UNIVERZITET U BEOGRADU

ELEKTROTEHNIČKI FAKULTET

KATEDRA ZA ELEKTRONIKU

PROJEKAT IZ RAČUNARSKE ELEKTRONIKE

A R K A N O I D

Projekat realizovali:
Turkmanović Haris 516/2014
Vukoje David 541/2014

Profesor
Milan Prokin

Asistent
Aleksandra Lekić

„Igrač u igrici kontroliše pedalu pomerajući je levo i desno tako da kuglica ne dodirne donju ivicu prozora. Kuglica i pedala se na početku nalaze na sredini donje ivice prozora i kuglica započinje svoje kretanje uvek pod istim uglom. Kada kuglica udari u neku od gornjih prepreka, ruši ih i igrač osvaja bodove. Udarom o bilo koju prepreku, izlazni ugao kuglice je jednak ulaznom u odnosu na normalu prepreke, pedale ili leve i desne ivice prozora. Prepreke u zavisnosti od boje nose različit broj poena: zelena - 80 poena, crvena - 90 poena, žuta - 120 poena, ljubičasta - 100 poena i plava - 50 poena. Siva prepreka se uništava ukoliko je dva puta udarena, a ako pedala uhvati srušenu sivu prepreku, povećava se dva puta. Igrica se završava kada se unište sve prepreke ili kada se izgubi 10 života kuglice. Na ekranu je sve vreme potrebno ispisivati preostali broj života kuglice i broj osvojenih poena. Igrica može da se prekine i pritiskom tastera ESC."

Sadržaj

1. Uvod 4
2. Grafički interfejs 7
2.1 Početni grafički interfejs 8
2.2 Glavni grafički interfejs 11
3.Algoritmi za pomeranje elemenata u igrici 13
3.2 Kretanje elemenata 14
3.2.2 Kretanje Lopte 14
3.2.3 Kretanje Bloka 15
4. MAIN procedura 16
5. Pokretanje igrice 18
6. Kodovi 19

1. Uvod

Slika 1. Izgled prozora neposredno pre početka igrice – Početni grafički interfejs

Pre samog pocetka izveštaja smatramo da je neophodno uvesti i objasniti pojmove koji predstavljaju osnovne gradivne elemente unutar same igrice , kao i objasniti načine na koje smo realizovali neke ideje i ciljeve koje smo sledili u realizaciji tih ideja.

Ono čemu smo težili jeste da u relativno grafički oskudnoj konzoli unesemo što veću dinamičnost da bi povećali interesantnost same igrice i privukli veću paznju onoga ko je igra. Takođe smo se trudili da što više napravimo igricu fleksibilnijom u smislu podešavanja i konfiguracije da bi se prilagodili većem broju igrača . Na primer , predpostavili smo da će postojati igrači koji će želeti da imaju mogućnost da ubrzaju ili uspore igricu, tj. da ubrzaju ili uspore pad, da će postojati igrači koji žele da određene blokove udaraju veći broj puta, da će postojati igrači koji žele da razni blokovi nose različit broj poena , ... *Default* podešavanja su nameštena tako da zadovolje specifikacije projekta.

Gore navedena fleksibilnost omogućava da se igrica nadogradi dodatnim grafickim interefejsima koji bi recimo igraču omogućavali da na početku igrice bira težinu igrice a svaka od težina bi sama podešavala ove parametre. Mi nismo bili u mogućnosti da odradimo to jer bi to isuviše otišlo od samog zahteva projekta ali smo ostavili mogućnost za dalju nadogradnju i rad na projektu u cilju njegovog usavršavanja.

Da bi smo, kao što smo već naveli, napravili igricu da bude što dinamičnija bilo je neophodno detaljnije istražiti funkcionisanje samog rada konzole i procedura koje omogućavaju odredjena ponašanja konzole.Ovde smo morali da odemo dalje od procedura i struktura koje pripadaju *Irvine32* biblioteci i da koristimo neke od procedura i struktura koje pripadaju Microsoft bibliotekama kao što su *kernel32.lib* i *user32.lib.* U ovim bibliotekama se nalaze procedure koje omogućavaju veću slobodu nad samom konzolom (inače funkcije *irvine32.lib* biblioteke su samo

apstrakcija funkcija *kernel32.lib* i *user32.lib* a u to se možemo uveriti ako pogledamo kodove funkcija iz *irvine32.lib* biblioteke). Deklaracija funkcija koje smo koristili a ne pripadaju *irvine32.lib* smeštene su u AdditionalDef.inc fajlu

```
;=================================================================================================================================
;---------------------------------------------------------------------------------------------------------------------------------
;------------------------------------------------- AdittionalDef.inc -------------------------------------------------------------
;---------------------------------------------------------------------------------------------------------------------------------
;------------------------------------------------------ Authors: -----------------------------------------------------------------
;---------------------------------------------- Turkmanovic Haris 516/2014 -------------------------------------------------------
;---------------------------------------------- Vukoje David 541/2014 ------------------------------------------------------------
;=================================================================================================================================
;Description:
;        This .inc file contains declarations of functions which are located in windows system Libraries such as
kernel32.lin and
;user32.lib . These declarations are necessary because they are not directly declared in Irvine32.lib
;=================================================================================================================================


INCLUDE Irvine32.inc
FreeConsole PROTO
AllocConsole PROTO

WriteConsoleOutputA PROTO ,
                            oHandle:HANDLE,
                            ArrayForRead:PTR _CHAR_INFO,
                            SizeOfArrayForRead:COORD,
                            StartPositionForRead:COORD,
                            ScreenBufferAreaForWrite:PTR SMALL_RECT
WriteConsoleOutputW PROTO ,
                            oHandle:HANDLE,
                            ArrayForRead:PTR _CHAR_INFO,
                            SizeOfArrayForRead:COORD,
                            StartPositionForRead:COORD,
                            ScreenBufferAreaForWrite:PTR SMALL_RECT
WriteConsoleOutputCharacterW PROTO,
                            hConsoleOutput:HANDLE,
                            lpCharacter:PTR WORD,
                            nLength:DWORD,
                            dwWriteCoord:COORD,
                            lpNumberOfCharsWritten:PTR DWORD
SetConsoleCursorPosition PROTO,
            hConsoleOutput:HANDLE,
                            wCursorPosition:COORD
ReadConsoleOutputA PROTO ,
                            oHandle:HANDLE,
                            ArrayForRead:PTR _CHAR_INFO,
                            SizeOfArrayForRead:COORD,
                            StartPositionForRead:COORD,
                            ScreenBufferAreaForWrite:PTR SMALL_RECT
CreateConsoleScreenBuffer PROTO,
                            dwDesiredAccess                         :DWORD ,
                            dwShareMode                                 :DWORD ,
                            lpSecurityAttributes        :PTR _SECURITY_ATTRIBUTES ,
            dwFlags                                                 :DWORD,
                            lpScreenBufferData          :PTR DWORD
SetConsoleActiveScreenBuffer PROTO,
                            handlee:HANDLE
FillConsoleOutputAttribute PROTO,
                            hConsoleOutput:HANDLE,
                            lpCharacter:WORD,
                            nLength:DWORD,
                            dwWriteCoord:COORD,
                            lpNumberOfCharsWritten:PTR DWORD
WriteConsoleOutputW PROTO ,
                            oHandle:HANDLE,
                            ArrayForRead:PTR _CHAR_INFO,
                            SizeOfArrayForRead:COORD,
                            StartPositionForRead:COORD,
                            ScreenBufferAreaForWrite:PTR SMALL_RECT
```

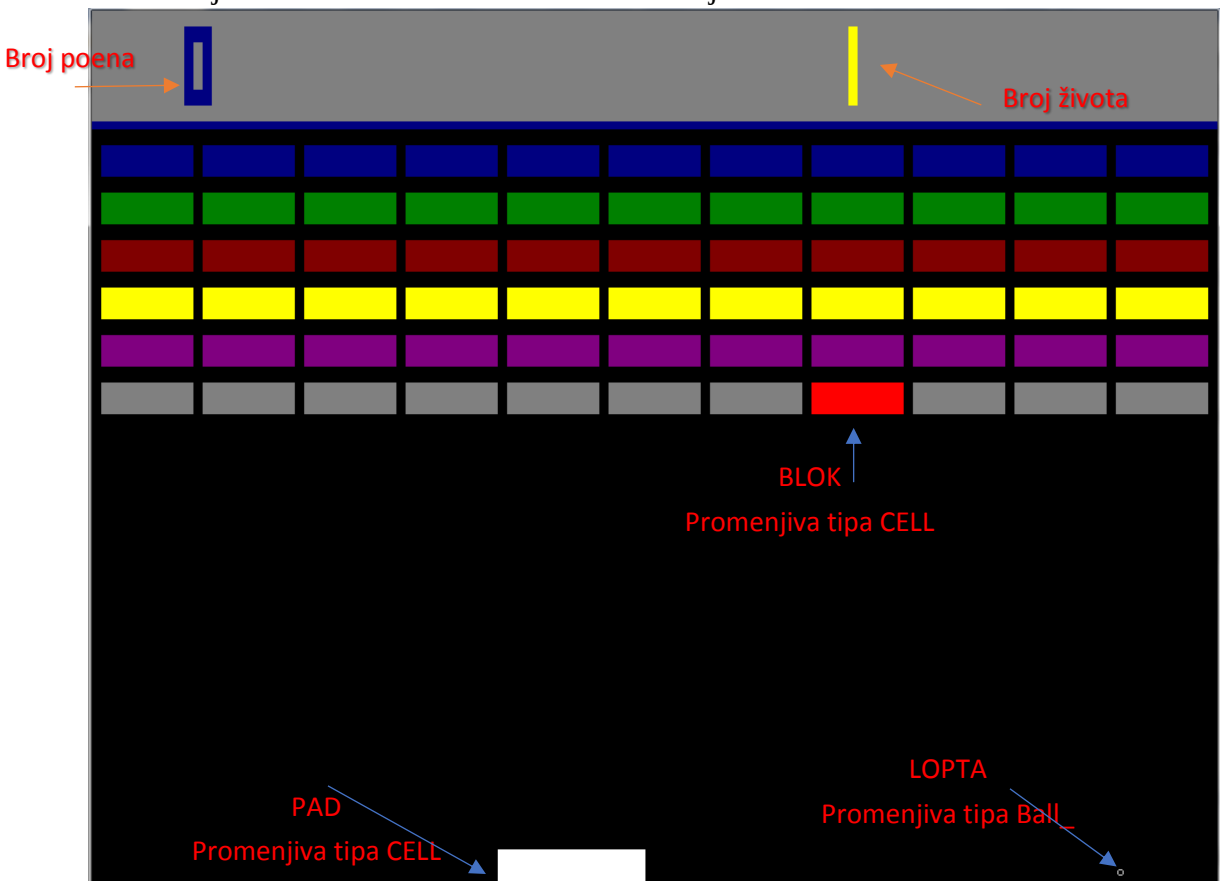
```
_CHAR_INFO STRUCT
                  AsciiChar WORD 219
                  Attributes WORD 0
_CHAR_INFO ENDS

_SECURITY_ATTRIBUTES STRUCT
                  nLength DWORD 0
                  lpSecurityDescriptor DWORD 0
   bInheritHandle DWORD 0
_SECURITY_ATTRIBUTES ENDS
```

Slika 2. Sadržaj Additional.inc fajla

Sada je potrebno da uvedemo pojmove koji definišu osnovne gradivne elemente same igrice, a to su *ćelija, lopta, blok, pad.*

Postoje dve osnovne strukture u samoj igrici a to su strukture CELL i BALL_.Ćelija je ustvari sama struktura CELL. *Lopta* je promenjiva tipa BALL_. *Blok* je promenjiva tipa CELL dok je i P*ad* je promenljiva tipa CELL.Razlika izmedju Pada i Bloka jeste u informaciji koju nose



Slika 3. Grafički interfejs glavnog dela igrice

Neophodno je naglasiti da su Lopta i Pad globalne promenjive dok su blokovi raspoređeni u memorijske lokacije koje posmatramo kao matricu . Matrica blokova je takodje globalna.

Deo grafickog interfejsa glavnog dela igrice koji se nalazi iznad plave linije predstavlja trenutnu informaciju o tome koliko igrač ima poena i koliko mu je preostalo života. O načinu i funkcijama koje su omogućile ispis brojeva i slova biće reči kasnije.

!NAPOMENA Preporuka je da se pre pokretanja igrice izvrši podešavanje konzole koje je objašnjeno u odeljku „Pokretanje igrice“

2. Grafički interfejs

Zbog boljeg razumevanja grafickih delova igrice neophodno je prvo objasniti kako funkcioniše ekran konzole i koje funkcije iz biblioteka *kernel32.lib* i *user32.lib* omogućavaju manipulaciju grafičkim delovima konzole.

Naime, prozor konzole je podeljen na veliki broj malih pravougaonika. U svaki od tih pravougaonika moguće je upisivati odredjeni karakter koji može biti zadat preko ASCII vrednosti ili Unicode vrednosti. Svakom od tih pravougaonika moguće je menjati boju pozadine i boju karaktera koji je upisan u taj pravougaonik.Takodje moguće je odjednom preko odredjenih funkcija manipulisati matricom tih pravougaonika a moguće je i editovati svaki od tih pravougaonika pojedinacno.

Pravougaonik je abstrahovan strukturom _CHAR_INFO koja ima dva polja:

- `AsciiChar`
- `Attributes`

Polje AsciiChar nosi informaciju o ASCII vrednosti karaktera koji upisujemo u pravougaonik a polje Attributes nosi informaciju o boji pozadine i karaktera.

Funkcije koje omogućavaju da upisujemo u bafer konzole (bafer- memorija iz koje se čita ono što će biti ispisano na ekranu) su *WriteConsoleOutputCharacterA i WriteConsoleOutputA* . Ovo su funkcije koje rade sa ASCII vrednostima ali postoje i funkcije *WriteConsoleOutputCharacterW i WriteConsoleOutputW* koje rade sa Unicode vrednostima ali one nisu korisćene u projektu. *WriteConsoleOutputA* je je funkcija koja je najviše korišćena u svrhu ispisivanja na ekran pa cemo je detaljno i objasniti.

Funkcija *WriteConsoleOutputA* omogućava ispisivanje matrice promenljivih tipa _CHAR_INFO na ekran konzole.To znači da će funkcija uzeti sa odredjene memoriske lokacije odredjen broj promenljivih tipa _CHAR_INFO i upisati ih u bafer konzole. Upisom u bafer konzole, konzola određenim mehanizmima ispisuje ASCII karakter definisan u polju strukture pod nazivom AsciiChar, u pravougaonik koji ima pozadinu definisanu poljem Attributes u strukturi _CHAR_INFO . Deklaracije funkcije *WriteConsoleOutputA* je sledeća

```
WriteConsoleOutputA PROTO ,
                    oHandle:HANDLE,
                    ArrayForRead:PTR _CHAR_INFO,
                    SizeOfArrayForRead:COORD,
                    StartPositionForRead:COORD,
                    ScreenBufferAreaForWrite:PTR SMALL_RECT
```

- *oHandle* – pokazivac tipa HANDLE na prozor konzole
- *ArrayForRead* – pokazivač na niz tipa _CHAR_INFO u kome su skladištene informacije o izgledu karaktera koji treba ispisati, boji karaktera i boji pozadine pravougaonika u koji se taj karakter upisuje
- *SizeOfArrayForRead* – argument tipa COORD koji nosi informaciju o veličini matrice iz koje se čita. X polje označava broj vrsta a Y polje označava broj kolona.
- *StartPositionForRead* – argument tipa COORD koji označava početno mesto sa koga se čita u odnosu na trenutnu poziciju kursora.
- *ScreenBufferAreaForWrite* –argument tipa pokazivača na SMALL_RECT koji označava oblast na ekranu, posmatrana u odnosu na trenutnu poziciju kursora, u koju treba upisati karaktere sa određenim atributima. Kao

povratnu vrednost ova funkcija modifikuje ovaj argument upisujući u njega vrednosti koje označavaju oblast koja je ispunjena.Ovo može biti korisno za proveru uspešnosti odrađene funkcije.

Naravno, pre upotrebe ove funkcije treba dovesti kursor na određenu poziciju da bi smo ostvarili ispis u određenu oblast na ekranu.

Upotreba ove funkcije omogućava vrlo efikasno ispisivanje grupe karaktera na ekranu kao što su blokovi, padovi, slova, brojevi.

Igrica ima dva grafička interfejsa:

- Početni grafički interfejs
- Glavni grafički interfejs

2.1 Početni grafički interfejs

Početni grafički interfejs je napravljen je kao pokazatelj fleksibilnosti rada sa već postojećim kodom, tačnije sa delom koda koji se odnosi na formiranje slova i brojeva.Sama konzola ne nudi mogućnost da se ispisuju karakteri veći od veličine jednog pravougaonika pa je trebalo osmisliti kod koji omogućava i tu opciju a sve u cilju obogaćenja grafičkih delova igrice. Ovde je zgodno mesto da objasnimo kako smo realizovali ispisivanje proizvoljnih simbola u različitoj boji i veličini.

Slika 4. Početni grafički interfejs

Dakle, svako slovo koje se ispisuje na ekranu je niz memorijskih lokacija sastavljen od „praznih“ i „punih“ pravougaonika čijom kombinacijom se kreira šablon svakog slova ponaosob. Prazan pravougaonik je označen sa *None* i predstavlja promenljivu tipa _CHAR_INFO čiji su atributi podešeni tako da odgovaraju boji pozadine a ASCII vrednost je 0 dok je pun pravougaonik karakter koji odgovara ASCII vrednosti 219 (vrednost za pravougaonik) a čiji atributi su podešeni za default boju koja je u ovom slučaju crvena.

