BouncingBall

Poročilo za 2. domačo nalogo pri predmetu Organizacija računalnikov

Avtor: Žan Reščič, 63220274

Mentor: viš. pred. dr. Robert Rozman

Datum: 26.8.2024

V sklopu predmeta Organizacija računalnikov smo imeli nalogo, da ustvarimo poljuben projekt na mikrokontrolerju STM32H750B-DK. Cilj naloge je bil, da ustvarim igrico z uporabo zaslona na dotik mikrokontrolerja. Ustvaril sem Bouncing Ball. Princip igre je, da žoga ostane v zraku čim več časa brez, da pade "izven zaslona". Deluje tako, da žoga zaradi gravitacije cel čas pada proti dnu in posledično izven zaslona, zato klikamo po njej, da jo odbiva v zrak. Žoga se giblje tudi horizontalno, smer je določena naključno. Če zadane levi, desni ali zgornji rob zaslona jo odbije v nasprotno smer, če gre pod spodnji rob zaslona se igra konča. Po končani igri se na zaslon izpiše število odbojev in čas igranja. Igra je bila napisana v okolju STM32CubeIDE v C/C++.

Začetni zaslon


Igranje igre

Končni zaslon

V datoteki Main.c je vsa programska koda za igro. Najprej so deklarirane vse spremenljivke, metode in strukture, med drugimi tudi struktura Ball od katere je odvisna cela igra. Ima parametra x, y, katera sta koordinate žoge, parametra vx, vy katera sta gravitacija horizontale in vertikale in še parameter radius. Uporabljeni so tudi parametri x_size in y_size katera sta velikost in širina zaslona, x in y katera sta koordinate pozicije dotika zaslona, px in py sta koordinate, ki beležita predhodni dotik zaslona. Start_time in time_elapsed se uporabljata za beleženje igralnega časa, time_str se uporablja za izpisovanje podatkov na zaslon in counter je števec odbojev žoge. TS_State je struktura preko katere inicializiramo in upravljamo oz. delamo z zaslonom.

```
uint32_t x_size;
uint32_t y_size;
uint16_t x = 0, y = 0;
uint16_t px = 0, py = 0;
uint32_t start_time;
uint32_t time_elapsed;
uint8_t time_str[20];
int counter = 0;

/*Touch screen struct to get data from*/
static TS_State_t TS_State;

UART_HandleTypeDef              UART3Handle;
TIM_HandleTypeDef               TIM3Handle;

// Define the ball structure
typedef struct {
    int x;
    int y;
    int vx;
    int vy;
    int radius;
} Ball;

Ball ball;
```

Deklaracija spremenljivk in struktur

V metodi main kličemo metode Init_LCD_TS() katera inicializira LCD in zaslon na dotik, Display_InitialContent() katera izriše začetni zaslon in DrawBall() katera inicializira in izriše žogo na zaslon.

```
static void Init_LCD_TS(void)
{
    /*LCD Init*/
    BSP_LCD_Init(0, LCD_ORIENTATION_LANDSCAPE);
    UTIL_LCD_SetFuncDriver(&LCD_Driver);
    BSP_LCD_GetXSize(0, &x_size);
    BSP_LCD_GetYSize(0, &y_size);

    /*TS Init*/
    TS_Init_t *TS_Init;
    TS_Init->Width = x_size;
    TS_Init->Height = y_size;
    TS_Init->Orientation = TS_SWAP_XY;
    TS_Init->Accuracy = 3;
    BSP_TS_Init(0, TS_Init);
}
```

