```
/*
 * DinoDeezRealLatest.c
 *
 * Created: 2023-05-10 13:02:48
 * Author : an1620ph-s
 */


#define F_CPU 8000000UL
#include <avr/io.h>
#include <stdlib.h>
#include <stdio.h>
#include <util/delay.h>
#include <avr/interrupt.h>

volatile uint16_t adc_value; // global adc variabel
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define SCALE 4 // Scale must divide SCREEN_WIDTH and SCREEN_HEIGHT
#define WIDTH ((SCREEN_WIDTH) / (SCALE))
#define HEIGHT ((SCREEN_HEIGHT) / (SCALE))
char virtual_display[8][128];
int airtime = 0;

int readMicrophone(){

// Set  ADC input to A0 pin

ADMUX = (ADMUX & 0xF0 ) | (0x00 & 0x0F);

//start ADC conversion
//		ADCSRA |= (1 << ADSC);
//
//		// wait for conversion to complete
//		while (ADCSRA & (1 << ADSC));
int adc_value_sum =0;

for (int i = 0; i < 4; i++){
		ADCSRA |= (1 << ADSC);
		while (ADCSRA & (1 << ADSC));
		adc_value_sum += ADC;
}
int value_clean = adc_value_sum /4;
```

```
return value_clean;


}



void disableDisplay()
{
	PORTB &= 0b1111110;
}

void chipSelect2Low()
{
	PORTD &= 0b11111101;
}

void chipSelect1High(){
	PORTD |= 0b00000001;
}

void chipSelect2High(){
	PORTD |= 0b00000010;
}

void chipSelect1Low(){
	PORTD &= 0b11111110;
}

void resetHigh(){
	PORTD |= 0b00000100;
}

void resetLow()
{
	PORTD &= 0b11111011;
}

void readWriteHigh(){
	PORTD |= 0b00001000; //high innebär read, den vill läsa från LCD
}

void readWriteLow(){
```

```
	PORTD &= 0b11110111; //low innebär write, vi skriver till LCD
}

void directionInputHigh(){ //rs
	PORTD |= 0b00010000;
}

void directionInputLow(){ //rs
	PORTD &= 0b11101111;
}

void enableHigh()
{
	PORTD |= 0b00100000; //hög till låg flank, data inläses
}

void enableLow()
{
	PORTD &= 0b11011111;
}

void dataDisplay(){ //writeData{
	enableLow();
	enableHigh();
}

void s1(){
	chipSelect1Low();
	chipSelect1High();
}

void s2(){
	chipSelect1High();
	chipSelect1Low();
}


void enableX(char x){
	directionInputLow();
	readWriteLow();
	PORTB = 0b01000000 | x;

	enableHigh();
	enableLow();
```

```
}



void readOn(){
	resetHigh();
	readWriteHigh();
	DDRB = 0x0;
	dataDisplay();
}

void readOff()
{
	DDRB = 0xFF;
}


void startDisplay()
{
	PORTB = 0b00111111;
	chipSelect1Low();
	chipSelect2Low();

	resetHigh();
	readWriteLow();
	directionInputLow();
	enableHigh();

	chipSelect1High();
	chipSelect2High();

	enableLow();
	enableHigh();


}

void setCell(char cell)
{
	directionInputLow();
	readWriteLow();

	PORTB = 0b10111000 | cell;
```

```
	enableHigh();
	enableLow();
}


void waitForDisplay()
{
	PORTB &= 0x7F;
	DDRB = 0x7F;

	directionInputLow();
	readWriteHigh();

	while (PINB & 0x80)
	;

	DDRB = 0xFF;
}

void draw()
{
	for (char cell = 1; cell < 8; cell++)
	{
		for (char x = 0; x < 128; x++)
		{
			waitForDisplay();
			char displayX;
			if (x < 64)
			{
				waitForDisplay();
				chipSelect1High();
				waitForDisplay();
				chipSelect2Low();
				waitForDisplay();
				displayX = x;
			}
			else
			{
				waitForDisplay();
				chipSelect2High();
				waitForDisplay();
				chipSelect1Low();
				waitForDisplay();
				displayX = x - 64;
```

```
                }

                waitForDisplay();
                enableX(displayX);
                waitForDisplay();
                setCell(cell);
                waitForDisplay();
                waitForDisplay();