Svako slovo koje se ispisuje kao i svaki broj predstavljaju određenu matricu karaktera koja kada se ispiše na ekran daje određeno slovo ili broj.Veličina te matrice predstavlja veličinu slova ili broja. Svaki broj ili slovo ima svoj jedinstven *„template“* koji je neophodno ispisati da bi na ekranu videli željeno slovo. Fajlovi u kojima su smešteni šabloni slova i brojeva su sledeći:

- fontemplate3x5
- fontemplate5x7
- fontemplate9x13

```
;This part of code contain default font information like font color and font backround
Font_Block          = 219
Font_Color          = Red
Font_Backround      = 80h

Point               TEXTEQU             <<Font_Block,Font_Color>>
None                TEXTEQU             <<0,Font_Backround>>

.data
Letter_Size1  COORD <3,5>

;This is template for SPACE char
Delete_font10   _CHAR_INFO None,None,None
Delete_font11   _CHAR_INFO None,None,None
Delete_font12   _CHAR_INFO None,None,None
Delete_font13   _CHAR_INFO None,None,None
Delete_font14   _CHAR_INFO None,None,None

;This is template for Digit 1
Number_One  _CHAR_INFO None,None,Point
Number_One1  _CHAR_INFO None,none,Point
Number_One2  _CHAR_INFO none,None,Point
Number_One3  _CHAR_INFO None,None,Point
Number_One4  _CHAR_INFO None,None,Point

;This is template for Digit 2
Number_Two  _CHAR_INFO  Point,Point,Point
Number_Two1  _CHAR_INFO None, None, Point
Number_Two2  _CHAR_INFO Point,Point,Point
Number_Two3  _CHAR_INFO point,None,None
Number_Two4  _CHAR_INFO Point,Point,Point

;This is template for Digit 3
Number_Three  _CHAR_INFO  Point,Point,Point
Number_Three1  _CHAR_INFO None, None, Point
Number_Three2  _CHAR_INFO Point,Point,Point
Number_Three3  _CHAR_INFO none, none, point
Number Three4   CHAR INFO Point,Point,Point
```

Slika 5. Deo jednog template fajla

U imenu šablona je sadržana veličina slova koja se nalaze u tom šablonu. Svaki od *template* fajlova sadrži samo šablone za ona slova i brojeve koji su bili neophodni u samoj igrici , međutim ukoliko želimo da dodamo neko slovo, broj ili proizvoljan simbol neophodno je dodati šablon za to. Ako dodajemo šablon iz ASCII tabele nije potrebno definisati kod za taj šablon, međutim ako dodajemo proizvoljne simbole neophodno je da dodamo i kod za taj simbol. Primer ovoga jeste simbol pod nazivom ETF_LOGO (slika 7.) koji predstavlja simbol Elektrotehničkog fakulteta a koji naravno ne postoji u ASCII tabeli pa je zbog toga bilo potrebno dodati deklaraciju kojom ovom simbolu dajemo jedinstven kod (neophodno je da kod bude jedinstven da se ne bi poklapao sa nekim od kodova iz ASCII tabele).

Sve funkcije koje su neophodne za rad sa slovima i brojevima nalaze nalaze se deklarisane u fajlu *font.inc* čiji sadržaj je sledeći.

```
ETF_LOGO_CODE               = 2000
DELETE_CODE                 = 2001

CreateLetter PROTO,
                            oHandle             :HANDLE,
                            Array_Size          :COORD,
                            Start_Position      :COORD,
                            Letter_Array  :PTR _CHAR_INFO,
                            Letter_Color  :DWORD
WriteLetter PROTO,
                            oHandle                     :HANDLE,
                            Ascii_Code                  :WORD,
                            Letter_Start_Position       :COORD,
                            Font_Group                  :WORD,
                            Color                       :DWORD
```

```
WriteBigString PROTO,
                            oHandle             :HANDLE,
                            String              :PTR WORD,
                            L                   :DWORD,
                            Font_Group          :WORD,
                            Array_Font_Color:PTR DWORD,
                            Start_Position      :COORD,
                            Font_Step           :DWORD
.data
Delete_Array         WORD 5 DUP(DELETE_CODE)
```

Slika 6. Sadržaj fajla font.inc

WriteBigString procedura je procedura koja prima pokazivač na *String* dužine *L* koji treba ispisati u odgovarajućoj boji (*Array_Font_Color*) sa odgovarajućim razmakom između slova (*Font_Step)* počevši od pozicije (*Start_Position). Font_Group* je broj template-a koji treba koristiti(svaki template koji se doda mora imati jedinsven ID) .Ova procedura zatim za svako slovo iz *Stringa* pozove proceduru *WriteLetter*. *WriteLetter* se poziva sve dok se ne dođe do kraja stringa. Ova procedura takođe vodi računa o tome da se svako slovo ispiše na odgovarajućoj poziciji. Inače pozicija slova je relativna i meri se u odnosu na početak niza.

WriteLetter funkcija na osnovu ID broja template-a (*Font_Group*) zna koji template treba uzeti. Na osnovu *Ascii_Code* odredjuje šablon simbola iz odgovarajućeg template-a. Nakon toga ova procedura poziva proceduru *CreateLetter*

CreateLetter procedura je zadužena za ispis slova na ekran na odgovarajućoj poziciji.Ova procedura ispisuje slovo odgovarajuće veličine na odgovarajućoj poziciji u odgovarajućoj boji a sve to radi pozivajući proceduru *WriteConsoleOutputA* čije je ponašanje već opisano.

Dakle, uz pomoć ove tri funkcije moguće je ispisati bilo šta bilo gde na ekranu u bilo kojoj veličini.Potrebno je samo napraviti odgovarajuće šablone i modifikovati proceduru *WriteLetter* tako da zna da u slučaju poziva nekog simbola taj simbol postoji. Potrebno je naglasiti da u slučaju ispisa brojeva, pre nego što se pozove procedura *WriteBigString* treba broj konvertovati u niz ASCII vrednosti .

```
Font_Block          = 219
Font_Color          = Red
Font_Backround      = 80h

Point         TEXTEQU <<Font_Block,Font_Color>>
None          TEXTEQU <<0,Font_Backround>>
.data
Letter_Size3 COORD <9,13>

ETF_Logo      _CHAR_INFO Point,Point,Point,Point,Point,Point,Point,Point,Point
ETF_logo1     _CHAR_INFO Point, None, None, None, None, None, None, None,Point
ETF_logo2     _CHAR_INFO Point, None,Point,Point,Point,Point,Point, None,Point
ETF_logo3     _CHAR_INFO Point, None, None, None,Point, None, None, None,Point
ETF_logo4     _CHAR_INFO Point, None, None,Point,Point,Point, None, None,Point
ETF_logo5     _CHAR_INFO Point, None,Point, None,Point, None,Point, None,Point
ETF_logo6     _CHAR_INFO Point,Point,Point,Point,Point,Point,Point,Point,Point
ETF_logo7     _CHAR_INFO Point, None,Point, None,Point, None,Point, None,Point
ETF_logo8     _CHAR_INFO Point, None, None,Point,Point,Point, None, None,Point
ETF_logo9     _CHAR_INFO Point, None, None, None,Point, None, None, None,Point
ETF_logo10    _CHAR_INFO Point, None,Point,Point,Point,Point,Point, None,Point
ETF_logo11    _CHAR_INFO Point, None, None, None, None, None, None, None,Point
ETF_logo12    _CHAR_INFO Point,Point,Point,Point,Point,Point,Point,Point,Point
```

Slika 7. Izgled šablona proizvoljnog simbola

2.2 Glavni grafički interfejs

Glavni grafički interfejs se sastoji od dva dela. Jedan deo se nalazi iznad plave linije čija je pozadina siva. Na tom delu se nalaze informacije o broju trenutnih poena (levo) i broju preostalih života (desno). Ovi brojevi su ispisani metodama koje su pomenute u prethodnom delu teksta. Ispisivanje poena se vrši pozivom procedure *WritePoints*.

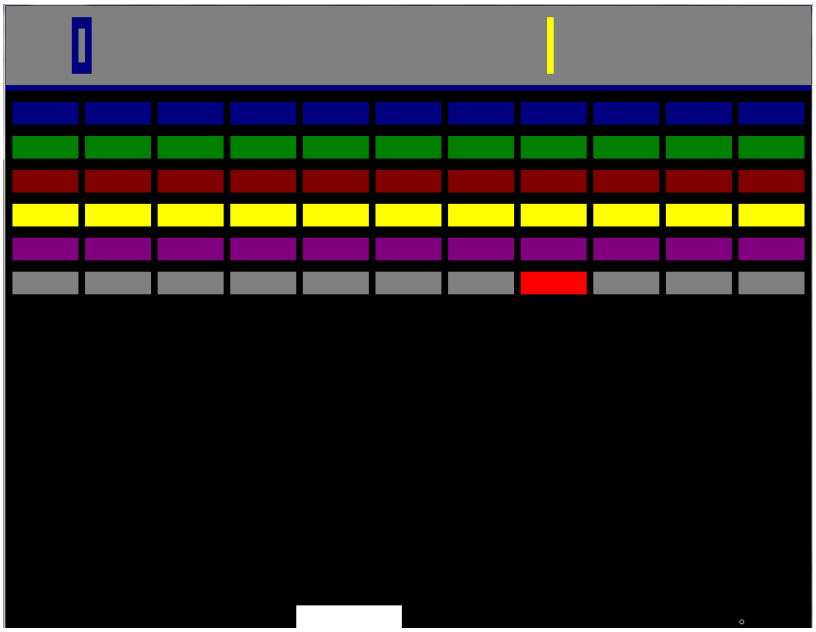
```
DecimalToAscIIm MACRO
                push ebx
                .WHILE eax!=0
                        mov edx,0
                        mov ebx,10
                        div ebx
                        add edx,48
                        mov WORD
PTR[ecx],dx
                        add ecx,2
                        INC Counter
                .ENDW
                pop ebx
ENDM
```

Slika 8. Makro koji vrši konverziju iz decimalnog zapisa u heksadecimalni niz

WritePoints procedura ispisuje broj poena u deo glavnog grafičkog interfejsa iznad plave linije. Funkciji se prosleđuje broj poena kao tip podatka DWORD. Nakon toga se na početku procedure konvertuje ta celobrojna vrednost u niz ASCII karaktera. To konvertovanje se vrši delom koda koji je izdvojen u MACRO i označen sa *DecimalToAsciim*. Nakon što se izvrši konverzija u niz ASCII karaktera, dobijeni niz je u obrnutom poretku pa ga treba rotirati tako da prvi element niza postane poslednji a poslednji element niza postane prvi itd. Nakon rotacije ova procedura poziva proceduru *WriteBigString* prosleđujući joj pokazivač na novi niz ASCII karaktera. Ispisivanje života se radi istim principom.

Ispod plave linije se nalazi drugi deo glavnog grafičkog interfejsa koji se sastoji od blokova, pada i lopte.

Blok je promenljiva strukture CELL koja se sastoji od matrice koja ima 3 vrste gornju, centralnu i donju. Gornju vrstu čini niz karaktera čija je ASCII vrednost 220, ispod nje se nalazi centralna vrsta čija je ASCII vrednost karaktera 219 a ispod nje se nalazi donja vrsta koja je sastavljena od ASCII karaktera vrednosti 223. Ove tri vrste su spakovane u jednu matricu čiji



Slika 9. Glavni grafički interfejs

broj kolona može biti proizvoljan. U našem slučaju broj kolona u jednoj matrici bloka jeste 10. Broj vrsta i kolona u jednom bloku se mogu menjati. Iscrtavanje jednog bloka na konzoli se vrši pozivom funkcije *WriteBlock.*

WriteBlock procedura na osnovu prosleđene informacije o bloku u promenljivoj Cell_Info ispisuje blok na ekranu tako što prvo podesi boju i ASCII vrednosti svake vrste a zatim pozove funkciju *WriteConsoleOutputA* koja ispiše blok na odgovarajućoj poziciji. Procedura koja vodi računa o tome da svaki blok bude ispisan na odgovarajućem mestu jeste procedura *WriteRow.*

WriteRow procedura je zadužena za ispis jedne vrste matrice blokova. Ova procedura kao argumente prima pokazivač na praznu vrstu matrice blokova *Cell_Array* kao i veličinu vrste izraženu u blokovima.Takođe prima argumente koji označavaju početnu poziciju reda (*Row_Start_Position*) , boju vrste (*Row_Color*), koliko poena nosi vrsta (*Row_point*) , informaciju o tome koliko puta je neophodno da bude udaren blok u vrsti da bi on nestao (*Hit_Counter*) kao i informaciju o tome da li je red padajući ili ne (*Fall_Flag*), Padajući red je onaj red čiji blokovi padaju nakon udara i ukoliko ih pad pokupi broj poena koji nosi taj blok se udvostručava.

Nakon poziva fukcije *WriteRow* ona vrši inicijalizaciju svakog bloka unutar vrste, zatim poziva funkciju *WriteBlock* koja ispisuje blok na određenoj poziciju. Pozicija je prethodno proračunata i njena vrednost je relativna u odnosu na početak vrste. *WriteRow* procedura se poziva svaki put kada treba ispisati red a poziva je procedura *MakeStartBlocksMatrix.*

MakeStartBlocksMatrix procedura je zadužena za ispravnu inicijalizaciju cele matrice blokova.Ona kao argumente prima praznu matricu blokova koju treba da ispuni, prima nizove koji sadrže informacije o boji svake vrste, poenima za svaku vrsu, da li je vrsta padajuća ili ne, kao i niz sa informacijama o tome koliko puta treba da bude udaren neki blok da bi nestao sa ekrana. Ovu proceduru poziva *main* procedura.

Dakle, redosled poziva procedura je sledeći:

1. Main procedura poziva *MakeStartBlocksMatrix*
2. *MakeStartBlocksMatrix* poziva *WriteRow* proceduru
3. *WriteRow* procedura poziva *WriteBlock*

Pored procedure *WriteBlock* postoji i procedura *DeleteBlock* koja briše blok sa određene pozicije upisivajući na tim pozicijama ASCII vrednost 0. Ova procedura ima naročit značaj u slučaju pomeranja bloka kada je blok neophodno prvo izbrisati, zatim pomeriti pa ponovo iscrtati .

Pad je promenljiva structure CELL koja je grafički slična bloku ali ima dve vrste ASCII karaktera vrednosti 219.Takodje postoje procedure za ispis (*WritePad*) i brisanje pada (*DeletePad*) koji imaju ulogu u kretanju pada. Kontrolu nad ovim procedurama ima procedura *MovPad* koja kontroliše kretanje pada na onsovu toga da li je pritisnuta leva ili desna strelica na tastaturi.

Lopta je promenljiva tipa BALL. Ispisivanje kružića koji predstavlja loptu se vrši pozivom procedure *WriteBallCell* a brisanje kružica pozivom procedure *DeleteBallCell*. Ove dve procedure u pozadini pozivaju procedure *WriteConsoleOutputCharacterW* i *WriteConsoleOutputAttribute* koje ustvari služe za ispis odnosno podešavanje atributa karaktera.

Veći deo elemenata u delu glavnog grafičkog interfejsa ispod plave linije ima mogućnost da se pomera. Načini, algoritmi i ideje koje su korišćene u realizaciji pomeranja objekata biće objašnjene u narednom poglavlju.

3.Algoritmi za pomeranje elemenata u igrici

3.1 Brzina kretanja elemenata

Da bi mogli da ispunimo zahtev projekta koji kaže da ukoliko igrač uhvati blok koji je dva puta udaren da dobija duplo više poena, bilo je neophodno omogućiti elementima koji se kreću da imaju mogućnost da se kreću proizvoljnom brzinom. Bzinu kretanja objekata u igrici smo uveli na sledeći način:

Pošto se procedure koje se odnose na dinamiku u igrici (kretanje, promena broja poena, promena broja života, …) pozivaju svaki put u main proceduri, ograničavanjem poziva procedura po nekom modulu daje efekat brzine kretanja objekata. Naime, main procedura se poziva svaki put. Kada program naiđe na proceduru pokrene je i u proceduri se proverava koliko puta je do sada pozvana ta procedura. Ukoliko je pozvana određeni broj put pristupa se njenom izvršavanju u suprotnom se preskače. Informacija o broju poziva neke procedure čuva se u vremenskim promenljivim koji su deklarisani pri dnu fajla *structure.inc*

```
Gray_Row_Time_Info          DWORD         Number_of_cell_in_Row       DUP(0)
Pad_Time_Info               DWORD        0
Ball_Time_Info              DWORD        0
```

Slika 10. Deklaracija vremenskih promenljivih u strcture.inc fajlu

Početna vrednost ovih promenljivih je nula. Pri pozivu neke procedure, koja je zadužena za kretanje bloka, pada ili lopte, se u tu proceduru iz main funkcije prosleđuje kao argument neka od gore navedenih vremenksih promenljivih . Vrednost ovih promenljivih se unutar procedure inkrementira i proverava da li je zadovoljena brzina koja je zadata od strane korisnika u fajlu *configuration.inc*. Ukoliko nije zadovoljena procedura skače na svoj kraj i izlazi se iz procedure . Ukoliko je zadovoljena, prvo se resetuje vremenski brojač a zatim ide algoritam zadužen za kretanje elemenata.

3.2 Kretanje elemenata

3.2.1 Kretanje Pad-a

U main proceduri se poziva procedura *MovPadCell* kojoj se kao argumenti prosleđuju pokazivač na *Pad* i pokazivač na vremensku promenljivu *Pad_Time_Info.* Pri ulasku u proceduru se prvo vrši provera da li je zadovoljen broj poziva procedure koji simulira brzinu kretanja pada. Ako jeste vrši se poziv procedure *MovPad* a zatim reset vremenske promenljive na 0

Nakon ulaska u *MovPad* proceduru prvo se iscrta pad na ekranu a zatim se provarava da li je pritisnuta neka od strelica na tastaturi.Ukoliko nije izlazi se iz procedure a ukoliko jeste skače se na odgovarajući deo koda u proceduri koji odgovara pritisnutoj strelici.

Nakon odlaska u deo koda koji odgovara pritisnutoj strelici vrši se brisanje pada procedurom *DeletePad,* zatim zadavanje nove pozicije pada . Onda sledi provera da li je pad stigao na ivice konzole, ako jeste padu se vraćaju stare koordinate a ukoliko pad nije stigao do ivica konzole vrši se crtanje pada sa novim koordinatama. Na ovaj način je izvršeno kretanje pada levo ili desno.

3.2.2 Kretanje Lopte

U main proceduri se poziva procedura *MovBall* kojoj se kao argumenti prosleđuju pokazivači na matricu blokova *Blocks_Cell_Array,* vremensku promenljivu *Ball_Time_Info* i pokazivač na promenljivu koja nosi informaciju o trenutnom broju života *Game_Lifes.* Na početku ove procedure se prvo proverava da li je zadovoljen broj poziva procedure kojim se simulira brzina kretanja lopte. Ukoliko je brzina zadovoljena nastavlja se sa izvršavanjem procedure a ukoliko nije izlazi se iz procedure.

Ukoliko je brzina zadovoljena prvo se briše lopta sa stare pozicije pozivom *DeleteBallCell* procedure a zatim se poziva procedura *NextBallPosition.*

NextBallPosition procedura određuje koordinate sledeće pozicije na kojoj lopta treba da se nađe. Proračun sledeće koordinate se vrši na osnovu prethodne i trenutne koordinate. U zavisnosti od sledeće koordinate skače se na odgovarajući deo koda u proceduri. Nakon skoka u odgovarajući deo vrši se poziv procedure *FindCollision* koja vrši proveru da li se sledeće koordinate lopte sudaraju sa nekim blokom ili padom. Funkcija *FindCollision* vraća 0 u eax registar ukoliko sudara nije bilo ili odgovarajuću vrednost u eax ukoliko je sudara bilo. Vrednost u eax registru koja je različita od 0 daje informaciju o tome sa koje strane se desio sudar. Ukoliko je bilo sudara ova funkcija vraća adresu memorijske lokacije bloka ili pada sa kojim se sudarila lopta.

Nakon što izađemo iz procedure *FindCollision* proverava se vrednost *eax* registra. Ukoliko je ta vrednost različita od 0 vrši se umanjenje *HitCounter* promenljive u bloku sa kojim se loptica sudarila. Ukoliko je nova vrednost jednaka 1 to znači da je u pitanju blok koji treba da bude udaren veći broj puta da bi nestao sa ekrana. Ovakvi blokovi dobijaju novu boju definisanu vrednošću *HitColor* promenljive koju takodje korisnik može sam da odabere. Ova mogućnost je dodata da bi se blokovi koje je neophodno udariti veći broj puta razlikovali od ostalih blokova.

