MiniProject Phase 2

Sampreety Pillai, M Shanmukha Priya

27 April 2023

Contents

1 Project introduction

Given below is a snake game created on the De1-Soc using NIOS II processors (VGA is used). The code is written in C. The rectangles represent the snake in the game. The bouncing ball represents the snake's meal/ball. The aim of the game is for the snake to eat(clash) with the ball, which increases the score.

1.1 Rules:

- Game ends when the snake clashes with the wall.
- User moves the snake around using push buttons. (Key[0] = move down, Key[1] = move up, Key[2] = move left, Key[3] = move right)
- Score increases when the ball hits the walls of the 'snake'.

2 Team Members

2.1 Sampreety Pillai

- Rollno: 2101CS71
- Webmail id: sampreety_2101cs71@iitp.ac.in
- Contribution: Responsible for creating the snake, the ball and its movements.

2.2 M Shanmukha Priya

- Rollno: 2101CS40
- Webmail id: priya_2101cs40@iitp.ac.in
- Contribution: Responsible for the shapes, score/game over logic, and colouring.

3 Code

```
   #define  MPCORE_PRIV_TIMER 0xff202000   // PERIPH_BASE + 0x0600
#define   swbutton         0xff200050


    void write_pixel(int x, int y, short colour) {
      volatile short *vga_addr=(volatile short*)(0x08000000 + (y<<10) + (x<<1));
      *vga_addr=colour;
    }

void write_char(int x, int y, char c) {
  // VGA character buffer
  volatile char * character_buffer = (char *) (0x09000000 + (y<<7) + x);
  *character_buffer = c;
}

void clear_char(){
     for(int x = 0; x<80;x++){
     for(int y = 0;y<60;y++){
        write_char(x, y, ' ');

     }
     }
```

```
   }

void drawCircle(int xc, int yc, int x, int y, int col)
{
    write_pixel(xc + x, yc + y, col);
    write_pixel(xc - x, yc + y, col);
    write_pixel(xc + x, yc - y, col);
    write_pixel(xc - x, yc - y, col);
    write_pixel(xc + y, yc + x, col);
    write_pixel(xc - y, yc + x, col);
    write_pixel(xc + y, yc - x, col);
    write_pixel(xc - y, yc - x, col);


}

void circleBres(int xc, int yc, int r, int col)
{
    int x = 0, y = r;
    int d = 3 - 2 * r;

    drawCircle(xc, yc, x, y, col);
    while (y >= x)
    {

        x++;

        if (d > 0)
        {
            y--;
            d = d + 4 * (x - y) + 10;
        }
        else
            d = d + 4 * x + 6;

        drawCircle(xc, yc, x, y, col);
    }
}


void create_square(int x, int y, short colour,int size){
   for(int l = 0; l<size;l++){
   for(int b = 0;b<size;b++){
   write_pixel(x+l, y+b, colour);
   }
   }
}
 void snake(int x, int y, short colour, int size){

   for(int l =0;l<size;l++){
      write_pixel(x+l, y, colour);
      write_pixel(x+l, y+size, colour);
   }
    for(int l =0;l<size;l++){
      write_pixel(x, y+l, colour);
      write_pixel(x+size, y+l, colour);
   }
 }
    /* use write_pixel to set entire screen to black (does not clear the character buffer) */
```

```
    void clear_screen() {
      int x, y;
      for (x = 0; x < 320; x++) {
        for (y = 0; y < 240; y++) {
        write_pixel(x,y,0);
      }
      }
    }
int game_over()
{

    clear_char();
    char* hw1 = "Game over!! ";
    int x = 30;
    while (*hw1) {
        write_char(x, 30, *hw1);
       x++;
       hw1++;
   }
    return 0;
}
int main(void)
{
   int s=48;
   clear_char();
   //char* sc= (char)s;
   char* hw1= "Score is: ";
   *(hw1+9) = s;
   //strcat(hw1, sc);

     int x = 10;
      while (*hw1) {
        write_char(x, 10, *hw1);
       x++;
       hw1++;
   }
    volatile int* PTR = (int*) swbutton;
    int flag, movement, status;
    volatile int * A9_priv_timer_ptr = (int *) MPCORE_PRIV_TIMER;
    *A9_priv_timer_ptr = 10000000;              // timeout = 1/(200 MHz) x 40x10^6 = 0.2 sec
    *(A9_priv_timer_ptr+2) = 0b011;         // set bits: mode = 1 (auto), enable = 1

    flag = 1;
   int snakey = 200;
   int snakex = 200;
   int snakex_1 = 180;
   int snakey_1 = 200;
   int snakex_2 = 160;
   int snakey_2 = 200;

   int count = 0;// clear flag for KEY pressed
    while (flag)
    {
       movement = *(PTR);

      clear_screen();
      int thisx = (6*count)%320;
      if((6*count)%640>310){
      thisx = 320 - (6*count)%320;
```

```
   }
   int thisy = (6*count)%240;
   if((6*count)%480 >230){
   thisy = 240 - (6*count)%240;
   }
   snakex_2 = snakex_1;
   snakey_2 = snakey_1;
   snakex_1 = snakex;
   snakey_1 = snakey;

   if(movement==2 ){
      if(snakey>10)
   snakey-=20;
      else
         flag=game_over();
   }else if(movement==1){
      if ( snakey<220)

      snakey+=20;
      else
         flag=game_over();

   }else if(movement==4 ){
      if(snakex >10)
   snakex-=20;
      else
         flag=game_over();
   }else if(movement==8 ){
      if(snakex<300)
   snakex+=20;
      else
         flag=game_over();
   }

   snake(snakex, snakey, 0x07e0,20);
   snake(snakex_1, snakey_1, 0x07e0,20);
   snake(snakex_2, snakey_2, 0x07e0,20);
       circleBres(thisx+2, thisy+2, 2, 0xf800);

    if(thisx>=snakex && thisx <=snakex+20 && thisy>=snakey && thisy <=snakey+20)
   {
      s=s+1;

//char* sc= (char)s;
char* hw1= "Score is: ";
*(hw1+9) = s;
//strcat(hw1, sc);

  int x = 10;
   while (*hw1) {
     write_char(x, 10, *hw1);
    x++;
    hw1++;
}
   }
   count+=1;
     /* wait for timer */
     do
         status = *(A9_priv_timer_ptr+3); // read timer status
```

```
        while (status == 0);
        *(A9_priv_timer_ptr+3) = status;    // reset timer flag bit
    }
}
```

4 Working

- The Snake: The snake is represented by a series of green rectangles. The user can use push buttons to move the snake either north, south, east or west. Direction is chosen according to which key is pressed (Key[0] = move down, Key[1] = move up, Key[2] = move left, Key[3] = move right). Two keys pressed simultaneously do not change the snake's coordinates. It shifts its position by 20 units with each clock cycle. The snake successfully achieves its aim/ eats the ball when its head, i.e. the first rectangle, collides with the ball. The remainder of its body changes position according to the previous location of the head(the middle portion places itself where the head was previously, and the tail sets itself to the previous location of the snake's middle). The game is considered over if the snake's head clashes with any of the four walls.

- The Ball: The red circle on the screen represents the ball. It remains bouncing throughout the game. It moves diagonally with a slope of +-1. Every time the ball goes through the surface of the snake's head, it increases the score by 1. The user cannot alter the position of the ball.

- Text Display and Screen: The screen constantly displays the current score. It clears all the pixels as well as characters at every new iteration. It increments itself according to when the snake has clashed with the ball. Whenever the game ends, i.e., the snake touches a wall, the "Game Over" message is displayed on the screen.