Elektrotehnički fakultet Univerziteta u Beogradu

RAČUNARSKA ELEKTRONIKA

-PROJEKAT br. 19-

Miloš Marković, 349/2014
Sanja Mihajlović, 123/2014

Postavka zadatka

Cilj je realizovati igricu Tetris. U igrici su na raspolaganju karakteristični oblici, od kojih se jedan, nasumično izabran, pojavljuje na sredini gornje ivice ‘‘terena‘‘ na početku igrice, i svaki put kada prethodna komponenta dodirne donju ivicu ekrana ili već naslagane komponente. Komponenta se pomera ka donjoj ivici terena unapred izabranom konstantnom brzinom, i može da se ubrza kursorom na dole na tastaturi. Leva i desna strelica pomeraju komponentu levo i desno, respektivno, dok je strelica na gore rotira u smeru suprotnom od smera kazaljke na satu, za 90 stepeni.

Potrebno je obezbediti iscrtavanje sledeće komponente koja će se pojaviti na terenu, u prozoru sa strane.

Takođe, potrebno je ispisivati broj poena osvojenih u toku igrice. Svaki put kada se jedan red u potpunosti popuni, on se briše, a broj poena se uvećava za 100. Igrica se prekida kada se pritisne taster ESC ili kada komponenta dodirne gornju ivicu prozora.

REALIZACIJA PROGRAMA

U ovom odeljku će biti objašnjena suština programskog koda implementiranog u cilju realizacije Tetrisa.

U promenljivama TEREN, SCORE i BUDUCI sadržani su izgledi redom, terena za igranje Tetrisa, prozora u koji će se upisivati trenutni rezultat i prozora u kome se pojavljuje komponenta koja je sledeća na redu. Izgledi ovih prozora se iscrtavaju funkcijom DRAW, a analogno je realizovana i funkcija DELETE, koja omogućava njihovo brisanje.

Promenljiva GOBJ sadži koordinate gameObject-a, odnosno, oblika koji se trenutno kreće po terenu. Unutar promenljive TABELA se pamti gObj koji je došao do donje ivice prozora ili već postojeće komponente unutar terena, te ne može dalje da se kreće, i to kao ‚1‘ na onim koordinatama na kojima se on nalazi. Prazan teren se obeležava ‚0‘. Funkcija koja realizuje pamćenje komponente u tabelu je TRANSFER2TABLE. Pomoću funkcije TABLE2GRAPHICS se u konzoli iscrtava trenutno stanje na terenu (‚0‘ kao , ‘ a 1 kao popunjen kvadratić).

U svakom trenutku kretanja po terenu, GOBJ se iscrtava funkcijom DRAWSHAPE, i, pre iscrtavanja na novim koordinatama, briše pomoću funkcije DELETESHAPE.

Pri pokretanju programa, ispisuje se naziv igre i korisniku se daje mogućnost da odabere njeno započinjanje (Enter) ili izlazak (Esc). Deo koda koji ovo realizuje se nalazi u okviru labela *introLabel* i *w1*.

Zatim se pomoću funkcije *InitTable* resetuju vrednosti promenljive TABELA (postavljaju na nulu), što predstavlja postavljanje praznog terena. Nakon toga, setuju se koordinate centra game Object-a koji će se stvoriti.
Nakon toga, iscrtavaju se teren, prozor za budući oblik koji će postati game Object i prozor sa rezultatom.
Zatim se nasumično generiše prvi oblik koji se potom upisuje u tačke game Object-a pomoću funkcije INITSHAPE.
Važno je napomenuti da su tačke tog objekta, sadržane u promenljivoj GOBJX2, tako raspoređene da je objekat duplo veći od finalnog, što je učinjeno zarad lakše implementacije rotacije.
Zatim se pomoću funkcije POINTS TO COORDS, pomoću definisanog centra komponente, njene tačke prevode u koordinate na terenu.
Nakon toga se nasumično generiše budući gameObject i iscrtava u okviru prozora BUDUCI. Konačno, prelazi se na labelu *start.* U okviru ove labele, u konzoli se iscrtava stvoreni game Object pomoću funkcije DRAW SHAPE, i prelazi se na narednu labelu – *waiting.*

Unutar nje, u zavisnosti od inputa na tastaturi (leva/desna/donja/gornja strelica, esc, nije pritisnut nijedan taster) se skače na odgovarajuću labelu i izvršava tražena operacija.