Metoda Init_LCD_TS()

```c
static void Display_InitialContent(void)
{
    /*Get width & height*/
    BSP_LCD_GetXSize(0, &x_size);
    BSP_LCD_GetYSize(0, &y_size);

    /* Clear the LCD */
    UTIL_LCD_SetBackColor(UTIL_LCD_COLOR_LIGHTBLUE);
    UTIL_LCD_Clear(UTIL_LCD_COLOR_LIGHTBLUE);
    BSP_LCD_FillRect(0, 0, 0, x_size, y_size, UTIL_LCD_COLOR_LIGHTBLUE);

    /* Set the LCD Text Color */
    UTIL_LCD_SetTextColor(UTIL_LCD_COLOR_WHITE);

    /* Display LCD messages */
    UTIL_LCD_SetFont(&Font24);
    UTIL_LCD_DisplayStringAt(0, 15, (uint8_t *)"BOUNCING BALL", CENTER_MODE);

    UTIL_LCD_SetFont(&Font20);
    UTIL_LCD_DisplayStringAt(0, 45, (uint8_t *)"Click on the ball to start", CENTER_MODE);

}
```

Metoda Display_InitialContent()

```c
static void DrawBall(bool b)
{
    if (b){
        /*Create ball object*/
        ball.radius = 22;
        ball.x = x_size/2;
        ball.y = y_size/2+50;
        ball.vx = 0;
        ball.vy = 0;
    }

    /*Draw ball*/
    UTIL_LCD_FillCircle(ball.x, ball.y, ball.radius, UTIL_LCD_COLOR_WHITE);
}
```

Metoda DrawBall()

V zanki while(1) preverjamo, če igralec pritisne na žogo, da se lahko začne igra. Pozicijo dotika zaslona dobimo z uporabo metode BSP_TS_GetState(0, &TS_State), katera zapiše pozicijo dotika v prej omenjeno strukturo TS_State. Iz strukture nato dobimo x in y koordinati dotika.

```c
/* Get Touch screen position */
BSP_TS_GetState(0, &TS_State);

/* Get the x & y coords */
x = TS_State.TouchX;
y = TS_State.TouchY;
```

Pridobivanje pozicije dotika zaslona

Z if stavkom preverimo če smo se dotaknili žoge, če je to res se začne beležiti čas igranja igre z HAL_GetTick() in sproži se metoda PlayGame() v kateri je celotna igra. Ko se igra konča se čas igranja ustavi, gremo iz while zanke in na zaslon se izpišeta čas igranja igre in število odbojev žoge.

```
while (1){

    /* Get Touch screen position */
    BSP_TS_GetState(0, &TS_State);

    /* Get the x & y coords */
    x = TS_State.TouchX;
    y = TS_State.TouchY;

    if ((y >= ball.y-10 && y <= ball.y+10) && (x >= ball.x-10 && x <= ball.x+10)) {
        UTIL_LCD_Clear(UTIL_LCD_COLOR_LIGHTBLUE);
        /* Start the game timer */
        start_time = HAL_GetTick(); // Get the current tick count

        PlayGame();

        /* Time played */
        time_elapsed = HAL_GetTick() - start_time;
        break;
    }
}


/* Game score displayed */
UTIL_LCD_Clear(UTIL_LCD_COLOR_LIGHTBLUE);

sprintf((char*)time_str, "Score: %d", counter);
UTIL_LCD_DisplayStringAt(0, 15, (uint8_t *)time_str, CENTER_MODE);

sprintf((char *)time_str, "Time: %lu.%lu s", (time_elapsed / 1000), (time_elapsed % 1000)/100);
UTIL_LCD_DisplayStringAt(0, 45, (uint8_t *)time_str, CENTER_MODE);

UTIL_LCD_DisplayStringAt(0, 75, (uint8_t *)"To play again click on", CENTER_MODE);
UTIL_LCD_DisplayStringAt(0, 95, (uint8_t *)"the reset button", CENTER_MODE);
```