                directionInputHigh();
                readWriteLow();

                PORTB = virtual_display[cell][x];       //virtual_display[cell][x]

                enableHigh();
                enableLow();

                virtual_display[cell][x] = 0; // Reset virtual_display afterwards
            }
        }
}

void draw2(){
            for (char cell = 0; cell < 8; cell++)
            {
                for (char x = 0; x < 128; x++)
                {
                    waitForDisplay();
                    char displayX;
                    if (x < 64)
                    {
                        waitForDisplay();
                        chipSelect1High();
                        waitForDisplay();
                        chipSelect2Low();
                        waitForDisplay();
                        displayX = x;
                    }
                    else
                    {
                        waitForDisplay();
                        chipSelect2High();
                        waitForDisplay();
                        chipSelect1Low();
                        waitForDisplay();
```

```
                displayX = x - 64;
            }

            waitForDisplay();
            enableX(displayX);
            waitForDisplay();
            setCell(cell);
            waitForDisplay();
            waitForDisplay();
            directionInputHigh();
            readWriteLow();

            PORTB = virtual_display[cell][x];       //virtual_display[cell][x]

            enableHigh();
            enableLow();

        }
    }
}
void init(){
    DDRA = 0;
    DDRB = 0xFF;
    DDRD = 0xFF;
    startDisplay();
    clear();

    ADMUX |= (1 << REFS0);
    ADMUX &= ~(1 << REFS1);
    ADCSRA |= (1 << ADEN) | (1 < ADPS2) | (1 < ADPS1) | (1 << ADPS0) | (1 << ADIE) | (1
<< ADSC); // Enable ADC, ADIE aktiverar innerupt: ADSC startar en conversion
}


void setPixel(char x, char y, int scale)
{
    x *= scale;
    y *= scale;
    for (int i = 0; i < scale; i++)
    {
        for (int j = 0; j < scale; j++)
        {
            // ABS to compensates for upside-down display :)
            char yDisplay = abs((y + i) - 63); //58
```

```
                        char xDisplay = abs((x + j) - 127); // 122

                        virtual_display[yDisplay / 8][xDisplay] |= (1 << (yDisplay % 8));
                }
        }
}


void setPixel2 (uint8_t x, uint8_t y, uint8_t color)
{
        if (color)
        {
                virtual_display[y / 8][x] |= (1 << (y % 8));
        }
        else
        {
                virtual_display[y / 8][x] &= ~(1 << (y % 8));
        }
}

void terain() {
        for (int i = 0; i < 33; i++)
        {
                setPixel(i,64,5);
        }
        draw2();

}


 void character (uint8_t y1){
          //ögon
          setPixel2(102, y1-3, 5);
          setPixel2(102, y1-4 ,5);
          setPixel2(102, y1-5, 5);
          setPixel2(102, y1-6 ,5);

                  setPixel2(105, y1-3, 5);
                  setPixel2(105, y1-4 ,5);
                  setPixel2(105, y1-5, 5);
                  setPixel2(105, y1-6 ,5);

          //mun
           setPixel2(103, y1-8, 5);
```

```
          setPixel2(103, y1-9 ,5);

setPixel2(104, y1-8, 5);
setPixel2(104, y1-9 ,5);



         //huvud

        setPixel2(100,y1,5);