Pri pritiskanju leve/desne/donje strelice, prvo se pomoću odgovarajućih funkcija za detekciju kolizije sa ivicama terena (COLISION LEFT/RIGHT/BOTTOM) proverava da li bi objekat izašao van terena po izvršenju zadate komande. Ukoliko bi, odlazi se na labelu *noKey,* što praktično znači da se tražena operacija neće izvršiti. Sama labela će biti naknadno objašnjena.
Ukoliko bi objekat ostao unutar terena, poziva se funkcija MOVE LEFT/RIGHT/DOWN.
Unutar ove funkcije se generiše privremeni game Object, koji se postavlja na koordinate koje bi originalni game Object imao nakon izvršenja tražene operacije. Zatim se on prosleđuje funkciji COLISION HIT, koja proverava da li se prosleđeni game Object nekim delom preklapa sa već postojećim komponentama na terenu, tako što gleda da li se na toj koordinati unutar tabele nalazi 0 ili 1. Ukoliko je 0, to znači da nema ničega, a ukoliko je 1, na toj koordinati već postoji neka komponenta, dakle, došlo bi do preklapanja.
Ukoliko nema preklapanja, privremeni objekat postaje stalni, a ukoliko dolazi do preklapanja, game Object se ne pomera.
Nakon toga se odlazi na labelu *KeyFound,* u okviru koje se iscrtava novi položaj game Object-a u konzoli. Sledi labela *NoKey.*
Unutar ove labele se proverava da li je isteklo vreme definisano promenljivom Tick (predefinisano vreme između dva pomeranja objekta na dole). Ukoliko nije, odlazi se na labelu *waiting,* koja je već opisana, a ako jeste, odlazi se na labelu *timer.*
U okviru labele *timer* poziva se funkcija BOTTOM STOP, koja proverava da li je objekat došao do donje ivice terena ili, pak, do već postojeće komponente na terenu. Ako nije, odlazi se na labelu *skip,* unutar koje se game Object briše sa postojeće pozicije, spušta se, i iscrtava se na novoj poziciji, u okviru labele *start.* Dalja procedura je opisana.
Ukoliko je pak, gameObject došao do prepreke, odlazi se na labelu *lbl,* unutar koje se poziva funkcija GAME END, u okviru koje se proverava da li je game Object došao do vrha terena. Ukoliko jeste, odlazi se na labelu *kraj*, a ukoliko nije, odlazi se na labelu *newShape*.
U okviru labele *newShape*, game Object koji više ne može da se pomera, se upisuje u promenljivu TABELA pomoću funkcije TRANSFER2TABLE, zatim se poziva funkcija CLEAR LINES koja briše linije koje su u potpunosti popunjene, ako postoje. Na osnovu vrednosti koju ona upiše u akumulator se određuje broj osvojenih poena, koji se potom upisuje u prozorčić SCORE. Zatim se iscrtava novo stanje na terenu pomoću funkcije TABLE2GRAPHICS, i resetuje pozicija centra budućeg game Object-a.

Na osnovu prethodno generisanog oblika u okviru prozora budući se pravi novi game Object, a unutar prozora BUDUCI se generiše novi, nasumično odabrani oblik, i ciklus se ponavlja odlaskom na labelu *start.*
U okviru labele *kraj* se odlazi na uvodni prozor, i zatim se odlazi na labelu *w2*, koja omogućava korisniku da ponovo započne igru pritiskom tastera Enter, ili izađe iz nje pritiskom tastera Esc.

Ukoliko je pritisnuta gornja strelica, poziva se funkcija ROTL, koja rotira objekat u smeru suprotnom od smera kazaljke na satu, ukoliko je to moguće.
Naime, spomenuta funkcija sadrži proveru da li je moguće izvršiti rotaciju, čija je suština slična kao i prethodno opisane provere – generiše se privremeni, rotirani objekat, i proverava se da li su sve njegove koordinate unutar terena i da li se preklapaju sa nekom od postojećih komponenti na terenu, pozivanjem funkcija OUT OF BOUNDS i COLISION HIT, respektivno. Ukoliko provere prođu uspešno, rotacija se izvršava, ukoliko ne, objekat ostaje u stanju u kom je.
Bitno je napomenuti princip izvršavanja rotacije – tačke game Object-a se množe matricom rotacije (pošto je u pitanju 2D sistem, i ugao od 90 stepeni, svodi se na to da je xRotirano=-y, a Yrotirano=x). Time je izbegnuto hard-codovanje novog položaja oblika, i omogućena potpuna proizvoljnost zadavanja oblika.

Pritiskom na taster ESC igra se okončava, i odlazi se na uvodni prozor (*introLabel*), a dalja procedura je opisana.

Pregled funkcija koje su implementirane zarad realizacije Tetrisa:

InitShape – kopira tačke objekta 1 u tačke objekta 2 (npr, nekog od prizvoljno izabranih oblika u tačke game Object-a)

PointsToCoords – na osnovu zadatih tačaka oblika i njegovog centra, određuje koordinate game Object-a na terenu

ROTL - rotira objekat u smeru suprotnom od smera kazaljke na satu, za 90 stepeni, ukoliko je moguće izvršiti rotaciju. Ako nije, objekat ostaje nepromenjen

DrawShape – iscrtava objekat u konzoli

DeleteShape – briše objekat iz konzole
Draw – služi za iscrtavanje svih prozora korišćenih u igri

Delete – pomoću nje može da se izbriše željeni prozor

ClearLines – briše popunjene linije, ako postoje, i u akumulator upisuje broj izbrisanih linija

Table2Graphics – sadržaj tabele popunjene 0 i 1 iscrtava u konzoli (gde nula predstavlja prazan prostor na terenu, a 1 označava postojanje komponente na tim koordinatama)

Transfer2Table – upisuje game Object koji višđe ne može da se pomera u tabelu (praktično, postaje deo terena)

ColisionLeft – ispituje da li bi se objekat sudario sa levom ivicom terena ukoliko bi se pomerio na levo
ColisionRight -ispituje da li bi se objekat sudario sa desnom ivicom terena ukoliko bi se pomerio na desno
ColisionBottom -ispituje da li bi se objekat sudario sa donjom ivicom terena ukoliko bi se pomerio na dole
ColisionHit – ispituje da li bi se objekat preklopio sa nekom već postojećom komponentom na terenu ukoliko bi se pomerio na zadati način
OutOfBounds – ispituje da li bi objekat ostao unutar terena nakon pomeranja (korosti se unutar ROTL)

