2013-01-03

PONG

TEIS 2013
C Kurs

Sammanfattning:

Ett spel , Pong har tagits fram till alteras DE2-115 utvecklings kort. Spelet går ut på att få en boll att studsa på motståndarens vägg och försvara sin egen.
Bsp inställningar:

- Reduced device drivers
- Small c lib
- C++ unabled

Kod storlek:
(Pong.elf) 31 KBytes program size (code + initialized data)

Jonas Torstensson
Jonas.torstensson@live.com

Innehållsförteckning:

1 Inledning

Ett spel , Pong har tagits fram till alteras DE2-115 utvecklings kort. Spelet går ut på att få en boll att studsa på motståndarens vägg och försvara sin egen. Till hjälp för att göra det har spelarna en liten bricka som flyttas så att bollen studsar på den istället för väggen.

2 Specifikation

Ett Spel skall skrivas till alteras DE2-115 utvecklings kort i språket c. Spelet är en enkel variation av pong . Spelet är väldigt enkelt och består av två brickor varav en som spelaren styr medan den andra styrs av datorn. En boll studsar mellan dessa brickor och det går ut på att få bollen att studsa in bakom motståndaren. Spelet kommer ha en meny där svårighets grad kan väljas och kommer att styras med trycknapparna.

3 Arkitektur mjukvara

3.1 Översikt

Alla funktioner i spelet ligger i de två filerna PONG.c och vga.c. Både Pong.c och vga.c har varsin header fil med definitioner och prototyper i. Det finns även en header fil som kallas font.h denna innehåller de tecken som används i spelet.

3.2 Filer

Spelet består av följande filer:

- Main.c

Här kallas alla funktioner ifrån

- Pong.c

Alla funktioner som styrspelet

- Vga.c

Alla funktioner som ritar grafik

- Pong.h

Fil huvud till pong.c prototyper definitioner mm..

- vga.h

Fil huvud till vga.c prototyper definitioner mm..

- font.h

Alla tecken som används i char_print() finns här.

3.3 Funktioner

Spelet består av följande funktioner:

