= 0) && ((yrow-d13)
<= 24)&&(role1>=4'd4)

        &&(m3>=11)&&(m3!=0)&&(((g1==0)&&(y13==105))||((g2==0)&&(y13==180))||((g3
==0)&&(y13==255))||((g4==0)&&(y13==330))||
        ((g5==0)&&(y13==405))))
        begin
        att4 <= 1;
        end
        else begin
        att4 <= 0;
        end

        always@(posedge clk)
        if (((xrow-z14)>= 0 )&& ((xrow-z14)<= 24 ) && ((yrow-d14)>= 0) && ((yrow-d14) <=
24)&&(role1>=4'd5)
        &&(m4>=11)&&(m4!=0)&&(((z14<g1+20)&&(y14==105)&&(g1!=0))||

        ((z14<g2+20)&&(y14==180)&&(g2!=0))||((z14<g3+20)&&(y14==255)&&(g3!=0))||((z
14<g4+20)&&(y14==330)&&(g4!=0))||
        ((z14<g5+20)&&(y14==405)&&(g5!=0))))
        begin
        att5 <= 1;
        end
        else if (((xrow-z14)>= 0 )&& ((xrow-z14)<= 24 ) && ((yrow-d14)>= 0) && ((yrow-d14)
<= 24)&&(role1>=4'd5)

        &&(m4>=11)&&(m4!=0)&&(((g1==0)&&(y14==105))||((g2==0)&&(y14==180))||((g3
==0)&&(y14==255))||((g4==0)&&(y14==330))||
        ((g5==0)&&(y14==405))))
        begin
        att5 <= 1;
        end
        else begin
        att5 <= 0;
        end

        always@(posedge clk)
        if                          (((xrow>=90)&&(xrow<=639)&&(yrow>=d31)&&(yrow-
24<=d31)&&(m31%100>=20)&&(m31%100<=40)&&(role3>=4'd1)&& (q6!=2))
```

```
	||((xrow>=90)&&(xrow<=639)&&(yrow>=d32)&&(yrow-
24<=d32)&&(m32%100>=20)&&(m32%100<=40)&&(role3>=4'd2)&& (q7!=2))
	||((xrow>=90)&&(xrow<=639)&&(yrow>=d33)&&(yrow-
24<=d33)&&(m33%100>=20)&&(m33%100<=40)&&(role3>=4'd3)&& (q8!=2))
	||((xrow>=90)&&(xrow<=639)&&(yrow>=d34)&&(yrow-
24<=d34)&&(m34%100>=20)&&(m34%100<=40)&&(role3>=4'd4)&& (q9!=2))
	||((xrow>=90)&&(xrow<=639)&&(yrow>=d35)&&(yrow-
24<=d35)&&(m35%100>=20)&&(m35%100<=40)&&(role3>=4'd5)&& (q10!=2)))
	begin
	fire<=1;
	end
	else begin
	fire<=0;
	end

	always@(posedge clk)
	if                              (((xrow-z51-10)*(xrow-z51-10)+(yrow-d51-10)*(yrow-d51-
10)<=100)&&(role5>=4'd1)
	&&(m51>=11)&&(m51!=0)&&(((z51<g1+20)&&(y51==105)&&(g1!=0))||

	((z51<g2+20)&&(y51==180)&&(g2!=0))||((z51<g3+20)&&(y51==255)&&(g3!=0))||((z
51<g4+20)&&(y51==330)&&(g4!=0))||
	((z51<g5+20)&&(y51==405)&&(g5!=0))))
	begin
	ice1 <= 1;
	end
	else                          if(((xrow-z51-10)*(xrow-z51-10)+(yrow-d51-10)*(yrow-d51-
10)<=100)&&(role5>=4'd1)

&&(m51>=11)&&(m51!=0)&&(((g1==0)&&(y51==105))||((g2==0)&&(y51==180))||((g3==0)
&&(y51==255))||((g4==0)&&(y51==330))||
	((g5==0)&&(y51==405))))
	begin
	ice1 <= 1;
	end
	else begin
	ice1 <= 0;
	end

	always@(posedge clk)
	if                          (((xrow-z52-10)*(xrow-z52-10)+(yrow-d52-10)*(yrow-d52-
10)<=100)&&(role5>=4'd2)
	&&(m52>=11)&&(m52!=0)&&(((z52<g1+20)&&(y52==105)&&(g1!=0))||

	((z52<g2+20)&&(y52==180)&&(g2!=0))||((z52<g3+20)&&(y52==255)&&(g3!=0))||((z
52<g4+20)&&(y52==330)&&(g4!=0))||
	((z52<g5+20)&&(y52==405)&&(g5!=0))))
	begin
	ice2 <= 1;
	end
	else                        if(((xrow-z52-10)*(xrow-z52-10)+(yrow-d52-10)*(yrow-d52-
10)<=100)&&(role5>=4'd2)

&&(m52>=11)&&(m52!=0)&&(((g1==0)&&(y52==105))||((g2==0)&&(y52==180))||((g3==0)
&&(y52==255))||((g4==0)&&(y52==330))||
	((g5==0)&&(y52==405))))
	begin
	ice2 <= 1;
	end
	else begin
	ice2 <= 0;
	end

	always@(posedge clk)
	if                          (((xrow-z53-10)*(xrow-z53-10)+(yrow-d53-10)*(yrow-d53-
10)<=100)&&(role5>=4'd3)
	&&(m53>=11)&&(m53!=0)&&(((z53<g1+20)&&(y53==105)&&(g1!=0))||

	((z53<g2+20)&&(y53==180)&&(g2!=0))||((z53<g3+20)&&(y53==255)&&(g3!=0))||((z
53<g4+20)&&(y53==330)&&(g4!=0))||
	((z53<g5+20)&&(y53==405)&&(g5!=0))))
	begin
	ice3 <= 1;
	end
	else                        if(((xrow-z53-10)*(xrow-z53-10)+(yrow-d53-10)*(yrow-d53-
10)<=100)&&(role5>=4'd3)

&&(m53>=11)&&(m53!=0)&&(((g1==0)&&(y53==105))||((g2==0)&&(y53==180))||((g3==0)
&&(y53==255))||((g4==0)&&(y53==330))||
	((g5==0)&&(y53==405))))
	begin
```

```
        ice3 <= 1;
        end
        else begin
        ice3 <= 0;
        end

        always@(posedge clk)
        if                    (((xrow-z54-10)*(xrow-z54-10)+(yrow-d54-10)*(yrow-d54-
10)<=100)&&(role5>=4'd4)
        &&(m54>=11)&&(m54!=0)&&(((z54<g1+20)&&(y54==105)&&(g1!=0))||

        ((z54<g2+20)&&(y54==180)&&(g2!=0))||((z54<g3+20)&&(y54==255)&&(g3!=0))||((z
54<g4+20)&&(y54==330)&&(g4!=0))||
        ((z54<g5+20)&&(y54==405)&&(g5!=0))))
        begin
        ice4 <= 1;
        end
        else                  if(((xrow-z54-10)*(xrow-z54-10)+(yrow-d54-10)*(yrow-d54-
10)<=100)&&(role5>=4'd4)