MoveLeft – pomera objekat za jedno mesto ulevo, ukoliko nema prepreke u vidu neke već postojeće komponente. Ako ima, objekat ostaje u nepromenjenom stanju

MoveRight - pomera objekat za jedno mesto udesno, ukoliko nema prepreke u vidu neke već postojeće komponente. Ako ima, objekat ostaje u nepromenjenom stanju
MoveDown - pomera objekat za jedno mesto na dole, ukoliko nema prepreke u vidu neke već postojeće komponente. Ako ima, objekat ostaje u nepromenjenom stanju

BottomStop – proverava da li je objekat došao do donje ivice terena ili se ispod njega nalazi neka ucrtana komponenta na terenu.
GameEnd – proverava da li je game Object došao do gornje ivice terena

Funkcije iz Irvinove i Windows biblioteke koje su korišćene pri izradi projekta:

SetConsoleCursorPosition
WriteConsole
GetStdHandle
GetConsoleCursorInfo
SetConsoleCursorInfo
Clrscr
WriteString
RandomRange
WriteInt
call Delay
call ReadKey

PROJEKTNI KOD

---------- FAJL externs.inc početak----------

```
include Irvine32.inc

;-----------------------------SHAPE PROTOS---------------------------
InitShape PROTO STDCALL orgShp:PTR POINT, destShp:PTR POINT

PointsToCoords PROTO STDCALL points:PTR POINT, coords:PTR COORD, center:COORD

ROTL PROTO STDCALL points:PTR POINT, coords:PTR COORD, center:COORD, table:PTR BYTE

DrawShape PROTO STDCALL coords:PTR COORD, hnd:DWORD, poc:COORD

DeleteShape PROTO STDCALL coords:PTR COORD, hnd:DWORD, poc:COORD

;-----------------------------crtanje PROTOS---------------------------

Draw PROTO STDCALL bafer: PTR BYTE, pocKoord: COORD, hnd:DWORD, vrste:DWORD, kolone:DWORD

Delete PROTO STDCALL pocKoord: COORD, hnd:DWORD, vrste:DWORD, kolone:DWORD

;-----------------------------GameLogic PROTOS---------------------------

ClearLines PROTO STDCALL table:PTR BYTE, top:DWORD

Table2Graphics PROTO STDCALL table:PTR BYTE, graphT:PTR BYTE

Transfer2Table PROTO STDCALL coords:PTR COORD, table:PTR BYTE

ColisionLeft PROTO STDCALL coords: PTR COORD
ColisionRight PROTO STDCALL coords: PTR COORD
ColisionBottom PROTO STDCALL coords: PTR COORD
ColisionHit PROTO STDCALL table:PTR BYTE, coords:PTR COORD
OutOfBounds PROTO STDCALL coords:PTR COORD

MoveLeft PROTO STDCALL objCent: PTR COORD, coords:PTR COORD, points:PTR POINT, table:PTR BYTE
```

```
MoveRight PROTO STDCALL objCent: PTR COORD, coords:PTR COORD, points:PTR POINT, table:PTR BYTE
MoveDown PROTO STDCALL objCent: PTR COORD, coords:PTR COORD, points:PTR POINT, table:PTR BYTE

BottomStop PROTO STDCALL table:PTR BYTE, coords:PTR COORD
GameEnd PROTO STDCALL coords:PTR COORD


InitTable PROTO STDCALL table:PTR BYTE

POINT STRUCT
        x SWORD ?
        y SWORD ?
POINT ENDS

;----------golobalne konstante----------
;KEY CODES IN AH
UP_ARROW EQU  48h
DOWN_ARROW EQU 50h
LEFT_ARROW EQU 4Bh
RIGHT_ARROW EQU 4Dh
ESC_KEY EQU 1h
ENTER_KEY EQU 1Ch
;
TEREN_WIDTH EQU 10
TEREN_HEIGHT EQU 22



;----------makroi----------


MoveCoord MACRO source, dest

ENDM

AddCoord MACRO source, dest

ENDM

SubCoord MACRO source, dest

ENDM
```

---------- FAJL externs.inc kraj----------

---------- FAJL crtanje.asm početak----------

```
include externs.inc

.code

Draw PROC USES AX ECX EBX,
        bafer: PTR BYTE, pocKoord: COORD, hnd:DWORD, vrste:DWORD, kolone:DWORD
        mov EBX, bafer
        mov ECX, vrste
loop1:
        push ECX
        INVOKE SetConsoleCursorPosition, hnd, pocKoord
        INVOKE WriteConsole, hnd, EBX, kolone, 0,0
        add EBX, kolone
        mov AX, pocKoord.y
        add AX, 1
        mov pocKoord.y, AX
        pop ECX
        loop loop1
        ret
```

```
Draw ENDP

Delete PROC USES AX ECX EBX,
	pocKoord: COORD, hnd:DWORD, vrste:DWORD, kolone:DWORD

	mov ECX, kolone
loop2:

push ' '
loop loop2
	mov EBX, ESP
	mov ECX, vrste
loop1:
	push ECX
	INVOKE SetConsoleCursorPosition, hnd, pocKoord
	INVOKE WriteConsole, hnd, EBX, kolone, 0,0
	mov AX, pocKoord.y
	add AX, 1
	mov pocKoord.y, AX
	pop ECX
	loop loop1
	ret
Delete ENDP
end
```

