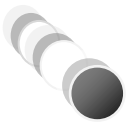


Projekat

Bouncing ball

Elektrotehnički fakultet
Univerziteta u Beogradu
Jun 2016.

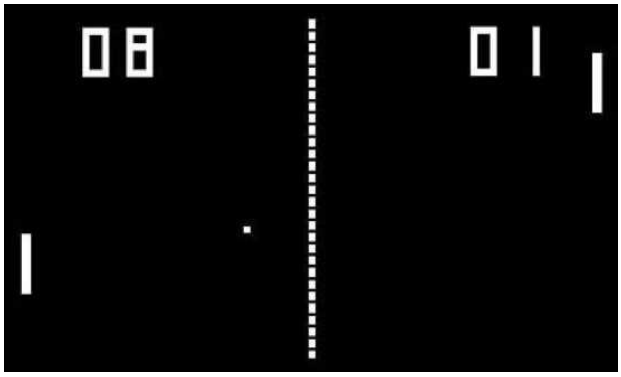
Predmet: Računarska elektronika — Jovanović Jelena 158/2012

Profesor: Dr Milan Prokin — Marinković Miloš 376/2012

Sadržaj

Uvod

Ovaj projekat je osmišljen tako da prikaže kako u programskom jeziku asembler može da se odradi igrica nalik jednoj od najjednostavnijih, najranijih i najpoznatijih arcade video igrica 70-ih – Pong.

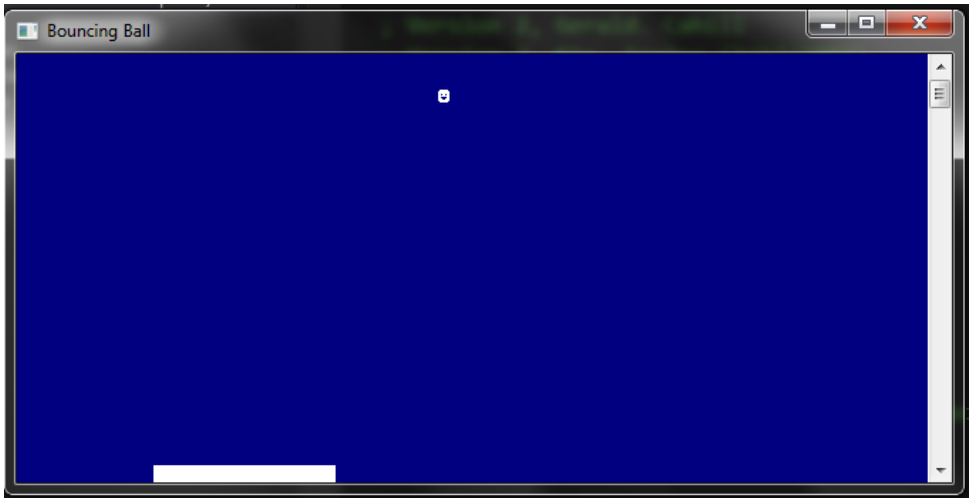


Pong je simulacija dvodimenzionalnog stoni tenis-a puštena 1972. godine koja je ubrzo postigla veliki uspeh. Igrač kontroliše paddle (reket) sa leve strane vertikalno i takmiči se protiv igrača desno koji je ili kompjuter ili drugi igrač. Uz pomoć reketa igrači pokušavaju da pošalju lopticu dalje od svoje strane na protivnikovu. Poen se osvaja kada protivnik ne stigne da vrati lopticu.

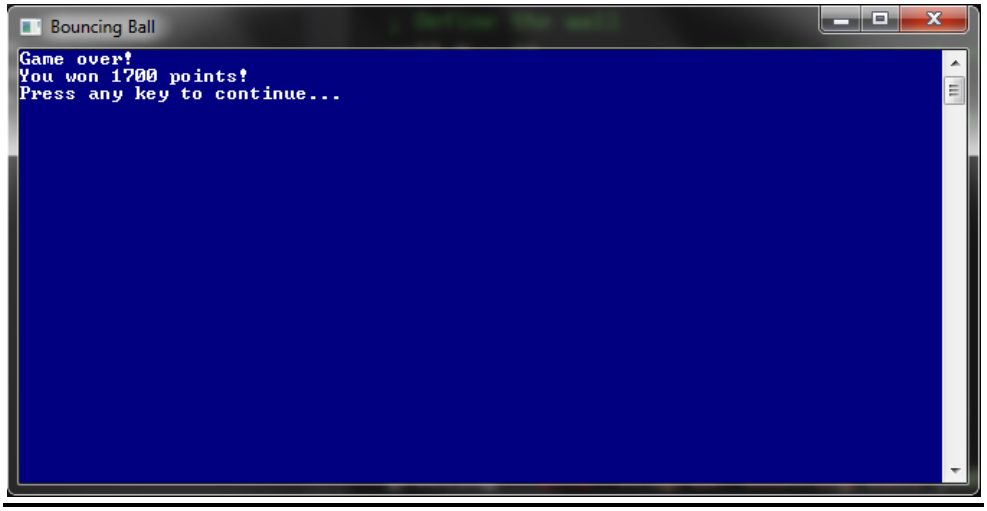
Pong je bio osnova za dobar deo arcade igrica od tad pa na dalje. Jedna od verzija igrica na tu temu je i naš projekat.