&&(m54>=11)&&(m54!=0)&&(((g1==0)&&(y54==105))||((g2==0)&&(y54==180))||((g3==0)
&&(y54==255))||((g4==0)&&(y54==330))||
        ((g5==0)&&(y54==405))))
        begin
        ice4 <= 1;
        end
        else begin
        ice4 <= 0;
        end

        always@(posedge clk)
        if                    (((xrow-z55-10)*(xrow-z55-10)+(yrow-d55-10)*(yrow-d55-
10)<=100)&&(role5>=4'd5)
        &&(m55>=11)&&(m55!=0)&&(((z55<g1+20)&&(y55==105)&&(g1!=0))||

        ((z55<g2+20)&&(y55==180)&&(g2!=0))||((z55<g3+20)&&(y55==255)&&(g3!=0))||((z
55<g4+20)&&(y55==330)&&(g4!=0))||
        ((z55<g5+20)&&(y55==405)&&(g5!=0))))
        begin
        ice5 <= 1;
        end
        else                            if(((xrow-z55-10)*(xrow-z55-10)+(yrow-d55-10)*(yrow-d55-
10)<=100)&&(role5>=4'd5)

&&(m55>=11)&&(m55!=0)&&(((g1==0)&&(y55==105))||((g2==0)&&(y55==180))||((g3==0)
&&(y55==255))||((g4==0)&&(y55==330))||
        ((g5==0)&&(y55==405))))
        begin
        ice5 <= 1;
        end
        else begin
        ice5 <= 0;
        end

        always@(posedge clk)                    // screen
        if ((xrow+1>= 0 )&& (xrow<= 639 ) && (yrow>= 0) && (yrow <= 479)&&(over==0))
        begin
        screen <= 1;
        end
        else begin
        screen <= 0;
        end

        always@(posedge clk)                    //gameover
        if ((gameoverx>= 0 )&& (gameoverx<= 399 ) && (gameovery>= 0) && (gameovery <=
49)&&(over==1))
        begin
        gameover <= 1;
        end
        else begin
        gameover <= 0;
        end

  always_comb begin
          {VGA_R, VGA_G, VGA_B} = {8'hff,8'hff,8'hff};
                if(screen==1)
                begin
    {VGA_R, VGA_G, VGA_B} = {8'h00,8'h00,8'h00};
                end
                if((campus==1)&&(over==0))begin
                {VGA_R, VGA_G, VGA_B} = {M_1[23:16], M_1[15:8], M_1[7:0]};
                end
                if((spriteon==1)&&(over==0))begin
                {VGA_R, VGA_G, VGA_B} = {M_bg[23:16], M_bg[15:8], M_bg[7:0]};
                end
                if((cu==1) && (M_cu >= 24'd3)&&(over==0))begin
                {VGA_R, VGA_G, VGA_B} = {M_cu[23:16], M_cu[15:8], M_cu[7:0]};
                end
                if ((flag==1)&&(over==0))  begin
                {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff, 8'hff};
                end
                if ((flag==2)&&(over==0)) begin
                {VGA_R, VGA_G, VGA_B} = {8'h00, 8'hff, 8'hff};
                end
                  if ((eat==1)&&(over==0)) begin
                {VGA_R, VGA_G, VGA_B} = {8'hff, 8'h00, 8'h00};
                end
                if((ch1==1) &&(M_ch1 != 24'd65280)&&(over==0))begin
                {VGA_R, VGA_G, VGA_B} = {M_ch1[23:16], M_ch1[15:8], M_ch1[7:0]};
                end
                if((ch1mov==1) && (M_ch1mov != 24'd65280)&&(over==0))begin
                {VGA_R,    VGA_G,    VGA_B}    =    {M_ch1mov[23:16],    M_ch1mov[15:8],
M_ch1mov[7:0]};
                end
                if((mon1==1) && (M_mon1 != 24'd65280)&&(over==0))begin
                {VGA_R,    VGA_G,    VGA_B}    =    {M_mon1[23:16],    M_mon1[15:8],
M_mon1[7:0]};
                end
                if((mon1mov==1) && (M_mon1mov != 24'd65280)&&(over==0))begin
                {VGA_R,  VGA_G,  VGA_B}  =  {M_mon1mov[23:16],  M_mon1mov[15:8],
M_mon1mov[7:0]};
                end
                if((bg2==1)&&(over==0)) begin
                {VGA_R, VGA_G, VGA_B} = {M_bg2[23:16], M_bg2[15:8], M_bg2[7:0]};
                end
                if((ch3==1) && (M_ch3 != 24'd9605778)&&(over==0))begin
                {VGA_R, VGA_G, VGA_B} = {M_ch3[23:16], M_ch3[15:8], M_ch3[7:0]};
                end
                if((ch3mov==1) && (M_ch3mov != 24'd9605778)&&(over==0))begin
                {VGA_R,    VGA_G,    VGA_B}    =    {M_ch3mov[23:16],    M_ch3mov[15:8],
M_ch3mov[7:0]};
                end
```

```
            if((ch4==1) && ((M_ch4 >= 24'd65536)||(M_ch4 <=
24'd60000))&&(over==0))begin
            {VGA_R, VGA_G, VGA_B} = {M_ch4[23:16], M_ch4[15:8], M_ch4[7:0]};
            end
            if((ch4mov==1) && ((M_ch4mov >= 24'd65536)||(M_ch4mov <=
24'd60000))&&(over==0))begin
            {VGA_R, VGA_G, VGA_B} = {M_ch4mov[23:16], M_ch4mov[15:8],
M_ch4mov[7:0]};
            end
            if((ch5==1) && (M_ch5 != 24'd65280)&&(over==0))begin
            {VGA_R, VGA_G, VGA_B} = {M_ch5[23:16], M_ch5[15:8], M_ch5[7:0]};
            end
            if((ch5mov==1) && (M_ch5mov != 24'd65280)&&(over==0))begin
            {VGA_R, VGA_G, VGA_B} = {M_ch5mov[23:16], M_ch5mov[15:8],
M_ch5mov[7:0]};
            end
            if((fire==1) && (M_fire > 24'd66000)&&(over==0))begin
            {VGA_R, VGA_G, VGA_B} = {M_fire[23:16], M_fire[15:8], M_fire[7:0]};
            end
            if((dead==1) && (M_dead > 24'd65535)&&(over==0))begin
            {VGA_R, VGA_G, VGA_B} = {M_dead[23:16], M_dead[15:8], M_dead[7:0]};
            end
            if((mon4==1) && (M_mon4 != 24'd65535)&&(over==0))begin
            {VGA_R, VGA_G, VGA_B} = {M_mon4[23:16], M_mon4[15:8],
M_mon4[7:0]};
            end
            if((mon4mov==1) && (M_mon4mov != 24'd65535)&&(over==0))begin
            {VGA_R, VGA_G, VGA_B} = {M_mon4mov[23:16], M_mon4mov[15:8],
M_mon4mov[7:0]};
            end
            if((mon5==1) && (M_mon5 != 24'd16711928)&& (M_mon5 !=
24'd16711929)&& (M_mon5 != 24'd16777215)&&(over==0))begin
            {VGA_R, VGA_G, VGA_B} = {M_mon5[23:16], M_mon5[15:8],
M_mon5[7:0]};
            end
            if((mon5mov==1) && (M_mon5mov != 24'd16711920)&&(over==0))begin
            {VGA_R, VGA_G, VGA_B} = {M_mon5mov[23:16], M_mon5mov[15:8],
M_mon5mov[7:0]};
            end
            if((att1==1) && (M_att1 != 24'd16749202)&& (q1!=2)&&(over==0))begin
```