Nakon što smo umanjili vrednost *HitCounter* promenljive unutar bloka , vrši se deo koda koji je izdvojen u MACRO a koji na osnovu pravca iz kog je došla loptica I na osnovu strane

sa koje se desio udar procenjuje sledeću vrednost koordinate lopte. Za ovaj proračun zadužena je funkcija *ReBoundBall* koja pravi nove koordinate lopte na osnovu zadatog pravca koji dobija kao argument.

Ukoliko nije bilo sudara sa nekim od elemenata unutar igrice moramo proveriti da li je loptica došla do ivica konzole. Za tu proveru pozivamo procedutu *IsAtScreenEdge* iza koje se krije ista ideja kao i iza funkcije *FindCollision.*

Ako se loptica nije sudarila sa blokom ili padom ili došla do ivica konzole postoji mogućnost da je pala van Pad-a i da smo izgibili život. Ukoliko se to desilo potrebno je oduzeti jedan život, staviti lopticu u neaktivno stanje stavljajuci vrednost *Active* promenljive unutar lopte na 0. Nakon ovoga poziva se procedura *LifeCheck* koja ima za zadatak da ispiše novi broj života na ekranu.

Na kraju ukoliko se nije desio nijedan sudar ni sa padom , ni sa blokom niti je loptica pala van ivica konzole potrebno je pozvati funkciju *SetNextBallPosition* koja stavlja nove koordinate lopte na osnovu prethodne i trenutne koordinate.

Nakon izlaska iz procedure *NextBallPosition* potrebno je sada iscrtati loptu sa novim koordinatama za šta je zadužena procedura *WriteBallCell*

3.2.3 Kretanje Bloka

U *main* proceduri se poziva *UpdateScreenMatrix* procedura. Unutar te procedure se prolazi ponaosob kroz matricu blokova i za svaki od tih blokova se proverava da li je blok aktivan proveravajuci flag *Activate* unutar svakog bloka. Ukoliko je blok neaktivan prelazi se na sledeći blok u matrici. Ukoliko je blok aktivan ispituje se da li je dostignut dovoljan broj udara loptice u blok. Ako nije, blok se ponovo ispisuje, tj. vrši se osvežavanje tog bloka u konzoli. Ukoliko je dostignut traženi broj udara vrši se provera da li je blok padajući ili nije.

Ukoliko je blok padajući poziva se procedura *MovBlock.* Unutar procedure *MovBlock* se vrši provera da li je dostignuta odgovarajuća brzina padanja, tj. da li je funkcija za pomeranje bloka pozvana dovoljan broj puta. Ako je zadovoljena brzina unutar ove funkcije se poziva funkcija *GoDown* koja ima zadatak da na konzoli grafički spusti blok za jedno mesto niže i da ukoliko je blok stigao na kraj konzole postavi odgovarajući flag (*DownFlag).*

Nakon što se blok spusti za jedno mesto niže proverava se da li je je došlo do sudara između Pad-a i bloka. Ukoliko je došlo do sudara prvo se setuje flag *BlockPadCollisionFlag.* Zatim se ukoliko je došlo do sudara vrednosti unutar Pad-a inicijalizuju na odgovarajuće vrednosti i to tako što se *Fall_Flag* unutar Pad-a setuje na vrednost 1 što daje informaciju da je došlo do sudara. Zatim se vrednost *Value* koja pripada Pad-u setuje na vrednost onog bloka sa kojim se pad sudario a zatim se *Fall_Flag* unutar bloka setuje na 0 čime padajući blok sada postaje kao i svaki drugi blok.

Nakon izlaska iz procedure *MovBlock* proveravamao *Fall_Flag* unutar Pad-a i ukoliko je na jedinici to je znak da je došlo do sudara sa nekim od blokova koji je imao vrednost koju smo upisali u polje Value samog Pad-a. Sada je potrebno tu vrednost dodati na trenutni broj poena i pozvati proceduru *WritePoints* zaduženu za ispis poena.

Ukoliko blok nije padajući a udaren je lopticom već dovoljan broj puta, potrebno je staviti vrednost *Activate* na 0 unutar odgovarajućeg bloka, na trenutni broj poena dodati broj poena bloka i izbrisati blok sa konzole.

4. Main procedura

```
main PROC
        ;This part of code move pointer to console to variable oHandle
        INVOKE GetSTDHandle , STD_OUTPUT_HANDLE
        mov OHandle,eax

        INVOKE WriteWelcomeScreen,oHandle,1

L2:     INVOKE CLRSCR

        INVOKE SetStartWindow,                  oHandle,GameBufferSize,                 ADDR
GameWindowSize, ADDR AppTitle
        INVOKE WriteStartLine,                  oHandle,ADDR StartLine
        INVOKE WriteConsoleOutputA,             ohandle,ADDR GameInfoBackround,  GameInfoSize,
         GameInfoStartPosition,         ADDR GameInfoA

        INVOKE MakeStartBlocksMatrix,    oHandle,ADDR Blocks_Cell_Array, Blocks_Matrix_Size,
ADDR Blocks_Collor_Array,        ADDR BLocks_Points_Array,        ADDR Blocks_Hit_Array,   ADDR
Blocks_Fall_Array
        INVOKE SetConsoleCursorInfo,     oHandle,ADDR Cursor_Info
        INVOKE SetStartPosition,         ADDR Pad,   ADDR Ball
        INVOKE WritePoints,                             ohandle,Game_Points
        INVOKE LifeCheck,                               oHandle,Game_Lifes , ADDR Ball

        mov edx, OFFSET Ball


L1: INVOKE MovPadCell,                          oHandle,ADDR Pad,                               ADDR
Pad_Time_Info
        INVOKE MovBall,                                 oHandle,ADDR Ball,
        ADDR Blocks_Cell_Array,  ADDR Ball_Time_Info, ADDR Game_Lifes
        INVOKE UpdateScreenMatrix,              oHandle,ADDR Blocks_Cell_Array,  ADDR Pad,
                        ADDR Game_Points

        mov edx, Offset ball
        movzx eax, (BALL_ PTR[edx]).Active
        .IF eax == 0
                mov edx,OFFSET Game_Points
                mov eax,0
                mov [edx],eax
                mov edx,OFFSET Game_Lifes
                mov eax,DWORD PTR [edx]
                .IF eax>0
                        mov eax,3000
                        INVOKE delay
                .ELSE
                        mov eax,1000
                        INVOKE delay
                .ENDIF
                mov eax,1
                mov edx, Offset ball
                mov (BALL_ PTR[edx]).Active,al
                JMP L2
        .ENDIF

SKIP:
         push ecx
         mov eax,KEY_TIME_PRESS
         INVOKE Delay
         add DelayBlock,Pad_Speed
         INVOKE GetKeyState,VK_ESCAPE
         AND eax,KEY_DOWN
         JNZ THEEND

         mov eax,Game_Lifes
         .IF eax == 0
                JMP THEEND
         .ENDIF
         pop ecx
         JMP L1
```

```
THEEND:
    INVOKE clrscr
        INVOKE WriteStartLine,                  oHandle,ADDR StartLine
        INVOKE WriteConsoleOutputA,             ohandle,ADDR GameInfoBackround,  GameInfoSize,
                GameInfoStartPosition,          ADDR GameInfoA
        INVOKE MakeStartBlocksMatrix,    Ohandle,ADDR Blocks_Cell_Array, Blocks_Matrix_Size,
        ADDR Blocks_Collor_Array,        ADDR BLocks_Points_Array,        ADDR Blocks_Hit_Array,   ADDR
Blocks_Fall_Array
        INVOKE WriteBigString,                  oHandle,ADDR GameOverString,     GameOverStringLengt,
        2,                                                  ADDR GameOverStringColor,
        GameOverStringPosition,  2

        mov eax, 3000
        INVOKE delay
        exit
main ENDP
END main
```

Slika 11. Kod u main proceduri

Procedura *WriteWelcomeScreen* ispisuje početni ekran konzole koji sadrži informacije o autorima i predmetu iz koga je projekat rađen

Nakon ove procedure pozivamo proceduru CLRSCR koja briše sve iz konzole. Nakon nje pozivamo proceduru *SetStartWindow* koja podešava veličinu bafera, prozora i naziv konzole. Posle nje sledi procedura zadužena da iscrta startu liniju ispod koje će biti igrački deo a iznad koje će biti informacije o trenutnom stanju igrice kao što su broj poena i života. Nakon ove procedure sledi procedura koja boji pozadinu iznad linije u sivo odvajajući je tako od ostatka igrice.

Procedura *MakeStartBlocksMatrix* kreira matricu blokova i inicijalizuje polja odgovarajućim vrednostima. Ova procedura takodje iscrtava red po red blokova na konzoli.

Procedura *SetConsolCursorInfo* pravi kursor nevidljivim i time onemogućava korišćenje kursora u igrici.

SetStartPositions procedura određuje početne koordinate Pad-a i lopte.

WritePoints i LifeCheck procedure ispisuju početni broj života i poena respektivno.

MovPadCell procedura je zadužena za kontrolu kretanja pada levo i desno, dok je procedura *MovBall* zadužena za ispravno kretanje i odbijanje loptice.

UpdateScreenMatrix procedura je zadužena za osvežavanje matrice blokova tako što svaki put ispisuje blok.

Slika 10. Izgled kraja igrice

Deo koda koji sledi do labele *SKIP* je zadužen za proveru preostalog broja života u igrici. Prvo se proverava flag *Activ* koji pripada Lopti. U slučaju da je taj flag na nuli to je znak da je došlo do gubutka jednog od života i u tom slučaju je potrebno staviti broj poena na nulu, pauzirati igricu oko 3 sekunde da bi igrač mogao da se pripremi za novu igru, i ponovo pokrenuti igricu skokom na labelu L2.

Slika 12. Izgled kraja igrice

Deo koda koji sledi posle *SKIP* labele pa sve do labele *THEEND* je deo koda koji proverava da li su ispunjeni uslovi za kraj igrice, tj. da li je pritisnut taster ESC ili je broj života na

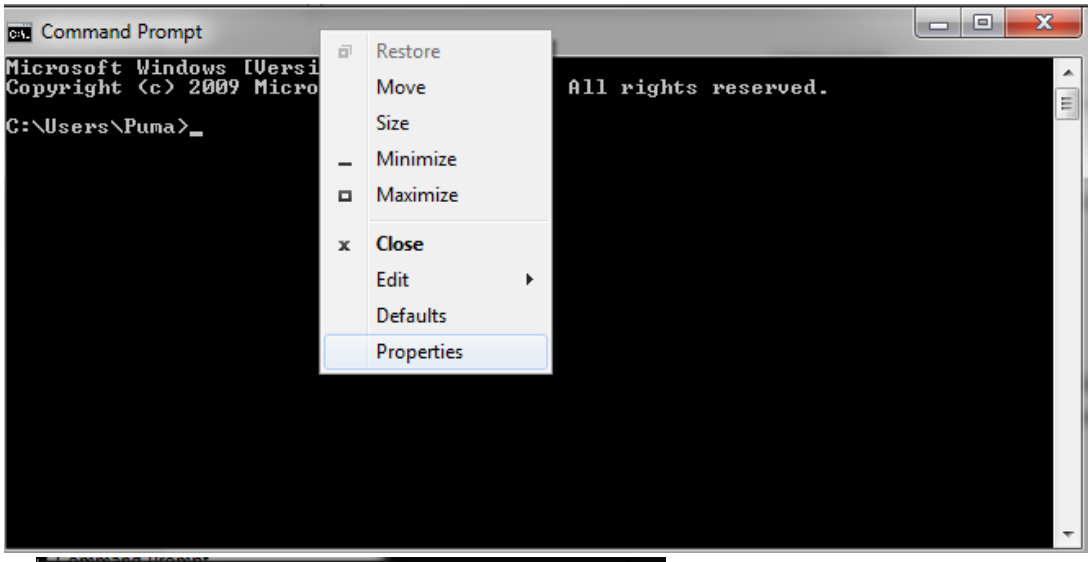
nuli. Ako je ispunjen neki od ovih uslova skače se na labelu *THEEND* a u suprotnom se skače na labelu *L1*.

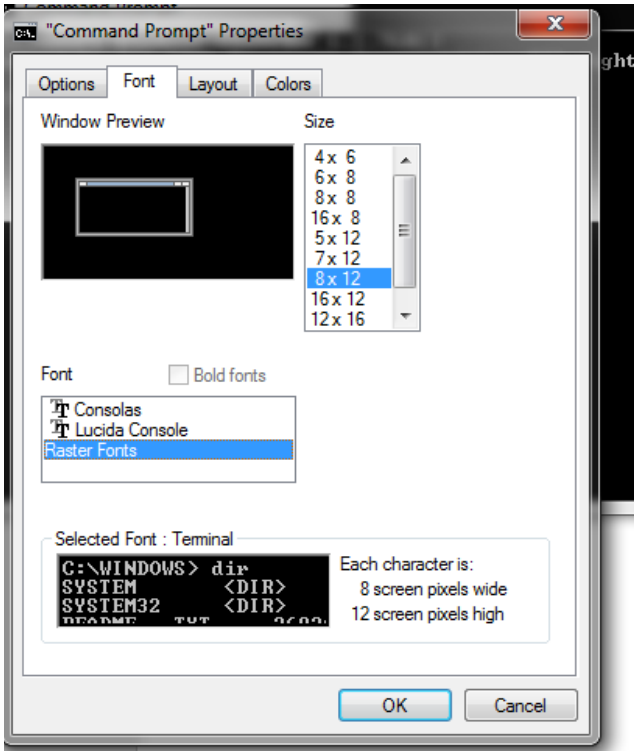
Kod posle labele THEEND je kod koji briše sve sa ekrana, ponovo iscrtava startnu liniju, pozadinu boji u sivo iznad startne linije i iscrtava matricu blokova, ispisuje "GAME OVER" i pauzira igricu na 3 sekunde.

5. Pokretanje igrice

Zbog toga što je na različitim računarima različito podešena konzola a igrica je pravljena prema jednoj rezoluciji konzole potrebno je izvršiti sledeća podešava u vašoj konzoli pre nego što pokrenete igricu.

Korak 1





Korak 2

Naravno , ova podešavanja će na nekim računarima već biti odrađena ali da bi se dobila puna efikasnost igrice najbolje je prvo proveriti ova podešavanja pa onda pokrenuti igricu.

6. Kodovi

- Sadržaj fajla *function.inc*

```
DecimalToAscIIm MACRO
                push ebx
                .WHILE eax!=0
                        mov edx,0
                        mov ebx,10
                        div ebx
                        add edx,48
                        mov WORD PTR[ecx],dx
                        add ecx,2
                        INC Counter
                .ENDW
                pop ebx
ENDM
ReverseArraym MACRO
                mov ecx,Counter
                LEA edi,AscII_Array
                mov esi,edi
                .WHILE ecx>1
                        add esi,TYPE WORD
                        dec ecx
                .ENDW
                mov ecx,Counter
                .WHILE ecx>0
                        movzx eax,WORD PTR[edi]
                        movzx edx,WORD PTR[esi]
                        mov WORD PTR[edi],dx
                        mov WORD PTR[esi],ax
                        sub esi,TYPE WORD
                        add edi,TYPE WORD
                        .IF edi>=esi
                                .BREAK
                        .ENDIF
                .ENDW
ENDM
mUP_Left MACRO
        .IF eax == Left_Collision
            INVOKE ReBoundBall, Ball_Cell,Up_Right
        .ELSEIF eax == TopBottom_Collision
            INVOKE ReBoundBall, Ball_Cell,Down_Left
        .ENDIF
ENDM
mUP_Right MACRO
        .IF eax == Right_Collision
            INVOKE ReBoundBall, Ball_Cell,Up_Left
        .ELSEIF eax == TopBottom_Collision
            INVOKE ReBoundBall, Ball_Cell,Down_Right
        .ENDIF
ENDM
mDown_Left MACRO
        .IF eax == Left_Collision
            INVOKE ReBoundBall, Ball_Cell,Down_Right
        .ELSEIF eax == TopBottom_Collision
            INVOKE ReBoundBall, Ball_Cell,UP_Left
        .ENDIF
ENDM
mDown_Right MACRO
        .IF eax == Right_Collision
            INVOKE ReBoundBall, Ball_Cell,Down_Left
        .ELSEIF eax == TopBottom_Collision
            INVOKE ReBoundBall, Ball_Cell,UP_Right
        .ENDIF
ENDM

SetColorAndASCII PROTO,
                        InArr:PTR _CHAR_INFO,
                        Color:DWORD,
                        ASCII:DWORD,
                        Siz:DWORD
```

```
SetStartWindow PROTO,
                        oHandle:HANDLE,
                        BuffSize:COORD,
                        WinSize:PTR SMALL_RECT,
                        GameName:PTR BYTE

WriteBlock PROTO ,
                        oHandle:HANDLE,
                        Cell_Info:PTR CELL
DeleteBlock PROTO ,
                        oHandle:HANDLE,
                        Cell_Info:PTR CELL
WritePad PROTO ,
                        oHandle:HANDLE,
                        Cell_Info:PTR CELL
DeletePad PROTO ,
                        oHandle:HANDLE,
                        Cell_Info:PTR CELL

WriteStartLine PROTO,
                        oHandle:HANDLE,
                        Line:PTR _CHAR_INFO

WriteRow PROTO,
                        oHandle:HANDLE,
                        Cell_Array:PTR CELL,
                        Cell_Array_Size:DWORD,
                        Row_Start_Position:WORD,
                        Row_Color:DWORD,
                        Row_Point:DWORD,
                        Hit_Count:WORD,
                        Fall_Ready:DWORD

MakeStartBlocksMatrix PROTO,
                        oHandle:HANDLE,
                        Cell_Matrix:PTR CELL,
                        Cell_Matrix_Size:COORD,
                        Collor_Array:PTR DWORD,
                        Points_Array:PTR DWORD,
                        Hit_Array:PTR WORD,
                        Fall_Array:PTR DWORD

MovPad PROTO,
                        oHandle:HANDLE,
                        Pad_:PTR CELL

GoDown PROTO,
                        oHandle:HANDLE,
                        Block:PTR CELL,
                        BottomFlag:PTR DWORD
MovBlock PROTO,
                        oHandle                         :HANDLE,
                        Block                           :PTR CELL,
                        Pad                                     :PTR CELL


UpdateScreenMatrix PROTO,
                        oHandle                 :HANDLE,
                        Matrix                  :PTR CELL,
                        Pad                             :PTR CELL,
                        Points                  :PTR DWORD

MovPadCell PROTO,
                        oHandle:HANDLE,
                        PadCell:PTR CELL,
                        Row_Time_Info:PTR DWORD

FindCollision PROTO,
                        Array:PTR CELL,
                        Cordinate:COORD,
                        Out_BLOCK:PTR DWORD

ReBoundBall PROTO, ;Funkcija koja generise naredne koordinate Ball strukture u slucaju odbijanja od objekta
                        Ball_Cell:PTR BALL_,
```

```
                        Direction:DWORD

NextBallPosition PROTO,
                        oHandle:HANDLE,         ;This Procedure calculate next ball position by Previos and current position
of BALL.Also this procedure check if Ball is at the window edge or fall from window.If there is Collision with the other block
                        Ball_Cell:PTR BALL_,
                        Blocks_Matrix: PTR CELL,
                        Lifes   :PTR DWORD

SetNextBallPosition PROTO, ;This Procedute set next ball position define by NextPosition
                        Ball_Cell:PTR BALL_,
                        NextPosition:COORD


WriteBallCell PROTO,
                        oHandle  :      HANDLE,
                        Ball_Cell:      PTR BALL_
DeleteBallCell PROTO,
                        oHandle  :      HANDLE,
                        Ball_Cell:      PTR BALL_
IsAtScreenEdge PROTO,                           ;this function return 0 in EAX if Ball is in window,1 if BALL is window edge
and 2 if ball fall
                        Position:COORD

MovBall PROTO,
                oHandle:HANDLE,
                BALL_CELL:PTR BALL_,
                Blocks_Matrix:PTR CELL,
                Time_Info:PTR DWORD,
                Lifes:PTR DWORD
WritePoints PROTO,
                ohandle                 :HANDLE,
                Points                  :DWORD
SetStartPosition PROTO,
                Pad_Cell:PTR CELL,
                Ball_Cell:PTR BALL_
LifeCheck PROTO,
                ohandle                 :HANDLE,
                Lifes                   :DWORD,
                Ball_Cell               :PTR BALL_
```

