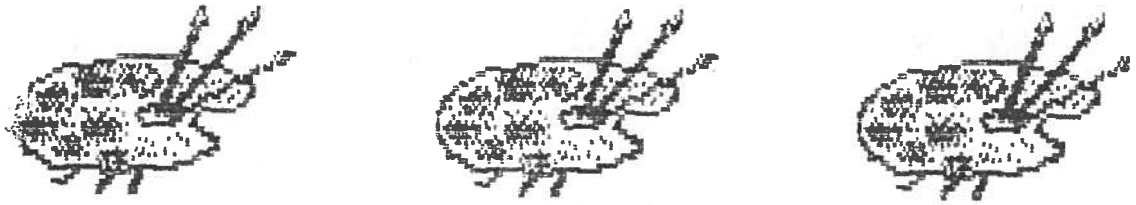


SPECTRUM GRAPHICS AND SOUND

Steve Money

KLUB VT KAROLINKA 1990

Obsah

Úvod

Tato příručka vznikla překladem knihy Spectrum Graphics and Sound autora Steva Moneye s vypuštěním krátké kapitoly o zvuku a doplněním posledních dvou kapitol k původnímu anglickému textu.

Příručka podává podrobný obraz o využití grafických schopností Spectra pomocí jazyka Basic. Jde vlastně o spoustu ukázkových programů, u nichž je zhruba osvětlena jejich činnost nebo alespoň její princip. Neslouží tedy jako nějaká učebnice, ale spíše jako receptář nápadů a postupů, které můžete náležitě upravit a vylepšit a vložit do svých programů. Mnoho postupů a metod je zde spíše naznačeno a je na vás si je aplikovat a vyšperkovat k vaší vlastní spokojenosti - tím pádem se vlastně stále učíte a zdokonalujete v programování.

Obsah jednotlivých kapitol:

1. kapitola obsahuje ty úplně základní informace o tom, co je to grafika, jak se využívá, jaké druhy grafiky Spectrum používá.
2. kapitola vás seznamuje s využitím té nejjednodušší grafiky, tedy mozaikové (jejíž využití v praxi je dosti omezené, ale možné).
3. kapitola podává základní informace o jemné grafice, funkci příkazů PLOT a DRAW a předvádí základní aplikace jemné grafiky.
4. kapitola vás naučí kreslit kruh několika způsoby, kreslit mnohoúhelníky a elipsy, rotaci obrázků.
5. kapitola předvádí využití uživatelských grafických znaků a tisk písmen na pozice jemné grafiky a jejich rotaci a zvětšování.
6. kapitola podává ucelený pohled na využití barev.
7. kapitola vás naučí kreslit různé typy grafů - sloupcový, kruhový, teploměr nebo graf funkce.
8. kapitola ukazuje základní postupy při pohybování bodem nebo obrazcem, odrážení od stěn nebo od pohybující se pálky.
9. kapitola předvádí, jak vložit do vašich obrázků zdání prostorovosti (třírozměrné grafy, perspektiva).
10. kapitola se zabývá problematikou přepsání originálních znaků (písmen, číslic apod.) na jiné dle vašeho přání.
11. kapitola vám umožní poznat, jak je v paměti uložena obrazová informace.

Překlad knihy byl proveden dosti volně, ponechal jsem tam jen důležitější věci a vypustil "omáčku" okolo. Při samotném "klepání" textu do D-writeru jsem se snažil vyhnout se chybám, jak pravopisným tak chybám v textech programů. Pokud mi ale přece jen nějaké unikly, pak se velmi omlouvám - snad to nebude mít vliv na celkový dojem z příručky a neznemožní vám to pochopit problematiku. Jinak programy nejsou natolik složité, aby se nedaly lehce opravit, případně samozřejmě modifikovat a vylepšit.

Po přečtení a prostudování této příručky musíte jen čekat na

další, kterou snad někdy někdo napíše a vydá a která se bude zabývat grafikou Spectra z pohledu programování ve strojovém kódu.

Poznámka vydavatele: u tohoto cyklostylového vydání se nám bohužel nepodařilo umístit do textu i grafické obrázky, které jsou v originálu knihy. Pokusíme se je nechat rozmnožit alespoň xeroxem na konci textu.
Prosíme čtenáře o pochopení....

Klub Karolinka

1. kapitola Základní informace o grafice

Jednou z nejatraktivnějších vlastností domácích počítačů je jejich schopnost vytvářet grafické obrazce, nejlépe barevné. Toto samozřejmě umí i váš ZX Spectrum.

Každý počítač chrakterizuje jeho rozlišovací schopnost, tzn. kolik bodů může zobrazit na obrazovce. Spectrum má rozlišovací schopnost jemné grafiky 256*176 bodů. Znamená to 256 bodů horizontálně a 176 vertikálně. Jeden každý z těchto bodů může počítač rozsvítit nebo zhasnout. V této jemné grafice jsou udělány všechny ty pěkné obrázky, které znáte z her.

Ale nevýhodou jemné grafiky v programech je vysoký nárok na paměť. Kdyby měl každý bod obrazovky uloženou v paměti informaci o svém rozsvícení či zhasnutí a ještě o barvě, zabralo by to skoro celou paměť počítače. U Spectra to jeho autoři řešili tak, že rozdělili obrazovku na čtverce 8*8 bodů (tedy 32*22 čtverců) a v každém tomto čtverci mohou být body pouze dvou barev.

Jistě jste už viděli spoustu krásných a rychlých her, ve kterých se grafické obrazce jenom míhají po obrazovce. Existují hrací stroje (produkované např. firmou Atari), na kterých se dají pouze hrát hry. V podstatě jsou to také počítače jako je ten váš, ale jejich vnitřní uspořádání je předurčuje pouze ke hraní her.