Bouncing ball

Program je napravljen u asembleru. Reket se nalazi na donjem delu ekrana i kontroliše se uz pomoć dva tastera levo – desno. Tasteri za kontrolu su levi i desni Ctrl. Cilj igrice je zadržati lopticu da ne padne pored reketa.



Loptica se kreće određenom brzinom koja se menja nakon određenog broja uspešnih odbijanja reketom dekrementacijom potrebnog delay-a za crtanje. Za svaki uspešan udarac dobija se po 100 poena. Kada se igrica završi, tj. kada loptica propadne kroz donji zid ekrana, na ekranu je ispisano koliko je poena osvojeno.



Bouncing ball – kod

U prilogu je detaljno iskomentarisan kod programa koji smo napisali u skladu sa traženim postavkama zadatka.

```
; Solution program.
; Original by Alejandro Presto - Feb 2003
; Version 2, Gerald. Cahill
; Version 3, Kip. Irvine (2/17/2003)
; Version 4, Jovanovic, Marinkovic (2016)

INCLUDE Irvine32.inc

; dl = current x
; dh = current y

; bl = next x
; bh = next y

ball = 2                ;a happy face (1) looks good too.


; Define the wall
wall_Y = 24             ;y position (row number)
wallLeft = 33           ;position of left side
wallRight = 48          ;position of right side
```

```
; Define the window size
xmin = 0                ;left edge
xmax = 79               ;right edge
ymin = 0                ;top
ymax = 24               ;bottom

.data
ddx BYTE 1              ;x increment per iteration
ddy BYTE 1              ;y increment per iteration
greeting   BYTE "Program: Bouncing Ball",0dh,0ah,
        "Close the window to end the program",0dh,0ah,0
goodbye BYTE "Game over!",0dh,0ah,0
goodbye1 BYTE "You won ",0
goodbye2 BYTE "00 points!",0dh,0ah,0

titleStr BYTE "Bouncing Ball",0
var1 BYTE 1
var2 BYTE 1
var3 DWORD 0 ;pad hits
var4 BYTE 0             ;potrebna promenljiva za ispis
var5 BYTE 0             ;potrebna promenljiva za ispis
drawDelay DWORD 50      ;milliseconds between redrawing the ball //dekrementiranje sa
                        brojem udaraca!!

.code
```

```
main PROC
;---------- intro stuff, just for my demo
        call Clrscr
        mov  edx,offset greeting
        call WriteString
        call WaitMsg
        call Clrscr

; PROGRAM STARTS HERE
;-----------------------------------------------------------
        mov eax,white + (blue * 16)
        call SetTextColor
        INVOKE SetConsoleTitle, ADDR titleStr
        call Clrscr

;----- hides the cursor ----------------------------------------
.data
cursorInfo CONSOLE_CURSOR_INFO <>
outHandle  DWORD ?
.code
        INVOKE GetStdHandle, STD_OUTPUT_HANDLE
        mov  outHandle,eax
        INVOKE GetConsoleCursorInfo, outHandle, ADDR cursorInfo
        mov  cursorInfo.bVisible,0
        INVOKE SetConsoleCursorInfo, outHandle, ADDR cursorInfo
;-----------------------------------------------------------
```

```
;------ Draw the Wall ---------------------------------------

        mov  dl,wallLeft
        mov  dh,wall_Y
        mov  ecx,wallRight - wallLeft + 1
        mov  al,0DBh            ; solid block character

DrawWall:
        call Gotoxy
        call WriteChar
        inc  dl
        loop DrawWall
;-------------------------------------------------------------


        mov dl,21               ;Initial value for X ball coordinate
        mov dh,8                ;Initial value for Y ball coordinate
        mov var1, wallLeft
        mov var2, wallRight
```

```
loop1:                          ;Infinite loop
        push dx
        INVOKE GetKeyState, VK_LCONTROL
        test eax,80000000h
        .IF !Zero?

        ;moveWallLeft
                mov  dl,var1
                mov  dh,wall_Y
                mov  ecx,wallRight - wallLeft + 1
                mov  al,' '       ; erasing wall

                DrawWall1:
                        call Gotoxy
                        call WriteChar
                        inc  dl
                        loop DrawWall1

                mov al,var1

                .IF al >= 0 + 3 && al <= 79
                        sub var1, 3
                        sub var2, 3
                .ENDIF

                mov  dl,var1
                mov  dh,wall_Y
```

```
        mov  ecx,wallRight - wallLeft + 1
        mov  al,0DBh            ; solid block character