        setPixel2(100,y1+1,5);

        setPixel2(101,y1,5);
        setPixel2(101, y1+1, 5);

        setPixel2(102,y1,5);
        setPixel2(102,y1+1,5);


                setPixel2(100,y1,5);
        setPixel2(100,y1+1,5);

        setPixel2(101,y1,5);
        setPixel2(101, y1+1, 5);

        setPixel2(102,y1,5);
        setPixel2(102,y1+1,5);




        setPixel2(103,y1,5);
        setPixel2(103,y1+1,5);

        setPixel2(104,y1,5);
        setPixel2(100,y1+1,5);

        setPixel2(104,y1,5);
        setPixel2(104,y1+1,5);

        setPixel2(105,y1,5);
        setPixel2(105,y1+1,5);

        setPixel2(106,y1,5);
```

```
setPixel2(106,y1+1,5);

setPixel2(107,y1,5);
setPixel2(107,y1+1,5);

setPixel2(99, y1-1, 5);
setPixel2(98, y1-1, 5);

setPixel2(99, y1-2, 5);
setPixel2(98, y1-2, 5);

        setPixel2(99, y1-3, 5);
        setPixel2(98, y1-3, 5);

                setPixel2(99, y1-4, 5);
                setPixel2(98, y1-4, 5);

                        setPixel2(99, y1-5, 5);
                        setPixel2(98, y1-5, 5);

                        setPixel2(99, y1-6, 5);
                        setPixel2(98, y1-6, 5);

                        setPixel2(99, y1-7, 5);
                        setPixel2(98, y1-7, 5);

                        setPixel2(99, y1-8, 5);
                        setPixel2(98, y1-8, 5);

                        setPixel2(99, y1-9, 5);
                        setPixel2(98, y1-9, 5);

                        setPixel2(100,y1-10,5);
                        setPixel2(100,y1-11,5);

                        setPixel2(101,y1-10,5);
                        setPixel2(101, y1-11, 5);

                        setPixel2(102,y1 -10,5);
                        setPixel2(102,y1-11,5);

                        setPixel2(103,y1 -10,5);
                        setPixel2(103,y1 -11,5);
```

```
setPixel2(104,y1 -10 ,5);
setPixel2(100,y1 -11 ,5);

setPixel2(104,y1 -10 ,5);
setPixel2(104,y1 -11,5);

setPixel2(105,y1 -10 ,5);
setPixel2(105,y1 -11,5);

setPixel2(106,y1 -10 ,5);
setPixel2(106,y1 - 11,5);

setPixel2(107,y1 - 10,5);
setPixel2(107,y1 - 11,5);

setPixel2(108, y1-1, 5);
setPixel2(109, y1-1, 5);

setPixel2(108, y1-2, 5);
setPixel2(109, y1-2, 5);

setPixel2(108, y1-3, 5);
setPixel2(109, y1-3, 5);

setPixel2(108, y1-4, 5);
setPixel2(109, y1-4, 5);

setPixel2(108, y1-5, 5);
setPixel2(109, y1-5, 5);

setPixel2(108, y1-6, 5);
setPixel2(109, y1-6, 5);

setPixel2(108, y1-7, 5);
setPixel2(109, y1-7, 5);

setPixel2(108, y1-8, 5);
setPixel2(109, y1-8, 5);

setPixel2(108, y1-9, 5);
setPixel2(109, y1-9, 5);
```

```
        draw2();


 }

void killGAME(){
                clear();
                GameOver();

                _delay_ms(1000);
                game();

}

void fence (){
        int air = 0;
        for (int i = 25; i > -1; i--){
                int x = readMicrophone();
                if (airtime == 0){
                        if (x > 600) {
                                character(3);
                                airtime = 2;
                                air = 1;

                        }
                        else{

                                character(8);
                                air = 0;
                        }
                }
                else{
                        airtime--;