---------- FAJL crtanje.asm kraj----------

---------- FAJL shape.asm početak----------

```
include externs.inc

.data
.code

;----------POMOCNI MAKROI----------
_rotl MACRO

ENDM
;kopira pointe
InitShape PROC USES EAX ESI EDI,
	orgShp:PTR POINT, destShp:PTR POINT

	mov ecx,4
	mov esi,orgShp
	mov edi,destShp
loop1:
	mov ax,(POINT PTR [esi]).x
	mov (POINT PTR [edi]).x,ax
	mov ax,(POINT PTR [esi]).y
	mov (POINT PTR [edi]).y,ax
	add esi,TYPE orgShp
	add edi,TYPE destShp
	loop loop1

	ret
InitShape ENDP

PointsToCoords PROC USES EAX ESI EDI,
		points:PTR POINT, coords:PTR COORD, center:COORD

	mov ecx,4
	mov esi,points
	mov edi,coords
loop2:
	mov ax,(POINT PTR [esi]).x
	sar ax,1
	add ax,center.x
```

```
		mov (COORD PTR [edi]).x,ax

		mov ax,(POINT PTR [esi]).y
		sar ax,1
		neg ax
		add ax,center.y
		mov (COORD PTR [edi]).y,ax

		add esi,TYPE points
		add edi,TYPE coords
		loop loop2

		ret
PointsToCoords ENDP

ROTL PROC USES EAX EBX ECX ESI EDI,
			points:PTR POINT, coords:PTR COORD, center:COORD, table:PTR BYTE
			LOCAL temp[4]:POINT, ctemp[4]:COORD

		mov esi,points
		lea edi, temp
		mov ecx,4
loop1:
		mov bx,(POINT PTR [esi]).x
		mov ax,(POINT PTR [esi]).y
		neg ax
		mov (POINT PTR [edi]).x,ax
		mov (POINT PTR [edi]).y,bx
		add esi,TYPE POINT
		add edi,TYPE POINT
		loop loop1

;-------------------Da li moze da se rotira---------------
		INVOKE PointsToCoords, ADDR temp, ADDR ctemp, center
		INVOKE ColisionHit, table, ADDR ctemp
		cmp eax,0
		jnz hit
		INVOKE OutOfBounds, ADDR ctemp
		cmp eax,0
		jnz hit


		INVOKE PointsToCoords, ADDR temp, coords, center

		lea esi,temp
		mov edi, points
		mov ecx,4
loop2:
		mov ax,(POINT PTR [esi]).x
		mov (POINT PTR [edi]).x,ax
		mov ax,(POINT PTR [esi]).y
		mov (POINT PTR [edi]).y,ax;moguca greska COORD UMESTO POINT
;
		add esi,TYPE POINT
		add edi,TYPE COORDS
		loop loop2
hit:
		ret
ROTL ENDP

DrawShape PROC USES EAX EBX ECX EDI,
			coords:PTR COORD, hnd:DWORD, poc:COORD
			LOCAL temp[4]:COORD

		mov ecx,4
```

```
        mov ebx,coords
        lea edi,temp
loop2:
        mov ax,(COORD PTR [ebx]).x
        add ax,poc.x
        mov (COORD PTR [edi]).x,ax
        mov ax,(COORD PTR [ebx]).y
        add ax,poc.y
        mov (COORD PTR [edi]).y,ax
        add ebx,TYPE COORD
        add edi,TYPE COORD
        loop loop2

        mov ecx,4
        lea ebx,temp
loop1:
        push ecx
        mov eax,0
        mov ax,(COORD PTR [ebx]).y
        sub ax,poc.y
        cmp ax,0
        jl skip
        INVOKE SetConsoleCursorPosition, hnd, COORD PTR [ebx]
        mov al, 254
        call WriteChar
skip:
        add ebx,type COORD
        pop ecx
        loop loop1
        ret
DrawShape ENDP

DeleteShape PROC USES EAX EBX ECX EDI,
                coords:PTR COORD, hnd:DWORD,poc:COORD
                LOCAL temp[4]:COORD

        mov ecx,4
        mov ebx,coords
        lea edi,temp
loop2:
        mov ax,(COORD PTR [ebx]).x
        add ax,poc.x
        mov (COORD PTR [edi]).x,ax
        mov ax,(COORD PTR [ebx]).y
        add ax,poc.y
        mov (COORD PTR [edi]).y,ax
        add ebx,TYPE COORD
        add edi,TYPE COORD
        loop loop2

        mov ecx,4
        lea ebx,temp
loop1:
        push ecx
        mov ax,(COORD PTR [ebx]).y
        sub ax,poc.y
        cmp ax,0
        jl skip
        INVOKE SetConsoleCursorPosition, hnd, COORD PTR [ebx]

        mov al, ' '
        call WriteChar
skip:
        add ebx,type COORD
        pop ecx
```

```
		loop loop1
		ret
DeleteShape ENDP