Ale Spectrum samozřejmě také umožňuje hraní her. Jistě jste jich už množství viděli a často jste obdivovali jejich grafické ztvárnění. Dá se říci, že v této oblasti je Spectrum jedním z nejlepších počítaču.

Výhodou počítače proti hernímu stroji je to, že si můžete nahrát velké množství her již hotových a nebo si vytvořit své vlastní, ať už v Basicu nebo s pomocí strojového kódu.

Existuje velké množství her a v každé z nich se objevuje jiné využití grafiky (ať už to jsou pasivní obrázky v dialogových hrách, potvůrky v akčních hrách nebo dokonce prostorová ztvárnění bludišť).

Ale mnoho uživatelů počítače nechce stále jenom hrát hry. Zde se nabízí další využití grafických možností počítače. Pro různé výpočty a statistiky je užitečné vidět nejen holé číselné údaje a hodnoty, ale moci si také prohlédnout graf.

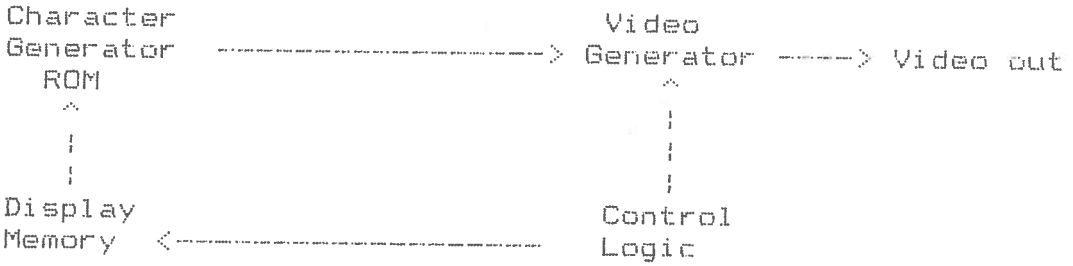
Dále můžeme pomocí jemné grafiky kreslit různá schémata, plošné spoje a nákresy součástek (CAD - computer aided design). Je zde možnost ukázat názorně průběh nějakého chemického nebo fyzikálního experimentu, nebo znázorňovat na obrazovce naměřené hodnoty, získané pomocí nějakého vnějšího zařízení.

Při pohledu na televizní obrazovku se vám zdá, že jde o jeden velký obrázek. Ve skutečnosti je to ale jenom jeden rychle se pohybující bod, který přejíždí zleva doprava a zhora dolu a rozsvěcuje jednotlivé body v TV řádcích. TV řádků je 625. Naše oko není schopno tento bod díky svý pomalým reakcím a jeho rychlému pohybu vůbec zaregistrovat. Totéž platí i pro barevnou obrazovku.

Když zapnete Spectrum, objeví se černý text na bílém pozadí a vy můžete v těchto barvách psát. Je to příjemné pro oči, protože je pro vás přirozené číst černý text na bílém papíře.
Obraz Spectra je rozdělen na čtverečky, v nichž můžete zobrazovat jednotlivé znaky. Těchto čtverečků je 32 v řádku a 22 ve sloupci. Jsou to tytéž čtverečky, o nichž jsem se už zmínil v souvislosti s barvami.
Při podrobnějším pohledu na některý znak na obrazovce zjistíte, že je složen z malých bodů. Těchto bodů je 8*8 a tvar znaku je tvořen rozsvícením určitých bodů v tomto rastru - viz obr.

```
........
..0000..
.0......
..0000..      znázornění bodového tvaru znaku
......0.
.0....0.      .  prázdný bod
..0000..      0  svítící bod
........
```

Obrázek na TV obrazovce je obnován každou 1/50 sekundy. Tedy 50* za sekundu vezme počítač z paměti inforaci o tvaru obrazu a vyšle ji do televizoru (nebo do monitoru). Tato část paměti se nazývá obrazová a každý byt v ní reprezentuje 8 bitů - bodů obrazovky (kromě barevné informace, ta je uložena jinak).
Princip obnovování obrazu dokumentuje toto schéma:



Bodová mapa znázorňující tvar znaku je uschována v oblasti paměti nazvané Character Generator. Tento je uložen v paměti ROM (Read Only Memory) a nemůže proto být změněn. Ale můžeme změnit adresu Character Generatoru a vytvořit si své vlastní znaky (viz dále).
Oproti jiným počítačům se Spectrum liší tím, že má pouze jednu obrazovkovou paměť a proto ukládá znaky i jemnou grafiku do jednoho prostoru a tak mohou vzniknout určité problémy při průniku znaků a jemné grafiky.

Není příliš pěkné vytvářet obrázky pouze z písmen a znaků, i když i to je možné. Ale Spectrum nám umožňuje využívat takzvanou hrubou grafiku - to je kreslit obrazce pomocí čtverečků o velikosti 4*4 body, které máme nadefinovány na klávesách 1-8 v grafickém módu.

Máme tedy 16 předdefinovaných znaků s těmito čtverečky (díky užití shiftu!). Při pečlivém promyšlení výsledného tvaru můžeme s těmito znaky velmi jednoduše vytvářet jednoduché obrazce. Máme vlastně nyní obrazovku rozdělenou na 64 sloupců a 42 řádků.
Můžeme ale tyto čtverečky zobrazovat v 8 barvách. Aby jste dostali nějaký základní obraz o hrubé grafice, zkuste tento program:

```
100  REM mosaic graphics demo
110  LET X=15
120  LET Y=10
130  FOR J=1 TO 15
140  INK INT (RND*7)
150  LET K=J
160  IF K>10 THEN LET K=10
170  FOR I=1 TO J
180  PRINT AT Y+K,X+I;CHR$ 130;
190  PRINT AT Y+K,X-I;CHR$ 129;
200  PRINT AT Y-K,X-I;CHR$ 132;
210  PRINT AT Y-K,X+I;CHR$ 136;
220  NEXT I
230  FOR I=1 TO K
240  PRINT AT Y+I,X+J;CHR$ 130;
250  PRINT AT Y+I,X-J;CHR$ 129;
260  PRINT AT Y-I,X-J;CHR$ 132;
270  PRINT AT Y-I,X+J;CHR$ 136;
280  NEXT I
290  NEXT J
300  FOR N=1 TO 500
310  LET I=INT (RND*16)
320  LET J=INT (RND*10)
330  INK INT (RND*7)
340  PRINT AT Y+J,X-I;CHR$ 130;
350  PRINT AT Y+J,X-I;CHR$ 129;
360  PRINT AT Y-J,X-I;CHR$ 132;
370  PRINT AT Y-J,X+I;CHT$ 136;
380  NEXT N
```

Obr. 1.4.

