RETRO KUTAK

Lekcija peta

Dobrodošli na peti čas "Male škole programiranja na Commodore 64 u BASIC-u". Danas ćemo nastaviti sa upoznavanjem mogućnosti i naredbi C64 računara. Dajemo kao i obično neke savete i primere za ono što učimo.

Počnimo kao i obično sa savetima i objašnjenjima

Sprajtovi

Šta su sprajtovi? Jednostavan odgovor je da su to sličice koje su definisane od strane programera i koje možemo pomerati po ekranu bez obzira u kom smo grafičkom modu. Kod C64 postoje tzv. hardverski sprajtovi. To znači da kreiranje i upravljanje ovim sličicama podržava sam sistem računara što značajno olakšava njihovo definisanje i korišćenje. Mogu se kreirati i tzv.softverski sprajtovi što je značajnije teže jer programer mora voditi računa o gomilu stvari koje su mu već omogućene postojanjem hardverskih sprajtova i isti nisu tako efikasni. Zato su kod C64 hardverski sprajtovi najbitniji. Uglavnom se koriste kod video igara (naročito arkadnih). Problem je korišćenje sprajtova putem standardnog bejzik programa zbog njegove izuzetne sporosti, pa se uglavnom koriste u mašinskim programima. Takođe kad je standardni bejzik u pitanju koriste se isključivo u tekstualnom modu iz razloga što ne bi imalo smisla koristiti ih u grafičkom modu visoke ili multikolor rezolucije (o njima u sledećoj lekciji), jer kada bi se istovremeno iscrtavala grafika dobili bi smo jedan frejm u 5 sekundi (i to ako imamo sreću). Kao i kod grafike i zvuka nema odgovarajuće standardne bejzik C64 naredbe za njihovo definisanje, uključivanje. pokretanje itd. (što bi nam i te kako olakšalo posao ali šta je tu je) pa ćemo morati koristiti memorijsku mapu za ono što želimo. Ima ih ukupno 8. Možemo ih pokretati po horizontali i vertikali na ekranu (320x200 tačaka),duplo uvećati po X ili Y osi, detektovati da li su se sudarili međusobno ili sa pozadinom itd.

Kako ih defenišemo?

Sprajtovi su sličice koje se u visokoj rezoluciji crtaju sa 24x21 tačaka, a u multikolor rezoluciji 12x21 tačaka. Jedan sprajt zauzima ukupno 63 bajta. Isti se iscrtava u matrici 3 x 21 bajta- 24 tačaka za horizontalu (3 x 8 tačaka/bita kod visoke rezolucije u dve boje ili kod multikolora 3 x 4 tačaka (dva bita po tački) što daje 12 tačaka multikolor u četri boje ili ukupno 24 bita) i 21 tačaka za vertikalu. Kombinacijom tih bajtova i unosom njihove vrednosti u memorijsku mapu zaduženu za sprajtove dobijamo željenu sličicu. Oni mogu biti i animirani (uglavnom i jesu) brzim učitavanjem različitih vrednosti u memoriju sprajta što se da videti u svim video igricama ali to nije moguće uraditi u standardnom bejziku da bi bilo iskoristivo zbog, kako smo to već napomenuli, velike sporosti istog.

Visokorezolucijski sprajt

Kao što smo rekli ovaj sprajt je dimenzije 24x21 tačaka u dve boje (u stvari boja lica bit 1 plus providnost-boja pozadine bit 0)

Preporučujem da napravite na papiru ili računaru ovakav šablon:

	B I N A R N E V R E D N O S T I																**decimalne vrednosti**										
red	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			
1																											
2																											
3																											
4																											
5																											
6																											
7																											
8																											
9																											
10																											
11																											
12																											
13																											
14																											
15																											
16																											
17																											
18																											
19																											
20																											
21																											

Da ne bude zabune bitovi u bajtu se obeležavaju od 0 do 7 počev od desne prema levoj strani po težini. To su sa nulom ukupno 8 bita i nose vrednosti (bit/vrednost) 0/1, 1/2, 2/4, 3/8, 4/16, 5/32, 6/64, 7/128. Preporučujemo da kada nacrtate željenu sličicu uzmete digitron, prebacite na binarni deo otkucate 8 bita iz šablona i vidite koliko to decimalno vredi. Takođe možete napraviti u excel tabeli šablon za obračunavanje decimalnih vrednosti ili koristiti neki program skinut sa interneta. Naravno možete i sami da izračunate bez upotrebe digitrona i računara (ako imate vremena i volje preporučujemo ovo poslednje, dobro je za vežbu moždanih sinapsi). U gore datom šablonu bitovi su odvojeni mrežom zadebljano, normalno, zadebljano, tako da je lako znati koji bit pripada kom bajtu.

Nacrtajmo sada sportska kola/formulu 1 i pretvorimo ih u decimalnu vrednost po bajtovima. Koristićemo slovo Q kao logičku jedinicu, a prazno polje kao logičku nulu. Naravno na vama je da odaberete kako ćete to uraditi

	B I N A R N E V R E D N O S T I																**decimalne vrednosti**		
red	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			
1																									0	0	0
2																									0	0	0
3			Q	Q	Q	Q													Q	Q	Q	Q			60	0	60
4			Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q			63	255	252
5			Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q			63	255	252
6			Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q			63	255	252
7			Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q			63	255	252
8							Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q							3	255	192
9							Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q							3	255	192
10							Q	Q	Q	Q				Q	Q	Q	Q								3	195	192
11							Q	Q	Q	Q				Q	Q	Q	Q								3	195	192
12							Q	Q	Q	Q				Q	Q	Q	Q								3	195	192
13							Q	Q	Q	Q				Q	Q	Q	Q								3	195	192
14							Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q							3	255	192
15							Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q							3	255	192
16									Q	Q	Q	Q	Q	Q	Q	Q									0	255	0
17			Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q			63	255	252
18			Q	Q	Q	Q													Q	Q	Q	Q			60	0	60
19			Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q			63	255	252
20			Q	Q	Q	Q													Q	Q	Q	Q			60	0	60
21																									0	0	0

Nisam baš vešt u crtanju ali se nadam da ovaj crtež može da prođe kao formula.