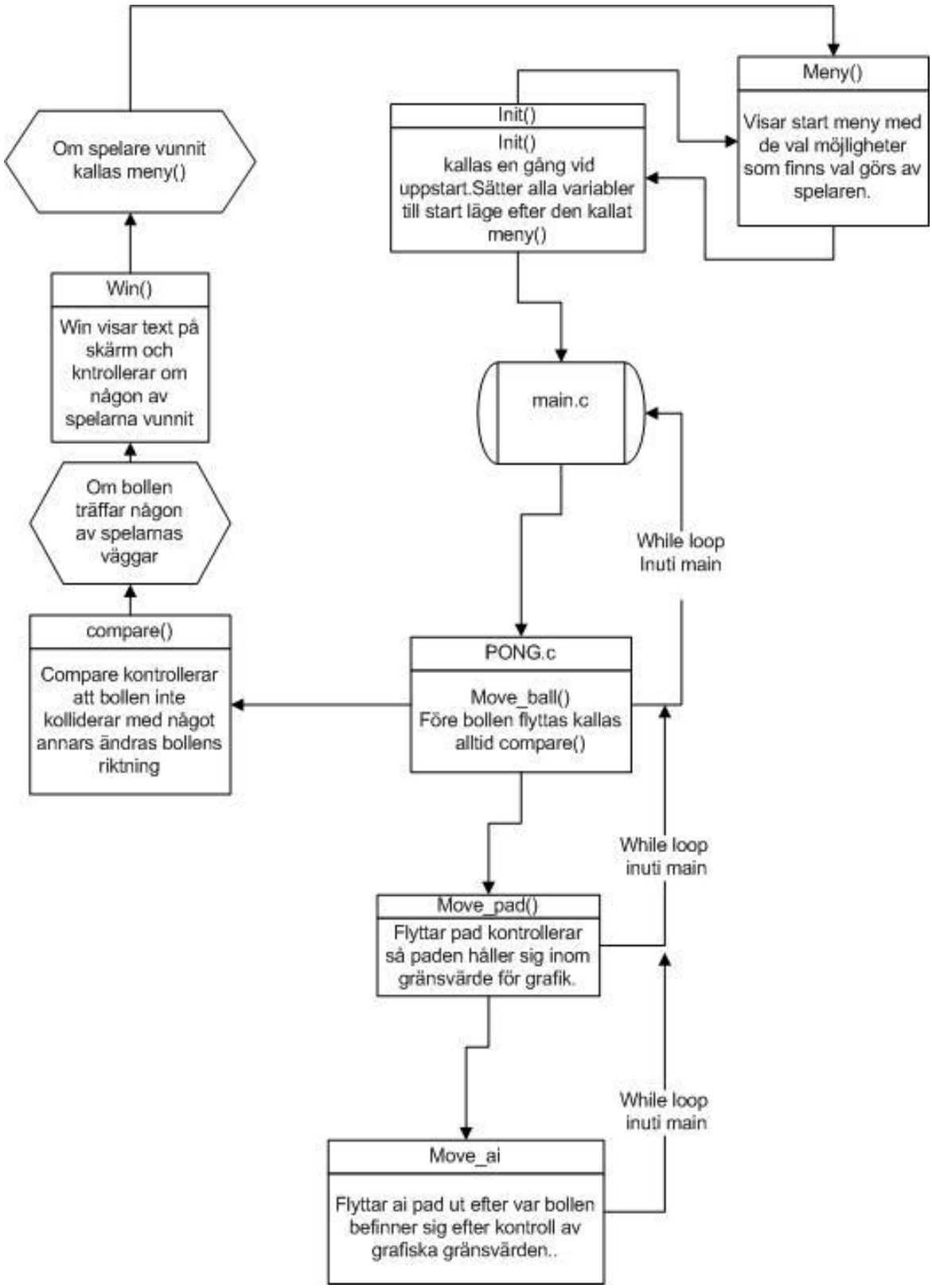
PONG.c

- **void meny**()
- **void init**()
- **void move_ball**()
- **void move_pad**(alt_u8 current)
- **void init**()
- **void AI**();

VGA.c

- **void print_pix**(alt_u16 x,alt_u16 y,alt_u8 rgb);
- **void swap**(alt_u16 *x,alt_u16 *y);
- **void line**(alt_u16 x0,alt_u16 x1,alt_u16 y0,alt_u16 y1,alt_u8 RGB);
- **void print_line**(alt_u16 x_start, alt_u16 y_start, alt_u16 x_end, alt_u16 y_end, alt_u16 thickness, alt_u8 RGB);
- **void print_circle**(alt_u16 radie, alt_u16 x_centrum , alt_u16 y_centrum, alt_u8 rgb);
- **void repaint**(alt_u8 rgb);
- **void print_word**(alt_u16 x, alt_u16 y, alt_u8 fg_RGB, alt_u8 bg_RGB, **char** *pek);
- **void print_char**(alt_u16 x,alt_u16 y,alt_u8 RGB,alt_u8 BG_RGB,alt_u8 Character);
- **alt_u8 read_pixel_ram_int**(alt_u16 x_start, alt_u16 y_start)

3.4 Flödes schema



Figur 1 Flödes schema för pong

Spelet börjar med att kalla init() som rensar skärm och därefter kallar meny().
Meny() skriver ut de val som finns och spelaren väljer med hjälp av tryck knapparna svårighetsgrad. Programmet går där efter in i mains while loop och stannar där resten av spelets gång. Inne i loopen kallas move_ball(), move_pad(), move_ai() i tur och ordning om och om igen. Funktionerna i sig kallar sedan andra funktioner de behöver för att fungera (se figur1).