Pomocí mozaikové grafiky tedy můžeme vytvářet jednoduché obrazce, jak ostatně uvidíte v další kapitole. Její výhodou je to, že se programuje podobně jako texty a nevzniká tedy problém míchání písmen a znaků. Nevýhodou je nízká rozlišovací schopnost a hrubost obrazců.

Jako většina počítačů umožňuje i SPectrum vytvoření vlastních grafických znaků, které uloží do své volné paměti RAM. Tyto znaky potom můžeme zobrazit místo těch normálních. Pokud by jste třeba ale potřebovali obrazec o rozkloze 16*16 bodů,musíte si ho složit ze čtyř znaků.

Jak jsme již zjistili, skládá se každý znak na obrazovce z jednotlivých bodů. Pomocí těchto bodů můžeme vytvořit jakýsi druhý obraz, který bude mít 32*8=256 bodů v řádku - X (připomeňme si, že každý znak je čtverec 8*8 bodů a máme 32

znaků v řádku). Ačkoliv existuje 24 textových řádků, tak spodní dva (tzv. dialogové) nejsou normálně pro nás využitelné. Proto máme 22*8=176 bodů ve slouci - Y. Tzn. že máme k dispozici 256*176=45056 bodů.

Ukázkou jemné grafiky je tento program:

```
100  REM DEMONSTRACE JEMNE GRAFIKY
110  LET X=0
120  LET Y=0
125  REM VERTIKALNI RADKY
130  FOR W=1 TO 22
140  PLOT X,Y
150  DRAW 0,175
160  LET X=X+W
170  NEXT W
180  LET X=0
185  REM HORIZONTALNI RADKY
190  FOR H=1 TO 19
200  PLOT X,Y
210  DRAW 255,0
220  LET Y=Y+H
230  NEXT H
235  REM DIAGONALY
240  PLOT 0,0
250  DRAW 255,175
260  PLOT 0,175
270  DRAW 255,-175
275  REM RAMECEK
280  PLOT 0,0
290  DRAW 255,0
300  DRAW 0,175
310  DRAW -255,0
320  DRAW 0,-175
```

Obr. 1.5.

Tyto jednotlivé body se obvykle nazývají pixely a grafika pixelová (nebo jemná). Bohužel díky našim omezeným možnostem není možné reprodukovat jednotlivé obrázky, proto zkoušejte vše to, co vám zde radíme.

V dalších kapitolách se podrobněji seznámíte s hrubou a jemnou (pixelovou) grafikou, s kreslením čar, využitím barev a tvorbou nových znaků a grafů.

2. kapitola Mozaiková grafika

Nyní se tedy podrobněji seznamme s mozaikovou grafikou, která pro svou činnost využívá textové rozdělení obrazovky (32*22 pozic).

Tento program vám ukáže znaky, které může Spectrum ihned zobrazovat:

```
100  REM CHARACTER SET
110  REM  ASCII KODY 32-64
120  CLS
130  PRINT
140  FOR N=32 TO 164
150  PRINT CHR$ N;" ";
160  NEXT N
```

Jsou to znaky s kódy 32 až 164. Znaky s kódem nižším než 32 se nazývají řídící a nelze je zobrazit (enter, pohyby kurzoru...). Znaky s kódem větším než 164 jsou takzvané tokeny - jsou to jednotlivá klíčová slova jazyka Basic. Každé z nich je pro úsporu paměti reprezentováno jedním znakem (tokenem). Jsou to např. PRINT, GOSUB...
Program nejprve smaže obrazovku a poté ukazuje jednotlivé znaky. Funkce CHR$ n dá znak s kódem n. Středník je tam kvůli tomun aby se znaky zobrazily vedle sebe a ne pod sebe (tak by se jich zobrazilo vždy jen 22 najednou).
Na konci výpisu si povšimněte několika znaků, složených z malých čtverečků. Jsou to znaky mozaikové grafiky, mají kódy 128 až 143. Získáte je pomocí grafického módu a kláves 1-8 (případně se shiftem - inverzní k normálnímu.) Grafický mód získáte stiskem CAPS SHIFT a 9.
Mozaikové znaky zobrazíte v programu instrukcí PRINT jako obyčejná písmena a znaky.

Černobílý obraz je sice užitečný při psaní a čtení textů, ale vy můžete ve svých obrázcích využít až 8 barev.
Pro změnu barvy černých čtverečků užijte příkaz INK spolu s číslem z intervalu 0-9, které označuje barvu. To, co nyní budete psát na obrazovku, bude jinak barevné než to, co tam již předtím bylo (přičemž to zůstane nezměněné).
Čísla 0-7 označují normální barvy, tak jak je to naznačeno na klávesnici Spectra. O funkci INK 8 a 9 si přečtěte v manuálu ke svému počítači.