Sada je vreme da ovaj sprajt unesemo u memoriju i prikažemo ga na ekranu i zato ćemo se poslužiti sledećim programom:

```
10 FOR I=832 TO 894
20 READ M
30 POKE I,M
40 NEXT I
```

U linijama od 10 do 40 smeštamo naš sprajt u memorijske adrese od 832 do 894 (63 bajta). Pogledajte u ovom tekstu naredbe READ i DATA

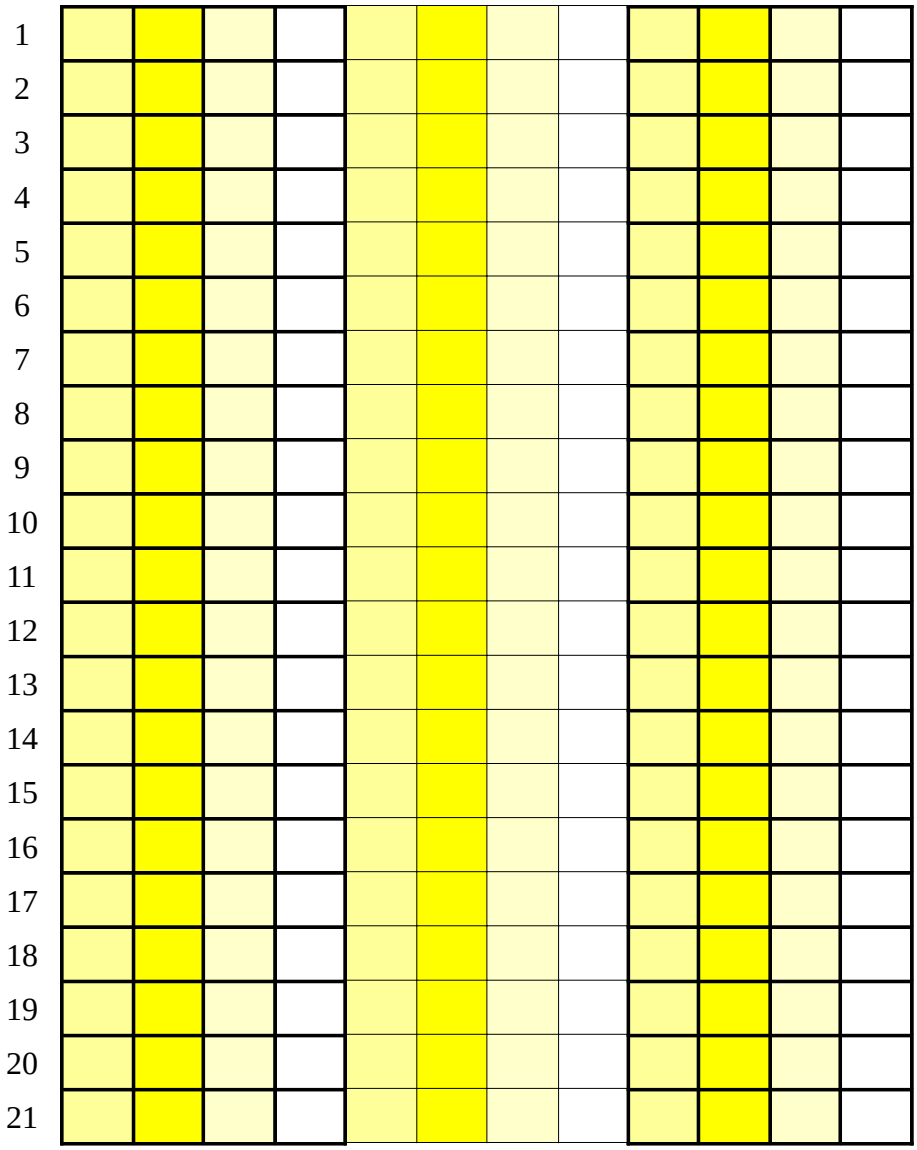
50 POKE 53269,1
Ovde preko adrese za kontrolu sprajta uključujemo prvi (svaki bit uključuje/isključuje jedan sprat od 0 do 7)
60 POKE 2040,13
Sprajtovi se smeštaju po blokovima od 64 bajta. U memorijsku lokaciju zaduženu za adresu prikaza prvog sprajta 2040 unosimo vrednost 13 (2040-prvi,... 2047-osmi sprajt). To znači da smo odredili memorijsku lokaciju za prikaz prvog sprajta adresu 832 (13*64) na kojoj smo učitali naš sprajt
70 POKE 53248,170:POKE 53264,0
Da bi smo za ovaj sprajt odredili horizontalni položaj na ekranu treba u adresu 53248 za X osu prvog sprajta uneti vrednost od 0 do 255 (53248-prvi,53250-drugi, ... 53262-osmi sprajt). Pošto je horizontala po kome se sprajt može kretati 320 tačaka fali nam još 65 tačaka. Konstruktori su to rešili dodeljivanjem još jednog bita u memorijskoj lokaciji 53264 za svaki od 8 sprajtova. Ako setujemo ovaj bit (53264,1 za prvi sprajt, svaki sprajt ima svoj bit od 0 do 7) koji nosi vrednost 256 i tu dodamo 255 iz osnovne X lokacije, onda je X kordinata istog povećana na 511 od kojih nam treba samo 320 (ako preteramo nećemo videti sprajt)
80 POKE 53249,70
Ovde dajemo vrednost Y kordinate vertikalnog položaja unosom vrednosti od 0 do 255 u memorijsku lokaciju 53249 za prvi sprajt (53249-prvi,53251-drugi... 53263-osmi sprajt). Pošto Y osa ima maksmalnu vrednost od 200 tačaka tu nema nikakvih problema. I ovde važi da ako preteramo sa vrednošću izačićemo sa ekrana
90 REM POKE 53277,1: POKE 53271,1
Ovaj programski red je neaktivan. Služi ukoliko želimo da dupliramo veličinu sprajta po X i/ili Y osi. Svaki sprajt ima svoj bit u ovim memorijskim adresama
100 POKE 53287,3
U adresu boje za prvi sprat unosimo vrednost za boju koju želimo (adrese se kreću od 53287-prvi do 53295-osmi sprajt)
110 END
Kraj programa
1000 DATA 0,0,0
1010 DATA 0,0,0
1020 DATA 60,0,60
1030 DATA 63,255,252
1040 DATA 63,255,252
1050 DATA 63,255,252
1060 DATA 63,255,252
1070 DATA 3,255,192
1080 DATA 3,255,192
1090 DATA 3,195,192
1100 DATA 3,195,192
1110 DATA 3,195,192
1120 DATA 3,195,192
1130 DATA 3,255,192
1140 DATA 3,255,192
1150 DATA 0,255,0
1160 DATA 63,255,252
1170 DATA 60,0,60
1180 DATA 63,255,252
1190 DATA 60,0,60
1200 DATA 0,0,0
Od linije 1000 do 1200 su podaci o izgledu našeg sprajta
Postoje još razne mogućnosti koje možemo uraditi sa sprajtovima (kao što je detekcija sudara dva sprajta ili sprajta sa pozadinom i td.). Videćemo ih u praktičnom radu