```
                {VGA_R, VGA_G, VGA_B} = {M_att1[23:16], M_att1[15:8], M_att1[7:0]};
                end
                if((att2==1) && (M_att2 != 24'd16749202) && (q2!=2)&&(over==0))begin
                {VGA_R, VGA_G, VGA_B} = {M_att2[23:16], M_att2[15:8], M_att2[7:0]};
                end
                if((att3==1) && (M_att3 != 24'd16749202) && (q3!=2)&&(over==0))begin
                {VGA_R, VGA_G, VGA_B} = {M_att3[23:16], M_att3[15:8], M_att3[7:0]};
                end
                if((att4==1) && (M_att4 != 24'd16749202) && (q4!=2)&&(over==0))begin
                {VGA_R, VGA_G, VGA_B} = {M_att4[23:16], M_att4[15:8], M_att4[7:0]};
                end
                if((att5==1) && (M_att5 != 24'd16749202) && (q5!=2)&&(over==0))begin
                {VGA_R, VGA_G, VGA_B} = {M_att5[23:16], M_att5[15:8], M_att5[7:0]};
                end
                if(((ice1==1)&& (q16!=2)&&(over==0))||((ice2==1)&& (q17!=2)&&(over==0))
                ||((ice3==1)&& (q18!=2)&&(over==0))||((ice4==1)&& (q19!=2)&&(over==0))
                ||((ice5==1)&& (q20!=2)&&(over==0))) begin
                {VGA_R, VGA_G, VGA_B} = {8'h00,8'h64,8'hff};
                end
                if(gameover==1)begin
                {VGA_R,   VGA_G,   VGA_B}   =   {M_gameover[23:16],   M_gameover[15:8],
M_gameover[7:0]};//for every graph, distribute the 24 bit mif data to R,G,B three colors
                end
        end

endmodule
```

Section3 Audio Part (Audio_Top.v, Audio_codec.sv, Audio_effects.v code)

```
//**********************Code of Audio_Top.v****************************//
module audio_top (
   input  OSC_50_B8A,
   inout  AUD_ADCLRCK,
   input  AUD_ADCDAT,
   inout  AUD_DACLRCK,
   output AUD_DACDAT,
   output AUD_XCK,
   inout  AUD_BCLK,
   output AUD_I2C_SCLK,
   inout  AUD_I2C_SDAT,
   output AUD_MUTE,
    input  [3:0] KEY,
    input  [3:0] SW,
    output [3:0] LED,
    input wire[15:0] flag_audio1,
    input wire[15:0] flag_audio2
);

wire reset = !KEY[0];
wire main_clk;
wire audio_clk;
wire [1:0] sample_end;
wire [1:0] sample_req;
wire [15:0] audio_output;
wire [15:0] audio_input;

PLL clock_pll (
    .refclk (OSC_50_B8A),
    .rst (reset),
    .outclk_0 (audio_clk),
    .outclk_1 (main_clk)
);

i2c_av_config av_config (
    .clk (main_clk),
    .reset (reset),
    .i2c_sclk (AUD_I2C_SCLK),
    .i2c_sdat (AUD_I2C_SDAT),
    .status (LED)
);

assign AUD_XCK = audio_clk;
assign AUD_MUTE = (SW != 4'b0);

audio_codec ac (
    .clk (audio_clk),
    .reset (reset),
    .sample_end (sample_end),
    .sample_req (sample_req),
    .audio_output (audio_output),
    .audio_input (audio_input),
    .channel_sel (2'b10),
    .AUD_ADCLRCK (AUD_ADCLRCK),
    .AUD_ADCDAT (AUD_ADCDAT),
    .AUD_DACLRCK (AUD_DACLRCK),
    .AUD_DACDAT (AUD_DACDAT),
    .AUD_BCLK (AUD_BCLK)
);

audio_effects sound (
    .clk (audio_clk),
    .sample_end (sample_end[1]),
    .sample_req (sample_req[1]),
    .audio_output (audio_output),
    .audio_input  (audio_input),
            .flag_audio1(flag_audio1),
            .flag_audio2(flag_audio2),
            .control(SW)

);

endmodule

//**************************Audio_codec.sv******************************//
module audio_codec (
    input  clk,
    input  reset,
    output [1:0]  sample_end,
    output [1:0]  sample_req,
    input  [15:0] audio_output,
    output [15:0] audio_input,
    // 1 - left, 0 - right
    input  [1:0] channel_sel,

    output AUD_ADCLRCK,
    input AUD_ADCDAT,
    output AUD_DACLRCK,
    output AUD_DACDAT,
    output AUD_BCLK
);

reg [7:0] lrck_divider;
reg [1:0] bclk_divider;
reg [15:0] shift_out;
reg [15:0] shift_temp;
reg [15:0] shift_in;

wire lrck = !lrck_divider[7];

assign AUD_ADCLRCK = lrck;
assign AUD_DACLRCK = lrck;
assign AUD_BCLK = bclk_divider[1];
assign AUD_DACDAT = shift_out[15];

always @(posedge clk) begin
      if (reset) begin
    lrck_divider <= 8'hff;
    bclk_divider <= 2'b11;
  end else begin
    lrck_divider <= lrck_divider + 1'b1;
    bclk_divider <= bclk_divider + 1'b1;
  end
end

assign sample_end[1] = (lrck_divider == 8'h40);
assign sample_end[0] = (lrck_divider == 8'hc0);
assign audio_input = shift_in;
assign sample_req[1] = (lrck_divider == 8'hfe);
assign sample_req[0] = (lrck_divider == 8'h7e);

wire clr_lrck = (lrck_divider == 8'h7f);
wire set_lrck = (lrck_divider == 8'hff);
wire set_bclk = (bclk_divider == 2'b10 && !lrck_divider[6]);
wire clr_bclk = (bclk_divider == 2'b11 && !lrck_divider[6]);

always @(posedge clk) begin
  if (reset) begin
    shift_out <= 16'h0;
    shift_in <= 16'h0;
    shift_in <= 16'h0;
  end else if (set_lrck || clr_lrck) begin
    if (channel_sel[set_lrck]) begin
      shift_out <= audio_output;
```