Aby jste viděli barevné možnosti Spectra, použijte tento program, který využívá některé mozaikové i normální znaky Spectra:

```
100  REM WALLPAPER PATTERNS
110  DIM A(7)
120  DIM C(7)
130  FOR S=1 TO 30
140  CLS
150  REM TVORBA OBRAZCE
160  FOR P=1 TO 7
170  LET A(P)=122+INT (RND*22)
180  LET C(P)=INT (RND*7)
190  NEXT P
200  REM VYTISTENI OBRAZCE
210  LET K=3+INT (RND*5)
220  FOR N=1 TO 672 STEP K
230  FOR J=1 TO K
240  PRINT INK C(J);CHR$ A(J);
250  NEXT J
260  NEXT N
270  PAUSE 200
280  NEXT S
```

Obr. 2.3.

Program funguje tak, že náhodně generuje 7 znaků a zobrazuje je v náhodně vygenerované barvě, až jimi zaplní celou obrazovku. Potom chvilku počká a všechno začne od začátku.

Stejně jako barvu inkoustu můžeme měnit i barvu pozadí zobrazovaných znaků - instrukcí PAPER. Ta funguje stejně jako INK, ale pro pozadí. Na znaky již dříve zobrazené nemá změna paperu vliv.

Někdy ale chceme zobrazit pouze několik znaků v odlišné barvě. Jistě, dá se to zařídit tak, že nejprve změníme INK nebo PAPER a potom jim navrátíme jejich původní hodnoty. Existuje ale pohodlnější řešení. Instrukce

```
200  PRINT INK 2;"RED TEXT"
```

způsobí vytištění slov "RED TEXT" červeně a další texty již budou psány normálně, dle nastavené hodnoty INK. V příkazu PRINT můžeme samozřejmě využívat všechny instrukce pro změnu barev, která ale bude platit pouze pro text psaný touto instrukcí PRINT.

Toto je tedy pozměněná forma předchozího programu, která funguje lépe:

```
100  REM BETTER WALLPAPER PROGRAM
110  REM MENI INK I PAPER
120  DIM A(7)
130  DIM C(7)
140  DIM P(7)
150  FOR S=1 TO 30
160  CLS
170  FOR I=1 TO 7
180  LET A(I)=122+INT (RND*22)
190  LET C(I)=INT (RND*8)
```

```
200  LET P(I)=INT (RND*8)
210  IF P(I)=C(I) THEN GOTO 190
220  NEXT I
230  REM TISK OBRAZCE
240  LET K=3+INT (RND*5)
250  FOR N=1 TO 672 STEP K
260  FOR J=1 TO K
270  PRINT INK C(J); PAPER P(J);CHR$ A(J);
280  NEXT J
290  NEXT N
300  PAUSE 200
310  NEXT S
```

Obr. 2.5.

Když už umíme tisknout v barvách, neškodilo by naučit se také tisknout na určité místo, které si na obrazovce zvolíme. K tomu slouží instrukce

```
100  PRINT AT R,S;"text"
```

Proměnné R a S označují souřadnice na obrazovce, odkud se "text" začne vypisovat. R označuje řádek, přičemž řádek 0 je ten nevyšší a 21 nejnižší (tzn. 22 řádků, řádky 22 a 23 nelze takto využít, protože jsou vyhraženy komunikaci počítače s uživatelem) S označuje sloupec (nultý sloupec je úplně vlevo na kraji, krajní pravý má číslo 31).

Takže obrazovku si vlastně pro příkaz PRINT AT můžeme představit jako obdélník o rozměrech 22*32 řádků a sloupců, dohromady tedy je zde místo pro 704 znaků. Pozice o souřadnicích 0,0 je levý horní roh. Pravý horní roh má souřadnice 0,31, pravý spodní 21,31 a levý spodní 21,0.

Následující program vypisuje náhodné znaky na náhodných pozicích na obrazovce. Pro zisk náhodné pozice se využívá instrukce RND, která sice sama o sobě dává číslo z intervalu 0 až 1, ale díky násobení a instrukci INT dostaneme celé číslo z intervalu, jaký jsme si zvolili.

```
100  REM VYPIS NAHODNYCH ZNAKU NA NAHODNYCH POZICICH
110  FOR N=1 TO 100
120  REM URCENI RADKU
130  LET R=INT (RND*22)
140  REM URCENI SLOUPCE
150  LET C=INT (RND*32)
160  REM NAHODNY INKOUST
170  INK INT (RND*7)
180  REM URCENI ZNAKU
190  LET S=96+INT (RND*48)
200  REM VYPSANI ZNAKU
210  PRINT AT R,C;CHR$ S
220  NEXT N
```

Obr. 2.7.

Jestliže užíváme mozaikovou grafiku, potom se každý znak vlastně rozdělí na čtyři body. To znamená, že dostaneme 44 bodových řádků a 64 bodových sloupců. Abychom je mohli efektivně využívat, musíme se naučit rozsvítit nebo zhasnout kterýkoliv z těchto bodů (každý z "bodů" mozaikové grafiky je vlastně čtvereček 4*4 body jemné grafiky).

Při podrobnějším prozkoumání zjistíme, že bodům ve znaku jsou přiřazeny tyto hodnoty:

```
   ¦
 2 ¦ 1
---+---
 8 ¦ 4
   ¦
```

Pokud tedy chceme vědět kód určitého tvaru, sečteme číslice bodů, které budou vypsány v barvě inkoustu (tedy "rozsvícené" body) a tento součet přičteme ke 128.

V našem souřadnicovém systému 64*44 bodů můžeme definovat polohu bodu souřadnicemi x jako sloupec a y jako řádek. Budeme potom mít sloupce 0 až 63 (bráno zleva doprava) a řádky 0-43 (zhora dolů).