- Sadržaj fajla *function.asm*

```
INCLUDE AdditionalDef.inc
INCLUDE Configuration.inc
INCLUDE structure.inc
INCLUDE Function.inc
INCLUDE Font.inc


.code

;-----------------------------------------------------------------------------------------------------------
;--------------------------------------Start condition procedure--------------------------------------------
;-----------------------------------------------------------------------------------------------------------
;In this section there are Procedure which cinfugure start condition of game such as windows size, start block
;position, ....
;This procedure set Window size,Window Title and Window Buffer size.Procedure call kernel32 procedure such as
;SetConsoleSreenBufferSize, SetConsoleWindowInfo, SetConsoleTitle . For understanding this Procedure visit MSDM WebSIte
SetStartWindow PROC ,
                        oHandle:HANDLE,
                        BuffSize:COORD,
                        WinSize:PTR SMALL_RECT,
                        GameName:PTR BYTE
                        pushad
                        INVOKE SetConsoleScreenBufferSize, oHandle, BuffSize
                        INVOKE SetConsoleWindowInfo, oHandle, TRUE, WinSize
                        INVOKE SetConsoleTitle, GameName
                        popad
                        ret
SetStartWindow ENDP

;This function make Start Matrix and Initializes BLOKCS CELL
```

```
MakeStartBlocksMatrix PROC,
                        oHandle                         :HANDLE,        ;Handle to window
                        Cell_Matrix                     :PTR CELL,              ;Pointer to Uninitialize Matrix
                        Cell_Matrix_Size:COORD,                 ;Size of matrix. X is number of rows and Y is number of
columns
                        Collor_Array            :PTR DWORD,             ;This Array contains color for each row
                        Points_Array            :PTR DWORD,             ;This Array contains points for each row
                        Hit_Array                       :PTR WORD,
                        Fall_Array                      :PTR DWORD

                        LOCAL RowStartPosition  :DWORD,
                                        CollorArrayPTR  :DWORD,
                                        PointsArrayPTR  :DWORD,
                                        HitArrayPTR             :DWORD,
                                        FallArrayPTR    :DWORD,
                                        ArraySize               :DWORD,
                                        RowColor        :DWORD,
                                        RowValue        :DWORD,
                                        Temp                    :DWORD,
                                        HitCounter              :WORD,
                                        FallFlag        :DWORD
                        pushad

                        movzx ecx, Cell_Matrix_Size.Y ;Initialize Counter

                        movzx edx, Cell_Matrix_Size.X
                        mov ArraySize, edx

                        mov edi, Cell_Matrix
                        mov esi,0
                        mov RowStartPosition, 0

                        mov eax,Collor_Array
                        mov CollorArrayPTR,eax

                        mov eax,Points_Array
                        mov PointsArrayPTR,eax

                        mov eax,Hit_Array
                        mov HitArrayPTR,eax

                        mov eax,Fall_Array
                        mov FallArrayPTR,eax

                L1:     mov eax,esi
                        mov edx,Block_Height
                    mul edx

                        ;Calculating rows start position
                        mov RowStartPosition,eax
                        add RowStartPosition,1
                        add RowStartPosition,StartLinePosition

                        mov edx,CollorArrayPTR
                        mov eax,[edx]
                        mov RowColor,eax

                        mov edx,PointsArrayPTR
                        mov eax,[edx]
                        mov RowValue,eax

                        mov edx,HitArrayPTR
                        movzx eax,WORD PTR [edx]
                        mov HitCounter,ax

                        mov edx,FallArrayPTR
                        mov eax,[edx]
                        mov FallFlag,eax

                        JMP L3
        L2:     JMP L1                                                          ;I need to cut this Because MASM
error told that this loop is too long
        L3:
                        INVOKE WriteRow, oHandle, edi, Arraysize, WORD PTR RowStartPosition, RowColor,
RowValue,HitCounter,FallFlag
```

```
                        mov eax,CollorArrayPTR
                        add eax,4
                        mov CollorArrayPTR,eax

                        mov eax,PointsArrayPTR
                        add eax,4
                        mov PointsArrayPTR,eax

                        mov eax,HitArrayPTR
                        add eax,2
                        mov HitArrayPTR,eax

                        mov eax,FaLLArrayPTR
                        add eax,4
                        mov FallArrayPTR,eax

                        inc esi

                        mov eax,TYPE CELL
                        mov edx,ArraySize
                        mul edx
                        add edi,eax

                        LOOP L2

                        popad
                        ret
MakeStartBlocksMatrix ENDP

;This procedure Initializes all blocks in one row
;This procedure is call by:
;               - MakeStartCellMatrix procedure
WriteRow PROC,
                        oHandle                         :HANDLE,
                        Cell_Array                      :PTR CELL,
                        Cell_Array_Size         :DWORD,
                        Row_Start_Position      :WORD,
                        Row_Color                       :DWORD,
                        Row_Point                       :DWORD,
                        Hit_Count                       :WORD,
                        Fall_Flag                       :DWORD

                        LOCAL   RowStartPositionYTOP    :WORD,
                                        RowStartPositionYBOTTOM  :WORD,
                                        RowStartPositionXLEFT    :WORD,
                                        RowStartPositionXRIGHT   :WORD
                        pushad

                        mov ecx,Cell_Array_Size
                        mov edi,Cell_Array

                        mov ax, Row_Start_Position
                        mov RowStartPositionYTOP, ax
                        add eax, Block_Height
                        sub eax, 1
                        mov RowStartPositionYBOTTOM,ax
                        mov ax,1

                        ;This Part of code calculating Left,Right,Top and Bottom position for each cell.
                        ;Than call WriteCell procedure for write that particular cell
                L1:     mov RowStartPositionXLEFT,ax
                        add eax, Block_Width
                        sub eax,1
                        mov RowStartPositionXRIGHT,ax
                        add ax,1
                        mov edx,Fall_Flag
                        mov (CELL PTR [edi]).Fall_Flag, edx
                        movzx edx,Hit_Count
                        mov (CELL PTR [edi]).HitCount, dx
                        mov edx,1
                        mov (CELL PTR [edi]).Activate, dl
                        mov esi,Row_Color
                        mov (CELL PTR [edi]).Color, esi
                        mov esi,Row_Point
```

```
                        mov (CELL PTR [edi]).Value, esi
                        mov dx,RowStartPositionYTOP
                        mov (CELL PTR [edi]).Area_position.TOP, dx
                        mov dx,RowStartPositionYBOTTOM
                        mov (CELL PTR [edi]).Area_position.BOTTOM, dx
                        mov dx,RowStartPositionXLEFT
                        mov (CELL PTR [edi]).Area_position.LEFT, dx
                        mov dx,RowStartPositionXRIGHT
                        mov (CELL PTR [edi]).Area_position.RIGHT, dx

                        add dx,Block_Height
                        sub dx,1
                        mov ax,dx

                        INVOKE WriteBlock, oHandle, edi

                        add edi,TYPE CELL

                        LOOP L1



                        popad
                        ret
WriteRow ENDP
;This procedute write start line to screen.Start line indicate new upper bound of GameScreen.Above this line is Game
information such as
;points and Remaining lifes
;This procedure is call by :
;               - Main procedure
WriteStartLine PROC ,
                                oHandle         :HANDLE,
                                Line            :PTR _CHAR_INFO

                                LOCAL   ArraySize                                       :COORD,
                                                ReadFromPosition                :COORD,
                                                WriteToScreenBufferPosition     :SMALL_RECT
                                pushad

                                mov ArraySize.X,GameWindow_Width
                                mov ArraySize.Y,1
                                mov ReadFromPosition.X,0
                                mov ReadFromPosition.Y,0
                                mov WriteToScreenBufferPosition.LEFT, 0
                                mov WriteToScreenBufferPosition.TOP, StartLinePosition
                                mov WriteToScreenBufferPosition.BOTTOM, StartLinePosition
                                mov WriteToScreenBufferPosition.RIGHT,GameWindow_Width-1

                                INVOKE WriteConsoleOutputA, oHandle, Line, ArraySize, ReadFromPosition, ADDR
WriteToScreenBufferPosition

                                popad
                                ret
WriteStartLine ENDP


;-----------------------------------------------------------------------------------------------------------
;-------------------------------------------------Block procedure-------------------------------------------
;-----------------------------------------------------------------------------------------------------------
;Write Block to Screen and remember information about cell in CELL structure
;This Procedure is call by next procedures:
;               - WriteRow
;               - GoDown
;               - UpdateScreenMatrix
WriteBlock PROC ,
                        oHandle         :HANDLE,
                        Cell_Info       :PTR CELL

                        LOCAL   CellArray[Max_Cell_Area] :_CHAR_INFO,    ;This array contains information(AsciiValue and
collor) about CHAR from which it was created one Block
                                        color                                           :DWORD,
                                        ArraySize                                       :COORD,                 ;This Data
give information about Array size for read
                                        ArrayStartPosition                      :COORD,
                                        RegionForWrite                          :SMALL_RECT
```

```
                        pushad

                        mov esi,Cell_Info
                        mov eax,(CELL PTR [esi]).Color
                        mov color,eax

                        ;Each individual block is composed of three sets(Arrays):
                        ;       - Up array
                        ;       - Central array
                        ;       - Down array
                        ;Each of them is array of AscII char.Length of Array is equal to Block_Width

                        INVOKE SetColorAndASCII,ADDR (sCellRow0),color,  220, Block_Width ;220 is AscII code for up block side
                        INVOKE SetColorAndASCII,ADDR (sCellRow1),color,  219, Block_Width ;219 is AscII code for cetral part
of the block
                        INVOKE SetColorAndASCII,ADDR (sCellRow2),color,  223, Block_Width ;223 is AscII code for Bottom part
of the block


                        mov Arraysize.X,Block_Width
                        mov Arraysize.Y,Block_Height

                        mov ArrayStartPosition.X,0
                        mov ArrayStartPosition.Y,0

                        mov ax,(CELL PTR [esi]).Area_Position.Left
                        mov  RegionForWrite.Left,ax
                        mov ax,(CELL PTR [esi]).Area_Position.Top
                        mov RegionForWrite.Top,ax
                        mov ax,(CELL PTR [esi]).Area_Position.Right
                        mov RegionForWrite.Right,ax
                        mov ax,(CELL PTR [esi]).Area_Position.Bottom
                        mov RegionForWrite.Bottom,ax

                        INVOKE WriteConsoleOutputA, OHandle, ADDR CellArray, Arraysize, ArrayStartPosition, ADDR
RegionForWrite
                        popad
                        ret
WriteBlock ENDP
;This procedure Initializes _Char_Info Structure
;This procedure is call by next procedures:
;               -WriteBlock
;               -DeleteBlock
SetColorAndASCII PROC,
                        InArr   :PTR _CHAR_INFO,
                        Color   :DWORD,
                        ASCII   :DWORD,
                        Siz             :DWORD
                        pushad

                        mov ecx,Siz
                        mov edi,InArr
                        mov eax, ASCII
                        mov edx, COLOR
        L1:         mov DWORD PTR [edi],eax
                        add edi,2
                        mov DWORD PTR [EDI],edx
                        add edi,2
                        LOOP L1

                        popad
                        ret
SetColorAndASCII ENDP

;This procedure delete Block from screen.Procedure set zero value of color to area of screen defined by Cell_Info.
;;This Procedure is call by next procedures:
;               - GoDown
;               - UpdateScreenMatrix
DeleteBlock PROC ,
                        oHandle         :HANDLE,
                        Cell_Info       :PTR CELL

                        LOCAL   CellArray[Max_Cell_Area] :_CHAR_INFO,
                                        color:DWORD,Array_Size          :COORD,
                                        Array_Start_Position            :COORD,
```

```
                                Region_For_Write                :SMALL_RECT,
                                temp:DWORD
                        pushad

                        mov esi,Cell_Info
                        mov eax,0
                        mov color,eax
                        mov temp,eax

                        INVOKE SetColorAndASCII,ADDR (sCellRow0),color,220, Block_Width
                        INVOKE SetColorAndASCII,ADDR (sCellRow1),color,219, Block_Width
                        INVOKE SetColorAndASCII,ADDR (sCellRow2),color,223, Block_Width

                        mov Array_size.X,Block_Width
                        mov Array_size.Y,Block_Height
                        mov Array_Start_Position.X,0
                        mov Array_Start_Position.Y,0
                        mov ax,(CELL PTR [esi]).Area_Position.Left
                        mov  Region_For_Write.Left,ax
                        mov ax,(CELL PTR [esi]).Area_Position.Top
                        mov Region_For_Write.Top,ax
                        mov ax,(CELL PTR [esi]).Area_Position.Right
                        mov Region_For_Write.Right,ax
                        mov ax,(CELL PTR [esi]).Area_Position.Bottom
                        mov Region_For_Write.Bottom,ax

                        INVOKE WriteConsoleOutputA, OHandle, ADDR CellArray, Array_size, Array_Start_Position, ADDR
Region_For_Write
                        ;After seting color to 0 we must return original color information to Block

                        INVOKE SetColorAndASCII,ADDR (sCellRow0),temp,220, Block_Width
                        INVOKE SetColorAndASCII,ADDR (sCellRow1),temp,219, Block_Width
                        INVOKE SetColorAndASCII,ADDR (sCellRow2),temp,223, Block_Width

                        popad
                        ret
DeleteBlock ENDP
;This procedure lowers the block position.This is used when we have Fall block
;This Procedure is call by next procedures:
;               - MovBlock

GoDown PROC,
                        oHandle         :HANDLE,
                        Block           :PTR CELL,
                        Bottom_Flag     :PTR DWORD

                        LOCAL   lSize                   :COORD,
                                StartPosition   :COORD,
                                AreaForWrite    :SMALL_RECT

                        pushad

                        mov edi,Bottom_Flag             ;Set bottom flag to 0
                        mov DWORD PTR [edi],0

                        mov edx, block                  ;set edx to be Pointer to block
                        mov lSize.X,Block_Width
                        mov lSize.Y,Block_Height
                        mov StartPosition.X,0
                        mov StartPosition.Y,0

                        ;Decreasing Block position
                        movzx eax, (CELL PTR[edx]).Area_Position.Bottom
                        INC eax
                        mov AreaForWrite.Bottom,ax
                        movzx eax, (CELL PTR[edx]).Area_Position.Top
                        INC eax
                        mov AreaForWrite.TOP,ax

                        ;Check if BLOCK is at the bottom
                        SUB eax,GameWindow_height
                        JNZ SKIP
                        mov DWORD PTR [edi],1                       ;Indicate that cell is Disappeared from screen
                SKIP:mov ax,(CELL PTR[edx]).Area_Position.Left
                        mov AreaForWrite.Left,ax
```

```
                        mov ax,(CELL PTR[edx]).Area_Position.Right
                        mov AreaForWrite.Right,ax

                        ;Frist we delete block from old position
                        INVOKE DeleteBlock,ohandle, Block

                        ;Mov new position coordinate to block
                        mov ax,AreaForWrite.Left
                        mov (CELL PTR[edx]).Area_Position.Left,ax
                        mov ax,AreaForWrite.Right
                        mov (CELL PTR[edx]).Area_Position.Right,ax
                        mov ax,AreaForWrite.Top
                        mov (CELL PTR[edx]).Area_Position.Top,ax
                        mov ax,AreaForWrite.Bottom
                        mov (CELL PTR[edx]).Area_Position.Bottom,ax

                        ;Call WriteBlock procedute to write block with new coordinate
                        INVOKE WriteBlock, oHandle, Block

                        popad
                        ret
GoDown ENDP

;This procedure move block down with desire speed.Also this procedure give information to outside world about collision
between block and pad.
;That information is remember in PAD cell
;Procedure is call by next procedures:
;               - UpdateScreenMatrix
MovBlock PROC,
                        oHandle                         :HANDLE,
                        Block                           :PTR CELL,
                        Pad_                            :PTR CELL


                        LOCAL   RowPTR                                  :DWORD,
                                DownFlag                        :DWORD,
                                Temp                                    :WORD,
                                BlockPadCollisionFlag   :DWORD
                        pushad

                        ;Frist we check if time condition is satisfied.
                        mov BlockPadCollisionFlag,0
                        mov DownFlag,0
                        mov edx,Block                                   ;edx is now pointer to Block
                        mov eax,(CELL PTR[edx]).Time
                        INC eax
                        mov (CELL PTR[edx]).Time,eax
                        .IF eax>=Fall_Speed
                                ;If time condition is satisfied than
                                ;Time reset
                                mov eax,0
                                mov (CELL PTR[edx]).Time,eax

                                ;Decrese Block Position
                                INVOKE GoDown,oHandle,Block,ADDR DownFlag

                                ;Did new block coordinates in collision with pad?
                                mov edi,Pad_                                    ;edi is now pointer to pad
                                ;Frist we check is Top side of pad equal to Bottom side of block
                                movzx eax,(CELL PTR [edx]).Area_Position.Bottom
                                movzx ecx,(CELL PTR[edi]).Area_Position.Top
                                .IF eax == ecx

                                        ;if Top side of pad equal to Bottom side of block than we check is pad left side
lower or eqal to block right side
                                        movzx eax,(CELL PTR [edx]).Area_Position.Right
                                        movzx ecx,(CELL PTR[edi]).Area_Position.Left
                                        .IF ecx<=eax

                                                ;if pad left side lower or eqal to block right side than we check is par left
side greater than block left side Reduced by pad  width
                                                movzx eax,(CELL PTR [edx]).Area_Position.Left
                                                sub eax,Pad_width
                                                movzx ecx,(CELL PTR[edi]).Area_Position.left
                                                cmp ecx,eax
```

```
                                        .IF !Sign? || Zero? ;This way to compare sign value.This is equal to ecx>=eax
                                                mov BlockPadCollisionFlag,1 ;
                                        .ENDIF
                                .ENDIF
                        .ENDIF
                        mov eax,BlockPadCollisionFlag
                        .IF eax == 1
                                        ;Now we set new value of fall fleg IN PAD CELL which indicate collision
betwen fall block and pad
                                        ;This remember information about collision In an area that's out of procedure
                                        mov (CELL PTR[edi]).Fall_Flag,1

                                        ;Mov bonus point to pad value
                                        ;This remember information about collision In an area that's out of procedure
                                        mov eax,(CELL PTR [edx]).Value
                                        mov (CELL PTR [edi]).Value,eax

                                        ;Now set fall flag in block to 0 which indicate that Falling block is now
ordinary block
                                        mov eax,0
                                        mov (CELL PTR [edx]).Fall_Flag,eax
                        .ENDIF
                        ;Is block at the bottom?
                        .IF DownFlag == 1
                                mov eax,0
                                mov (CELL PTR [edx]).Fall_Flag,eax
                        .ENDIF
                .ENDIF
                popad
                ret
MovBlock ENDP



;-----------------------------------------------------------------------------------------------------------
;-------------------------------------------------Pad procedure---------------------------------------------
;-----------------------------------------------------------------------------------------------------------


;This procedure write PAD cell to screen
;This Procedure is call by next procedures:
;       - MovPad

WritePAD PROC ,
                oHandle:HANDLE,
                Cell_Info:PTR CELL
                LOCAL Cell_Array[Max_Cell_Area]:_CHAR_INFO,
color:DWORD,Array_Size:COORD,Array_Start_Position:COORD,Region_For_Write:SMALL_RECT
                pushad

                mov esi,Cell_Info
                mov eax,(CELL PTR [esi]).Color
                mov color,eax

                INVOKE SetColorAndASCII,ADDR (PadRow0),color,219, Pad_Width
                INVOKE SetColorAndASCII,ADDR (PadRow1),color,219, Pad_Width
                mov Array_size.X,Pad_Width
                mov Array_size.Y,Pad_Height
                mov Array_Start_Position.X,0
                mov Array_Start_Position.Y,0
                mov ax,(CELL PTR [esi]).Area_Position.Left
                mov  Region_For_Write.Left,ax
                mov ax,(CELL PTR [esi]).Area_Position.Top
                mov Region_For_Write.Top,ax
                mov ax,(CELL PTR [esi]).Area_Position.Right
                mov Region_For_Write.Right,ax
                mov ax,(CELL PTR [esi]).Area_Position.Bottom
                mov Region_For_Write.Bottom,ax
                INVOKE WriteConsoleOutputA, OHandle, ADDR Cell_Array, Array_size, Array_Start_Position, ADDR
Region_For_Write
                popad
                ret
WritePAD ENDP

;This procedure write PAD cell to screen
```

```
;This Procedure is call by next procedures:
;       - MovPad
DeletePAD PROC ,
                        oHandle:HANDLE,
                        Cell_Info:PTR CELL
                        LOCAL Cell_Array[Max_Cell_Area]:_CHAR_INFO,
color:DWORD,Array_Size:COORD,Array_Start_Position:COORD,Region_For_Write:SMALL_RECT
                        pushad

                        mov esi,Cell_Info
                        mov eax,0
                        mov color,eax

                        INVOKE SetColorAndASCII,ADDR (PadRow0),color,220, Pad_Width
                        INVOKE SetColorAndASCII,ADDR (PadRow1),color,223, Pad_Width
                        mov Array_size.X,Pad_Width
                        mov Array_size.Y,Pad_Height
                        mov Array_Start_Position.X,0
                        mov Array_Start_Position.Y,0
                        mov ax,(CELL PTR [esi]).Area_Position.Left
                        mov  Region_For_Write.Left,ax
                        mov ax,(CELL PTR [esi]).Area_Position.Top
                        mov Region_For_Write.Top,ax
                        mov ax,(CELL PTR [esi]).Area_Position.Right
                        mov Region_For_Write.Right,ax
                        mov ax,(CELL PTR [esi]).Area_Position.Bottom
                        mov Region_For_Write.Bottom,ax
                        INVOKE WriteConsoleOutputA, OHandle, ADDR Cell_Array, Array_size, Array_Start_Position, ADDR
Region_For_Write
                        popad
                        ret
DeletePAD ENDP
;This procedure check is pressed left or is pressed right key and move pad to appropriate side if key is down
;This procedure is call by next procedures:
;       - MovPadCell
MovPad PROC,
                oHandle :HANDLE,
                Pad_            :PTR CELL

                pushad

                INVOKE WritePad, oHandle, Pad_
                ;Check if left key arrow is down?
START:  INVOKE GetKeyState,VK_LEFT
                AND eax, KEY_DOWN       ;If key is pressed , the highest bit in EAX register is equal to 1
                JNZ LEFT                ;If left key is pressed than we jump to part of code where MOVE PAD TO LEFT

                INVOKE GetKeyState,VK_RIGHT
                AND eax, KEY_DOWN
                JNZ RIGHT                       ;If right key is pressed than we jump to part of code where MOVE PAD TO right

                JMP GO_EXIT                     ;If no one is pressed than exit from procedure

                ;Code where procces left move for PAD cell
LEFT:   INVOKE DeletePad, oHandle, Pad_
                mov edx, Pad_           ;EDX in now pointer to PAD
                sub (CELL PTR[edx]).Area_Position.Right,Pad_Step ;Decrease Right PAD side
                sub (CELL PTR[edx]).Area_Position.Left, Pad_Step ;Decrease LEFT PAD side
                JLE CORRECTION_LEFT                     ;if left side is lower than screen border JMP to the part of the code
where we return pad to screen

M_LEFT: INVOKE WritePad,oHandle,Pad_

                JMP GO_EXIT

                ;Code where procces right move for PAD cell.This is same like left PAD move
RIGHT:  INVOKE DeletePad,oHandle,Pad_
                mov edx, Pad_
                add (CELL PTR[edx]).Area_Position.Right,Pad_Step
                add (CELL PTR[edx]).Area_Position.Left,Pad_Step
                MOVZX eax,(CELL PTR[edx]).Area_Position.Right
                mov edi,GameWindow_Width
                SUB edi,1
                SUB eax,edi
                JGE CORRECTION_RIGHT
```

```
M_RIGHT:INVOKE WritePad,oHandle,Pad_
                JMP GO_EXIT

CORRECTION_LEFT:
                add (CELL PTR[edx]).Area_Position.Right,Pad_Step
                add (CELL PTR[edx]).Area_Position.Left,Pad_Step
                JMP M_LEFT


CORRECTION_RIGHT:
                sub (CELL PTR[edx]).Area_Position.Right,Pad_Step
                sub (CELL PTR[edx]).Area_Position.Left,Pad_Step
                JMP M_RIGHT

 GO_EXIT:popad
                ret
MovPad ENDP
;This procedure mov PAD with define speed
;This procedure is call by next procedures:
;       - Main procedure
MovPadCell PROC,
                        oHandle                 :HANDLE,
                        Pad_                    :PTR CELL,
                        Row_Time_Info   :PTR DWORD

                        pushad

                        mov edx, Row_Time_Info
                        mov eax, [edx]
                        INC eax
                        mov [edx],eax
                        CMP eax,Pad_Speed               ;Check if speed is satisfied
                        JL GO_END
                                INVOKE MovPad, Ohandle, Pad_
                                mov eax,0
                                mov [edx], eax
         GO_END:popad

                        ret
MovPadCell ENDP

;-------------------------------------------------------------------------------------------------------------------
;-------------------------------------------------Screen procedure--------------------------------------------------
;-------------------------------------------------------------------------------------------------------------------

;This Complex function Update screen blocks matrix.Update is Re-writing of blocks cell if it becomes inactive.Blocks becomes
inactive
;if ball hit block enough number of times.HitCount is a number that shows how many times a block should be hit.If that number
is reach
;than block becomes inactive.
;This procedure is call by next procedures:
;               - Main procedure
UpdateScreenMatrix PROC,
                        oHandle                 :HANDLE,
                        Matrix                  :PTR CELL,
                        Pad_                    :PTR CELL,
                        Points                  :PTR DWORD

                        pushad
                        mov edx,Matrix  ;now is edx pointer to matrix
                        mov ecx,Start_Blocks_Array_Size
START:          movzx eax,(CELL PTR [edx]).Activate;Check if block active?
                        .IF eax ==1;If block is active than we check if rich enough hit number
                                movsx eax,(CELL PTR [edx]).HitCount
                                cmp eax,0
                                .IF SIGN? || ZERO? ;If HitCount <= 0 .This is way for compare two sign number
                                        mov eax,(CELL PTR [edx]).Fall_Flag
                                        .IF eax == 1
                                                INVOKE MovBlock,ohandle,edx,Pad_
                                                mov edi,Pad_
                                                mov eax,(CELL PTR[edi]).Fall_Flag
                                                .IF eax == 1
                                                        mov eax,(CELL PTR[edi]).Value
                                                        mov esi,points
```

```
                                                add [esi],eax
                                                mov eax,[esi]
                                                JMP SKIP
CUT:                                        JMP START
SKIP:
                                                INVOKE WritePoints,ohandle,eax
                                                mov eax,0
                                                mov (CELL PTR[edi]).Fall_Flag,eax
                                                mov (CELL PTR[edi]).Value,eax
                                        .ENDIF
                                .ELSE
                                        mov eax,0
                                        mov (CELL PTR[edx]).Activate,al
                                        mov eax,(CELL PTR[edx]).Value
                                        mov edi,Points
                                        add [edi],eax
                                        mov eax,[edi]
                                        INVOKE WritePoints,ohandle,eax
                                        INVOKE DeleteBlock, ohandle,edx
                                        JMP END_LOOP
                                .ENDIF
                        .ENDIF
                        INVOKE WriteBlock,ohandle,edx
                .ENDIF
END_LOOP:       add edx,Cell_Size
                LOOP CUT

                popad
                ret
UpdateScreenMatrix ENDP



WritePoints PROC,
                ohandle                 :HANDLE,
                Points                  :DWORD
                LOCAL  Ascii_Array[5]   :WORD,
                                        Counter         :DWORD,
                                        temp            :DWORD
                pushad
                mov Counter,0
                mov eax,Points
                LEA ecx,AscII_Array
                .IF eax ==0
                        add eax,48
                        mov WORD PTR [ecx],ax
                        mov Counter,1
                        JMP WRITE_POINTS
                .ENDIF

                DecimalToAscIIm
                ReverseArraym

WRITE_POINTS:
                INVOKE WriteBigString, oHandle, ADDR Delete_array, 5       , 1, ADDR Points_Color_Array,
Points_Start_Position, 1
                INVOKE WriteBigString, ohandle,  ADDR Ascii_Array , Counter, 1, ADDR Points_Color_Array,
Points_Start_Position, 1

                popad
                ret
WritePoints ENDP

LifeCheck PROC,
                ohandle                 :HANDLE,
                Lifes                   :DWORD,
                Ball_Cell               :PTR BALL_
                LOCAL  Ascii_Array[5]   :WORD,
                                        Counter         :DWORD,
                                        temp            :DWORD
                pushad
                mov Counter,0
                mov eax,Lifes
                LEA ecx,AscII_Array
```

```
                        DecimalToAscIIm
                        ReverseArraym

                        INVOKE WriteBigString, oHandle, ADDR Delete_array, 5       , 1, ADDR Life_Color_Array,
Life_Start_Position     , 1
                        INVOKE WriteBigString, ohandle,  ADDR Ascii_Array , Counter, 1, ADDR Life_Color_Array,
Life_Start_Position     , 1
                        mov edx,Ball_Cell
                popad
                ret
LifeCheck ENDP
WriteBallCell PROC,
                        oHandle  :      HANDLE,
                        Ball_Cell:      PTR BALL_
                        LOCAL char:WORD,Attribute:WORD,WrittenChar:DWORD,Position:COORD
                        pushad
                        mov char,25CBh
                        mov Attribute,Ball_color
                        mov edx, Ball_Cell
                        movzx eax, (BALL_ PTR [edx]).CurrentPos.x
                        mov Position.x,ax
                        movzx eax, (BALL_ PTR [edx]).CurrentPos.y
                        mov Position.y,ax
                        INVOKE WriteConsoleOutputCharacterW,oHandle,ADDR char,1,Position,ADDR WrittenChar
                        INVOKE WriteConsoleOutputAttribute,oHandle,ADDR Attribute,1,Position,ADDR WrittenChar

                        popad
                        ret
WriteBallCell ENDP
DeleteBallCell PROC,
                        oHandle  :      HANDLE,
                        Ball_Cell:      PTR BALL_
                        LOCAL char:WORD,Attribute:WORD,WrittenChar:DWORD,Position:COORD
                        pushad
                        mov char,25CBh
                        mov Attribute,0
                        mov edx, Ball_Cell
                        movzx eax, (BALL_ PTR [edx]).CurrentPos.x
                        mov Position.x,ax
                        movzx eax, (BALL_ PTR [edx]).CurrentPos.y
                        mov Position.y,ax
                        INVOKE WriteConsoleOutputCharacterW,oHandle,ADDR char,1,Position,ADDR WrittenChar
                        INVOKE WriteConsoleOutputAttribute,oHandle,ADDR Attribute,1,Position,ADDR WrittenChar


                        popad
                        ret
DeleteBallCell ENDP

FindCollision PROC,   ;This function return 0 in EAX register if dont find collision betwen Ball and Block or Ball and Pad.In
other case return 1 in EAX register
                        Array:PTR CELL,
                        Coordinate:COORD,
                        Out_BLOCK:PTR DWORD
                        LOCAL Counter:DWORD,Top:DWORD,Left:DWORD,Right:DWORD,Bottom:DWORD
                        mov Counter,0
                        .WHILE (Counter<Start_Blocks_Array_Size+1)
                                mov eax,counter
                                mov edx,Cell_Size
                                mul edx
                                add eax,array
                                mov edx,eax
                                movzx  eax,(CELL PTR [edx]).Area_Position.Left
                                mov Left ,eax
                                movzx  eax,(CELL PTR [edx]).Area_Position.Right
                                mov Right ,eax
                                movzx  eax,(CELL PTR [edx]).Area_Position.Top
                                mov Top ,eax
                                movzx  eax,(CELL PTR [edx]).Area_Position.Bottom
                                mov Bottom ,eax
                                movzx eax, Coordinate.Y
                                .IF (eax>=Top) && (eax<=Bottom)                         ;Check is BALL in range?
                                        movzx eax, Coordinate.X
                                        .IF (eax>=Left) && (eax<=Right)
                                                movzx eax, Coordinate.Y
```

```
                                        .IF(eax == Bottom) || (eax == Top)
                                                mov eax,TopBottom_Collision
                                                JMP Check_Active
                                        .ENDIF
                                        movzx eax, Coordinate.X
                                        .IF ( eax == Left)
                                                 mov eax,Right_Collision
                                                 JMP Check_Active
                                        .ELSEIF (eax == Right)
                                                 mov eax,Left_Collision
                                                 JMP Check_Active
                                        .ENDIF


                                                .ENDIF
                                .ENDIF
                                INC Counter
                                .CONTINUE
Check_Active:   movzx edi, (CELL PTR [edx]).Activate
                                .IF  edi == 1
                                        mov edi, Out_Block
                                        mov [edi],edx
                                        JMP END_PROC
                                .ELSE
                                        mov eax, NoCollision
                                        JMP END_PROC
                                .ENDIF
                        .ENDW
                        mov eax,NoCollision
        END_PROC:
                        ret
FindCollision ENDP

ReBoundBall PROC,
                        Ball_Cell:PTR BALL_,
                        Direction:DWORD
                        LOCAL x_curr:DWORD,y_curr:DWORD,NextPosition:COORD
                        pushad
                        mov eax,Direction
                        mov edx,Ball_Cell
                        .IF eax==UP_Left
                                JMP UP_LEFTL
                        .ELSEIF eax==UP_RIGHT
                                JMP UP_RIGHTL
                        .ELSEIF eax==DOWN_RIGHT
                                JMP DOWN_RIGHTL
                        .ELSEIF eax==DOWN_LEFT
                                JMP DOWN_LEFTL
                        .ENDIF
                        JMP END_PROC
UP_RIGHTL:  movzx eax,(BALL_ PTR[edx]).CurrentPos.X
                        INC eax
                        mov NextPosition.X,ax
                        movzx eax,(BALL_ PTR[edx]).CurrentPos.Y
                        DEC eax
                        mov NextPosition.Y,ax
                        INVOKE SetNextBallPosition, edx, NextPosition
                        JMP END_PROC
UP_LEFTL:   movzx eax,(BALL_ PTR[edx]).CurrentPos.X
                        DEC eax
                        mov NextPosition.X,ax
                        movzx eax,(BALL_ PTR[edx]).CurrentPos.Y
                        DEC eax
                        mov NextPosition.Y,ax
                        INVOKE SetNextBallPosition, edx, NextPosition
                        JMP END_PROC
DOWN_RIGHTL:movzx eax,(BALL_ PTR[edx]).CurrentPos.X
                        INC eax
                        mov NextPosition.X,ax
                        movzx eax,(BALL_ PTR[edx]).CurrentPos.Y
                        INC eax
                        mov NextPosition.Y,ax
                        INVOKE SetNextBallPosition, edx, NextPosition
                        JMP END_PROC
DOWN_LEFTL: movzx eax,(BALL_ PTR[edx]).CurrentPos.X
```

```
                        DEC eax
                        mov NextPosition.X,ax
                        movzx eax,(BALL_ PTR[edx]).CurrentPos.Y
                        INC eax
                        mov NextPosition.Y,ax
                        INVOKE SetNextBallPosition, edx, NextPosition
                        JMP END_PROC

END_PROC:   popad
                        ret

ReBoundBall ENDP

NextBallPosition PROC,          ;This Procedure calculate next ball position by Previos and current position of BALL.Also this
procedure check if Ball is at the window edge or fall from window
                        oHandle:HANDLE,
                        Ball_Cell:PTR BALL_,
                        BLocks_Matrix:PTR CELL,
                        Lifes   :PTR DWORD
                        LOCAL NextPosition:COORD, CurrentPosition:COORD,CollisionBlockPTR:DWORD
                        pushad
                        mov edx,Ball_Cell
                        movzx eax, (BALL_ PTR [edx]).CurrentPos.X
                        mov CurrentPosition.X,ax
                        movzx eax, (BALL_ PTR [edx]).CurrentPos.Y
                        mov CurrentPosition.Y,ax
                        movzx eax,(BALL_ PTR [edx]).PrevPos.X
                        .IF ax > CurrentPosition.X
                                movzx eax,(BALL_ PTR [edx]).PrevPos.Y
                                JMP LEFTL
                        .ELSE
                                movzx eax,(BALL_ PTR [edx]).PrevPos.Y
                                JMP RIGHTL
                        .ENDIF
LEFTL:          .IF ax > CurrentPosition.Y
                                JMP UP_LEFTL
                        .ELSE
                                JMP DOWN_LEFTL
                        .ENDIF
RIGHTL:         .IF ax > CurrentPosition.Y
                                JMP UP_RIGHTL
                        .ELSE
                                JMP DOWN_RIGHTL
                        .ENDIF
UP_LEFTL:       DEC CurrentPosition.X
                        DEC CurrentPosition.Y
                        INVOKE FindCollision,Blocks_Matrix, CurrentPosition,ADDR CollisionBlockPTR
                        .IF eax != NoCollision
                                mov edx,CollisionBlockPTR
                                DEC (CELL PTR [edx]).HitCount
                                movzx edi, (CELL PTR [edx]).HitCount
                                .IF edi == 1
                                        mov (CELL PTR [edx]).Color,HitColor
                                .ENDIF
                                mUP_Left
                        .ELSEIF
                                INVOKE IsAtScreenEdge,CurrentPosition
                                .IF eax!=NoCollision && eax != DeadBall
                                        mUP_Left
                                .ELSEIF eax == DeadBall
                                        mov edx,Ball_Cell
                                        mov (BALL_ PTR[edx]).Active,0
                                        mov edx,Lifes
                                        mov eax,DWORD PTR [edx]
                                        dec eax
                                        mov DWORD PTR [edx],eax
                                        INVOKE LifeCheck,                                       oHandle,eax,            ADDR Ball
                                .ELSE
                                        INVOKE SetNextBallPosition,Ball_Cell,CurrentPosition
                                .ENDIF
                        .ENDIF
                        JMP END_PROC
UP_RIGHTL:      INC CurrentPosition.X
                        DEC CurrentPosition.Y
                        INVOKE FindCollision,Blocks_Matrix, CurrentPosition,ADDR CollisionBlockPTR
```

```
                .IF eax != NoCollision
                        mov edx,CollisionBlockPTR
                        DEC (CELL PTR [edx]).HitCount
                        movzx edi, (CELL PTR [edx]).HitCount
                        .IF edi == 1
                                mov (CELL PTR [edx]).Color,HitColor
                        .ENDIF
                        mUP_Right
                .ELSEIF
                        INVOKE IsAtScreenEdge,CurrentPosition
                        .IF eax!=NoCollision && eax!= DeadBall
                                mUP_Right
                        .ELSEIF eax == DeadBall
                                mov edx,Ball_Cell
                                mov (BALL_ PTR[edx]).Active,0
                                mov edx,Lifes
                                mov eax,DWORD PTR [edx]
                                dec eax
                                mov DWORD PTR [edx],eax
                                INVOKE LifeCheck,                                oHandle,eax,            ADDR Ball
                        .ELSE
                                INVOKE SetNextBallPosition,Ball_Cell,CurrentPosition
                        .ENDIF
                .ENDIF
                JMP END_PROC
DOWN_LEFTL:     DEC CurrentPosition.X
                INC CurrentPosition.Y
                INVOKE FindCollision,Blocks_Matrix, CurrentPosition,ADDR CollisionBlockPTR
                .IF eax != NoCollision
                        mov edx,CollisionBlockPTR
                        DEC (CELL PTR [edx]).HitCount
                        movzx edi, (CELL PTR [edx]).HitCount
                        .IF edi == 1
                                mov (CELL PTR [edx]).Color,HitColor
                        .ENDIF
                        mDown_Left
                .ELSEIF
                        INVOKE IsAtScreenEdge,CurrentPosition
                        .IF eax!=NoCollision && eax != DeadBall
                                mDown_Left
                        .ELSEIF eax == DeadBall
                                mov edx,Ball_Cell
                                mov (BALL_ PTR[edx]).Active,0
                                mov edx,Lifes
                                mov eax,DWORD PTR [edx]
                                dec eax
                                mov DWORD PTR [edx],eax
                                INVOKE LifeCheck,                                oHandle,eax,            ADDR Ball
                        .ELSE
                                INVOKE SetNextBallPosition,Ball_Cell,CurrentPosition
                        .ENDIF
                .ENDIF
                JMP END_PROC
DOWN_RIGHTL:    INC CurrentPosition.X
                INC CurrentPosition.Y
                INVOKE FindCollision,Blocks_Matrix, CurrentPosition,ADDR CollisionBlockPTR
                        .IF eax != NoCollision
                        mov edx,CollisionBlockPTR
                        DEC (CELL PTR [edx]).HitCount
                        movzx edi, (CELL PTR [edx]).HitCount
                        .IF edi == 1
                                mov (CELL PTR [edx]).Color,HitColor
                        .ENDIF
                        mDown_Right
                .ELSEIF
                        INVOKE IsAtScreenEdge,CurrentPosition
                        .IF eax!=NoCollision && eax != DeadBall
                                mDown_Right
                        .ELSEIF eax == DeadBall
                                mov edx,Ball_Cell
                                mov (BALL_ PTR[edx]).Active,0
                                mov edx,Lifes
                                mov eax,DWORD PTR [edx]
                                dec eax
                                mov DWORD PTR [edx],eax
```

```
                                        INVOKE LifeCheck,                                       oHandle,eax,            ADDR Ball
                                .ELSE
                                        INVOKE SetNextBallPosition,Ball_Cell,CurrentPosition
                                .ENDIF
                        .ENDIF
END_PROC:   popad
                        ret
NextBallPosition ENDP
SetNextBallPosition PROC,
                        Ball_Cell:PTR BALL_,
                        NextPosition:COORD
                        pushad
                        mov edx,Ball_Cell
                        movzx eax, (BALL_ PTR [edx]).CurrentPos.X
                        mov (BALL_ PTR [edx]).PrevPos.X, ax
                        movzx eax, (BALL_ PTR [edx]).CurrentPos.Y
                        mov (BALL_ PTR [edx]).PrevPos.Y, ax
                        movzx eax, NextPosition.X
                        mov (BALL_ PTR [edx]).CurrentPos.X, ax
                        movzx eax, NextPosition.Y
                        mov (BALL_ PTR [edx]).CurrentPos.Y, ax
                        popad
                        ret
SetNextBallPosition ENDP


IsAtScreenEdge PROC,                            ;this function return 0 in EAX if Ball is in window,1 if BALL is window edge
and 2 if ball fall
                        Position:COORD
                        mov eax,0
                        .IF Position.x>=GameWindow_Width
                                mov eax,Right_Collision
                        .ELSEIF Position.X<=0
                                mov eax,Left_Collision
                        .ELSEIF Position.Y<=StartLinePosition
                                mov eax,TopBottom_Collision
                        .ELSEIF Position.Y>=GameWindow_height
                                mov eax,4
                        .ENDIF
                        ret
IsAtScreenEdge ENDP
MovBall PROC,
                oHandle:HANDLE,
                BALL_CELL:PTR BALL_,
                Blocks_Matrix:PTR CELL,
                Time_Info:PTR DWORD,
                Lifes:PTR DWORD
                pushad
                mov edx,Time_info
                mov eax,[edx]
                .IF eax>=Ball_Speed
                        INVOKE DeleteBallCell,ohandle,BaLL_CELL
                        INVOKE NextBallPosition,ohandle, BAll_CeLL,Blocks_Matrix,Lifes
                        INVOKE WriteBallCell,ohandle,Ball_CeLL
                        mov edx,Time_info
                        mov eax,[edx]
                        mov WORD PTR[edx],0
                        JMP END_P
                .ENDIF
                mov edx,Time_info
                mov eax,[edx]
                INC eax
                mov [edx],eax
END_P:
                popad
                ret
MovBall ENDP
SetStartPosition PROC,
                Pad_Cell:PTR CELL,
                Ball_Cell:PTR BALL_
                LOCAL   tempc:COORD,
                                tempp:COORD,
                                randomVal:DWORD
                pushad
                mov eax,GameWindow_Width
```

```
		INVOKE RandomRange
		mov randomVal,eax
		mov edx,Pad_Cell
		mov edi,Ball_Cell
		mov (CELL PTR [edx]).Area_Position.Left,ax
		mov ebx,Pad_Width
		add eax,ebx
		dec eax
		mov (CELL PTR [edx]).Area_Position.Right,ax
		dec eax
		dec eax
		mov (BALL_ PTR [edi]).CurrentPos.X,ax
		dec eax
		mov (BALL_ PTR [edi]).PrevPos.X,ax
		mov eax,GameWindow_Height
		dec eax
		mov (CELL PTR [edx]).Area_Position.Bottom,ax
		sub eax,Pad_Height
		mov (BALL_ PTR [edi]).CurrentPos.Y,ax
		inc eax
		mov (CELL PTR [edx]).Area_Position.Top,ax
		mov (BALL_ PTR [edi]).PrevPos.Y,ax


		popad
		ret
SetStartPosition ENDP



END
```