                        character(3);

                }
                x = 0;



        setPixel(i, 61, 5);
        setPixel(i, 62, 5);
        if (i == 5 && air == 0 ){
```

```
                killGAME();
        }


        draw();


        }
}


ISR(ADC_vect) {
        adc_value = ADC;
}

void StartGame(){
int x = 10;
int y = 6;
int s = 3;


//S

setPixel(x+ 4, y +0, s);
setPixel(x+ 3, y + 0, s);
setPixel(x + 2, y + 0 , s);
setPixel(x+2, y +1, s);
setPixel(x+2, y+2, s);
setPixel(x+3, y + 2, s);
setPixel (x+4, y+ 2, s);
setPixel( x+ 4, y +2, s);
setPixel ( x + 4, y + 3,s);
setPixel (x+4, y+4, s);
setPixel(x+3, y+4, s);
setPixel(x+2, y+ 4, s);

x+=2;



//T
```

```
setPixel(x+4, y + 0 , s);
setPixel (x+ 5, y + 0, s );
setPixel (x+ 6, y+ 0, s );
setPixel(x+5, y+1, s);
setPixel( x+ 5, y+2, s);
setPixel(x+5, y+3, s);
setPixel(x+5, y+4, s);

x+=4;

//A
setPixel(x+5, y + 0 , s);
setPixel (x+4, y+0, s);
setPixel(x+6, y + 0, s);
setPixel(x+4, y+1, s);
setPixel(x+6, y+1, s);
setPixel(x+4, y+2 , s);
setPixel(x+6, y+2, s);
setPixel(x+4, y+3, s);
setPixel(x+6, y+3, s);
setPixel(x+5, y+3, s);
setPixel(x+4, y+4, s);
setPixel(x+6, y+4, s);

x+=4;

//R
setPixel(x+4, y+0, s);
setPixel(x+5, y+0, s);
setPixel(x+6, y+0, s);
setPixel(x+4, y+1, s);
setPixel(x+4, y+2, s);
setPixel(x+5, y+2, s);
setPixel(x+6, y+2, s);
setPixel(x+6, y+1, s);
setPixel(x+4, y+ 3, s);
setPixel(x+4, y+ 4, s);
setPixel(x+5, y+ 3, s);
setPixel(x+6, y+4, s);

x+=4;
setPixel(x+4, y + 0 , s);
setPixel (x+ 5, y + 0, s );
setPixel (x+ 6, y+ 0, s );
```

```
setPixel(x+5, y+1, s);
setPixel( x+ 5, y+2, s);
setPixel(x+5, y+3, s);
setPixel(x+5, y+4, s);


draw();




}

void GameOver() {
int x = 0;
int y = 6;
int s = 3;

//G

setPixel(x+4, y+0 ,s);
setPixel (x+5, y+0, s);
setPixel(x+6, y+0, s);
setPixel(x+7, y+0, s);
setPixel(x+4, y+1, s);
setPixel(x+4, y+2,s);
setPixel(x+4, y+3, s);
setPixel(x+4, y+4,s);
setPixel(x+5, y+4, s);
setPixel(x+6, y+4, s);
setPixel(x+7, y+4,s);
setPixel(x+7, y+3, s);
setPixel(x+7, y+2, s);
setPixel(x+6, y+2, s);

x+=5;

// A
setPixel(x+5, y + 0 , s);
setPixel (x+4, y+0, s);
setPixel(x+6, y + 0, s);
setPixel(x+4, y+1, s);
setPixel(x+6, y+1, s);
setPixel(x+4, y+2 , s);
```

```
setPixel(x+6, y+2, s);
setPixel(x+4, y+3, s);
setPixel(x+6, y+3, s);
setPixel(x+5, y+3, s);
setPixel(x+4, y+4, s);
setPixel(x+6, y+4, s);

x+=4;

//M
setPixel(x+4, y + 0 , s);
setPixel (x+4, y+1, s);
setPixel(x+4, y + 2, s);
setPixel(x+4, y+3, s);
setPixel(x+4, y+4, s);
setPixel(x+5, y + 1 , s);
setPixel (x+7, y+1, s);
setPixel(x+6, y+2, s);
setPixel(x+8, y + 0 , s);
setPixel (x+8, y+1, s);
setPixel(x+8, y + 2, s);
setPixel(x+8, y+3, s);
setPixel(x+8, y+4, s);

x+=6;
//E
setPixel(x+4, y+0, s);
setPixel(x+5, y+0,s);
setPixel (x+6 , y+0, s);
setPixel(x+4, y+1, s);
setPixel(x+4, y+2, s);
setPixel(x+5, y+2, s);
setPixel(x+6, y+2, s);
setPixel(x+4, y+ 3, s);
setPixel(x+4, y+4,s);
setPixel(x+5, y+4,s);
setPixel(x+6, y+4,s);


x+=6;

//O
setPixel(x+4, y + 0 , s);
setPixel( x+5, y+0, s);
```

```
setPixel(x+6 , y+0, s);
setPixel(x+4, y + 1 , s);
setPixel(x+6 , y+1, s);
setPixel(x+4 , y+2, s);
setPixel(x+6 , y+2, s);
setPixel(x+4 , y+3, s);
setPixel(x+6 , y+3, s);
setPixel(x+4, y + 4 , s);
setPixel( x+5, y+4, s);
setPixel(x+6 , y+4, s);

x+=3;

//V
setPixel(x+5, y+0 ,s);
setPixel(x+7 , y + 0, s);