Pro nalezení těchto souřadnic v textovém souř. systému (32*22) použijeme celou hodnotu z podílu dvěmi. Tedy například pro hodnoty x=33 a y=25 bude řádek:

r=INT (y/2)=INT (12.5)=12

a pro sloupec:

c =INT (x/2)=INT (16.5)=16

No a teď už jenom poznat, na kterou ze čtyř pozic v textové pozici patří mozaikový bod. To zjistíme porovnáním celočíselných hodnot s nezaokrouhleným podílem. Tedy pokud INT (x/2)=X/2, potom bod leží vlevo a hodnota kódu je 2, když INT (x/2)<>x/2, leží bod vpravo a hodnota kódu je 1. Stejně tak pro sloupce - když se hodnoty rovnají, leží bod v horní pozici a když ne, tak dole. Blíže to osvětlí tento podprogram:

```
500  LET C=INT (X/2)
510  LET R=INT (Y/2)
520  LET P=1
530  IF C=X/2 THEN LET P=P*2
540  IF R>Y/2 THEN LET P=P*4
550  PRINT AT R,C;CHR$ (128+P)
```

Proměnná P obsahuje kód mozaikového znaku, tvořený podle předešlého schématu.

Tohle sice funguje, ale pouze když máme prázdnou obrazovku nebo alespoň tiskovou pozici, do které zapisujeme. Nově zobrazený bod totiž smaže ostatní, vypsané již předtím do jeho tiskové pozice. Musíme tedy nějak zjistit, jestli už v tiskové pozici, kam

chceme umístit další bod, něco není.

Protože Spectrum ukládá do obrazovkové paměti pouze tvary znaků a ne jejich kódy jako některé jiné počítače, nemůžeme pomocí příkazu PEEK zjistit, co už jsme na obrazovku vypsali. Existuje sice funkce SCREEN$, která zjistí kód znaku zobrazeného v daných souřadnicích, ta ale bohužel nefunguje pro mozaikové symboly.

Poradíme si tedy tak, že do číselného pole o rozměrech p(c,r) budeme ukládat kódy vypisovaných mozaikových znaků a při dalších bodech budeme konfrontovat odpovídající hodnotu z pole s bodem, který chceme napsat.

V poli tedy budou uloženy kódy mozaikových znaků zmenšené o 128, tedy hodnoty které získáme sečtením plných bodů dle předešlého schématu. Programem zjistíme, jestli bod, který chceme vyplnit, již svítí nebo ne a případně připočteme i jeho pozici a znovu mozaikový znak vytiskneme na obrazovku.

Získáme tedy podprogram, pomocí kterého můžeme na libovolnou pozici vytisknout jeden mozaikový bod, aniž bychom ovlivnili ostatní již vytisklé body. Tento podprogram je využit v následujícím prográmku:

```
100  REM NAHODNE ROZMISTENE GRAFICKE BODY
110  DIM P(32,22)
120  PRINT AT 0,0;"ZAPLNUJI POLE"
130  FOT Y=1 TO 22: FOR X=1 TO 32
140  LET P(X,Y)=0
150  NEXT X: NEXT Y: CLS
160  REM KRESLENI BODU
170  FOR N=1 TO 500
180  LET X=INT (RND*64)
190  LET Y=INT (RND*44)
200  INK INT (RND*8)
210  GOSUB 500
220  NEXT N
230  STOP
490  REM PROCEDURA TISKNOUCI BOD
500  LET R=INT (Y/2)
510  LET C=INT (X/2)
520  LET P1=1
530  IF C=X/2 THEN P1=P1*2
540  IF R<>Y/2 THEN LET P1=P1*4
550  LET P2=P(C+1,R+1)
560  IF P2<8 AND P1=8 THEN GOTO 640
570  IF P2>=8 THEN LET P2=P2-8
580  IF P2<4 AND P1=4 THEN GOTO 640
590  IF P2>=4 THEN LET P2=P2-4
600  IF P2<2 AND P1=2 THEN GOTO 640
610  IF P2>=2 THEN LET P2=P2-2
620  IF P2<1 AND P1=1 THEN GOTO 640
630  GOTO 650
640  LET P(C+1,R+1)=P(C+1,R+1)+P1
650  PRINT AT R,C;CHR$ (128+P(C+1,R+1))
660  RETURN
```

Obr. 2.9.

Poznámka : omluvte našeho překladatele, který někdy v textu označuje sloupce písmenem "S", správně má být podle originálu "C", jak je uvedeno i ve výpisech programů.

Pokud bychom chtěli kreslit čáry, bylo by asi nejlehčí nakreslit si celou čáru na čtverečkovaný papír s označenými sloupci a řádky a pozice jednotlivých bodů zadat např. do DAT, s tím že si je počítač sám načte a vytiskne. Jestliže chceme čáry diagonální, tedy šikmé, bude tato metoda asi užitečná. Ale horizontální a vertikální čáry dokáže počitač velmi jednoduše kreslit sám.

Chceme tedy nakreslit nejprve horizontální čáru mezi dvěma body X1 a X2. Jelikož se jedná o horizontální čáru, souřadnice y se nebude měnit - to máme 50% práce ušetřeno, takže vlastně uděláme x2-x1 kroků.

Tento program kreslí horizontální čáru:

```
100  REM HORIZONTALNI MOZAIKOVA CARA
110  DIM P(32,22)
120  PRINT AT 0,0;"ZAPLNUJI POLE"
130  FOT Y=1 TO 22: FOR X=1 TO 32
140  LET P(X,Y)=0
150  NEXT X: NEXT Y: CLS
160  REM KRESLENI CAR
170  FOR N=1 TO 50
175  REM POCATEK A KONEC
180  LET X1=INT (RND*64)
190  LET X2=INT (RND*64)
200  LET Y1=INT (RND*64)
205  REM URCENI SMERU KRESLENI
210  LET XS=SGN (X2-X1)
215  REM DELKA CARY
220  LET NS=ABS (X2-X1)
230  FOR S=0 TO NS
240  LET X=X1+S*XS
250  LET Y=Y1
260  GOSUB 500
270  NEXT S
280  INK INT (RND*7)
290  NEXT N
300  STOP
500  LET R=INT (Y/2)
510  LET C=INT (X/2)
520  LET P1=1
530  IF C=X/2 THEN LET P1=P1*2
540  IF R<>Y/2 THEN LET P1=P1*4
550  LET P2=P(C+1,R+1)
560  IF P2<8 AND P1=8 THEN GOTO 640
570  IF P2>=8 THEN LET P2=P2-8
580  IF P2<4 AND P1=4 THEN GOTO 640
590  IF P2>=4 THEN LET P2=P2-4
600  IF P2<2 AND P1=2 THEN GOTO 640
610  IF P2>=2 THEN LET P2=P2-2
620  IF P2<1 AND P1=1 THEN GOTO 640
630  GOTO 650
640  LET P(C+1,R+1)=P(C+1,R+1)+P1
650  PRINT AT R,C;CHR$ (128+P(C+1,R+1))
660  RETURN
```