end
```
---------- FAJL shape.asm kraj----------
---------- FAJL gameLogic.asm početak----------
```
include externs.inc

.code
;vraca da li je red popunjen u EAX
_checkLine PROC USES ECX ESI,
		tmp:PTR BYTE

		mov ecx,TEREN_WIDTH
		mov esi,tmp
loop1:
		mov al,BYTE PTR [esi]
		cmp al,0
		jz izlaz
		inc esi
		loop loop1
izlaz:
		ret
_checkLine ENDP

_clearLine PROC USES ECX ESI,
		tmp:PTR BYTE

		mov ecx,TEREN_WIDTH
		mov esi,tmp
loop1:
		mov (BYTE PTR [esi]),0
		inc esi
		loop loop1
		ret
_clearLine ENDP

;Slaze tablicu kada se red popuni, Vraca broj obrisanih redova u EAX, top predstavlja optimizaciju 0 pregleda sve, 5 ne pregleda prvih 5
redova
ClearLines PROC USES EBX ECX ESI,
		table:PTR BYTE, top:DWORD

		mov eax,0
		mov al,TEREN_WIDTH
		mul BYTE PTR top
		mov esi,table
		add esi,eax
		mov ecx,TEREN_HEIGHT
		sub ecx,top
		mov ebx,0
loop1:
		INVOKE _checkLine, esi
		cmp eax,0
		jz skip
		INVOKE _clearLine, esi
		mov eax, esi  ; koliko ima elemenata do vrha (table[0])
		sub eax, table
		CALL _sort
		add ebx,1
skip:
		add esi,TEREN_WIDTH*TYPE BYTE
		loop loop1
		mov eax,ebx
```

```
		ret
ClearLines ENDP

;uzima esi i broj ciklusa u eax
_sort PROC USES EAX ECX ESI EDI,

		mov ecx,eax
		dec esi
		mov edi,esi
		add edi, TEREN_WIDTH
		loop1:
			mov al,(BYTE PTR[esi])
			mov (BYTE PTR[edi]),al
			dec edi
			dec esi
		loop loop1
		ret
_sort ENDP

;prebacuje sadrzaj  tabele u baffer karaktera
Table2Graphics PROC USES EAX ECX ESI EDI,
		table:PTR BYTE, graphT:PTR BYTE

		mov ecx,TEREN_WIDTH*TEREN_HEIGHT
		mov esi,table
		mov edi,graphT
loop1:
		mov al,(BYTE PTR[esi])
		cmp al,0
		jz skok
		mov (BYTE PTR[edi]),254
		jmp uslov
skok:
		mov (BYTE PTR[edi]),' '
uslov:
		inc esi
		inc edi
		loop loop1

		ret
Table2Graphics ENDP

;upisuje gObj u tabelu
Transfer2Table PROC USES EAX EBX ECX EDI ESI,
		coords:PTR COORD, table:PTR BYTE

		mov ecx,4
		mov esi,coords

loop1:
		mov eax,TEREN_WIDTH
		mov ebx,0
		mov bx,(COORD PTR [esi]).y
		add bx,2;zbog 220
		mul bx
		add ax,(COORD PTR [esi]).x
		mov ebx,eax
		cmp eax,220
		jge skip
		add ebx,table
		mov (BYTE PTR [ebx]), 1
skip:
		add esi,TYPE COORD
		loop loop1
```

```
        ret
Transfer2Table ENDP

ColisionLeft PROC USES ECX ESI,
        coords: PTR COORD

        mov eax,0
        mov ecx,4
        mov esi,coords
loop1:
        ;levo
        mov ax, (COORD PTR [esi]).x
        sub ax,1
        cmp ax,-1
        jle hit

        add esi,TYPE COORD
        loop loop1

        mov eax,1
        ret
hit:
        mov eax,0
        ret

ColisionLeft ENDP

ColisionRight PROC USES ECX ESI,
        coords: PTR COORD
        LOCAL desno:WORD

        mov eax,0
        mov ecx,4
        mov esi,coords
loop1:
        ;desno
        mov ax, (COORD PTR [esi]).x
        add ax,1
        cmp ax,TEREN_WIDTH
        jge hit

        add esi,TYPE COORD
        loop loop1

        mov eax,1
        ret
hit:
        mov eax,0
        ret

ColisionRight ENDP

ColisionBottom PROC USES ECX ESI,
        coords: PTR COORD

        mov eax,0
        mov ecx,4
        mov esi,coords
loop1:
        ;dole
        mov ax, (COORD PTR [esi]).y
        add ax,1
        cmp ax,TEREN_HEIGHT-2
        jge hit
```

```
		add esi,TYPE COORD
		loop loop1

		mov eax,1
		ret
hit:
		mov eax,0
		ret

ColisionBottom ENDP

;vraca sudar u eax!=0
ColisionHit PROC USES EBX ECX ESI EDI,
		table:PTR BYTE, coords:PTR COORD

		mov edi, coords
		mov ecx, 4

loop2:
		mov esi, table
		mov eax,0
		mov eax,TEREN_WIDTH
		mov ebx,0