- **void meny**()
meny() kallas alltid av init() meny() kallar där efter de funktioner som behövs för att skriva ut den text som finns i menyn.
- **void init**()
void init() blankar skärm och kallar meny() därefter sätts start värdena till spelet.
- **void move_ball**()
Kallar först compare så att bollens koordinater jämförs med omgivningen. Flyttar där efter boll och använder sig sedan av print_cirkle för att skriva ut en cirkel.
- **void compare** ( )
compare() kallas av move_ball() och jämför bolls koordinater med omgivning. Om kollision konstateras ändras riktning. Funktionen kallar read_pixel_ram_int som läser pixelen rakt framför boll för att konstatera om bollen håller på att kollidera med något.
- **void move_pad**(alt_u8 current)
Flyttar spelarens pad om knapptrycks, efter kontroll av gräns värde för grafik.
- **void AI**();
Flyttar pad utefter var boll befinner sig på skärmen. Funktionen är väldigt simpel och imiterar bara bollens y koordinat. Svårighets graden består i hur närma bollen måste vara innan paden får röra sig.

3.5 VGA funktioner

- **void print_char**(unsigned int **x**,unsigned int **y**,unsigned int **rgb**,unsigned BG_RGB,char Character);

Prints the Character with the color *rgb* and the background color BG_RGB at the coordinate (x, y). Hämtar information om tecken ifrån font.h.

- **void print_circle**(alt_u16 radie, alt_u16 x_centrum , alt_u16 y_centrum, alt_u8 rgb)

Funktionen tar fyra argument som i tur och ordning är radie(radie) centrum koordinater x, y(x_centrum,y_centrum) och färg(rgb). När funktionen kallas ritar den en cirkel fylld av färgen rgb. Cirkeln ritas med x_centrum och y_centrum som mittpunkt med radien radie.

- **void print_line**(alt_u16 x_start, alt_u16 y_start, alt_u16 x_end, alt_u16 y_end, alt_u16 thickness, alt_u8 RGB)
-

Funktionen tar 6 argument var av de 2 första av dessa x_start och y_start. Dessa två värden bestämmer var linjen startar precis som x_end och y_end bestämmer var linjen slutar. Linjen kan ritas med en viss tjocklek som bestäms av thickness och har en färg som bestäms av RGB.

- **Void print_word**(*alt_u16 x,alt_u16 y,alt_u8 RGB, alt_u8 bg_RGB,**char** *pek*)

Skriver ut ett helt ord med hjälp av print_char. En pekare till en sträng ges som argument därefter matas print_char() bokstav för bokstav med innehållet av strängen.

- **alt_u8 read_pixel_ram_int**(alt_u16 x_start, alt_u16 y_start)

Läser pixel ifrån minnet. Tar x och y koordinat som argument och returnerar en alt_u8 som innehåller rgb data.

- **void repaint**(alt_u8 rgb);

Repaint ritar om hela skärmen i en färg som anges i funktionens argument.

- **void print_pix**(alt_u16 x,alt_u16 y,alt_u8 rgb);

Skriver ut en pixel på bestämda koordinater. Tar x och y samt färg som argument.

- **void swap**(alt_u16 *x,alt_u16 *y);

Tar två argument x och y dessa byter värde med varann.

3.6 Kod storlek/reducering

Koden har reducerats genom att support för c++ har plockats bort och att reduced drivers används samt att small c library används.

Kod storlek före reducering

```
                CONTENTS, ALLOC, LOAD, READONLY, CODE
 2 .text         0000f690  000801b4  000801b4  000011b4  2**2
                CONTENTS, ALLOC, LOAD, READONLY, CODE
 3 .rodata       00000690  0008f844  0008f844  00010844  2**2
                CONTENTS, ALLOC, LOAD, READONLY, DATA
 4 .rwdata       00002388  0008fed4  0009225c  00010ed4  2**2
                CONTENTS, ALLOC, LOAD, DATA, SMALL_DATA
 5 .bss          00000314  000945e4  000945e4  000135e4  2**2
                  ALLOC, SMALL_DATA
```