Obr. 2.10.

Pokud bude rozdíl x2-x1 kladný, bude hodnota proměnné xs rovna 1. Když bude x2-x1 záporné, bude xs=-1 (funkce SGN). Toto potřebujeme proto, že ve smyčce určující počet bodů musí tento počet být kladný (ns=ABS (x2-x2)).
Čára je vykreslována pomocí jednoduché smyčky o s krocích jdoucí od 0 do ns. X-ová hodnota pro každý bod se získá přičtením čísla s*xs k x1 (pokud bude xs -1, pak se x bude zmenšovat). S jde od nuly, abychom začali bodem X1.

Pokud chceme nakreslit vertikální čáru, je to velmi podobné, jak poznáte na následujícím programu. Zde se nemění x-ová ale y-ová souřadnice a x zůstává stále stejné.

```
100  REM VERTIKALNI MOZAIKOVA CARA
110  DIM P(32,22)
120  PRINT AT 0,0;"ZAPLNUJI POLE"
130  FOT Y=1 TO 22: FOR X=1 TO 32
140  LET P(X,Y)=0
150  NEXT X: NEXT Y: CLS
160  REM KRESLENI CAR
170  FOR N=1 TO 50
175  REM POCATEK A KONEC
180  LET X1=INT (RND*64)
190  LET Y1=INT (RND*44)
200  LET Y2=INT (RND*44)
205  REM URCENI SMERU KRESLENI
210  LET YS=SGN (Y2-Y1)
215  REM DELKA CARY
220  LET NS=ABS (Y2-Y1)
230  FOR S=0 TO NS
240  LET Y=Y1+S*YS
250  LET X=X1
260  GOSUB 500
270  NEXT S
280  INK INT (RND*7)
290  NEXT N
300  STOP
500  LET R=INT (Y/2)
510  LET C=INT (X/2)
520  LET P1=1
530  IF C=X/2 THEN LET P1=P1*2
540  IF R<>Y/2 THEN LET P1=P1*4
550  LET P2=P(C+1,R+1)
560  IF P2<8 AND P1=8 THEN GOTO 640
570  IF P2>=8 THEN LET P2=P2-8
580  IF P2<4 AND P1=4 THEN GOTO 640
590  IF P2>=4 THEN LET P2=P2-4
600  IF P2<2 AND P1=2 THEN GOTO 640
610  IF P2>=2 THEN LET P2=P2-2
620  IF P2<1 AND P1=1 THEN GOTO 640
630  GOTO 650
640  LET P(C+1,R+1)=P(C+1,R+1)+P1
650  PRINT AT R,C;CHR$ (128+P(C+1,R+1))
660  RETURN
```

Obr. 2.11.

Jistě jste si všimli, že někdy nově jedoucí řádek mění barvu čar položených nad nebo pod ním. To je způsobeno uložením barev v obrazové paměti (každá tisková pozice (2*2 mozaikové body) může mít pouze jednu barvu inkoustu - již jsme se o tom několikrát zmínili).

Pokud chcete vytvořit celý obrázek pomocí mozaikové grafiky, je to velmi pracné. Existuje ale postup, jak si ušetřit mnoho práce a ještě se u toho i pobavit.

Vytvoříme totiž jednoduchý kreslící program, který bude umět pohybovat jakýmsi perem po obrazovce a buď psát nebo mazat mozaikové body.

Pro ovládání pera využijeme klávesy 5,6,7 a 8 - můžete se na nich orientovat podle kurzorových šipek a klávesy U a D pro Zdvižení a Spuštění pera. Takže 5 bude doleva, 8 doprava, 6 dolu a 7 nahoru. Klávesou Q program zastavíme.

Stisklou klávesu poznáme pomocí instrukce INKEY$ - viz manuál Spectra.

Pokud pohybujeme perem doleva (5), snižuje se hodnota X. Pokud doprava (8), tak se X zvětšuje. Stejně tak pohyb nahoru (7) je zmenšování Y a dolu (6) jeho zvětšování. Program hlídá, aby kurzor "nevyjel" z obrazovky (došlo by k chybovému hlášení).