Multikolorni sprajt

Ovaj sprajt je dimenzije 12x21 tačaka u četri boje (boja lica, dodatna boja 1, dodatna boja 2 i providnost-boja pozadine)

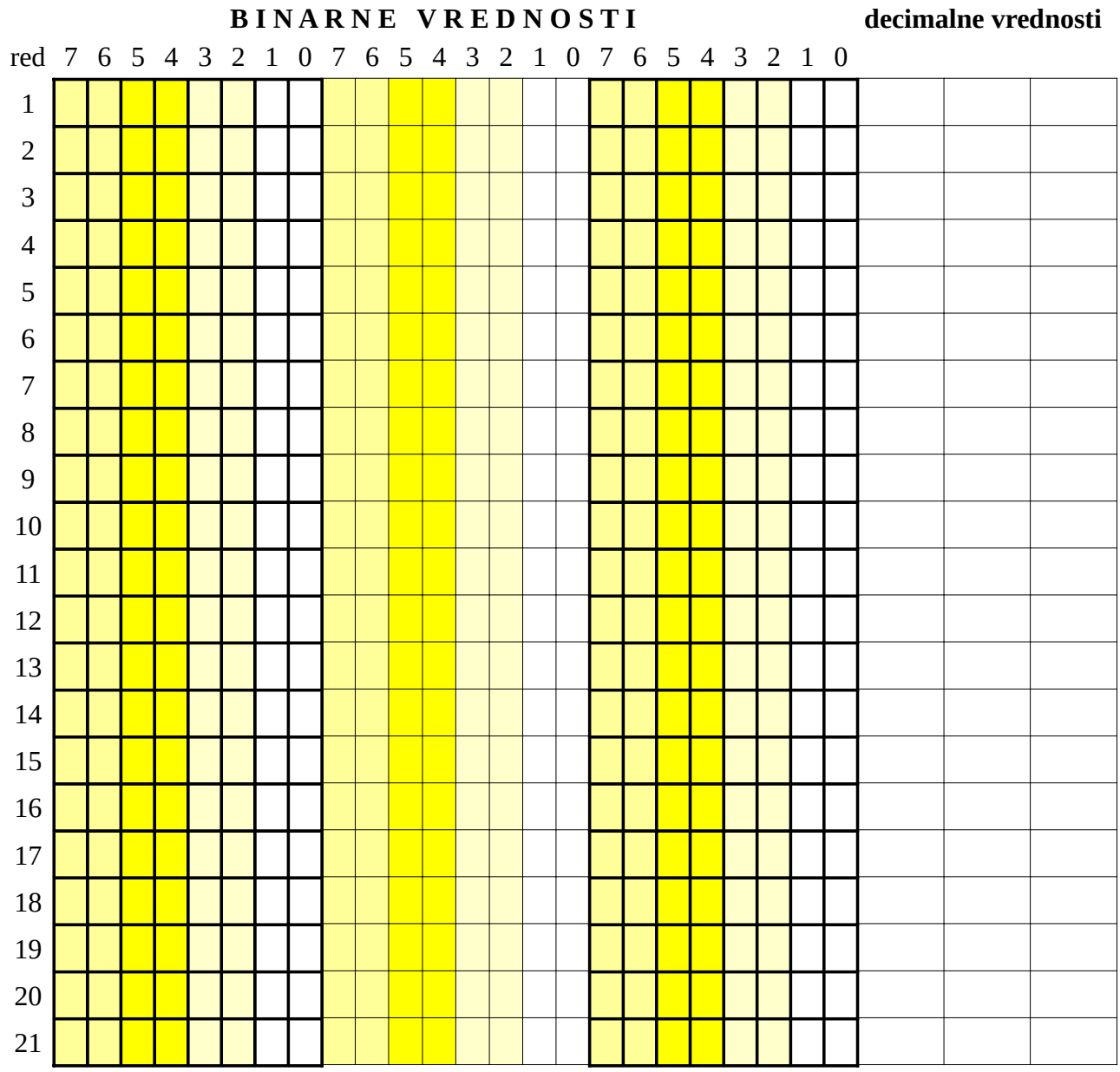
Što se tiče rada sa multikolornim sprajtom najlakše je ako napravimo dva šablona (glavni i pomoćni). Takođe radi lakšeg orentisanja redovi šablona su obojeni. Sve to ovako izgleda:

Glavni šablon:



Prvo možete primetiti da je šablon duplo manji po horizontali od šablona za sprajt visoke rezolucije. Razlog je što se odvajaju po dva bita za jednu tačku i ova dva bita daju 4 kombinacije (tri boje plus transparencija). To takođe znači da je multikolor tačka dva puta veća po horizontali od tačke visoke rezolucije.
U ovoj formi uneli bismo kako želimo da izgleda naš multikolorni sprajt i upisivali bi smo oznake za boje npr. P-plava, C-crvena, B-crna itd. Zatim bi smo dodelili kombinacije P-01, C-11, B-10. Četvrta kombinacija 00 je transparentna - vidi se pozadina.