```
            shift_temp <= audio_output;
            shift_in <= 16'h0;
        end else shift_out <= shift_temp;
    end else if (set_bclk == 1) begin
        if (channel_sel[lrck])
            shift_in <= {shift_in[14:0], AUD_ADCDAT};
    end else if (clr_bclk == 1) begin
        shift_out <= {shift_out[14:0], 1'b0};
    end
end

endmodule

//*****************************Audio_effects.v************************//
module audio_effects (
    input  clk,
    input  sample_end,
    input  sample_req,
    output [15:0] audio_output,
    input  [15:0] audio_input,
    input [3:0] control,
   input wire[15:0] flag_audio1, flag_audio2
);

reg [15:0] romdata;
reg [14:0]  index;
reg [15:0] last_sample;
reg [15:0] dat;
wire[15:0] exp;
reg [1:0] over=0;
assign audio_output =dat;
reg [15:0] flag1,flag2;
parameter SINE =0;
parameter FEEDBACK=1;

explosion explosion (
            .address(index),
            .clock(clk),
            .q(exp));
```

```
always @(posedge clk) begin
   flag1<=flag_audio1;
          flag2<=flag_audio2;
   if (sample_end) begin
      last_sample <= audio_input;
   end

   if (sample_req) begin
      if ((control[SINE])&&(flag1==1))
                           begin
         dat <= exp;
         if (index == 15'd 20000)begin
                  index<= 15'd00;
           over<=1;
           end
         else
                  index <= index + 1'b1;
      end else
         dat <= 16'd0;
   end
end

endmodule
```

Code of hello.c in Software

```
//********************************hello.c********************************//

#include <stdlib.h>
#include <stdio.h>
#include "vga_led.h"
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <unistd.h>
#include "usbkeyboard.h"

struct libusb_device_handle *keyboard;
```

```
uint8_t endpoint_address;
int vga_led_fd;

/* Read and print the segment values */
void print_xyf_info() {
  vga_axis_arg_t vla;
  int i;

  for (i = 0 ; i < 201 ; i++) {
    vla.axis = i;
    if (ioctl(vga_led_fd, VGA_LED_READ_xy, &vla)) {
      perror("ioctl(VGA_LED_READ_xy) failed");
      return;
    }
    printf("%02x ", vla.xyf);
  }
  printf("\n");
}


/* Write the contents of the array to the display */
void write_xyf(const int xyf_[201])
{
  vga_axis_arg_t vla;
  int i;
  for (i = 0 ; i < 201 ; i++) {
    vla.axis = i;
    vla.xyf = xyf_[i];
    if (ioctl(vga_led_fd, VGA_LED_WRITE_xy, &vla)) {
      perror("ioctl(VGA_LED_WRITE_xy) failed");
      return;
    }
  }
}

int main()
{
 struct usb_keyboard_packet packet;
            int transferred;
char keystate[12];
if ( (keyboard = openkeyboard(&endpoint_address)) == NULL ) {
```

```
                fprintf(stderr, "Did not find a keyboard\n");
                exit(1);
        }
vga_axis_arg_t vla;
printf("01\n");
static const char filename[] = "/dev/vga_led";
 printf("02\n");

static int message[201] = { 35, 105,584,105,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                    0,0,0,0,0,0,0,0,0,0,          //-------------mess29
                    0,0,0,0,0,0,0,0,0,0,
                    0,0,0,0,0,0,0,0,0,0,
                    0,0,0,0,0,0,0,0,0,0,
                    0,0,0,0,0,0,0,0,0,0,
                    0,0,0,0,0,0,0,0,0,0,
                    0,0,0,0,0,0,0,0,0,0,
                    0,0,0,0,0,0,0,0,0,0,            //--------------99
                    0,0,0,0,0,0,0,0,0,0,
                    0,0,0,0,0,0,0,0,0,0,
                    0,0,0,0,0,0,0,0,0,0,            //---------129
                    0,0,642,642,642,642,0,642,642,642,642,0,
                    642,642,642,642,0,642,642,642,642,0,
                    642,642,642,642,0,0,0,0,0,0,0,0,0,0,0,
                    0,0,0,0,0,0,0,0,0,0,0,0,
                    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};    // every message need to transmit



printf("VGA axis Userspace program started\n");

if ( (vga_led_fd = open(filename, O_RDWR)) == -1) {
  fprintf(stderr, "could not open %s\n", filename);
  return -1;
}

printf("initial state: ");
print_xyf_info();

write_xyf(message);                         //write the initial value to the register

printf("current state: ");
```

```
  print_xyf_info();

  int i,j;
  int movestep = 5;
  int deadstop=0;                          // the value for counting when monster dead
  static int role[6]={0,0,0,0,0,0};
  int k,t,u;
  static int flag1[5]={0,0,0,0,0,0};
  int flag2=0;
  int flag3=0;
  int flag4=0;
  int flag5=0;
  static int flag6[5]={0,0,0,0,0};
  static int monstepmax[20]={10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10};  //
initial move speed of monster
  static int stopsign[20]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
  static int monstep[20]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};          // two vectors to record
monster move and stop
  int roundtime=0;
  int static stopchara[5]={105,180,255,330,405};
  static int blood[20]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};         // the blood of monsters
  static int number[5]={0,0,0,0,0};
  static int ahead[5]={0,0,0,0,0};                        // two vector to help record the first monster
of each line

  static int bite[20]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};              // to record the life of
defender


//---------------part1:control the movement of choosing square---------------------------------//


if(message[200]==0)
{

while(1) {
    libusb_interrupt_transfer(keyboard, endpoint_address,(unsigned char *) &packet,
                              sizeof(packet),&transferred, 10000);
                        if (transferred == sizeof(packet)) {
      sprintf(keystate, "%02x %02x %02x %02x\n", packet.modifiers, packet.keycode[0],
              packet.keycode[1],packet.keycode[2],packet.keycode[3]);
    if (packet.keycode[3]==0x00 && message[1] != 105){  // move up
    message[1]=message[1]-75;
    }
  else if (packet.keycode[3]==0x04 && message[1] != 405){ //move down
    message[1]=message[1]+75;
    }
  else if (packet.keycode[3]==0x06 && message[0] != 35){  // move left
    message[0]=message[0]-55;
    }
  else if (packet.keycode[3]==0x02 && message[0] != 585){// move right
    message[0]=message[0]+55;
    }

  write_xyf(message);

//-----------------------------------part2:choose defender---------------------------- --------------------//


    if  ((packet.keycode[3]==0x2F)  &&  (message[1]==105)  &&  (message[4]  !=  1)
&&(message[0]=35)){  // select
    message[4]=1;
    message[5]=1;
  if(role[1]==5)
   {role[1]=0;}
    role[1]=role[1]+1;
    }
 else  if  ((packet.keycode[3]==0x2F)  &&  (message[1]==180)  &&  (message[4]  !=
1)&&(message[0]=35)){  // select
    message[4]=1;
    message[5]=3;
  if(role[3]==5)
   {role[3]=0;}
    role[3]=role[3]+1;
    }

 else  if  ((packet.keycode[3]==0x2F)  &&  (message[1]==330)  &&  (message[4]  !=
1)&&(message[0]=35)){  // select
    message[4]=1;
    message[5]=4;
  if(role[4]==5)
   {role[4]=0;}
```

```
    role[4]=role[4]+1;
    }
  else   if   ((packet.keycode[3]==0x2F)   &&   (message[1]==405)   &&   (message[4]   !=
1)&&(message[0]=35)){  // select
    message[4]=1;
    message[5]=5;
  if(role[5]==5)
   {role[5]=0;}
    role[5]=role[5]+1;
    }