- Sadržaj fajla *Additionaldef.inc*

```
;=================================================================================================================================
;---------------------------------------------------------------------------------------------------------------------------------
;------------------------------------------------- Function.inc ----------------------------------------------------
;---------------------------------------------------------------------------------------------------------------------------------
;------------------------------------------------- Authors: ----------------------------------------------------
;---------------------------------------------- Turkmanovic Haris 516/2014 ----------------------------------------------
;---------------------------------------------- Vukoje David 541/2014 ----------------------------------------------
;=================================================================================================================================
;Description:
;	This .inc file contains declaration of function which is located in windows system Libraries such as kernel32.lin and
;user32.lib . These declaration are necessary because they are not directly declared in Irvine32.lib
;=================================================================================================================================


INCLUDE Irvine32.inc
FreeConsole PROTO
AllocConsole PROTO

WriteConsoleOutputA PROTO ,
			oHandle:HANDLE,
			ArrayForRead:PTR _CHAR_INFO,
			SizeOfArrayForRead:COORD,
			StartPositionForRead:COORD,
			ScreenBufferAreaForWrite:PTR SMALL_RECT
WriteConsoleOutputW PROTO ,
			oHandle:HANDLE,
			ArrayForRead:PTR _CHAR_INFO,
			SizeOfArrayForRead:COORD,
			StartPositionForRead:COORD,
			ScreenBufferAreaForWrite:PTR SMALL_RECT
WriteConsoleOutputCharacterW PROTO,
			hConsoleOutput:HANDLE,
			lpCharacter:PTR WORD,
			nLength:DWORD,
			dwWriteCoord:COORD,
			lpNumberOfCharsWritten:PTR DWORD
SetConsoleCursorPosition PROTO,
	    hConsoleOutput:HANDLE,
```

```
                        wCursorPosition:COORD
ReadConsoleOutputA PROTO ,
                        oHandle:HANDLE,
                        ArrayForRead:PTR _CHAR_INFO,
                        SizeOfArrayForRead:COORD,
                        StartPositionForRead:COORD,
                        ScreenBufferAreaForWrite:PTR SMALL_RECT
CreateConsoleScreenBuffer PROTO,
                        dwDesiredAccess                 :DWORD ,
                        dwShareMode                             :DWORD ,
                        lpSecurityAttributes    :PTR _SECURITY_ATTRIBUTES ,
            dwFlags                                     :DWORD,
                        lpScreenBufferData              :PTR DWORD
SetConsoleActiveScreenBuffer PROTO,
                        handlee:HANDLE
FillConsoleOutputAttribute PROTO,
                        hConsoleOutput:HANDLE,
                        lpCharacter:WORD,
                        nLength:DWORD,
                        dwWriteCoord:COORD,
                        lpNumberOfCharsWritten:PTR DWORD
WriteConsoleOutputW PROTO ,
                        oHandle:HANDLE,
                        ArrayForRead:PTR _CHAR_INFO,
                        SizeOfArrayForRead:COORD,
                        StartPositionForRead:COORD,
                        ScreenBufferAreaForWrite:PTR SMALL_RECT



_CHAR_INFO STRUCT
                AsciiChar WORD 219
                Attributes WORD 0
_CHAR_INFO ENDS

_SECURITY_ATTRIBUTES STRUCT
                nLength DWORD 0
                lpSecurityDescriptor DWORD 0
   bInheritHandle DWORD 0
_SECURITY_ATTRIBUTES ENDS
```