Onda bi smo sve te kombinacije po bojama uz glavnog šablona uneli u pomoćni koji izgleda ovako:



Da vidimo primer kako bi to sve izgledalo:

C - CRVENA - 01
Z - ZELENA - 10
P - PLAVA - 11
" " - TRANSPARENTNA (BOJA KOJA JE ODREĐENA POZADINOM) - 00

1												
2		Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	
3		Z	P	P	P	P	P	P	P	P	Z	
4		Z	P	P	P	P	P	P	P	P	Z	
5		Z	P	C	C	C	C	C	C	P	Z	
6		Z	P	C	C	C	C	C	C	P	Z	
7		Z	P	C					C	P	Z	
8		Z	P	C					C	P	Z	
9		Z	P	C					C	P	Z	
10		Z	P	C					C	P	Z	
11		Z	P	C					C	P	Z	
12		Z	P	C					C	P	Z	
13		Z	P	C					C	P	Z	
14		Z	P	C					C	P	Z	
15		Z	P	C	C	C	C	C	C	P	Z	
16		Z	P	C	C	C	C	C	C	P	Z	
17		Z	P	P	P	P	P	P	P	P	Z	
18		Z	P	P	P	P	P	P	P	P	Z	
19		Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	
20												
21												

Sada ćemo zameniti ove oznake sa bitovima koje smo malopre dali za svaku boju C- crvena - 01, Z- zelena - 10, P- plava - 11, prazna mesta sa 00 i sve ovo uneti u pomoćnoj formi. Zatim ćemo tako dobijene kombinacije prevesti u decimalne brojeve

	B I N A R N E V R E D N O S T I																								**decimalne vrednosti**		
red	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0	42	170	168
3	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	47	255	248
4	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	47	255	248
5	0	0	1	0	1	1	0	1	0	1	0	1	0	1	0	1	0	1	1	1	1	0	0	0	45	85	120
6	0	0	1	0	1	1	0	1	0	1	0	1	0	1	0	1	0	1	1	1	1	0	0	0	45	85	120
7	0	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	45	0	120
8	0	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	45	0	120
9	0	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	45	0	120
10	0	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	45	0	120
11	0	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	45	0	120
12	0	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	45	0	120
13	0	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	45	0	120
14	0	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	45	0	120
15	0	0	1	0	1	1	0	1	0	1	0	1	0	1	0	1	0	1	1	1	1	0	0	0	45	85	120
16	0	0	1	0	1	1	0	1	0	1	0	1	0	1	0	1	0	1	1	1	1	0	0	0	45	85	120
17	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	47	255	248
18	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	47	255	248
19	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0	42	170	168
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Ostaje nam još da ovaj sprajt prenesemo u memoriju računara i prikažemo ga. To postižemo na sledeći način:

```
10 FOR I=832 TO 894
20 READ M
30 POKE I,M
40 NEXT I
50 POKE 53269,1
60 POKE 2040,13
70 POKE 53248,170:POKE 53264,0
80 POKE 53249,70
90 REM POKE 53264,1: POKE 53271,1
```

Od programske linije 10 do 90 je potpuno identično kao kod sprajta visoke rezolucije

```
100 POKE 53276,1
```

U ovoj memorijskoj adresi uključujemo multikolor za prvi sprajt (svaki bit jedan sprajt 0-7)

110 POKE 53287,5
Ova adresa određuje boju kod kombinacije 10 za multikolor tačku. Takođe ova boja jedina može biti različita od ostalih multikolor sprajtova (adrese se kreću od 53287-prvi do 53295-osmi sprajt). Kao što vidite to su iste adrese koje koristimo i za jedinu boju kod visokorezolucijskog sprajta.
120 POKE 53285,2
Ova adresa određuje boju kod kombinacije 01 za multikolor tačku
130 POKE 53286,3
Ova adresa određuje boju kod kombinacije 11 za multikolor tačku
Kombinacija 00 je transparentna (vidi se pozadina) i samim tim nema memorijsku adresu za boju
1000 DATA 0,0,0
1010 DATA 42,170,168
1020 DATA 47,255,248
1030 DATA 47,255,248
1040 DATA 45,85,120
1050 DATA 45,85,120
1060 DATA 45,0,120
1070 DATA 45,0,120
1080 DATA 45,0,120
1090 DATA 45,0,120
1100 DATA 45,0,120
1110 DATA 45,0,120
1120 DATA 45,0,120
1130 DATA 45,0,120
1140 DATA 45,85,120
1150 DATA 45,85,120
1160 DATA 47,255,248
1170 DATA 47,255,248
1180 DATA 42,170,168
1190 DATA 0,0,0
1200 DATA 0,0,0
Od linije 1000 do 1200 su podaci o izgledu našeg multikolornog sprajta. Sve ostalo što je rečeno za visokorezolucijski sprajt važi i za multikolorni.

Naravno tu postoje i razne programske cake. Naprimer ako nam treba jedna sličica za koju želimo da bude visokorezolucije ali da ime više boja možemo uključiti više visokorezolucijskih sprajtova raznih boja i staviti jedan preko drugog i koristeći transparenciju izmeštati boje po želji i još mnogo interesantnih stvari.