//-------------------------------part3: place role-------------------------------------------------------------
----//

if (packet.keycode[3]==0x2F && message[4] != 0 && message[0] != 35 && role[1]==1 &&
message[5]==1 )
  {
    message[4]=0;
    message[6]=message[0];
    message[7]=message[1];
    message[10]=message[0]+55;
    message[11]=message[1]+25;
    message[0]=35;
    message[1]=105;
    message[8]=7;}
   else if (packet.keycode[3]==0x2F && message[4] != 0 && message[0] != 35 && role[1]==2
&& message[5]==1)
   {message[4]=0;
    message[12]=message[0];
    message[13]=message[1];
   message[20]=message[0]+55;
    message[21]=message[1]+25;
   message[0]=35;
  message[1]=105;
    message[28]=7;
    }
   else if (packet.keycode[3]==0x2F && message[4] != 0 && message[0] != 35 && role[1]==3
&& message[5]==1)
   {message[4]=0;
    message[14]=message[0];
    message[15]=message[1];
   message[22]=message[0]+55;
    message[23]=message[1]+25;
   message[0]=35;
   message[1]=105;
    message[29]=7;
    }
   else if (packet.keycode[3]==0x2F && message[4] != 0 && message[0] != 35 && role[1]==4
&& message[5]==1)
   {message[4]=0;
    message[16]=message[0];
    message[17]=message[1];
   message[24]=message[0]+55;
    message[25]=message[1]+25;
    message[0]=35;
    message[1]=105;
    message[30]=7;

    }
   else if (packet.keycode[3]==0x2F && message[4] != 0 && message[0] != 35 && role[1]==5
&& message[5]==1)
   {message[4]=0;
    message[18]=message[0];
    message[19]=message[1];
   message[26]=message[0]+55;
    message[27]=message[1]+25;
   message[0]=35;
    message[1]=105;
    message[31]=7;
    }                          //-----------------place defender type 1


for(k=3;k<=5;k++)
{
if (packet.keycode[3]==0x2F && message[4] != 0 && message[0] != 35 && role[k]==1 &&
message[5]==k)  // select
  {
    message[4]=0;
    message[32+25*(k-2)]=message[0];
    message[33+25*(k-2)]=message[1];
    if(k==3)
     {message[68]=message[1]+55;}
    else if(k==5)
    {message[117]=message[0]+55;
     message[118]=message[1]+20;}
     message[0]=35;
    message[1]=105;
    message[52+25*(k-2)]=7;
    }
   else if (packet.keycode[3]==0x2F && message[4] != 0 && message[0] != 35 && role[k]==2
&& message[5]==k)
   {message[4]=0;
    message[34+25*(k-2)]=message[0];
    message[35+25*(k-2)]=message[1];
    if(k==3)
    {message[70]=message[1]+55;}
    else if(k==5)
    {message[119]=message[0]+55;
     message[120]=message[1]+20;}
    message[0]=35;
    message[1]=105;
    message[53+25*(k-2)]=7;
    }
   else if (packet.keycode[3]==0x2F && message[4] != 0 && message[0] != 35 && role[k]==3
&& message[5]==k)
   {message[4]=0;
    message[36+25*(k-2)]=message[0];
    message[37+25*(k-2)]=message[1];
    if(k==3)
    {message[72]=message[1]+55;}
    else if(k==5)
    {message[121]=message[0]+55;
     message[122]=message[1]+20;}
    message[0]=35;
   message[1]=105;
    message[54+25*(k-2)]=7;
    }
   else if (packet.keycode[3]==0x2F && message[4] != 0 && message[0] != 35 && role[k]==4
&& message[5]==k)
   {message[4]=0;
    message[38+25*(k-2)]=message[0];
    message[39+25*(k-2)]=message[1];
    if(k==3)
     {message[74]=message[1]+55;}
    else if(k==5)
     {message[123]=message[0]+55;
      message[124]=message[1]+20;}
   message[0]=35;
   message[1]=105;
    message[55+25*(k-2)]=7;
     }
   else if (packet.keycode[3]==0x2F && message[4] != 0 && message[0] != 35 && role[k]==5
&& message[5]==k)
   {message[4]=0;
    message[40+25*(k-2)]=message[0];
    message[41+25*(k-2)]=message[1];
    if(k==3)
     {message[76]=message[1]+55;}
    if(k==5)
     {message[125]=message[0]+55;
      message[126]=message[1]+20;}
    message[0]=35;
    message[1]=105;
    message[56+25*(k-2)]=7;
     }
}                              //---------------------place defender type 2 to 4


//------------------------------------------part4:move and stop of monster----------------------------------
--------//
   if(roundtime==0)
   {message[132] = 640;}
   if(roundtime==300)
   {message[142] = 640;}
   if(roundtime==600)
   {message[137] = 640;}
   if(roundtime==900)
   {message[147] = 640;}
   if(roundtime==1200)
   {message[152] = 640;}

   if(roundtime==1500)
   {message[133] = 640;message[138] = 640;}
   if(roundtime==1700)
   {message[143] = 640;message[148] = 640;}
   if(roundtime==2100)
   {message[153] = 640;}
   if(roundtime==2180)
   {message[154] = 640;}

   if(roundtime==2500)
   {message[134] = 640;message[139] = 640;message[144] = 640;}

   if(roundtime==3000)
   {message[135] = 640;message[140] = 640;message[145] = 640;message[149] = 640;}


for(j=0;j<=4;j++)
{
for(t=0;t<=3;t++)
{

if((message[132+5*j+t]-55<=message[6])                &&                (message[7]==stopchara[j])
&&(message[132+5*j+t]!=642)&&(stopsign[4*j+t]!=1)&&(message[179]!=2))
{stopsign[4*j+t]=1;
 message[179]=1;}
if(message[179]==2)
{stopsign[4*j+t]=0;}

if((message[132+5*j+t]-55<=message[12])                                                      &&
(message[13]==stopchara[j])&&(message[132+5*j+t]!=642)&&(stopsign[4*j+t]!=1)&&(messa
ge[180]!=2))
{stopsign[4*j+t]=1;
 message[180]=1;}
if(message[180]==2)
{stopsign[4*j+t]=0;}

if((message[132+5*j+t]-55<=message[14])                                                      &&
(message[15]==stopchara[j])&&(message[132+5*j+t]!=642)&&(stopsign[4*j+t]!=1)&&(messa
ge[181]!=2))
{stopsign[4*j+t]=1;
 message[181]=1;}
if(message[181]==2)
{stopsign[4*j+t]=0;}
```