- Sadržaj fajla *Configuration.inc*

```
;#########################################################################################################################
;-------------------------------------------------------------------------------------------------------------------------
;------------------------------------------------ COnfiguration.inc ------------------------------------------------------
;-------------------------------------------------------------------------------------------------------------------------
;------------------------------------------------------ Authors: ---------------------------------------------------------
;---------------------------------------------- Turkmanovic Haris 516/2014 -----------------------------------------------
;---------------------------------------------- Vukoje David 541/2014 ----------------------------------------------------
;#########################################################################################################################
;Description:
;       This include file contains game properties like number of lives,ball speed,pad speed, ....
;   Also , there is game windows properties
;       Thanks to this .inc file game is more flexible and the player can adjust the game according to his affinities
;       Part of .inc file mark as private can't be modified by player
;#########################################################################################################################


;Windows size
GameWindow_Width        = 122 ;Start width of game window
GameWindow_Height               = 55  ;Start Hight of game window
WelcomeScreen_Width             = 70
WelcomeScreen_Hight             = 35

;GameGraphic
StartLinePosition               = 7 ;+1 position where is Frist Block Matrix row start. Unit is CHAR
StartLineColor                  = Blue
Block_Width                     = 10
BLock_Height                    = 3
Pad_Width                       = 16
Pad_Height                      = 2
```

```
Pad_Color                       = White
Ball_Color                      = 7
HitColor                        = lightRed
Points_color                    = BLUE
Life_color                      = Yellow



;GameOptions
Start_Life_Number               = 2
Pad_speed                       = 0
Ball_Speed                      = 1
Pad_step                        = 1
Fall_Speed              = 4
Number_Of_Colors        = 6
Number_of_Cell_in_Row   = 11
Max_Cell_Area           = 100; This is max area which cell can obtain. Unit is CHAR
WS_Time_Duration        = 50


AppName TEXTEQU <"Arkanoid",0>

;-------------------------------------
;-----------Private-----------------
KEY_DOWN EQU 80000000h
KEY_TIME_PRESS          = 10
Cell_Size               = 27
Dec_LIFE                = 0 ;Flag which indicate that Ball is fall

sCellRow0 EQU CellArray+0*4*Block_Width
sCellRow1 EQU CellArray+1*4*Block_Width
sCellRow2 EQU CellArray+2*4*Block_Width

PadRow0 EQU Cell_Array+0*4*Pad_Width
PadRow1 EQU Cell_Array+1*4*Pad_Width

MatrixRow0 EQU Blocks_Cell_Array+0*Number_of_Cell_in_Row*Cell_Size
MatrixRow1 EQU Blocks_Cell_Array+1*Number_of_Cell_in_Row*Cell_Size
MatrixRow2 EQU Blocks_Cell_Array+2*Number_of_Cell_in_Row*Cell_Size
MatrixRow3 EQU Blocks_Cell_Array+3*Number_of_Cell_in_Row*Cell_Size
MatrixRow4 EQU Blocks_Cell_Array+4*Number_of_Cell_in_Row*Cell_Size
MatrixRow5 EQU Blocks_Cell_Array+5*Number_of_Cell_in_Row*Cell_Size

Pad_X_Start_Position    = GameWindow_width/2-Pad_Width/2;Private
Pad_Y_Start_Position    = GameWindow_height-1-Pad_Height;Private

Row EQU StartLinePosition+1+esi*Cell_Height
Start_Blocks_Array_Size EQU Number_Of_Colors*Number_of_Cell_in_Row

NoCollision             =       0
TopBottom_Collision     =       1
Left_Collision          =       2
Right_Collision         =       3
UP_LEFT                 =       1
UP_RIGHT                =       2
DOWN_LEFT               =       3
DOWN_RIGHT              =       4
DeadBall                =       4

GameInfoArea EQU GameWindow_width*StartLinePosition
```

- Sadržaj fajla *font.inc*

```
ETF_LOGO_CODE                   = 2000
DELETE_CODE                     = 2001


CreateLetter PROTO,
                                oHandle                 :HANDLE,
                                Array_Size              :COORD,
                                Start_Position          :COORD,
                                Letter_Array            :PTR _CHAR_INFO,
```

```
                                Letter_Color            :DWORD
WriteLetter PROTO,
                                oHandle                                 :HANDLE,
                                Ascii_Code                              :WORD,
                                Letter_Start_Position                   :COORD,
                                Font_Group                              :WORD,
                                Color                                   :DWORD

WriteBigString PROTO,
                                oHandle                 :HANDLE,
                                String                  :PTR WORD,
                                L                       :DWORD,
                                Font_Group              :WORD,
                                Array_Font_Color        :PTR DWORD,
                                Start_Position          :COORD,
                                Font_Step               :DWORD


.data
Delete_Array            WORD 5 DUP(DELETE_CODE)
```