Krenimo sa današnjom lekcijom

Danas ćemo naučiti još neke funkcije

LEFT$ - Uzimanje iz tekstualne promenljive željeni broj karaktera počev od leve strane sadržaja zadate promenljive

Format funkcije je:
LEFT$ (A$,B)
Često nam je potrebno da iz neke tekstualne promenljive izdvojimo željene karaktere. Ova funkcija nam omogućava da to uradimo počev od leve strane teksta u zadatoj promenljivoj i da uzmemo onoliko karaktera koliko navedemo pod Argumentom "B" (Argument može da bude broj ili promenljiva). Rezultat možemo odmah prikazati ili ga smestiti u neku željenu tekstualnu promenljivu

Primer:
10 A$="PRVI DRUGI TRECI"
20 PRINT LEFT$(A$,4)
Posle pokretanje programa na ekranu će se ispisati "PRVI"

RIGHT$ - Uzimanje iz tekstualne promenljive željeni broj karaktera počev sa desne strane sadržaja zadate promenljive

Format funkcije je:
RIGHT$ (A$,B)
Ova naredba izvlači iz zadate tekstualne promenljive željene karaktere počev od njene desne strane. Sve ostalo što važi za predhodnu funkciju važi i ovde
Primer:
10 A$="PRVI DRUGI TRECI"
20 PRINT RIGHT$(A$,5)
Posle pokretanje programa na ekranu će se ispisati "TRECI"

MID$ - Uzimanje iz tekstualne promenljive željeni broj karaktera počev od sredine sadržaja zadate promenljive

Format funkcije je:
MID$ (A$,A,B)
Ova naredba će nam iz zadate tekstualne promenljive izvući željene karaktere počev od karaktera zadatog argumentom "A" za željeni broj karaktera zadatih argumentom "B". Ostalo je kao i kod dve predhodne funkcije
Primer:
10 A$="PRVI DRUGI TRECI"
20 PRINT MID$(A$,6,5)
Posle pokretanje programa na ekranu će se ispisati "DRUGI"

Vreme je za nove naredbe i primere.

READ, DATA - čitanje podataka upisanih u program

Ove dve naredbe idu uvek zajedno. Naredba READ čita podatke koje su navedene u naredbi DATA i smešta ih u željene numeričke ili tekstualne promenljive. Veoma korisno kada moramo da unesemo veliku količinu podataka u raznim promenljivima iz programa. Ukoliko pokušamo da unesemo pogrešnu vrednost (tekstualnu umesto numeričku) ili ne bude više podataka prilikom pozivanja naredbe READ dobićemo poruke o grešci "Syntax error" odnosno "Out of data"
Formati naredbe su:
READ A - DATA 5
READ X$ - DATA "PROBNO" - možemo upisati znake navoda kod teksta ali nije obavezno
READ A,X$,A(I),X$(M,5)... - DATA 5,PROBNO,1,PRVI...
Primer:
10 PRINT "IME","PREZIME","BR.BODOVA"
20 FOR I=1 TO 3
30 READ I$,P$,B
40 PRINT I$,P$,B
50 NEXT I
60 DATA ZOKA,ZOKIC,25
70 DATA MIKI,MIKIC,70
80 DATA PETAR,PETROVIC,55

Kada startujemo program u prvom redu 10 će se ispisati zaglavlje. Zatim u programskom redu 20 aktiviramo petlju sa 3 ciklusa. U redu 30 čitamo podatke iz DATA naredbe i iste se smeštaju u tekstualne promenljive I$ (ime),P$ (Prezime) i numeričku promenljivu B (broj bodova). Red 40 štampa ove podatke sa razmakom. Red 50 odrađuje 3 ciklusa i svakim prolazom se učitavaju i štampaju sledeći podaci iz DATA naredbi. Od reda 60 do 80 su DATA podaci koje koristimo. Imajte u vidu da ne moraju da se pišu u posebnom redu za svakog korisnika. To je sve zbog preglednosti inače mogli smo da napišemo u 60 redu DATA ZOKA,ZOKIC,25,MIKI,MIKIC,70,PETAR, PETROVIC,55 dok god može da stane u dva programska reda
Kada se program startuje dobijemo sledeće:

IME	PREZIME	BR.BODOVA
ZOKA	ZOKIC	25
MIKI	MIKIC	70
PETAR	PETROVIC	55

RESTORE - Ova naredba omogućava da se čitanje podataka naredbom READ iz naredbe DATA ponovo vrati na prvi podatak u DATA naredbi

Oblik naredbe:
RESTORE
Primer:

```
10 FOR I=1TO 10
20 READ A
30 PRINT A,
40 NEXT I
50 RESTORE
60 GOTO 10
70 DATA 3,7,8,9,2,99,2,2,55,77,88,77
```

U liniji 10 postavljamo petlju sa 10 ciklusa, Programski red 20 čita numerički podatak. Linija 30 štampa taj podatak. Linija 40 završava petlju kada se prestigne zadata krajnja vrednost. To znači da će prikazati 10 podataka datih u data liniji i zanemariti ostale ukoliko ih ima. U programskom redu 50 dajemo naredbu RESTORE kojom čitanje podataka iz DATA naredbe vraćamo na prvi (u našem slučaju na prvi broj 3). Programski red 60 stvara beskonačnu petlju i da nismo dali naredbu RESTORE dobili bi smo poruku o nedostatku podataka "Out of data". U liniji 70 se nalazi DATA naredba koja sadrži podatke koji će biti pročitani naredbom READ

GET - Očitavanje pretisnutog tastera

Ova naredba očitava tastaturu i rezultat očitavanja (jedan karakter) smešta u zadate numeričke i tekstualne promenljive.
Ukoliko se traži numerički a pretisne se tekstualni taster dobija se poruka o grešci "Syntax error"
Format naredbe:

```
GET A$
GET A
GET A$,B$,C$,A,B...
```

Naredba GET može primiti maksimalno 10 promenljivih.
Primer:

```
10 GET A$
15 IF A$="A" THEN PRINT " PRITISNUO SI A";
20 PRINT A$;:GOTO 10
```

Kada startujemo ovaj program računar će čekati da se pritisne neki taster i kada se to desi ispisaće ga na ekranu osim ako to ne bude slovo A. Ukoliko jeste pojaviće se poruka "Pritisnuo si A" i program će nastaviće dalje sa ispisom slova koje pretiskate.