```
 if((message[132+5*j+t]-55<=message[16])                                              &&
(message[17]==stopchara[j])&&(message[132+5*j+t]!=642)&&(stopsign[4*j+t]!=1)&&(messa
ge[182]!=2))
{stopsign[4*j+t]=1;
 message[182]=1;}
if(message[182]==2)
{stopsign[4*j+t]=0;}

 if((message[132+5*j+t]-55<=message[18])                                              &&
(message[19]==stopchara[j])&&(message[132+5*j+t]!=642)&&(stopsign[4*j+t]!=1)&&(messa
ge[183]!=2))
{stopsign[4*j+t]=1;
 message[183]=1;}
if(message[183]==2)
{stopsign[4*j+t]=0;}

if(message[132+5*j+t]<=90)
{message[200]=1;
 message[177]=1;
  }


for(k=3;k<=5;k++)
   {
    for(u=0;u<=4;u++)
    {
    if(message[132+5*j+t]-55<=message[32+25*(k-2)+2*u]         &&         (message[32+25*(k-
2)+1+2*u]==stopchara[j])&&(message[132+5*j+t]!=642)&&(stopsign[4*j+t]!=1)&&(message[
184+5*(k-3)+u]!=2))
       {
         stopsign[4*j+t]=1;
         message[184+5*(k-3)+u]=1;
       }
     if(message[184+5*(k-3)+u]==2)
      {stopsign[4*j+t]=0;}
     }
   }
```

```
if((message[132+5*j+t]!=642)&&(stopsign[4*j+t]==0)&&(monstep[4*j+t]==monstepmax[4*j+t
])&&(message[157+t+4*j]==0))
          {
      message[132+5*j+t] = message[132+5*j+t]-movestep;
      monstep[4*j+t]=0;
      }
   else if(monstep[4*j+t]!=monstepmax[4*j+t])
    {message[132+5*j+t] = message[132+5*j+t];
     monstep[4*j+t]=monstep[4*j+t]+1;}
}
}
   roundtime++;

//-----------------------------------------part5:attack for different defender---------------------------------
-----//

for(t=0;t<=4;t++)
{
for(k=0;k<=3;k++)
{
  if((message[132+k+5*t]!=0)&&(message[132+k+5*t]<=640))
    {message[136+5*t]=message[132+5*t]; }
}

for(k=0;k<=3;k++)
{  if(message[136+5*t]>=message[132+k+5*t])
   {message[136+5*t]=message[132+k+5*t];
   number[t]=k+157+4*t;
   ahead[t]=132+k+5*t;}
}

}

if(message[179]!=2)
{
for (k=0;k<=4;k++)
{

if((message[7]==stopchara[k]) && (message[136+5*k]!=0))
{
  if ( ( (message[6]!=0) && (message[8]%40!=10) &&(message[8]>10) &&
(message[10]<message[136+5*k]+20)) ||
((message[6]!=0)  && (message[8]<=10) && (message[10]<message[136+5*k]+20)))
    {
    message[8]=message[8]+1;
      }
    else if( (message[6]!=0) && (message[8]%40==10) &&
(message[10]<message[136+5*k]+20))
     {
    message[8]=message[8];
      }
    else if((message[6]!=0) && (message[8]%40==10) &&
(message[10]>=message[136+5*k]+20) &&(message[number[k]]==0)&& (blood[number[k]-
157]<=200))
     {
    message[8]=message[8]+1;
    message[10]=message[6]+55;
    if(flag1[k]!=1)
    {blood[number[k]-157] = blood[number[k]-157] +30;}
    flag1[k]=0;
      }
    else if((message[6]!=0) && (message[8]%40!=10) &&
(message[10]>=message[136+5*k]+20)&& (message[number[k]]==0)&&(blood[number[k]-
157]<=200))
     {
    message[8]=message[8]+1;
    if(flag1[k]!=1)
    {blood[number[k]-157]=blood[number[k]-157]+30;}
    flag1[k]=1;
      }
}
}

if (((message[7]==105) && (message[136]==0)) || ((message[7]==180) && (message[141]==0))
|| ((message[7]==255) && (message[146]==0)) ||
((message[7]==330) && (message[151]==0)) || ((message[7]==405) && (message[156]==0)))

{  if ( ( (message[6]!=0) && (message[8]%40!=10) &&(message[8]>10) &&
(message[10]<639)) ||
((message[6]!=0)  && (message[8]<=10) && (message[10]<639)))
```

```
    {
    message[8]=message[8]+1;
      }
    else if( (message[6]!=0)  && (message[8]%40==10) && (message[10]<639))
     {
     message[8]=message[8];
      }
    else if((message[6]!=0) && (message[8]%40==10) && (message[10]>=639))
     {
    message[8]=message[8]+1;
    message[10]=message[6]+55;
      }
    else if((message[6]!=0)  && (message[8]%40!=10) && (message[10]>=639))
     {
    message[8]=message[8]+1;
      }
}
}


if(message[180]!=2)
{
for (k=0;k<=4;k++)
{
if((message[13]==stopchara[k]) && (message[136+5*k]!=0))
{
   if ( ( (message[12]!=0) && (message[28]%40!=10) &&(message[28]>10) &&
(message[20]<message[136+5*k]+20)) ||
((message[12]!=0)  && (message[28]<=10) && (message[20]<message[136+5*k]+20)))
    {
    message[28]=message[28]+1;
      }
    else if( (message[12]!=0) && (message[28]%40==10) &&
(message[20]<message[136+5*k]+20))
     {
    message[28]=message[28];
      }
    else if((message[12]!=0) && (message[28]%40==10) &&
(message[20]>=message[136+5*k]+20)&&(message[number[k]]==0)&&(blood[number[k]-
157]<=200))
     {
```

```
    message[28]=message[28]+1;
    message[20]=message[12]+55;
      if(flag2!=1)
    {blood[number[k]-157]=blood[number[k]-157]+30;}
    flag2=0;
       }
    else if((message[12]!=0) && (message[28]%40!=10) &&
(message[20]>=message[136+5*k]+20)&&
(message[number[k]]==0)&&(blood[number[k]-157]<=200))
     {
    message[28]=message[28]+1;
      if(flag2!=1)
    {blood[number[k]-157]=blood[number[k]-157]+30;}
    flag2=1;
       }
}
}

if (((message[13]==105) && (message[136]==0)) || ((message[13]==180) &&
(message[141]==0)) || ((message[13]==255) && (message[146]==0)) ||((message[13]==330)
&& (message[151]==0)) || ((message[13]==405) && (message[156]==0)))
{ if ( ( (message[12]!=0) && (message[28]%40!=10) &&(message[28]>10) &&
(message[20]<639)) ||
((message[12]!=0) && (message[28]<=10) && (message[20]<639)))
    {
    message[28]=message[28]+1;
       }
    else if( (message[12]!=0) && (message[28]%40==10) && (message[20]<639))
     {
    message[28]=message[28];
       }
    else if((message[12]!=0) && (message[28]%40==10) && (message[20]>=639))
     {
    message[28]=message[28]+1;
    message[20]=message[12]+55;
       }
    else if((message[12]!=0) && (message[28]%40!=10) && (message[20]>=639))
     {
    message[28]=message[28]+1;
       }
}
```