        DrawWall2:
                call Gotoxy
                call WriteChar
                inc  dl
                loop DrawWall2

.ENDIF

INVOKE GetKeyState, VK_RCONTROL
test eax,80000000h
.IF !Zero?

;moveWallRight
        mov  dl,var1
        mov  dh,wall_Y
        mov  ecx,wallRight - wallLeft + 1
        mov  al,' '             ; erasing wall

        DrawWall3:
                call Gotoxy
                call WriteChar
                inc  dl
                loop DrawWall3
```

```
        mov al,var2

        .IF al <= 79 - 3 && al >= 0
                add var1, 3
                add var2, 3
        .ENDIF

        mov  dl,var1
        mov  dh,wall_Y
        mov  ecx,wallRight - wallLeft + 1
        mov  al,0DBh            ; solid block character

        DrawWall4:
                call Gotoxy
                call WriteChar
                inc  dl
                loop DrawWall4

.ENDIF

pop dx

mov bl,dl
add bl,ddx                      ;get potential next x
mov bh,dh
add bh,ddy                      ;get potential next y
```

```
.IF bh != wall_Y || bl > var2 || bl < var1
        jmp Check_rectangle_boundaries
    .ENDIF

    ; striking the left or right of the wall?
    .IF bl == var1 || bl == var2
            neg ddy
            add var3, 1
            .IF drawDelay >10
                    dec drawDelay
            .ENDIF
            jmp redraw

    .ELSE                       ; striking the middle of the wall
            neg ddy
            add var3, 1
            .IF drawDelay >14
                    dec drawDelay
            .ENDIF
            jmp redraw
    .ENDIF
```

```
Check_rectangle_boundaries:
        .IF bl < xmin || bl > xmax
          neg ddx
        .ENDIF

        .IF bh > ymax && bh!=255 && bh < 200
                jmp kraj
        .ENDIF

        .IF bh < ymin || bh> 200
                 neg ddy
        .ENDIF

redraw:
        call Gotoxy                 ;erase the ball
        mov  al,' '
        call WriteChar

        add  dl,ddx                 ;get new x
        add  dh,ddy                 ;get new y

        call Gotoxy                 ;print the ball
        mov  al,ball
        call WriteChar

        mov  eax,drawDelay          ;delay
        call Delay

        jmp  loop1
;----------------------------------------
```

```
kraj:

        call Clrscr
        mov  edx,offset goodbye
        call WriteString
        mov edx, offset goodbye1
        call WriteString



        .IF var3 >=100              ;odredjivanje stotine
                mov eax, var3
                mov drawDelay, eax
                mov bl, 100
                div bl
                mov var5, ah        ;ostakat desetica i jedinica
                mov var4, al        ;stotine
                        mov ah, 0   ;pocetak bloka za ispis stotine
                        mov var3, eax
                        add var3, 48
                        mov edx, offset var3
                        call WriteString
        .ELSE
                mov eax, var3       ;obezbedjivanje kompatibilnosti sa ostalim ispitivanjima
                                        kada je broj udaraca manji od 100
                mov var5, al
        .ENDIF
```

```
.IF var5 >= 10                  ;ispitivanje desetice
        mov eax, 0
        mov al, var5
        mov bl, 10
        div bl
        mov var5, ah            ;ostatak jedinica
        mov var4, al            ;desetice
            mov ah, 0           ;pocetak bloka za ispis desetica
            mov var3, eax
            add var3, 48
            mov edx, offset var3
            call WriteString
    .ELSE

        .IF drawDelay>=100 ;provera da li je broj udaraca bio izmedju 100 i 109
            mov var3, 48 ;umetanje 0 na mesto desetice
            mov edx, offset var3
            call WriteString
        .ENDIF
    .ENDIF
```

```
mov eax, 0                          ;pocetak bloka za ispis jedinica
        mov al, var5
        mov var3, eax
        add var3, 48
        mov edx, offset var3
        call WriteString



        mov edx, offset goodbye2
        call WriteString
        call WaitMsg

        call Clrscr
        exit

;---------------------------------------

main ENDP

END main
```