Proměnná W indikuje stav "pera" - W=1 znamená pero píšící, W=0 je pero nahoře.

```
100  REM PROGRAM PRO KRESLENI
110  REM MOZAIKOVYMI ZNAKY
120  PRINT AT 0,0;"NACITAM DO POLE"
130  DIM P(32,22)
140  FOR Y=1 TO 22: FOR X=1 TO 32
150  LET P(X,Y)=0
160  NEXT X: NEXT Y :CLS
170  LET X1=32: LET X2=32
180  LET Y1=22: LET Y2=22
190  LET W=1: LET X=X1: LET Y=Y1: GOSUB 700
195  REM OVLADACI SMYCKA
200  PRINT AT 0,0;"X=";X1;" Y=";Y1;"          "
210  LET A$=INKEY$: IF A$="" THEN GOTO 210
220  IF A$="5" THEN LET X1=X1-1: GOTO 300
230  IF A$="8" THEN LET X1=X1+1: GOTO 300
240  IF A$="6" THEN LET Y1=Y1+1: GOTO 300
250  IF A$="7" THEN LET Y1=Y1-1: GOTO 300
260  IF A$="D" THEN LET W=1: GOTO 300
270  IF A$="U" THEN LET W=0: GOTO 300
280  IF A$="Q" THEN STOP
290  GOTO 200
295  REM KONTROLA X A Y
300  IF X1<0 THEN LET X1=0
310  IF X1>63 THEN LET X1=63
320  IF Y1<2 THEN LET Y1=2
330  IF Y1>43 THEN LET Y1=43
340  LET LP=PC
350  LET X=X1:LET Y=Y1: GOSUB 700
360  LET X=X2: LET Y=Y2
```

```
370  IF W=0 AND LP=1 THEN GOSUB 900
380  LET X2=X1: LET Y2=Y1
390  GOTO 200
690  REM KRESLICI PODPROGRAM
700  LET R=INT (INT (Y)/2
710  LET C=INT (INT (X)/2
720  LET P1=1: LET PC=0
730  IF C=INT (X)/2 THEN LET P1=P1*2
740  IF R<>INT (Y)/2 THEN LET P1=P1*4
750  LET P2=P(C+1,R+1)
760  IF P2<8 AND P1=8 THEN GOTO 840
770  IF P2>=8 THEN LET P2=P2-8
780  IF P2<4 AND P1=4 THEN GOTO 840
790  IF P2>=4 THEN LET P2=P2-4
800  IF P2<2 AND P1=2 THEN GOTO 840
810  IF P2>=2 THEN LET P2=P2-2
820  IF P2<1 AND P1=1 THEN GOTO 840
830  GOTO 850
840  LET P(C+1,R+1)=P(C+1,R+1)+P1
850  PRINT AT R,C; CHR$ (128+P(C+1,R+1))
860  RETURN
895  REM MAZANI BODU
900  LET R=INT (INT (Y)/2
910  LET C=INT (INT (X)/2
920  LET P1=1
930  IF C=INT (X)/2 THEN LET P1=P1*2
940  IF R<>INT (Y)/2 THEN LET P1=P1*4
950  LET P2=P(C+1,R+1)
960  IF P2>=8 AND P1=8 THEN GOTO 1040
970  IF P2>=8 THEN LET P2=P2-8
980  IF P2>=4 AND P1=4 THEN GOTO 1040
990  IF P2>=4 THEN LET P2=P2-4
1000 IF P2>=2 AND P1=2 THEN GOTO 1040
1010 IF P2>=2 THEN P2=P2-2
1020 IF P2>=1 AND P1=1 THEN GOTO 1040
1030 GOTO 1050
1040 LET P(C+1,R+1)=P(C+1,R+1)-P1
1050 PRINT AT R,C;CHR$ (128+P(C+1,R+1))
1060 RETURN
```

Obr. 2.12.

Pokud je pero spuštěno, nový bod se vykreslí na jeho pozici díky podprogramu na řádku 500. Pokud je pero nahoře, je bod vymazán podoprogramem na řádku 700. Na nejvrchnějším řádku je udána aktuální pozice bodu.

Nyní už tedy umíme kreslit obrazce, jednotlivé body i celé čáry, ale někdy budeš chtít vykreslit na obrazovku celé obrázky. Potom je nejjednodušší vzít si čtverečkovaný papír, nakreslit na něj siluetu obrázku a napsat si kódy jednotlivých mozaikových znaků, které v obrázku využijeme.

Tak například budeme chtít vykreslit dopstřed obrazovky siluetu panáčka velikosti 8*8 mozaikových bodů, tzn 4*4 znaky. Kódy znaků uložíme do dat a odtud je načteme a pomocí PRINT AT vytiskneme na obrazovku. Dostatečně to snad osvětlí následující program:

```
100  REM VYKRESLENI PANACKA
110  REM MOZAIKOVOU GRAFIKOU
120  DIM A(4,4)
130  FOR J=1 TO 4
140  FOR I=1 TO 4
150  READ A(I,J)
160  NEXT I
170  NEXT J
180  DATA 128,133,143,128
190  DATA 132,140,142,140
200  DATA 128,132,142,128
210  DATA 128,138,128,138
220  LET R=10
230  LET C=15
240  FOR J=1 TO 4
250  FOR I=1 TO 4
260  PRINT AT R+J-1,C+I-1;CHR$ A(I,J)
270  NEXT I
280  NEXT J
290  STOP
```

Pokud ale můžeme vytisknout jednho panáčka uprostřed obrazovky, můžeme jich samozřejmě vytisknout i více. Následující program tiskne panáčky přes celou obrazovku, ve čtyřech řádcích. První panáček je na pozici 0,0 (nahoře vlevo). Mezi jednotlivými řadami je vynechaný jeden řádek, aby nebyli "nalepení" na sebe.

```
100  REM CELA OBRAZOVKA PANACKU
110  REM POMOCI MOZAIKOVE GRAFIKY
120  DIM A(4,4)
130  FOR J=1 TO 4
140  FOR I=1 TO 4
150  READ A(I,J)
160  NEXT I
170  NEXT J
180  DATA 128,133,143,128
190  DATA 132,140,142,140
200  DATA 128,132,142,128
210  DATA 128,138,128,138
220  FOR R=0 TO 16 STEP 5
230  FOR C=0 TO 28 STEP 4
240  FOR J=1 TO 4
250  FOR I=1 TO 4
```

```
260  PRINT AT R+J-1,C+I-1;CHR$ A(I,J)
270  NEXT I
280  NEXT J
290  NEXT C
300  NEXT R
310  STOP
```

Obr. 2.15.