```
}


if(message[181]!=2)
{
for (k=0;k<=4;k++)
{
if ((message[15]==stopchara[k])&& (message[136+5*k]!=0))
{
   if ( ( (message[14]!=0) && (message[29]%40!=10) &&(message[29]>10) &&
(message[22]<message[136+5*k]+20)) ||
((message[14]!=0) && (message[29]<=10) && (message[22]<message[136+5*k]+20)))
    {
    message[29]=message[29]+1;
      }
    else if( (message[14]!=0) && (message[29]%40==10) &&
(message[22]<message[136+5*k]+20))
     {
    message[29]=message[29];
      }
    else if((message[14]!=0) && (message[29]%40==10) &&
(message[22]>=message[136+5*k]+20) &&(message[number[k]]==0)&&(blood[number[k]-
157]<=200))
     {
    message[29]=message[29]+1;
    message[22]=message[14]+55;
     if(flag3!=1)
    {blood[number[k]-157] = blood[number[k]-157] +30;}
    flag3=0;
      }
    else if((message[14]!=0) && (message[29]%40!=10) &&
(message[22]>=message[136+5*k]+20)&&(message[number[k]]==0)&&(blood[number[k]-
157]<=200))
     {
    message[29]=message[29]+1;
     if(flag3!=1)
    {blood[number[k]-157]=blood[number[k]-157]+30;}
    flag3=1;
      }
}
}
```

```
if (((message[15]==105) && (message[136]==0)) || ((message[15]==180) &&
(message[141]==0)) || ((message[15]==255) && (message[146]==0)) ||((message[15]==330)
&& (message[151]==0)) || ((message[15]==405) && (message[156]==0)))

{ if ( ( (message[14]!=0) && (message[29]%40!=10) &&(message[29]>10) &&
(message[22]<639)) ||
((message[14]!=0) && (message[29]<=10) && (message[22]<639)))
    {
    message[29]=message[29]+1;
      }
    else if( (message[14]!=0) && (message[29]%40==10) && (message[22]<639))
     {
    message[29]=message[29];
      }
    else if((message[14]!=0) && (message[29]%40==10) && (message[22]>=639))
     {
    message[29]=message[29]+1;
    message[22]=message[14]+55;
      }
    else if((message[14]!=0) && (message[29]%40!=10) && (message[22]>=639))
     {
    message[29]=message[29]+1;
      }
}
}


if(message[182]!=2)
{
for (k=0;k<=4;k++)
{
if((message[17]==stopchara[k]) && (message[136+5*k]!=0))
{
   if ( ( (message[16]!=0)    && (message[30]%40!=10) &&(message[30]>10) &&
(message[24]<message[136+5*k]+20)) ||
((message[16]!=0)  && (message[30]<=10) && (message[24]<message[136+5*k]+20)))
    {
    message[30]=message[30]+1;
      }
    else if( (message[16]!=0) && (message[30]%40==10) &&
(message[24]<message[136+5*k]+20))
     {
    message[30]=message[30];
      }
    else if((message[16]!=0) && (message[30]%40==10) &&
(message[24]>=message[136+5*k]+20) &&(message[number[k]]==0)&&(blood[number[k]-
157]<=200))
     {
    message[30]=message[30]+1;
    message[24]=message[16]+55;
     if(flag4!=1)
    {blood[number[k]-157]=blood[number[k]-157]+30;}
    flag4=0;
      }
    else if((message[16]!=0) && (message[30]%40!=10) &&
(message[24]>=message[136+5*k]+20)&&
(message[number[k]]==0)&&(blood[number[k]-157]<=200))
     {
    message[30]=message[30]+1;
     if(flag4!=1)
    {blood[number[k]-157]=blood[number[k]-157]+30;}
    flag4=1;
      }
}
}

if (((message[17]==105) && (message[136]==0)) || ((message[17]==180) &&
(message[141]==0)) || ((message[17]==255) && (message[146]==0)) ||((message[17]==330)
&& (message[151]==0)) || ((message[17]==405) && (message[156]==0)))

{ if ( ( (message[16]!=0) && (message[30]%40!=10) &&(message[30]>10) &&
(message[24]<639)) ||
((message[16]!=0) && (message[30]<=10) && (message[24]<639)))
    {
    message[30]=message[30]+1;
      }
    else if( (message[16]!=0) && (message[30]%40==10) && (message[24]<639))
     {
    message[30]=message[30];
      }
    else if((message[16]!=0)  && (message[30]%40==10) && (message[24]>=639))
     {
    message[30]=message[30]+1;
    message[24]=message[16]+55;
      }
    else if((message[16]!=0) && (message[30]%40!=10) && (message[24]>=639))
     {
    message[30]=message[30]+1;
      }
}
}


if(message[183]!=2)
{
for (k=0;k<=4;k++)
{
if((message[19]==stopchara[k]) && (message[136+5*k]!=0))
{
   if  (  ((message[18]!=0)  &&  (message[31]%40!=10)  &&(message[31]>10)  &&
(message[26]<message[136+5*k]+20)) ||
((message[18]!=0) && (message[31]<=10) && (message[26]<message[136+5*k]+20)))
    {
    message[31]=message[31]+1;
      }
    else     if(     (message[18]!=0)&&     (message[31]%40==10)     &&
(message[26]<message[136+5*k]+20))
     {
    message[31]=message[31];
      }
    else     if((message[18]!=0)     &&     (message[31]%40==10)     &&
(message[26]>=message[136+5*k]+20)   &&(message[number[k]]==0)&&(blood[number[k]-
157]<=200))
     {
    message[31]=message[31]+1;
    message[26]=message[18]+55;
     if(flag5!=1)
    {blood[number[k]-157]=blood[number[k]-157]+30;}
    flag5=0;
      }
    else if((message[18]!=0) && (message[31]%40!=10) &&
(message[26]>=message[136+5*k]+20)&&
(message[number[k]]==0)&&(blood[number[k]-157]<=200))
     {
    message[31]=message[31]+1;
     if(flag5!=1)
    {blood[number[k]-157]=blood[number[k]-157]+30;}
    flag5=1;
      }
}
}

if (((message[19]==105) && (message[136]==0)) || ((message[19]==180) &&
(message[141]==0)) || ((message[19]==255) && (message[146]==0)) ||((message[19]==330)
&& (message[151]==0)) || ((message[19]==405) && (message[156]==0)))

{
if (((message[18]!=0) && (message[31]%40!=10) &&(message[31]>10) &&
(message[26]<639)) ||((message[18]!=0) && (message[31]<=10) && (message[26]<639)))
    {
    message[31]=message[31]+1;
      }
    else if( (message[18]!=0) && (message[31]%40==10) && (message[26]<639))
     {
    message[31]=message[31];
      }
    else if((message[18]!=0) && (message[31]%40==10) && (message[26]>=639))
     {
    message[31]=message[31]+1;
    message[26]=message[18]+55;
      }
    else if((message[18]!=0) && (message[31]%40!=10) && (message[26]>=639))
     {
    message[31]=message[31]+1;
      }
}
}                                   //--------------attack1

for(k=0;k<=4;k++)
{
if(message[184+k]!=2)
```

```
{
for (j=0;j<=3;j++)
{
if((message[58+2*k]==stopchara[k]) && (message[132+5*k+j]<=640))
{
   if      (      (message[77+k]%100==20)       &&       (blood[number[k]-157]<=200)       &&
(message[number[k]]==0))
     {
     blood[number[k]-157]=blood[number[k]-157]+50;
      }
}
}
}
}

for(k=1;k<=5;k++)
{message[77+k-1]=message[77+k-1]+1;
}                                        //--------------attack2



for(k=0;k<=4;k++)
{
if(message[194+k]!=2)
{
for(t=0;t<=4;t++)
{
if((message[108+2*k]==stopchara[t]) && (message[136+5*t]!=0))
{
   if ( ( (message[107+2*k]!=0) && (message[127+k]%30!=10) &&(message[127+k]>10) &&
(message[117+2*k]<message[136+5*t]+20)) ||
((message[107+2*k]!=0)                   &&             (message[127+k]<=10)             &&
(message[117+2*k]<message[136+5*t]+20)))
     {
     message[127+k]=message[127+k]+1;
       }
     else      if(      (message[107+2*k]!=0)            &&       (message[127+k]%30==10)       &&
(message[117+2*k]<message[136+5*t]+20))
      {
     message[127+k]=message[127+k];
       }
```