Na kraju dana dajmo sebi oduška i napravimo jednu arkadu. Ništa nije bolje od aktiviranje možda-nih čelija od ovoga. Zamislite formulu kojoj je cilj da ostane što duže na stazi i samim tim pobere što je moguće više poena. Naravno da ne bude sve ružičasto pobrinule su se prepreke u vidu ograde puta koje ne smete da udarite. Ukoliko se vaša formula sudari sa ogradom 50 puta igri je kraj i daje se ukupni broj poena koji je do tada postignut.
Pa da počnemo sa izradom naše prve, jedinstvene, nikad viđene arkade (obično preterivanje. ali malo marketinga ne može da škodi)
Šalu na stranu krećemo sa programiranjem i pritom ćemo objašnjavati korak po korak

```
10 PRINT "(SHIFT+CLR/HOME)"
```
Prvo naravno brišemo ekran
```
20 XK=170:SU=0:PO=0
```
Postavljamo početnu vrednost za horizontalni položaj sprajta i brišemo promenljive za detekciju sudara i broja poena
```
30 GOSUB 11000
```
Odlazimo na podprogram gde se ispisuje početni ekran i malo uputstvo.zatim se dodeljuju boje okvira,pozadine i karaktera. Takođe setujemo zvuk koji će, koliko toliko, imitirati rad motora
```
40 GOSUB 10000
```
Idemo na podprogram gde se učitava i aktivira sprajt kojim ćemo upravljati tokom ove vožnje
```
50 PRINT "(SHIFT+CLR/HOME)"
```
Ponovo brišemo ekran
```
60 L=15:D=15
```
Ovde imamo početno određivanje veličine leve i desne ivice ograde
```
70 X$=" ########################################"
```
U tekstualnu promenljivu X$ smeštamo jedan razmak i 38 # karaktera koji sačinjavaju ogradu preko cele staze. Ekran može prikazati u jednom redu maksimalno 40 karaktera ali ostavićemo jedan prazan karakter sa leve i desne strane ekrana. Razlog je ukoliko se red ispuni za svih maksimanih 40 karaktera sledeća naredba PRINT će ostaviti jedan prazan red što ne želimo
```
80 L$=LEFT$(X$,L)
```
Kod promenljive L$ unosimo levu ivicu ograde koja u početku ima 15 karaktera uzetih iz promenljive X$ sa leve strane
```
90 M$="        "
```
Promenljiva M$ sadrži 9 praznih (space) karaktera i predstavlja put po kome vozimo.
```
100 D$=RIGHT$(X$,D)
```
Ovde u promenljivi D$ unosimo desnu ivicu ograde koja takođe u početku ima 15 karaktera uzetih iz promenljive X$ ovog puta sa desne strane. Kada uzmemo u obzir levu ivicu ograde, put i desnu ivicu ograde dobijamo ukupno 39 karaktera u redu. Gore smo napomenuli zašto ne želimo da ih bude 40
```
110 T=INT(RND(0)*2)+1
```
U ovom redu formiramo slučajan broj od 1 do 2 i isti unosimo u promenljivu T
```
120 IF T=1 THEN L=L-1:D=D+1
```
U slučaju da je vrednost promenljive T = 1 skraćujemo levu ivicu ograde za 1 i povećavamo desnu za 1 (put skreće ulevo)
```
130 IF T=2 THEN L=L+1:D=D-1
```
U slučaju da je vrednost promenljive T = 2 uvećavamo levu ivicu ograde za 1 i skraćujemo desnu za 1 (put skreće udesno)
```
140 IF L=0 THEN L=L+1:D=D-1
```
U slučaju da smo došli do same krajnje leve ivice ograde uvećavamo levu ivicu za 1 i umanjujemo desnu za 1 (sprečavamo da put skroz nestane na levoj strani ekrana)
```
150 IF D=0 THEN D=D+1:L=L-1
```
U slučaju da smo došli do same krajnje desne ivice ograde uvećavamo desnu ivicu za 1 i umanjujemo levu za 1 (sprečavamo da put skroz nestane na desnoj strani ekrana)