		mov bx,(COORD PTR [edi]).y
		add bx,2;zbog 220
		mul bl
		add ax,(COORD PTR [edi]).x
		add esi,eax
		mov al,BYTE PTR [esi]
		cmp al,0
		jnz hit
		add edi,TYPE COORD
		loop loop2

		mov eax,0
		ret
hit:
		mov eax,1
		ret
ColisionHit ENDP

OutOfBounds PROC USES ECX ESI,
		coords:PTR COORD

		mov esi,coords
		mov ecx,4
loop1:
		mov ax, (COORD PTR [esi]).x
		cmp ax, -1
		jle hit
		mov ax, (COORD PTR [esi]).x
		cmp ax, TEREN_WIDTH
		jge hit
		mov ax, (COORD PTR [esi]).y
		cmp ax, TEREN_HEIGHT-2
		jge hit
		add esi,TYPE COORD
		loop loop1

		mov eax,0
		ret
hit:
		mov eax,1
		ret
```

```
OutOfBounds ENDP

MoveLeft PROC USES EAX EBX ECX ESI,
	objCent: PTR COORD, coords:PTR COORD, points:PTR POINT, table:PTR BYTE
	LOCAL temp[4]:COORD

	mov esi,objCent
	mov ax,(COORD PTR [esi]).x
	sub ax,1
	mov (COORD PTR [esi]).x,ax

	INVOKE PointsToCoords, points, ADDR temp, COORD PTR [esi]

	INVOKE ColisionHit, table, ADDR temp
	cmp eax,0
	jnz hit

	INVOKE PointsToCoords, points, coords, COORD PTR [esi]

	ret
hit:
	mov esi,objCent
	mov ax,(COORD PTR [esi]).x
	add ax,1
	mov (COORD PTR [esi]).x,ax
	ret

MoveLeft ENDP

MoveRight PROC USES EAX EBX ECX ESI,
	objCent: PTR COORD, coords:PTR COORD, points:PTR POINT, table:PTR BYTE
	LOCAL temp[4]:COORD

	mov esi,objCent
	mov ax,(COORD PTR [esi]).x
	add ax,1
	mov (COORD PTR [esi]).x,ax

	INVOKE PointsToCoords, points, ADDR temp, COORD PTR [esi]

	INVOKE ColisionHit, table, ADDR temp
	cmp eax,0
	jnz hit

	INVOKE PointsToCoords, points, coords, COORD PTR objCent

	ret
hit:
	mov esi,objCent
	mov ax,(COORD PTR [esi]).x
	sub ax,1
	mov (COORD PTR [esi]).x,ax
	ret

MoveRight ENDP

MoveDown PROC USES EBX ECX ESI,
	objCent: PTR COORD, coords:PTR COORD, points:PTR POINT, table:PTR BYTE
	LOCAL temp[4]:COORD

	mov esi,objCent
	mov ax,(COORD PTR [esi]).y
	add ax,1
	mov (COORD PTR [esi]).y,ax
```

```
        INVOKE PointsToCoords, points, ADDR temp, COORD PTR [esi]

        INVOKE ColisionHit, table, ADDR temp
        cmp eax,0
        jnz hit

        INVOKE PointsToCoords, points, coords, COORD PTR objCent

        mov eax,0
        ret
hit:
        mov esi,objCent
        mov ax,(COORD PTR [esi]).y
        sub ax,1
        mov (COORD PTR [esi]).y,ax
        mov eax,1
        ret

MoveDown ENDP

BottomStop PROC USES EBX ECX ESI EDI,
        table:PTR BYTE, coords:PTR COORD

        mov edi, coords
        mov ecx, 4
loop2:
        mov eax,0
        mov ax,(COORD PTR [edi]).y
        add eax,1
        cmp eax,TEREN_HEIGHT-2
        jz          hit

        mov esi, table
        mov eax,0
        mov eax,TEREN_WIDTH
        mov ebx,0
        mov bx,(COORD PTR [edi]).y
        add bx,3;zbog 220+jedna nize test
        mul bl
        add ax,(COORD PTR [edi]).x
        add esi,eax
        mov al,BYTE PTR [esi]
        cmp al,0
        jnz hit
        add edi,TYPE COORD
        loop loop2

        mov eax,0
        ret
hit:
        mov eax,1
        ret

BottomStop ENDP

GameEnd PROC USES ECX ESI,
        coords:PTR COORD

                mov esi,coords
        mov ecx,4
loop1:
        mov ax, (COORD PTR [esi]).y
        cmp ax, -2
        jz hit
        add esi,TYPE COORD
```

```
        loop loop1

        mov eax,0
        ret
hit:
        mov eax,1
        ret
GameEnd ENDP

InitTable PROC USES EAX ECX ESI,
        table:PTR BYTE

        mov ecx,TEREN_WIDTH*TEREN_HEIGHT
        mov esi,table
loop1:
        mov eax,0
        mov BYTE PTR[esi],al
        inc esi
loop loop1

        ret
InitTable ENDP

end
```