- Sadržaj fajla *font.asm*

```
INCLUDE Additionaldef.INC
INCLUDE Font.Inc
INCLUDE fontemplate3x5.INC
INCLUDE fontemplate5x7.INC
INCLUDE fontemplate9x13.INC

.code
CreateLetter PROC,
                                oHandle                 :HANDLE,
                                Array_size              :COORD,
                                Start_Position          :COORD,
                                Letter_Array            :PTR _CHAR_INFO,
                                Letter_Color            :DWORD

                                LOCAL   ToWriteArea                     :SMALL_RECT,
                                                WriteStartPosition      :COORD

                                pushad

                                mov eax,Letter_Color
                                CMP eax,0
                                .IF !SIGN?;If Color >=0 than we change color
                                        mov edx,Letter_Array
                                        movzx eax,Array_Size.x
                                        movzx ecx,Array_Size.Y
                                        mul ecx
                                        mov ecx,eax
                                        mov edx,Letter_Array
                                        .WHILE ecx>0
                                                 mov eax,Letter_Color
                                                 movzx edi,(_CHAR_INFO PTR [edx]).Attributes
                                                 .IF edi != Font_Backround
                                                 mov (_CHAR_INFO PTR [edx]).Attributes,ax
                                                 .ENDIF
                                                 dec ecx
                                                 add edx,TYPE _CHAR_INFO
                                        .ENDW

                                .ENDIF


                                movzx eax,Start_Position.X
                                mov ToWriteArea.Left,ax
                                movzx eax,Start_Position.Y
                                mov ToWriteArea.Top,ax
                                movzx eax,Array_Size.x
                                movzx ecx,Start_Position.X
                                add eax,ecx
                                mov ToWriteArea.Right,ax
                                movzx eax,Array_Size.Y
                                movzx ecx,Start_Position.Y
```

```
                        add eax,ecx
                        mov ToWriteArea.Bottom,ax
                        mov WriteStartPosition.X,0
                        mov WriteStartPosition.Y,0
                        INVOKE WriteConsoleOutputA,ohandle,Letter_Array,Array_Size,WriteStartPosition,ADDR ToWriteArea
                        popad
                        ret
CreateLetter ENDP

WriteLetter PROC,
                        oHandle                                 :HANDLE,
                        Ascii_Code                              :WORD,
                        Letter_Start_Position   :COORD,
                        Font_Group                              :WORD,
                        Color                                   :DWORD
                        pushad

                        movzx eax,Ascii_Code
                        .IF eax == DELETE_CODE
                                movzx eax,Font_Group
                                .IF eax == 1
                                        INVOKE CreateLetter,ohandle,Letter_Size1,Letter_Start_Position,ADDR
Delete_font10,color
                                .ELSEIF eax == 2
                                        INVOKE CreateLetter,ohandle,Letter_Size1,Letter_Start_Position,ADDR
Delete_font20,color
                                .ENDIF
                        .ELSEIF eax == ETF_LOGO_CODE
                                movzx eax,Font_Group
                                .IF eax == 3
                                        INVOKE CreateLetter,ohandle,Letter_Size3,Letter_Start_Position,ADDR
ETF_Logo,color
                                .ENDIF
                        .ELSEIF eax == '1'
                                movzx eax,Font_Group
                                .IF eax == 1
                                        INVOKE CreateLetter,ohandle,Letter_Size1,Letter_Start_Position,ADDR
NUMBER_ONE,color
                                .ELSEIF eax == 2

                                .ENDIF
                        .ELSEIF eax == '2'
                                movzx eax,Font_Group
                                .IF eax == 1
                                        INVOKE CreateLetter,ohandle,Letter_Size1,Letter_Start_Position,ADDR
NUMBER_TWO,color
                                .ELSEIF eax == 2

                                .ENDIF
                        .ELSEIF eax == '3'
                                movzx eax,Font_Group
                                .IF eax == 1
                                        INVOKE CreateLetter,ohandle,Letter_Size1,Letter_Start_Position,ADDR
NUMBER_THREE,color
                                .ELSEIF eax == 2

                                .ENDIF
                        .ELSEIF eax == '4'
                                movzx eax,Font_Group
                                .IF eax == 1
                                        INVOKE CreateLetter,ohandle,Letter_Size1,Letter_Start_Position,ADDR
NUMBER_FOUR,color
                                .ELSEIF eax == 2

                                .ENDIF
                        .ELSEIF eax == '5'
                                movzx eax,Font_Group
                                .IF eax == 1
                                        INVOKE CreateLetter,ohandle,Letter_Size1,Letter_Start_Position,ADDR
NUMBER_FIVE,color
                                .ELSEIF eax == 2

                                .ENDIF
                        .ELSEIF eax == '6'
                                movzx eax,Font_Group
```

```
                                        .IF eax == 1
                                                INVOKE CreateLetter,ohandle,Letter_Size1,Letter_Start_Position,ADDR
NUMBER_SIX,color
                                        .ELSEIF eax == 2

                                        .ENDIF
                                .ELSEIF eax == '7'
                                        movzx eax,Font_Group
                                        .IF eax == 1
                                                INVOKE CreateLetter,ohandle,Letter_Size1,Letter_Start_Position,ADDR
NUMBER_SEVEN,color
                                        .ELSEIF eax == 2

                                        .ENDIF
                                .ELSEIF eax == '8'
                                        movzx eax,Font_Group
                                        .IF eax == 1
                                                INVOKE CreateLetter,ohandle,Letter_Size1,Letter_Start_Position,ADDR
NUMBER_EIGHT,color
                                        .ELSEIF eax == 2

                                        .ENDIF
                                .ELSEIF eax == '9'
                                        movzx eax,Font_Group
                                        .IF eax == 1
                                                INVOKE CreateLetter,ohandle,Letter_Size1,Letter_Start_Position,ADDR
NUMBER_NINE,color
                                        .ELSEIF eax == 2

                                        .ENDIF
                                .ELSEIF eax == '0'
                                        movzx eax,Font_Group
                                        .IF eax == 1
                                                INVOKE CreateLetter,ohandle,Letter_Size1,Letter_Start_Position,ADDR
NUMBER_ZERO,color
                                        .ELSEIF eax == 2

                                        .ENDIF
                                .ELSEIF eax == 'A'
                                movzx eax,Font_Group
                                        .IF eax == 1
                                        .ELSEIF eax == 2
                                                INVOKE CreateLetter,ohandle,Letter_Size2,Letter_Start_Position,ADDR
A_Letter_20     ,color
                                        .ENDIF
                                .ELSEIF eax == 'R'
                                movzx eax,Font_Group
                                        .IF eax == 1
                                        .ELSEIF eax == 2
                                                INVOKE CreateLetter,ohandle,Letter_Size2,Letter_Start_Position,ADDR
R_Letter_20     ,color
                                        .ENDIF
                                .ELSEIF eax == 'K'
                                movzx eax,Font_Group
                                        .IF eax == 1
                                        .ELSEIF eax == 2
                                                INVOKE CreateLetter,ohandle,Letter_Size2,Letter_Start_Position,ADDR
K_Letter_20     ,color
                                        .ENDIF
                                .ELSEIF eax == 'N'
                                movzx eax,Font_Group
                                        .IF eax == 1
                                        .ELSEIF eax == 2
                                                INVOKE CreateLetter,ohandle,Letter_Size2,Letter_Start_Position,ADDR
N_Letter_20     ,color
                                        .ENDIF
                                .ELSEIF eax == 'O'
                                movzx eax,Font_Group
                                        .IF eax == 1
                                        .ELSEIF eax == 2
                                                INVOKE CreateLetter,ohandle,Letter_Size2,Letter_Start_Position,ADDR
O_Letter_20     ,color
                                        .ENDIF
                                .ELSEIF eax == 'I'
                                movzx eax,Font_Group
```

```
                                        .IF eax == 1
                                        .ELSEIF eax == 2
                                                INVOKE CreateLetter,ohandle,Letter_Size2,Letter_Start_Position,ADDR
I_Letter_20     ,color
                                        .ENDIF
                                .ELSEIF eax == 'D'
                                movzx eax,Font_Group
                                        .IF eax == 1
                                        .ELSEIF eax == 2
                                                INVOKE CreateLetter,ohandle,Letter_Size2,Letter_Start_Position,ADDR
D_Letter_20,color
                                        .ENDIF
                                .ELSEIF eax == 'G'
                        movzx eax,Font_Group
                                        .IF eax == 1
                                        .ELSEIF eax == 2
                                                INVOKE CreateLetter,ohandle,Letter_Size2,Letter_Start_Position,ADDR
G_Letter_20,color
                                        .ENDIF
                                .ELSEIF eax == 'M'
                        movzx eax,Font_Group
                                        .IF eax == 1
                                        .ELSEIF eax == 2
                                                INVOKE CreateLetter,ohandle,Letter_Size2,Letter_Start_Position,ADDR
M_Letter_20,color
                                        .ENDIF
                                .ELSEIF eax == 'E'
                        movzx eax,Font_Group
                                        .IF eax == 1
                                        .ELSEIF eax == 2
                                                INVOKE CreateLetter,ohandle,Letter_Size2,Letter_Start_Position,ADDR
E_Letter_20,color
                                        .ENDIF
                                .ELSEIF eax == 'V'
                        movzx eax,Font_Group
                                        .IF eax == 1
                                        .ELSEIF eax == 2
                                                INVOKE CreateLetter,ohandle,Letter_Size2,Letter_Start_Position,ADDR
V_Letter_20,color
                                        .ENDIF
                                .ENDIF

                                popad
                                ret
WriteLetter ENDP
WriteBigString PROC,
                                oHandle                 :HANDLE,
                                String                  :PTR WORD,
                                L                               :DWORD,
                                Font_Group              :WORD,
                                Array_Font_Color:PTR DWORD,
                                Start_Position  :COORD,
                                Font_Step               :DWORD

                                LOCAL   Letter_Start_Position   :COORD,
                                        Char                                            :WORD,
                                        Color                                           :DWORD
                                pushad
                                movzx eax,Start_Position.X
                                mov Letter_Start_Position.X,ax
                                movzx eax,Start_Position.Y
                                mov Letter_Start_Position.Y,ax
                                mov edx,String
                                mov edi,Array_Font_Color
                                mov ecx,0
                                .WHILE ecx<L
                                        movzx eax,WORD PTR [edx]
                                        mov char,ax
                                        mov eax,DWORD PTR [edi]
                                        mov Color,eax
                                        INVOKE WriteLetter,ohandle,Char,Letter_Start_Position,Font_Group,Color
                                        .IF Font_Group == 1
                                                movzx eax,Letter_Size1.x
                                        .ELSEIF Font_Group == 2
                                                movzx eax,Letter_Size2.x
```

```
                                        .ELSEIF Font_Group == 3
                                                movzx eax,Letter_Size3.x
                                        .ENDIF
                                        add eax,Font_Step
                                        add Letter_Start_Position.X,ax
                                        add edx,TYPE WORD
                                        add edi,TYPE DWORD
                                        INC ecx
                                .ENDW
                                popad
                                ret
WriteBigString ENDP
END
```

- Sadržaj fajla *fontemplate3x7.inc*

```
;=================================================================================================================================
;---------------------------------------------------------------------------------------------------------------------------------
;---------------------------------------------- fontemplate3x5.inc -----------------------------------------------------------
;---------------------------------------------------------------------------------------------------------------------------------
;------------------------------------------------------ Authors: -----------------------------------------------------------------
;---------------------------------------------- Turkmanovic Haris 516/2014 ---------------------------------------------------
;---------------------------------------------- Vukoje David 541/2014 --------------------------------------------------------
;=================================================================================================================================
;Description:
;       This include file contain font templates which size is 3 screen point width and 5 screen point hight
;       This include file is create to storage tempaltes for digit.This digit is used with points but can be used by other
text
;       In this include file you can import template for other font which size is 3x5
;=================================================================================================================================

;This part of code contain default font information like font color and font backround
Font_Block              = 219
Font_Color              = Red
Font_Backround  = 80h

Point                   TEXTEQU         <<Font_Block,Font_Color>>
None                    TEXTEQU         <<0,Font_Backround>>

.data
Letter_Size1    COORD <3,5>

;This is template for SPACE char
Delete_font10   _CHAR_INFO None,None,None
Delete_font11   _CHAR_INFO None,None,None
Delete_font12   _CHAR_INFO None,None,None
Delete_font13   _CHAR_INFO None,None,None
Delete_font14   _CHAR_INFO None,None,None

;This is template for Digit 1
Number_One  _CHAR_INFO None,None,Point
Number_One1  _CHAR_INFO None,none,Point
Number_One2  _CHAR_INFO none,None,Point
Number_One3  _CHAR_INFO None,None,Point
Number_One4  _CHAR_INFO None,None,Point

;This is template for Digit 2
Number_Two  _CHAR_INFO Point,Point,Point
Number_Two1  _CHAR_INFO None,None,Point
Number_Two2  _CHAR_INFO Point,Point,Point
Number_Two3  _CHAR_INFO point,None,None
Number_Two4  _CHAR_INFO Point,Point,Point

;This is template for Digit 3
Number_Three  _CHAR_INFO Point,Point,Point
Number_Three1  _CHAR_INFO None,None,Point
Number_Three2  _CHAR_INFO Point,Point,Point
Number_Three3  _CHAR_INFO none,none,point
Number_Three4  _CHAR_INFO Point,Point,Point

;This is template for Digit 4
Number_Four  _CHAR_INFO Point,none,Point
Number_FOur1  _CHAR_INFO point,None,Point
Number_Four2  _CHAR_INFO Point,Point,Point
```

```
Number_Four3  _CHAR_INFO none,none,point
Number_Four4  _CHAR_INFO none,none,Point

;This is template for Digit 5
Number_Five  _CHAR_INFO Point,Point,Point
Number_Five1  _CHAR_INFO Point,None,None
Number_Five2  _CHAR_INFO Point,Point,Point
Number_Five3  _CHAR_INFO None,None,Point
Number_Five4  _CHAR_INFO Point,Point,Point

;This is template for Digit 6
Number_Six  _CHAR_INFO Point,Point,Point
Number_Six1  _CHAR_INFO Point,None,None
Number_Six2  _CHAR_INFO Point,Point,Point
Number_Six3  _CHAR_INFO Point,None,Point
Number_Six4  _CHAR_INFO Point,Point,Point

;This is template for Digit 7
Number_Seven  _CHAR_INFO Point,Point,Point
Number_Seven1  _CHAR_INFO none,none,Point
Number_Seven2  _CHAR_INFO none,none,Point
Number_Seven3  _CHAR_INFO none,none,Point
Number_Seven4  _CHAR_INFO none,none,Point

;This is template for Digit 8
Number_Eight  _CHAR_INFO Point,Point,Point
Number_Eight1  _CHAR_INFO Point,None,Point
Number_Eight2  _CHAR_INFO Point,point,point
Number_Eight3  _CHAR_INFO Point,None,Point
Number_Eight4  _CHAR_INFO Point,point,Point

;This is template for Digit 9
Number_Nine  _CHAR_INFO Point,Point,Point
Number_Nine1  _CHAR_INFO Point,None,Point
Number_Nine2  _CHAR_INFO Point,point,point
Number_Nine3  _CHAR_INFO none,None,Point
Number_Nine4  _CHAR_INFO none,none,Point

;This is template for Digit 0
Number_Zero  _CHAR_INFO Point,Point,Point
Number_Zero1  _CHAR_INFO Point,None,Point
Number_Zero2  _CHAR_INFO Point,none,point
Number_Zero3  _CHAR_INFO Point,None,Point
Number_Zero4  _CHAR_INFO Point,point,Point
```

- Sadržaj fajla *fontemplate5x7.inc*

```
;=================================================================================================================================
;---------------------------------------------------------------------------------------------------------------------------------
;------------------------------------------------ fontemplate5x7.inc -----------------------------------------------
;---------------------------------------------------------------------------------------------------------------------------------
;------------------------------------------------------ Authors: -----------------------------------------------------
;---------------------------------------------- Turkmanovic Haris 516/2014 ---------------------------------------------
;---------------------------------------------- Vukoje David 541/2014 --------------------------------------------------
;=================================================================================================================================
;Description:
;       This include file contain font templates which size is 5 screen point width and 7 screen point hight
;       This include file is create to storage templates for Letter.This letter is used in welcomeScreen
;       In this include file you can import template for other font which size is 5x7
;       Here are the fonts that are used but can be added afterwards
;=================================================================================================================================

;This part of code contain default font information like font color and font backround

Font_Block              = 219
Font_Color              = Red
Font_Backround   = 80h

Point                   TEXTEQU <<Font_Block,Font_Color>>
None                    TEXTEQU <<0,Font_Backround>>

.data
```

```
Letter_Size2 COORD <5,7>

;This is template for SPACE char
Delete_Font20 _CHAR_INFO None,None,None,None,None
Delete_Font21 _CHAR_INFO None,None,None,None,None
Delete_Font22 _CHAR_INFO None,None,None,None,None
Delete_Font23 _CHAR_INFO None,None,None,None,None
Delete_Font24 _CHAR_INFO None,None,None,None,None
Delete_Font25 _CHAR_INFO None,None,None,None,None
Delete_Font26 _CHAR_INFO None,None,None,None,None

;This is template for Letter A
A_Letter_20 _CHAR_INFO Point,Point,Point,Point,Point
A_Letter_21 _CHAR_INFO Point,None,None,None,Point
A_Letter_22 _CHAR_INFO Point,None,None,None,Point
A_Letter_23 _CHAR_INFO Point,Point,Point,Point,Point
A_Letter_24 _CHAR_INFO Point,None,None,None,Point
A_Letter_25 _CHAR_INFO Point,None,None,None,Point
A_Letter_26 _CHAR_INFO Point,None,None,None,Point

;This is template for Letter R
R_Letter_20 _CHAR_INFO Point,Point,Point,None,None
R_Letter_21 _CHAR_INFO Point,None,None,Point,None
R_Letter_22 _CHAR_INFO Point,None,None,None,Point
R_Letter_23 _CHAR_INFO Point,None,None,Point,None
R_Letter_24 _CHAR_INFO Point,Point,Point,None,None
R_Letter_25 _CHAR_INFO Point,None,None,Point,None
R_Letter_26 _CHAR_INFO Point,None,None,None,Point

;This is template for Letter K
K_Letter_20 _CHAR_INFO Point,None,None,None,Point
K_Letter_21 _CHAR_INFO Point,None,None,Point,None
K_Letter_22 _CHAR_INFO Point,None,Point,None,None
K_Letter_23 _CHAR_INFO Point,Point,None,None,None
K_Letter_24 _CHAR_INFO Point,None,Point,None,None
K_Letter_25 _CHAR_INFO Point,None,None,Point,None
K_Letter_26 _CHAR_INFO Point,None,None,None,Point

;This is template for Letter N
N_Letter_20 _CHAR_INFO Point,None,None,None,Point
N_Letter_21 _CHAR_INFO Point,None,None,None,Point
N_Letter_22 _CHAR_INFO Point,Point,None,None,Point
N_Letter_23 _CHAR_INFO Point,None,Point,None,Point
N_Letter_24 _CHAR_INFO Point,None,None,Point,Point
N_Letter_25 _CHAR_INFO Point,None,None,None,Point
N_Letter_26 _CHAR_INFO Point,None,None,None,Point

;This is template for Letter O
O_Letter_20 _CHAR_INFO Point,Point,Point,Point,Point
O_Letter_21 _CHAR_INFO Point,None,None,None,Point
O_Letter_22 _CHAR_INFO Point,None,None,None,Point
O_Letter_23 _CHAR_INFO Point,None,None,None,Point
O_Letter_24 _CHAR_INFO Point,None,None,None,Point
O_Letter_25 _CHAR_INFO Point,None,None,None,Point
O_Letter_26 _CHAR_INFO Point,Point,Point,Point,Point

;This is template for Letter I
I_Letter_20 _CHAR_INFO None,None,Point,None,None
I_Letter_21 _CHAR_INFO None,None,Point,None,None
I_Letter_22 _CHAR_INFO None,None,Point,None,None
I_Letter_23 _CHAR_INFO None,None,Point,None,None
I_Letter_24 _CHAR_INFO None,None,Point,None,None
I_Letter_25 _CHAR_INFO None,None,Point,None,None
I_Letter_26 _CHAR_INFO None,None,Point,None,None

;This is template for Letter D
D_Letter_20 _CHAR_INFO Point,Point,Point,None,None
D_Letter_21 _CHAR_INFO Point,None,None,Point,None
D_Letter_22 _CHAR_INFO Point,None,None,None,Point
D_Letter_23 _CHAR_INFO Point,None,None,None,Point
D_Letter_24 _CHAR_INFO Point,None,None,None,Point
D_Letter_25 _CHAR_INFO Point,None,None,Point,None
D_Letter_26 _CHAR_INFO Point,Point,Point,None,None

;This is template for Letter G
```

```
G_Letter_20 _CHAR_INFO None, None, Point,Point, None
G_Letter_21 _CHAR_INFO None, Point, None, None,Point
G_Letter_22 _CHAR_INFO Point, None, None, None,Point
G_Letter_23 _CHAR_INFO Point, None, None, None, None
G_Letter_24 _CHAR_INFO Point, None, None,Point,Point
G_Letter_25 _CHAR_INFO None ,Point, None, None,Point
G_Letter_26 _CHAR_INFO None , None, Point,Point,None

;This is template for Letter M
M_Letter_20 _CHAR_INFO Point, None, None, None,Point
M_Letter_21 _CHAR_INFO Point,Point, None,Point,Point
M_Letter_22 _CHAR_INFO Point, None,Point, None,Point
M_Letter_23 _CHAR_INFO Point, None, None, None,Point
M_Letter_24 _CHAR_INFO Point, None, None, None,Point
M_Letter_25 _CHAR_INFO Point, None, None, None,Point
M_Letter_26 _CHAR_INFO Point, None, None, None,Point

;This is template for Letter E
E_Letter_20 _CHAR_INFO Point,Point,Point,Point,Point
E_Letter_21 _CHAR_INFO Point, None, None, None, None
E_Letter_22 _CHAR_INFO Point, None, None, None, None
E_Letter_23 _CHAR_INFO Point,Point,Point,Point, None
E_Letter_24 _CHAR_INFO Point, None, None, None, None
E_Letter_25 _CHAR_INFO Point, None, None, None, None
E_Letter_26 _CHAR_INFO Point,Point,Point,Point,Point

;This is template for Letter V
V_Letter_20 _CHAR_INFO Point, None, None, None,Point
V_Letter_21 _CHAR_INFO Point, None, None, None,Point
V_Letter_22 _CHAR_INFO Point, None, None, None,Point
V_Letter_23 _CHAR_INFO  None,Point, None,Point, None
V_Letter_24 _CHAR_INFO  None,Point, None,Point, None
V_Letter_25 _CHAR_INFO  None, None,Point, None, None
V_Letter_26 _CHAR_INFO  None, None,Point, None, None
```