160 PRINT L$+M$+D$
Ova programska linija prikazuje trenutnu levu ivicu ograde,put i trenutnu desnu ivicu ograde i samim tim daje nam stazu po kojoj vozimo.
170 POKE 54273,20:POKE 54273,25
U ovoj programskoj liniji povećavamo i smanjujemo visinu tona i na taj način imitiramo zvuk motora
180 GOSUB 9000
Odlazimo na podprogram u kojem upravljamo vozilom, proveravamo da li je došlo do sudara i ako ih je više od 50 završavamo vožnju
190 GOTO 80
Ovde se ponovo vraćamo na programski red 80 i time pravimo beskonačnu petlju sve dok ne bude ispunjen uslov za kraj programa
9000 A=PEEK(197)
Nalazimo se u prvom podprogramu. Ova linija ispituje vrednost u memorijskoj adresi 197 koja služi za očitavanje tastature. Svaki pritisak na taster daje različit broj vezan za isti koji se upisuje na ovoj memorjskoj lokaciji i koji možemo da ispitamo. Vrednost očitane tastature smeštamo u promenljivu A
9010 IF A=10 THEN XK=XK-5
Ovde proveravamo da li je očitana vrednost jednaka broju 10 što znači da je pritisnut taster A. Ako jeste umanjujemo horizontalnu poziciju sprajta za 5 - skretanje ulevo (Ako je umanjimo samo za 1 vozilo se presporo kreće da bi moglo da izbegne prepreku)
9020 IF XK<20 AND IN=0 THEN XK=20
U slučaju da je XK promenljiva (X hotizontalna kordinata vozila) manja od 20 i indikator IN je nula u XK stavljamo vrednost 20 da vozilo ne bi više moglo da ide u levo i tako nestane sa ekrana. Zbog čega nam je potreban indikator IN? Ovaj indikator će biti jednak 1 kada budemo pokretali vozilo na desnu stranu i pređemo vrednost X kordinate od 255. Kada je na nuli znači da nije pređena ova vrednost
9025 IF XK<0 AND IN=1 THEN XK=255:POKE 53264,0:IN=0
Ovde imamo ispitivanje da li je kordinata vozila manja od nule i indikator IN jednak jedinici što znači da je vozilo prešlo granicu od 255 tačaka. Iz tog razloga gasimo deveti bajt prvog sprajta koji teži 256, za kordinatu vozila stavljamo vrednost 255 i poništavamo indikator
9030 IF A=18 THEN XK=XK+5
U slučaju da je očitana vrednost tastature jednaka broju 18 znači da je pritisnut taster D. Onda uvećavamo horizontalnu poziciju sprajta za 5 - skretanje udesno
9040 IF XK>255 THEN XK=0:POKE 53264,1:IN=1
U slučaju da je XK promenljiva veća od 255 to znači da izlazimo iz opseka mogućnosti osmobitnog broja pa je potrebno uključiti deveti bajt sprajta koji se nalazi na adresi 53264 (svaki bit za jedan sprajt 0-7) i postaviti promenljivu XK na nulu (deveti sprajt nosi težinu 256 + vrednost promenljive/X kordinate). Inače kao što rekosmo sprajt po horizontali može ići do 320 tačaka
9050 IF XK>64 AND IN=1 THEN XK=XK-5
Ovde proveramo da li je horizontalna vrednost veća od 64 i da li je indikator IN jednak jedinici. Ako jeste to znači da smo prešli granicu od 320 tačaka (256+64) i da vozilo ne bi dalje išlo udesno i samim tim nestalo sa ekrana umanjujemo kordinatu za 5.
9200 POKE 53248,XK
Unosimo u memorijsku adresu horizontalne kodinate prvog sprajta vrednost XK promenljive i samim tim pokrećemo vozilo levo/desno zavisno od ispunjenja gore datih uslova
9210 PO=PO+1
Povećavamo vrednost poena igrača za 1
9300 IF PEEK(53279)=1 THEN PRINT "SUDAR";SU:SU=SU+1
Ovde testiramo da li je došlo do sudara sprajta sa pozadinom (koju čini bilo koji karakter) i ukoliko jeste setuje se bit sprajta kod koga je došlo do sudara (u našem slučaju prvi sprajt, svaki sprajt ima svoj bit od 0 do 7) u memorijskoj lokaciji 53279 zaduženoj za detekciju sudara sa pozadinom.

Ukoliko je došlo do sudara ispisuje se "SUDAR" i broj koliko je sudara do tada došlo (Ako vam smeta ovo ispisivanje izbacite ga. Lično mislim da je interesantno i doprinosi tenziji igre pogotovo kada je put gotovo na samoj levoj ivici jer onda ima veoma mali prostor za manervisanje). Zatim se promenljiva SU povećava za 1

9310 IF SU>50 THEN GOTO 12000

Ukoliko je promenljva SU veća od 50 došlo je do više od 50 sudara i program odlazi na programsku liniju 12000 gde se ispisuju poeni, postavlja pitanje za novu igru i ukoliko odgovorimo sa N program se završava

9500 RETURN

Povratak na naredbu posle GOSUB koja je pozvala ovaj podprogram

10000 FOR I=832 TO 894

U ovom podprogramu formiramo,uključujemo i podešavamo sprajt. Prvo aktiviramo petlju koja će promenljvoj I dodeliti početnu vrednost 832 (memorijska adresa od koje upisujemo sprajt) i ići do krajnje vrednosti 894 (memorijska adresa kojom završavamo upis sprajta)

10010 READ M

Ovde čitamo podatke iz DATA naredbe i vrednost dodeljujemo promenljivoj M

10020 POKE I,M

Unosimo u promenljivoj I koja sadrži lokaciju memorije u kojoj se unosi sprajt vrednost promenljive M koja je očitana iz DATA naredbi

10030 NEXT I

Završavamo petlju unosom svih 63 bajta opisa sprajta u memoriju rezervisanu za čuvanje istog

10140 POKE 53269,1

Memorijska adresa 53269 uključuje prvi sprajt unosom vrednosti 1 (svaki sprajt ima svoj bit od 0 do 7)

10150 POKE 2040,13

Adresa 2040 određuje u kom bloku memorije od 64 bajta se nalazi upisan sprajt. U ovom slučaju to je blok 13 (13*64=832) (2040-prvi,... 2047-osmi sprajt)

10160 POKE 53248,170:POKE 53264,0

Postavljamo poćetnu horizontalnu vrednost X kordinate sprajta u memorjskoj lokaciji 53248 za prvi sprajt (53248-prvi,53250-drugi, ... 53262-osmi sprajt) na 170. Deveti bit u memoriskoj lokaciji 53264 je resetovan (na nuli)

10170 POKE 53249,70

Postavljamo vrednost Y kordinate prvog sprajta na 70 (53249-prvi,53251-drugi... 53263-osmi sprajt)

10180 POKE 53287,3

Postavljamo vrednost boje za prvi sprajt. (53287-prvi,53288-drugi,... 53294-osmi sprajt)