-- FAJL gameLogic.asm kraj--

------------------------------------ FAJL main.asm početak--

```
include externs.inc

.data

tabela BYTE 220 DUP(0)
tabelaG BYTE 220 DUP(?)

pocKoord COORD <11,2>
pocKoord1 COORD <12,3>

prompt BYTE "PRESS ENTER TO START/ESC TO QUIT...",0
prompt2 BYTE "PRESS ANY KEY TO CONTINUE...",0
promptK2 COORD <1,25>
introK COORD <1,1>
introK2 COORD <1,7>
intro   BYTE  178,177,178,176,254,  32,178,254,254,178,  32,178,176,178,178,254,  32,178,176,176,  32,  32,254,254,177,  32,  32,
32,178,254,176,178
                BYTE  32, 32,178, 32, 32, 32,176, 32, 32, 32, 32, 32, 32,254, 32, 32, 32,177, 32, 32,178, 32, 32,178, 32, 32, 32,178,
32, 32, 32, 32
                BYTE  32, 32,176, 32, 32, 32,176,177,254, 32, 32, 32, 32,254, 32, 32, 32,177,254,177, 32, 32, 32,176, 32, 32, 32,
32,176,178,254, 32
                BYTE  32, 32,178, 32, 32, 32,178, 32, 32, 32, 32, 32, 32,178, 32, 32, 32,176, 32,254, 32, 32, 32,177, 32, 32, 32, 32,
32, 32, 32,254
                BYTE  32,  32,254,  32,  32,  32,177,177,254,178,  32,  32,  32,178,  32,  32,  32,254,  32,  32,177,  32,178,176,178,  32,
32,178,176,254,178, 32
                ;5X32

teren   BYTE 201,205,205,205,205,205,205,205,205,205,205,187; 22x12
                BYTE 186,' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',186
                BYTE 186,' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',186
                BYTE 186,' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',186
                BYTE 186,' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',186
                BYTE 186,' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',186
                BYTE 186,' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',186
                BYTE 186,' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',186
```

```
                    BYTE 186,' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',186
                    BYTE 186,' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',186
                    BYTE 186,' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',186
                    BYTE 186,' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',186
                    BYTE 186,' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',186
                    BYTE 186,' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',186
                    BYTE 186,' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',186
                    BYTE 186,' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',186
                    BYTE 186,' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',186
                    BYTE 186,' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',186
                    BYTE 186,' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',186
                    BYTE 186,' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',186
                    BYTE 186,' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',186
                    BYTE 200,205,205,205,205,205,205,205,205,205,205,188

scoreK1 COORD <25,2>
scoreK2 COORD <26,5>
rez DWORD ?
score BYTE 201,205,205,205,205,205,205,205,205,205,205,187       ;8x12
          BYTE 186, 83, 67, 79, 82, 69, 58,' ',' ',' ',' ',186
          BYTE 186,' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',186
          BYTE 186,' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',186
          BYTE 186,' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',186
          BYTE 186,' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',186
          BYTE 186,' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',186
          BYTE 200,205,205,205,205,205,205,205,205,205,205,188

buduciK COORD <1,2>
buduci BYTE 201,205,205,205,205,205,205,187  ;8x8
          BYTE 186,' ',' ',' ',' ',' ',' ',186
          BYTE 186,' ',' ',' ',' ',' ',' ',186
          BYTE 186,' ',' ',' ',' ',' ',' ',186
          BYTE 186,' ',' ',' ',' ',' ',' ',186
          BYTE 186,' ',' ',' ',' ',' ',' ',186
          BYTE 186,' ',' ',' ',' ',' ',' ',186
          BYTE 200,205,205,205,205,205,205,188

Ishape POINT <-3,0>,<-1,0>,<1,0>,<3,0>
Zshape POINT <-1,-2>,<-1,0>,<1,0>,<1,2>
Sshape POINT <1,-2>,<1,0>,<-1,0>,<-1,2>
Lshape POINT <0,2>,<0,0>,<0,-2>,<2,-2>
Jshape POINT <0,2>,<0,0>,<0,-2>,<-2,-2>
Tshape POINT <-2,0>,<0,0>,<2,0>,<0,2>
Oshape POINT <-1,-1>,<-1,1>,<1,1>,<1,-1>


sADDR EQU OFFSET Ishape

gObjX2 POINT 4 DUP(<>)
gObj COORD 4 DUP(<>)
gObjCenter COORD <5,-2>

gObjBuduciX2 POINT 4 DUP(<>)
gObjBuduci COORD 4 DUP(<>)
gObjCenterB COORD <4,4>

consHandle DWORD ?
cInfo CONSOLE_CURSOR_INFO <>

ranNum DWORD ?



clear BYTE 500 DUP(' '),0
```

```
time DWORD 0
Tick=1000

.code
main proc
	INVOKE GetStdHandle, STD_OUTPUT_HANDLE
	mov consHandle, EAX

	INVOKE GetConsoleCursorInfo, consHandle,ADDR cInfo
	mov eax,0
	mov cInfo.bVisible,eax
	INVOKE SetConsoleCursorInfo, consHandle, ADDR cInfo



	;----------------------intro-------------------------
introLabel:
	call Clrscr
	INVOKE Draw, ADDR intro, introK, consHandle, 5, 32
	INVOKE SetConsoleCursorPosition, consHandle, introK2
	mov  edx,OFFSET prompt
   call WriteString
w1:
	mov eax,20                                      ;wait 20ms
	call Delay
	call ReadKey                              ; look for keyboard input