(Pong.elf) 73 KBytes program size (code + initialized data)

kod storlek efter reducering

```
                CONTENTS, ALLOC, LOAD, READONLY, CODE
 2 .text         00006a3c  000801b4  000801b4  000011b4  2**2
                CONTENTS, ALLOC, LOAD, READONLY, CODE
 3 .rodata       00000204  00086bf0  00086bf0  00007bf0  2**2
                CONTENTS, ALLOC, LOAD, READONLY, DATA
 4 .rwdata       00000ef8  00086df4  00087cec  00007df4  2**2
                CONTENTS, ALLOC, LOAD, DATA, SMALL_DATA
 5 .bss          00000178  00088be4  00088be4  00009be4  2**2
```

(Pong.elf) 31 KBytes program size (code + initialized data)

4 Hårdvara

Hårdvaran som skickats med av kunden består av en processor, ett ram, LCD display, VGA controller, interface till knappar lysdioder timer och ir.
Hårdvaran finns i filen trn_teis_c_time_limited.sof.

Tabell 1 visar adresserna till de olika periferierna på kortet

Slave Descriptor	Address Range	Size	Attributes/PIN
Terasic_IrDA_Rec_0	0x001010E0 - 0x001010E3	4	IR enheten
sysid	0x001010D8 - 0x001010DF	8	-
jtag_uart_0	0x001010D0 - 0x001010D7	8	Printf, scanf
Expansion_in	0x001010C0 - 0x001010CF	16	GPIO[7..0]
Expansion_out	0x001010B0 - 0x001010BF	16	GPIO[17..10]
lcd_0	0x001010A0 - 0x001010AF	16	LCD display
VGA_data_r	0x00101090 - 0x0010109F	16	VGA
VGA_Control_w	0x00101080 - 0x0010108F	16	VGA
VGA_data_w	0x00101070 - 0x0010107F	16	VGA
VGA_adress_w	0x00101060 - 0x0010106F	16	VGA
PERIPH_SYS_BUTTON_PIO	0x00101050 - 0x0010105F	16	Button 0-3
PERIPH_SYS_LEDS_PIO	0x00101040 - 0x0010104F	16	De längst till höger
PERIPH_HIGH_RES_TIMER	0x00101020 - 0x0010103F	32	timer
PERIPH_SYS_CLK_TIMER	0x00101000 - 0x0010101F	32	timer
onchip_memory2_0	0x00080000 - 0x000CFFFF	327680	memory