10210 RESTORE

Vraćamo čitač DATA podataka na prvi podatak

10220 RETURN

Povratak na glavni program

10500 DATA 0,0,0
10510 DATA 0,0,0
10520 DATA 60,0,60
10530 DATA 63,255,252
10540 DATA 63,255,252
10550 DATA 63,255,252
10560 DATA 63,255,252
10570 DATA 3,255,192
10580 DATA 3,255,192
10590 DATA 3,195,192
10600 DATA 3,195,192
10610 DATA 3,195,192

```
10620 DATA 3,195,192
10630 DATA 3,255,192
10640 DATA 3,255,192
10650 DATA 0,255,0
10660 DATA 63,255,252
10670 DATA 60,0,60
10680 DATA 63,255,252
10690 DATA 60,0,60
10700 DATA 0,0,0
```
Od programske linije 10500 do 10700 su podaci izgleda našeg sprajta
```
11000 PRINT "(SHIFT+CLR/HOME)(CTRL+2)"
```
Ovaj podprogram ispisuje početni ekran. Prvo se ekran briše i boja karaktera postavlja na belu
```
11010 PRINT "TRKA FORMULE 1";
11020 PRINT "KOMANDE SU A - LEVO, D - DESNO"
11030 PRINT "TREBA IZDRZATI STO DUZE I ZARADITI STO VISE POENA"
11040 PRINT "MOZETE NAPRAVITI NAJVISE 50 SUDARA"
11045 PRINT "BILO KOJI TASTER ZA POCETAK IGRE"
```
Od programske linije 11010 do 11045 ispisuje se naslov i uputstvo za igrače
```
11050 POKE 53280,3:POKE 53281,12
```
Postavljamo boju okvira na svetlo plavu a pozadinu na svetlo sivu (imitacija asfalta, ima smisla zar ne?)
```
11060 GET A$:IF A$="" THEN GOTO 11060
```
U ovoj programskoj liniji očitavamo tastaturu pomoću GET naredbe i smeštamo vrednost pritisnutog tastera u promenljivu A$. Ukoliko nije pretisnut nijedan taster odlazimo ponovo na istu liniju (dok ne pretisnemo neki taster ne idemo dalje)
```
11070 FOR I=54272 TO 54300:POKE I,0:NEXT I
```
Ovde brišemo sve parametre za zvuk i setujemo isti za korišćenje
```
11080 POKE 54296,8
```
Jačina zvuka na 8 (maksimalno 15)
```
11090 POKE 54278,240
```
Aktiviranje stalnog zvuka prvog kanala
```
11100 POKE 54276,33
```
Postavljanje oblika zvuka prvog kanala na testerasti
```
11120 POKE 54272,100: POKE 54273,20
```
Početna visina tona prvog kanala (niži i viši bajt)
```
11200 RETURN
```
Povratak na glavni program
```
12000 PRINT "(SHIFT+CLR/HOME)FORMULA JE SUVISE OSTECENA. IGRI JE KRAJ"
```
Od ove linje počinje završetak programa. Brišemo ekran, ispisujemo poruku o oštećenju vozila
```
12010 PRINT "ZARADIO SI UKUPNO";PO;"POENA":POKE 53269,0
```
Ispisujemo koliko je poena ostvareno i gasimo prvi i jedini sprajt. Ako ovo ne uradimo sprajt ostaje na ekranu i posle završetka programa
```
12020 FOR I=54272 TO 54300:POKE I,0:NEXT I
```
Prekidamo zvuk (brišemo sve njegove parametere). Ako ovo ne učinimo čućemo zvuk i posle završetka programa
```
12030 PRINT "MOZE JOS JEDNA IGRA? (M/N)"
```
Ovde postavljamo pitanje za još jednu igru. Pitate se zašto umesto D/N (DA/NE) stoji M/N (MOŽE/NEMOŽE). Tako je, pogodili ste. Pošto za kretanje koristimo A i D tastere velika je verovatnoća da ćemo držati pritisnut taster D kada dođe do poslednjeg 50 sudara i samim tim odmah startovti novu igru

```
12040 GET A$:IF A$="" THEN GOTO 12040
```
Pomoću naredbe A$ očitavamo tastaturu. Ako nije pretisnut nijedan taster vrtiće se u petlji. Možda ste se pitali zašto smo za upravljanje sprajtom koristili čitanje memorijske adrese 197 i ispitivali njenu vrednost a ne naredbu GET. Odgovor je prost, naredba GET čeka da se taster stisne i u stalnoj je petlji dok se to ne desi, a memorijska adresa 197 vrši stalno očitavanje tastature bez obzira da li je taster pritisnut ili ne i samim tim program se ne prikida dok se ne pretisne željeni taster
```
12050 IF A$="M" THEN GOTO 10
```
Ukoliko je pritisnut taster M idemo na novu igru
```
12060 IF A$="N" THEN END
```
Ukoliko je pritisnut taster N igri je kraj
```
12070 GOTO 12040
```
Ukoliko nije pritisnut nijedan od traženih tastera ponovo se vraćamo na očitavanje tastature

Kao što ste videli uz malo truda može se napraviti arkadna igrica i u bejziku. Naravno sve je to sporo i nezgrapno, pa zamislte da još ima više od jednog sprajta koji kontrolišete... Ali ako išta više morate se složiti da je bilo zabavno. Pokušajte da dodate nešto ovom programu. Npr. umesto jednobojnog sprajta ubacite multikolor (formula u tri boje + transparentna). Prilikom krajnjeg sudara učinite da formula izgleda kao da je pretrpela veliko oštećenje, stavite drugi karakter koji više liči na ogradu puta, promenite zvuk da bolje imitira rad motora i još mnogo toga šta vam padne na pamet.

Ovim završavamo za danas.

Uživajte u programiranju na starom dobrom debeljku (ili C verziji ili emulatoru ili ...)