- Sadržaj fajla *fontemplate9x13.inc*

```
;=================================================================================================================================
;---------------------------------------------------------------------------------------------------------------------------------
;------------------------------------------------ fontemplate9x13.inc ------------------------------------------------------------
;---------------------------------------------------------------------------------------------------------------------------------
;------------------------------------------------------ Authors: -----------------------------------------------------------------
;---------------------------------------------- Turkmanovic Haris 516/2014 -------------------------------------------------------
;---------------------------------------------- Vukoje David 541/2014 ------------------------------------------------------------
;=================================================================================================================================
;Description:
;       This include file contain font templates which size is 9 screen point width and 13 screen point hight
;       This include file is create special for draw ETF logo
;       In this include file you need to import template for other font which size is 9x13
;=================================================================================================================================

;This part of code contain default font information like font color and font backround

Font_Block              = 219
Font_Color              = Red
Font_Backround  = 80h

Point           TEXTEQU <<Font_Block,Font_Color>>
None            TEXTEQU <<0,Font_Backround>>
.data
Letter_Size3 COORD <9,13>

ETF_Logo_CHAR_INFO Point,Point,Point,Point,Point,Point,Point,Point,Point
ETF_logo1       _CHAR_INFO Point, None, None, None, None, None, None,Point
ETF_logo2   _CHAR_INFO Point, None,Point,Point,Point,Point, None,Point
ETF_logo3       _CHAR_INFO Point, None, None, None,Point, None, None, None,Point
ETF_logo4       _CHAR_INFO Point, None, None,Point,Point,Point, None, None,Point
ETF_logo5       _CHAR_INFO Point, None,Point, None,Point, None,Point, None,Point
ETF_logo6       _CHAR_INFO Point,Point,Point,Point,Point,Point,Point,Point,Point
ETF_logo7       _CHAR_INFO Point, None,Point, None,Point, None,Point, None,Point
ETF_logo8       _CHAR_INFO Point, None, None,Point,Point,Point, None, None,Point
ETF_logo9       _CHAR_INFO Point, None, None, None,Point, None, None, None,Point
ETF_logo10      _CHAR_INFO Point, None,Point,Point,Point,Point,Point, None,Point
```

```
ETF_logo11      _CHAR_INFO Point, None, None, None, None, None, None, None,Point
ETF_logo12      _CHAR_INFO Point,Point,Point,Point,Point,Point,Point,Point,Point
```

- Sadržaj fajla *structure.inc*

```
;=================================================================================================================================
;---------------------------------------------------------------------------------------------------------------------------------
;---------------------------------------------------- structure.inc ----------------------------------------------------------
;---------------------------------------------------------------------------------------------------------------------------------
;---------------------------------------------------- Authors: ---------------------------------------------------------------
;-------------------------------------------- Turkmanovic Haris 516/2014 -----------------------------------------------------
;---------------------------------------------- Vukoje David 541/2014 --------------------------------------------------------
;=================================================================================================================================
;Description:
;       Include file contain data and properties which is important for game.In this file is defined CELL structure which is
basic
; part of the game.Also here is defined BALL_ structure which represent moving ball in game.Here are also data which is
imortant
; for game window and game itself. Here we can set how much points is Assigned to row , row color, ball start position , ....
;       Thanks to this file we can change the performance of the game itself
;=================================================================================================================================

;-----------------------------------------
;-------------Graphic part----------------
CELL STRUCT
        ActivateBYTE    1
        Value           DWORD   0
        Color           DWORD   0
        HitCountWORD    0
        AREA_Position SMALL_RECT <0,0,0,0> ; Position of cell in console. This is set by PROCEDURE which use this STRUCTURE
        FALL_Flag       DWORD 0 ;This flag mark Falling Block
        TIME            DWORD 0;
CELL ENDS

BALL_ STRUCT
        CurrentPos      COORD <>
        PrevPos         COORD <>
        Active          BYTE  1
BALL_ ENDS


.data
;-----------------------------------------------------
;-------------Welcome screen Window data--------------
;-----------------------------------------------------
WelcomeScreenBufferSize                         COORD           <WelcomeScreen_Width     ,WelcomeScreen_Hight>
WelcomeScreenWindowSize                         SMALL_RECT      <0,0,WelcomeScreen_Width-1,WelcomeScreen_Hight-1>



;-------------GameWindow data--------------

GameBufferSize                          COORD                                   <GameWindow_width,GameWindow_Height>
GameWindowSize                          SMALL_RECT                              <0,0,GameWindow_width-1,GameWindow_height-1>
AppTitle                                BYTE                                    AppName
Cursor_Info                                     CONSOLE_CURSOR_INFO             <100,0>

;-----------------------------------------
;-------------Graphic data----------------

StartLine                               _CHAR_INFO      GameWindow_width
                                DUP(<223,StartLineColor>)
Blocks_Collor_Array             DWORD           BLUE, GREEN, RED, YELLOW, MAGENTA, GRAY
;!!! Blocks_Cell_Array And Pad_Cell NEED TO BE ADJACENT beacouse sometimes PROCEDURE look at this part of memory as one PART
Blocks_Cell_Array               CELL            Start_Blocks_Array_Size  DUP(<>)
Pad                                             CELL
        <1,0,Pad_Color,0,<Pad_X_Start_Position,Pad_Y_Start_Position,Pad_X_Start_Position+Pad_width-
1,Pad_Y_Start_Position+Pad_Height-1>>
Blocks_Points_Array     DWORD           50, 80, 90, 120, 100, 60
Ball                                    BALL_           <<0,0>,<0,0>>

;-----------------------------------------
;---------------GAME data-----------------
```

```
Blocks_Hit_Array        WORD                    1,       1,  1,   1,   1,  2
Blocks_Fall_Array               DWORD                   0,       0,  0,   0,   0,  1
Blocks_Matrix_size              COORD                   <Number_of_cell_in_Row,Number_Of_Colors>
Points_Start_Position   COORD                   <10,1>
Points_Color_Array              DWORD                   10                              DUP(Points_Color)
Life_Start_Position             COORD                   <80,1>
Life_Color_Array        DWORD                   10                              DUP(Life_Color)
GameInfoBackround               _CHAR_INFO              GameInfoArea    DUP(<0,80h>)
GameInfoSize                    COORD                   <GameWindow_width,StartLinePosition>
GameInfoA                               SMALL_RECT              <0,0,GameWindow_width,StartLinePosition>
GameInfoStartPosition   COORD                   <0,0>
Game_Points                             DWORD                   0
Game_Lifes                              DWORD                   Start_Life_Number
GameOverString                  WORD                    'G','A','M','E',DELETE_CODE,'O','V','E','R'
GameOverStringLengt             WORD                    9
GameOverStringCOlor             DWORD                   9 DUP(RED)
GameOverStringPosition  COORD                   <30,0>

;---------------Time data-----------------
Gray_Row_Time_Info              DWORD           Number_of_cell_in_Row                                   DUP(0)
Pad_Time_Info                   DWORD       0
Ball_Time_Info                  DWORD       0
```

- Sadržaj fajla *WelcomeScreen.asm*

```
INCLUDE Additionaldef.inc
INCLUDE WelcomeScreen.inc
INCLUDE Configuration.inc
INCLUDE Function.inc
INCLUDE Structure.inc
INCLUDE Font.inc


.code

;This is procedure which create WelcomeScreen.
WriteWelcomeScreen PROC,
                oHandle                 :HANDLE,
                StartColor              :DWORD
                LOCAL temp:DWORD,ArrPTR:DWORD
.data
WSBackround                             _CHAR_INFO              WSBackroundAreaSize     DUP(<0,WSBackroundColor>)
WSBackroundSize                 COORD                   <WelcomeScreen_Width,WelcomeScreen_Hight>
WSBackroundStartPosition COORD                  <0,0>
WSBackroundArea                 SMALL_RECT              <0,0,WelcomeScreen_Width-1,WelcomeScreen_Hight-1>
WSTemp                                  DWORD                   0
WSString1                               BYTE                    "U N I V E R Z I T E T   U   B E O G R A D U",0
WSString2                               BYTE                    "E L E K T R O T E H N I C K I   F A K U L T E T",0
WSString3                               BYTE                    "K A T E D R A   Z A   E L E K T R O N I K U",0
WSString4                               BYTE                    "I g r i c u   r e a l i z o v a l i :",0
WSString5                               BYTE                    "T U R K M A N O V I C   H A R I S ",0
WSString6                               BYTE                    "V U K O J E   D A V I D ",0
WSString1Size                   DWORD                   0
WSString2Size                   DWORD                   0
WSString3Size                   DWORD                   0
WSString4Size                   DWORD                   0
WSString5Size                   DWORD                   0
WSString6Size                   DWORD                   0
WSString1Attribute              WORD                    Blue
WSString2Attribute              WORD                    Red
WSString3Attribute              WORD                    Red
WSString4Attribute              WORD                    Blue
WSString5Attribute              WORD                    Red
WSString6Attribute              WORD                    Red
WSString1StartPosition  COORD                   <StartTextAreaPositionX+2,StartTextAreaPositionY+2>
WSString2StartPosition  COORD                   <StartTextAreaPositionX,StartTextAreaPositionY+7>
```

```
WSString3StartPosition   COORD                   <StartTextAreaPositionX+2,StartTextAreaPositionY+9>
WSString4StartPosition   COORD                   <StartTextAreaPositionX+1,WelcomeScreen_Hight-5>
WSString5StartPosition   COORD                   <StartTextAreaPositionX+1,WelcomeScreen_Hight-3>
WSString6StartPosition   COORD                   <StartTextAreaPositionX+7,WelcomeScreen_Hight-1>
WSLine                                  _CHAR_INFO              100 DUP(<223,WSLineColor>)
WSLineSize                              COORD                   <0,1>
WSLineStartPosition              COORD                  <0,0>
WSLineArea1                             SMALL_RECT
<StartTextAreaPositionX,StartTextAreaPositionY,0,StartTextAreaPositionY+1>
WSLineArea2                             SMALL_RECT
<StartTextAreaPositionX,StartTextAreaPositionY+4,0,StartTextAreaPositionY+5>
WSLineArea3                             SMALL_RECT
<StartTextAreaPositionX,StartTextAreaPositionY+11,0,StartTextAreaPositionY+12>
MaxStringSize                    DWORD                  0
WSETFLogoStartPosition  COORD           <0,0>
WSETFLogoColor                   DWORD                  Red
WSArkanoidString        WORD                    'A','R','K','A','N','O','I','D'
WSArkanoidStringSize     DWORD                  8
WSArkanoidColor                  DWORD                  Blue,Red,Green,Yellow,Blue,Red,Green,Yellow
WSArkanoidStartPosition COORD                   <0,0>
WSArkanoidXindent                DWORD                  3
WSArkanoidYindent                DWORD                  4


.code

                pushad
                INVOKE SetConsoleCursorInfo,ohandle,ADDR Cursor_Info
                INVOKE SetStartWindow, oHandle, WelcomeScreenBufferSize,ADDR WelcomeScreenWindowSize, ADDR AppTitle
                INVOKE WriteConsoleOutputA,      oHandle,ADDR WSBackround,       WSBackroundSize,WSBackroundStartPosition,ADDR
WSBackroundArea

                mov eax,StartTextAreaPositionY
                sub eax,1
                mov WSETFLogoStartPosition.Y,ax
                mov eax,StartTextAreaPositionX
                sub eax,11
                mov WSETFLogoStartPosition.X,ax
                add eax,WSArkanoidXindent
                mov WSArkanoidStartPosition.X,ax
                movzx eax,WSLineArea3.Bottom
                add eax,WSArkanoidYindent
                mov WSArkanoidStartPosition.Y,ax

                INVOKE WriteLetter,      ohandle, ETF_LOGO_CODE,WSETFLogoStartPosition,3 ,WSETFLogoColor

                mov eax,WSBackroundColor
                or  ax,WSString1Attribute
                mov WSString1Attribute,ax
                mov eax,WSBackroundColor
                or  ax,WSString2Attribute
                mov WSString2Attribute,ax
                mov eax,WSBackroundColor
                or  ax,WSString3Attribute
                mov WSString3Attribute,ax
                mov eax,WSBackroundColor
                or  ax,WSString4Attribute
                mov WSString4Attribute,ax
                mov eax,WSBackroundColor
                or  ax,WSString5Attribute
                mov WSString5Attribute,ax
                mov eax,WSBackroundColor
                or  ax,WSString6Attribute
                mov WSString6Attribute,ax


                mov eax,LENGTHOF WSString1
                mov WSString1Size,eax
                .IF eax>MaxStringSize
                        mov MaxStringSize,eax
                .ENDIF
                mov eax,LENGTHOF WSString2
                mov WSString2Size,eax
                .IF eax>MaxStringSize
                        mov MaxStringSize,eax
```

```
		.ENDIF
		mov eax,LENGTHOF WSString3
		mov WSString3Size,eax
		.IF eax>MaxStringSize
			mov MaxStringSize,eax
		.ENDIF
		mov eax,MaxStringSize
		mov WSLineSize.X,ax
		movzx ecx,WsLineArea1.Left
		add eax,ecx
		mov WSLineArea1.Right,ax
		mov WSLineArea2.Right,ax
		mov WSLineArea3.Right,ax

		mov eax,LENGTHOF WSString4
		mov WSString4Size,eax
		mov eax,LENGTHOF WSString5
		mov WSString5Size,eax
		mov eax,LENGTHOF WSString6
		mov WSString6Size,eax



		INVOKE WriteConsoleOutputA,	oHandle,ADDR WSLine,	WSLineSize,WSLineStartPosition,ADDR WSLineArea1
		INVOKE WriteConsoleOutputA,	oHandle,ADDR WSLine,	WSLineSize,WSLineStartPosition,ADDR WSLineArea2
		INVOKE WriteConsoleOutputA,	oHandle,ADDR WSLine,	WSLineSize,WSLineStartPosition,ADDR WSLineArea3

		INVOKE SetConsoleCursorPosition,ohandle,WSString1StartPosition
		mov edx,OFFSET WSString1
		INVOKE WriteString
		INVOKE FillConsoleOutputAttribute, ohandle, WSString1Attribute	,WSString1Size,WSString1StartPosition,ADDR
WSTemp

		INVOKE SetConsoleCursorPosition,ohandle,WSString2StartPosition
		mov edx,OFFSET WSString2
		INVOKE WriteString
		INVOKE FillConsoleOutputAttribute, ohandle, WSString2Attribute	,WSString2Size,WSString2StartPosition,ADDR
WSTemp

		INVOKE SetConsoleCursorPosition,ohandle,WSString3StartPosition
		mov edx,OFFSET WSString3
		INVOKE WriteString
		INVOKE FillConsoleOutputAttribute, ohandle, WSString3Attribute	,WSString3Size,WSString3StartPosition,ADDR
WSTemp

		INVOKE SetConsoleCursorPosition,ohandle,WSString4StartPosition
		mov edx,OFFSET WSString4
		INVOKE WriteString
		INVOKE FillConsoleOutputAttribute, ohandle, WSString4Attribute	,WSString4Size,WSString4StartPosition,ADDR
WSTemp

		INVOKE SetConsoleCursorPosition,ohandle,WSString5StartPosition
		mov edx,OFFSET WSString5
		INVOKE WriteString
		INVOKE FillConsoleOutputAttribute, ohandle, WSString5Attribute	,WSString5Size,WSString5StartPosition,ADDR
WSTemp

		INVOKE SetConsoleCursorPosition,ohandle,WSString6StartPosition
		mov edx,OFFSET WSString6
		INVOKE WriteString
		INVOKE FillConsoleOutputAttribute, ohandle, WSString6Attribute	,WSString6Size,WSString6StartPosition,ADDR
WSTemp

		mov edx,WS_Time_Duration
		.WHILE !SIGN?
			mov eax,WSFontRefreshTime
			INVOKE Delay
			Invoke WriteBigString,ohandle,ADDR WSArkanoidString ,WSArkanoidStringSize,2,ADDR WSArkanoidColor
	,WSArkanoidStartPosition,2

			push edx
			mov edi,OFFSET WSArkanoidColor
			mov ArrPTR,edi
			add edi,28
```

```
                        mov esi,edi
                        sub esi,TYPE DWORD
                        mov eax,DWORD PTR [edi]
                        mov temp,eax

                        .WHILE edi!=ArrPTR
                                mov eax,DWORD PTR[esi]
                                mov DWORD PTR[edi],eax
                                SUB esi,TYPE DWORD
                                sub edi,TYPE DWORD
                        .ENDW
                        mov eax,temp
                        mov DWORD PTR [edi],eax
                        pop edx
                        dec edx
                        CMP edx,0
                .ENDW
                popad
                Ret
WriteWelcomeScreen ENDP
END
```

- Sadržaj fajla *WelcomeScreen.inc*

```
;=====================================================================================================================================
;-------------------------------------------------------------------------------------------------------------------------------------
;------------------------------------------------ WelcomeScreen.inc ---------------------------------------------------
;-------------------------------------------------------------------------------------------------------------------------------------
;------------------------------------------------------ Authors: -----------------------------------------------------
;---------------------------------------------- Turkmanovic Haris 516/2014 -----------------------------------------------
;---------------------------------------------- Vukoje David 541/2014 ----------------------------------------------
;=====================================================================================================================================
;Description:
;       This .inc file is important for WelcomeScreen.asm . Here is located some properties of welcome screen.
;=====================================================================================================================================


StartTextAreaPositionX   = 15
StartTextAreaPositionY   = 3
WSBackroundColor         = 80h
WSLineColor              = 84h
WSFontRefreshTime        = 100
WSBackroundAreaSize                     EQU WelcomeScreen_Width*WelcomeScreen_Hight

   WriteWelcomeScreen PROTO,
                oHandle                 :HANDLE,
        StartColor              :DWORD
```