	cmp ah,ENTER_KEY
	jz newGame
	cmp ah,ESC_KEY
	jz izl
	jmp w1

	;----------------------Iscrtavanje terena----------------------------
newGame:

	INVOKE InitTable,ADDR tabela
	mov eax,5
	mov gObjCenter.x,ax
	mov eax,-2
	mov gObjCenter.y,ax

	call Clrscr
	INVOKE Draw, ADDR teren, pocKoord, consHandle, 22, 12
	INVOKE Draw, ADDR buduci, buduciK, consHandle, 8, 8

	;----------------------Inicijacija Scora---------------------------------
	INVOKE Draw, ADDR score, scoreK1, consHandle, 8, 12
	mov eax,0
	mov rez,eax
	INVOKE SetConsoleCursorPosition, consHandle, scoreK2
	mov eax,rez
	call WriteInt
	;----------------------Inicijalizacija prvog oblika-------------------------
	;call Randomize zabada funkcija
	mov eax,7;                        ;prvi shape
	call RandomRange ;
	sal eax,4
	add eax, sADDR
	mov  ranNum,eax
	INVOKE InitShape, ranNum, ADDR gObjX2
	INVOKE PointsToCoords, ADDR gObjX2, ADDR gObj, gObjCenter
```

```
	;----------------------Inicijalizacija buduceg oblika--------------------
	mov eax,7;			;prvi shape
	call RandomRange ;
	sal eax,4
	add eax, sADDR
	mov  ranNum,eax
	INVOKE InitShape, ranNum, ADDR gObjBuduciX2
	INVOKE PointsToCoords, ADDR gObjBuduciX2, ADDR gObjBuduci, gObjCenterB
	INVOKE DrawShape, ADDR gObjBuduci, consHandle, buduciK

	jmp start

	;INVOKE Draw, ADDR intro, pocKoord, consHandle, 5, 32
	;invoke Delete, pocKoord, consHandle, 22, 11
	;INVOKE Table2Graphics,ADDR tabela, ADDR tabelaG
	INVOKE Draw, ADDR tabelaG+20, pocKoord1, consHandle, 20, 10
	;INVOKE Transfer2Table,ADDR gObj,ADDR tabela
	;INVOKE ClearLines, ADDR tabela, 20

	INVOKE Table2Graphics,ADDR tabela, ADDR tabelaG
	INVOKE Draw, ADDR tabelaG+20, pocKoord1, consHandle, 20, 10

newShape:
	INVOKE Transfer2Table,ADDR gObj,ADDR tabela
	INVOKE ClearLines, ADDR tabela, 0
	;score++
	add eax,rez
	mov rez,eax
	INVOKE SetConsoleCursorPosition, consHandle, scoreK2
	mov eax,rez
	call WriteInt
		;crtanje tabele
	INVOKE Table2Graphics,ADDR tabela, ADDR tabelaG
	INVOKE Draw, ADDR tabelaG+20, pocKoord1, consHandle, 20, 10

	mov eax,5
	mov gObjCenter.x,ax
	mov eax,-2
	mov gObjCenter.y,ax

	INVOKE InitShape, ADDR gObjBuduciX2, ADDR gobjX2
	;----------------------Inicijalizacija buduceg oblika--------------------
	mov eax,7;			;prvi shape
	call RandomRange ;
	sal eax,4
	add eax, sADDR
	mov  ranNum,eax
	INVOKE DeleteSHAPE, ADDR gObjBuduci, consHandle, buduciK
	INVOKE InitShape, ranNum, ADDR gObjBuduciX2
	INVOKE PointsToCoords, ADDR gObjBuduciX2, ADDR gObjBuduci, gObjCenterB
	INVOKE DrawShape, ADDR gObjBuduci, consHandle, buduciK
	jmp start

timer:
	INVOKE BottomStop, ADDR tabela, ADDR gObj
	cmp eax,0
	jz skip
lbl:
	INVOKE GameEnd, ADDR gObj
	cmp eax,0
	jnz kraj
	jmp newShape
skip:
	;crtanje oblika
	INVOKE DeleteSHAPE, ADDR gObj, consHandle, pocKoord1
```

```
        mov ax,gObjCenter.y
        inc ax
        mov gObjCenter.y,ax
start:
        INVOKE PointsToCoords, ADDR gObjX2, ADDR gObj, gObjCenter
        INVOKE DrawShape, ADDR gObj, consHandle, pocKoord1

waiting:

;-----------------------input resovling----------------------
        mov eax,20                              ;wait 20ms
        call Delay
        call ReadKey                    ; look for keyboard input
   jz   NoKey                           ; no key pressed yet

        cmp ah,UP_ARROW
        jz up
        cmp ah,DOWN_ARROW
        jz down
        cmp ah,LEFT_ARROW
        jz left
        cmp ah,RIGHT_ARROW
        jz right
        cmp ah,ESC_KEY
        jz introLabel
        jmp NoKey                               ;wrong key go to nokey

up:
        INVOKE DeleteSHAPE, ADDR gObj, consHandle, pocKoord1
        INVOKE ROTL, ADDR gObjX2, ADDR gObj, gObjCenter,ADDR tabela
        jmp keyFound

down:
        ;check can move down
        INVOKE ColisionBottom, ADDR gObj
        jz NoKey

        INVOKE DeleteSHAPE, ADDR gObj, consHandle, pocKoord1
        INVOKE MoveDown, ADDR gObjCenter, ADDR gObj, ADDR gObjX2, ADDR tabela
        cmp eax,0
        jnz lbl
        jmp keyFound
left:
        ; check can move left
        INVOKE ColisionLeft, ADDR gObj
        jz NoKey

        INVOKE DeleteSHAPE, ADDR gObj, consHandle, pocKoord1
        INVOKE MoveLeft, ADDR gObjCenter, ADDR gObj, ADDR gObjX2, ADDR tabela
        jmp keyFound

right:
        ; check can move right
        INVOKE ColisionRight, ADDR gObj
        jz NoKey

        INVOKE DeleteSHAPE, ADDR gObj, consHandle, pocKoord1
        INVOKE MoveRight, ADDR gObjCenter, ADDR gObj, ADDR gObjX2, ADDR tabela
        jmp keyFound

        jmp NoKey                       ;wrong key
keyFound:
        INVOKE PointsToCoords, ADDR gObjX2, ADDR gObj, gObjCenter
        INVOKE DrawShape, ADDR gObj, consHandle, pocKoord1
```

```
NoKey:



;------------timeing logic------------------
            call GetTickCount
            mov ebx,eax
            sub eax,time
            cmp eax,Tick
            JB waiting
            mov time,ebx
            jmp timer

kraj:
            INVOKE SetConsoleCursorPosition, consHandle, promptK2
            mov  edx,OFFSET prompt2
    call WriteString
w2:
            mov eax,20                                  ;wait 20ms
            call Delay
            call ReadKey
            jz w2
            jmp introLabel

izl:
            invoke ExitProcess,0
main endp
end
```

---------- FAJL main.asm kraj----------