5 Verifiering

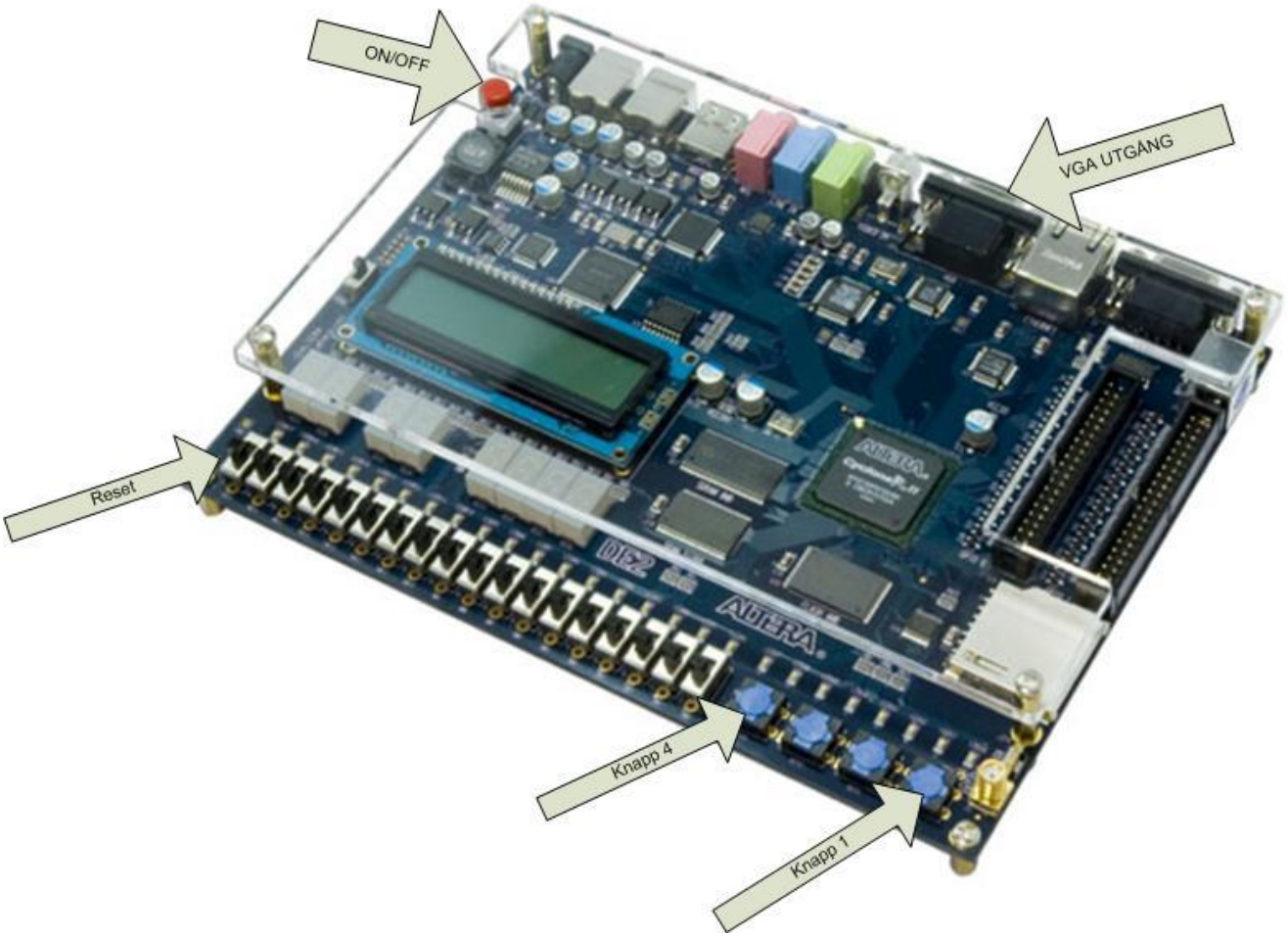
Spelet har verifierats och Debuggats under tiden som det har konstruerats. Sluttest har gjorts så att spelet fungerar som det ska att bollen inte fastnar och att objekten inte kan gå utan för skärm. Ett fel som uppmärksammats men inte kunnat åtgärdas ännu är att bilden deformeras lite när objekten ska flyttas på skärmen.

6 Bilagor

a.demonstrationsfil

```
// $Workfile : C_Ingenjörsjobb_1.c
//
// Programmer(s):Jonas Torstensson
// Date Created : 2012-12-25
// updated:2012-12-30
// Description :main huvud funktionen som är överst i
program strukturen.
//
//

#include "altera_avalon_pio_regs.h"
#include <unistd.h>
#include "PONG.h"


int main(){


init();

while(1){
move_ball();
move_pad(0);
AI();
usleep(FRAMERATE);
}


}
```

b.Spelmanual



Meny

Spelet inleds i en meny där svårighets grad väljs det finns tre nivåer att välja mellan easy medium och hard.
Val görs genom knapp 1 knapp 2 eller knapp3 först trycks därefter startas spelet med knapp 4.
Omspelet startas direkt är default värde för svårighets grad satt till medium.
Knapp 1 = Easy
Knapp 2 = Medium
Knapp 3= Hard
Knapp 4= Start

Spel

Spelet styrs sedan med tryckknapp ett och två .
Knapp 1 = ner
Knapp 2 = upp