While zanka in izpis podatkov na zaslon

Znotraj metode PlayGame() se najprej določijo meje generatorja naključnih števil, kateri se uporablja za določanje horizontalne gravitacije, kar premika žogo levo ali desno. Nato, če se je zgodil dotik zaslona dobimo položaj dotika, z uporabo metode BSP_TS_GetState(0, &TS_State) iz BSP knjižnice. Nato povečamo vertikalno gravitacijo za 1 ball.vy += 1 in posodobimo položaj žoge tako, da prištejemo horizontalno gravitacijo x koordinati ball.x += ball.vx in vertikalno gravitacijo y koordinati ball.y += ball.vy. Sledi čiščenje zaslon z metodo UTIL_LCD_Clear() in znova narišemo števec odbojev in žogo. Sledi preverjanje ali smo žogo dotaknili, če smo jo, spremenimo vertikalno gravitacijo, da jo dvigne ball.vy += -10, horizontalno gravitacijo da jo pomakne v levo oz. desno stran ball.vx += random_number in števec odbojev se poveča. Za štetje odbojev si beležimo tudi prejšnje pozicije dotika zaslona, da lahko pravilno določimo odboje. Če sta prejšnji in trenutni dotik zaslona enaka, števca ne povečamo. Preveriti moramo še, če je žoga znotraj zaslona oz. če je zadela katerega od robov zaslona. To naredimo tako, da z if stavki preverimo če se x oz. y koordinata strukture ball dotika katerega od robov zaslona. Če je znotraj zaslona se igra nadaljuje. Prav tako se igra nadaljuje, če je zadela levi, zgornji ali desno rob samo, da se še žoga odbije v nasprotno smer. Če je zadela spodnji rob se igra konča in se vrnemo v main() metodo.

```c
static void PlayGame(void)
{
    srand(time(0));
    int lower = -5, upper = 5;
    int random_number;
    while(1){
        /* Get Touch screen position */
        BSP_TS_GetState(0, &TS_State);

        /*Get the x & y coords*/
        x = TS_State.TouchX;
        y = TS_State.TouchY;

        /* Update the ball's position */
        ball.vy += 1; // gravity
        ball.x += ball.vx;
        ball.y += ball.vy;

        UTIL_LCD_Clear(UTIL_LCD_COLOR_LIGHTBLUE);

        /*Draw ball and score*/
        sprintf((char*)time_str, "%d", counter);
        UTIL_LCD_DisplayStringAt(0, 15, (uint8_t *)time_str, CENTER_MODE);
        DrawBall(false);

        /*Check if the ball is touched*/
        if ((y >= ball.y-ball.radius/2 && y <= ball.y+ball.radius/2) && (x >= ball.x-ball.radius/2 && x <= ball.x+ball.radius/2)) {
            /*Random generator for horizontal bounce*/
            random_number = (rand() % (upper - lower + 1)) + lower;

            ball.vy += -10; // kick the ball up
            ball.vx += random_number; // kick the ball sideways
            if(x != px && y != py){
                counter += 1; // count the bounces
                px = x;
                py = y;
            }
        }
```

Prvi del metode PlayGame()

```c
        /* Check for collisions with screen boundaries*/
        if (ball.x - ball.radius < 0) {
            ball.vx += 4; // bounce horizontally
        }
        if (ball.x + ball.radius > x_size) {
            ball.vx += -4; // bounce horizontally
        }
        if (ball.y - ball.radius < 0) {
            ball.vy = 4; // bounce vertically
        }
        if (ball.y + ball.radius/2 > y_size+20) {
            break; //game over
        }
        HAL_Delay(26); // control frame rate
    }
}
```

Drugi del metode PlayGame()

Naloga mi je bila zanimiva, najprej sem nekaj časa porabil, da sem razumel kako deluje LCD zaslon na dotik, ko sem to razumel je projekt bilo dokaj enostavno narediti. Logika igre mi je delala malce težav, saj sem prvič naredil takšen tip igre. Za lažje razumevanje in implementiranje sem se spomnil na igro Flappy Bird, katera ima podobno logiko. S končnim projektom sem zadovoljen in pri pisanju igre sem se tudi naučil nekaj novih stvari.

Github repozitorij
Video delovanja