setPixel(x+5, y+1, s);
setPixel(x+7, y+1, s);
setPixel(x+5, y+2, s);
setPixel(x+7, y+2,s);
setPixel(x+5, y+3, s);
setPixel (x+7, y+3, s);
setPixel(x+6, y+4, s);

x+=5;
//E
setPixel(x+4, y+0, s);
setPixel(x+5, y+0,s);
setPixel (x+6 , y+0, s);
setPixel(x+4, y+1, s);
setPixel(x+4, y+2, s);
setPixel(x+5, y+2, s);
setPixel(x+6, y+2, s);
setPixel(x+4, y+ 3, s);
setPixel(x+4, y+4,s);
setPixel(x+5, y+4,s);
setPixel(x+6, y+4,s);


x+=4;
//R
setPixel(x+4, y+0, s);
setPixel(x+5, y+0, s);
setPixel(x+6, y+0, s);
```

```
setPixel(x+4, y+1, s);
setPixel(x+4, y+2, s);
setPixel(x+5, y+2, s);
setPixel(x+6, y+2, s);
setPixel(x+6, y+1, s);
setPixel(x+4, y+ 3, s);
setPixel(x+4, y+ 4, s);
setPixel(x+5, y+ 3, s);
setPixel(x+6, y+4, s);



draw();
}
void game(){

        StartGame();

        while(1){
        int blowstart = readMicrophone();
                if(blowstart > 700){
                        while(1){

                                terain();
                                fence();
                        }
                }
                blowstart = 0;
        }







}
void clear(){
        for (char cell = 0; cell < 8; cell++)
        {
                for (char x = 0; x < 128; x++)
                {
                        waitForDisplay();
                        char displayX;
                        if (x < 64)
```

```
                        {
                                waitForDisplay();
                                chipSelect1High();
                                waitForDisplay();
                                chipSelect2Low();
                                waitForDisplay();
                                displayX = x;
                        }
                        else
                        {
                                waitForDisplay();
                                chipSelect2High();
                                waitForDisplay();
                                chipSelect1Low();
                                waitForDisplay();
                                displayX = x - 64;
                        }

                        waitForDisplay();
                        enableX(displayX);
                        waitForDisplay();
                        setCell(cell);
                        waitForDisplay();
                        waitForDisplay();
                        directionInputHigh();
                        readWriteLow();

                        PORTB = virtual_display[cell][x];       //virtual_display[cell][x]

                        enableHigh();
                        enableLow();

                        virtual_display[cell][x] = 0; // Reset virtual_display afterwards
                }
        }
}

int main(void)
{
        init();

        character(15);

        //sei();
```

```
    //game();

        //StartGame();


// int x = 5;
// int y = 7;
// int micInput = readMicrophone();
// character(8,9,10);
//        setPixel(4,5,5);
//        setPixel(5,5,5);
//        setPixel(6,5,5);
//        setPixel(7,5,5);
//        setPixel(7,5,5);
//                        draw();

//    while (1) {
//                terain();
//
//                fence();
//

//                setPixel(x,y,5);
//                setPixel(x,y +1,5);
//                setPixel(x,y +2,5);




        // }
}
```