```
    else if((message[107+2*k]!=0) && (message[127+k]%30==10) &&
(message[117+2*k]>=message[136+5*t]+20)&&(message[number[t]]==0)&&(blood[number[t]
-157]<=200))
     {
    message[127+k]=message[127+k]+1;
    message[117+2*k]=message[107+2*k]+55;
    monstepmax[number[t]-157]=30;
     if(flag6[k]!=1)
    {blood[number[t]-157] = blood[number[t]-157] +40;}
    flag6[k]=0;
      }
    else if((message[107+2*k]!=0) && (message[127+k]%30!=10) &&
(message[117+2*k]>=message[136+5*t]+20)&&(message[number[t]]==0)&&(blood[number[t]
-157]<=200))
     {
    message[127+k]=message[127+k]+1;
    monstepmax[number[t]-157]=30;
     if(flag6[k]!=1)
    {blood[number[t]-157] = blood[number[t]-157] +40;}
    flag6[k]=1;
      }
}
}
}


if (((message[108+2*k]==105) && (message[136]==0)) || ((message[108+2*k]==180) &&
(message[141]==0)) || ((message[108+2*k]==255) && (message[146]==0))
||((message[108+2*k]==330) && (message[151]==0)) || ((message[108+2*k]==405) &&
(message[156]==0)))
{ if ( ( (message[107+2*k]!=0) && (message[127+k]%30!=10) &&(message[127+k]>10) &&
(message[117+2*k]<639)) ||
((message[107+2*k]!=0) && (message[127+k]<=10) && (message[117+2*k]<639)))
    {
    message[127+k]=message[127+k]+1;
      }
    else if( (message[107+2*k]!=0) && (message[127+k]%30==10) &&
(message[117+2*k]<639))
     {
    message[127+k]=message[127+k];
      }
    else        if((message[107+2*k]!=0)        &&        (message[127+k]%30==10)        &&
(message[117+2*k]>=639))
     {
    message[127+k]=message[127+k]+1;
    message[117+2*k]=message[107+2*k]+55;
      }
    else        if((message[107+2*k]!=0)        &&        (message[127+k]%30!=10)        &&
(message[117+2*k]>=639))
     {
    message[127+k]=message[127+k]+1;
      }
}
}                                   //--------------attack3


for(k=1;k<=5;k++)
{
if(message[193+k]!=2)
{
for(t=0;t<=4;t++)
{
if((message[83+2*k-2]==stopchara[t]) && (message[136+5*t]!=0))
{
   if   ((message[82+2*k-2]!=0)   &&   (message[82+2*k-2]>=message[136+5*t]-55)   &&
(blood[number[t]-157]<=200) && (message[102+k-1]%30==10))

    {
    message[102+k-1]=message[102+k-1]+1;
    blood[number[t]-157]=blood[number[t]-157]+10;
      }
   else  if  ((message[82+2*k-2]!=0)  &&  (message[82+2*k-2]>=message[136+5*t]-55)  &&
(blood[number[t]-157]<=200) && (message[102+k-1]%30!=10))
   {
    message[102+k-1]=message[102+k-1]+1;
      }
    else  if  ((message[82+2*k-2]!=0)  &&  (message[82+2*k-2]<message[136+5*t]-55)  &&
(message[102+k-1]%30!=10))
    {
    message[102+k-1]=message[102+k-1]+1;
      }
     else    if  ((message[82+2*k-2]!=0)  &&  (message[82+2*k-2]<message[136+5*t]-55)  &&
(message[102+k-1]%30==10))
     {
     message[102+k-1]=message[102+k-1];
       }
    else if(blood[number[t]-157]>200)
    {message[102+k-1]=10;}
}
}
}
}

if  (((message[83+2*k-2]==105)  &&  (message[136]==0))  ||  ((message[83+2*k-2]==180)  &&
(message[141]==0))        ||        ((message[83+2*k-2]==255)        &&        (message[146]==0))
||((message[83+2*k-2]==330)  &&  (message[151]==0))  ||  ((message[83+2*k-2]==405)  &&
(message[156]==0)))
{
   message[102+k-1]=message[102+k-1];
}                                          //------------attack4


//-----------------part6:calculate the blood of monster and defender-----------------------------------//

for(k=0;k<=4;k++)
{
if (blood[number[k]-157]>200)
{
blood[number[k]-157]=0;
message[number[k]]=1;
 }              // attack

   if((deadstop !=25)&&(message[number[k]]==1))
       {message[number[k]] =1;
        deadstop=deadstop+1;
        }
    else if((deadstop ==25)&&(message[number[k]]==1))
    { message[number[k]] =2;
     deadstop=0;
    }

 if((message[number[k]]==2))
```

```
{message[ahead[k]]=642;
 message[number[k]]=2;
message[136+5*k]=0;
stopsign[number[k]-157]=1;
}
}

for(k=179;k<=198;k++)
 {
     if((message[k]==1) &&(bite[k-179]==100))
         { message[k]=2;
           bite[k-179]=0;
          }
     if((message[k]==1) &&(bite[k-179]<100))
       {
        bite[k-179]++;  }
}


//-----------------------------------------------------part7:bullet move---------------------------------------
-----//
 if(message[8]>=11)
{
     message[10]=message[10]+7;
}

 if(message[28]>=11)
{
     message[20]=message[20]+7;
}

 if(message[29]>=11)
{
     message[22]=message[22]+7;
}

 if(message[30]>=11)
{
     message[24]=message[24]+7;
}
```

```
 if(message[31]>=11)
{
    message[26]=message[26]+7;
}

  if(message[127]>=11)
{
    message[117]=message[117]+7;
}
  if(message[128]>=11)
{
    message[119]=message[119]+7;
}
  if(message[129]>=11)
{
    message[121]=message[121]+7;
}
  if(message[130]>=11)
{
    message[123]=message[123]+7;
}
  if(message[131]>=11)
{
    message[125]=message[125]+7;
}
        write_xyf(message);
    if (packet.keycode[1] == 0x00)/* ESC pressed? */
        break;
    }
   usleep(70000);
  }
}
  printf("VGA LED Userspace program terminating\n");
  return 0;
}
```