Samozřejmě můžete panáčka vytisknout na kteroukoliv pozici na obrazovce - záleží na proměnných r a c v prvním programu. Můžete také změnit čísla v datech nebo i rozměr obrázku - to vše už záleží jen na vašich nápadech.

Nyní tedy víte to podstatné o mozaikové grafice, přejdeme tedy k jistě zajímavější a užitečnější jemné grafice.

Kapitola 3: Jemná grafika

Pro seriozní práci s grafikou Spectra ale samozřejmě nevystačíme s mozaikovou grafikou a proto využijeme možnosti Spectra ovládat samostatně každý bod obrazovky. Každá znaková pozice na obrazovce se skládá s 8*8 bodů. Máme 32 znaků na řádku -tzn. 32*8=256 bodů na řádku; a 22 řádků, to je 22*8=176 bodů na výšku.

V paměti počítače jsou body uloženy pro úsporu po osmi bodech v jednom bajtu, tzn. že celková paměť pro bodový obraz je 256*176 to celé děleno 8, tedy 5632 bajtů, v nichž každý bit může mít buď barvu INK nebo PAPER. (Ve skutečnosti je to více, protože musíme počítat s dvěmi spodními dialogovými řádky a s pamětí pro barevnou informaci).

Barevná podoba obrázku je uložena za bodovou a Spectrum narozdíl od jiných počítačů nemůže mít každý bod v jiné barvě (to by zabralo příliš mnoho paměti), ale vždy 8*8 bodů (jeden znak) může mít svůj INK a PAPER. To může dělat určité problémy, jak uvidíme později.

Rozsvícení a zhasnutí jednoho bodu

V jemné grafice můžeme jednoduše ovládat jeden každý bod na obrazovce. Instrukce PLOT ve tvaru:

```
100  PLOT x,y
```

rozsvítí jeden bod na pozicích x,y. (Tzn. že tento bod bude v barvě INK.)

Máme nyní na obrazovce mnohem více bodů než v mozaikové grafice. Hodnota x může nabývat velikost od 0 do 255 (sloupec) a y od 0 do 175 (řádek). Bod o souřadnicích 0,0 je vlevo dole. To je odlišnost od příkazu PRINT AT, kde navíc uvádíme nejprve řádek a potom sloupec (u PLOT naopak).

Takže levý horní roh obrazovky má souřadnice 0,175, pravý spodní 255,0 a pravý horní 255,175. Následující program kreslí body na náhodných pozicích po celé obrazovce:

```
100  REM NAHODNE BODY
110  FOR N=1 TO 1000
120  LET X=INT (RND*256)
130  LET Y=INT (RND*176)
140  PLOT X,Y
150  NEXT N
```

Stejně jako v textovém módu i zde Spectrum sleduje pozici jakéhosi kurzoru, který není nijak zobrazován a je uložen v paměti. Po každé grafické operaci se kurzor přesouvá do nové pozice, dané provedenou operací.

Když Spectrum zapnete nebo po instrukci CLS je bodový kurzor automaticky nastaven na pozici 0,0. Příkaz PLOT x,y změní pozici kurzoru na hodnoty X a Y.

Pro smazání bodu na obrazovce se užívá instrukce INVERSE 1 v příkazu PLOT. Tato instrukce přehodí INK za PAPER a naopak, takže pokud byl bod na pozici x,y v barvě INK, přepne se po PLOT INVERSE 1 na PAPER. Takže instrukce pro smazání bodu může vypadat následovně:

```
150  PLOT INVERSE 1;X,Y: INVERSE Ø
```

Více se příkazem INVERSE budeme zabývat v kapitole 6.

Užití barev v jemné grafice

Stejně jako text, i jednotlivé body můžeme kreslit v různých barvách. Bod je tištěn barvou INK a proto jej nebude PAPER ovlivňovat. Tento program kreslí náhodné body v náhodných barvách:

```
100  REM NAHODNE BAREVNE BODY
110  FOR N=1 TO 1000Ø
120  LET I=INT (RND*7)
130  LET X=INT (RND*256)
140  LET Y=INT (RND*176)
150  INK I
160  PLOT X,Y
170  NEXT N
```

Jistě jste si všimli, že bod je nakreslen nějakou barvou a pak se náhle někde vedle něj vykreslí jiný bod jinou barvou a tento náš starý bod se přepne do nové barvy. To je důsledek oné nedokonalosti Spectra a je to určité omezení pro kreslení barevných obrázků.

Míchání textu a jemné grafiky

Některé počítače ukládají text a body do jiné oblasti paměti a jsou u nich určité problémy při míchání textu a bodů. Spectrum toto ukládá všechno do jedné paměti a je proto jednoduché míchat tyto dva styly.

Musíte si pamatovat, že pozici na řádku (tedy vlastně sloupec) získáte snadno: x=8*c, kde x je bodový sloupec a c textový. Pro výpočet řádku musíte užít rovnici y=175-8*r pro výpočet řádku y.

Pokud chcete opačný převod (tedy z bodových na textové), musíte zaokrouhlovat instrukcí INT, asi takto:

```
r=INT ((175-y)/8)
c=INT (x/8)
```

Znovu připomínám, že PRINT AT vyžaduje nejprve řádek a potom sloupec.

Malým problémem je to, že instrukce PRINT AT při nastaveném OVE R Ø smaže body v místě, kam píše. Instrukce PLOT nic nemaže.

Kreslení čar

Možnost kreslit jednotlivé body je sice pěkná, ale často chceme udělat čáru. Zde nemusíme postupovat tak složitě jako u mozaikové grafiky, počítač je sám schopný udělat čáru, jakou potřebujeme. Pro toto slouží instrukce

```
DRAW x,y
```

Ale pozor! Hodnoty x a y tentokrát neoznačují absolutní bod na obrazovce, ale jsou relativní vůči bodovému kurzoru. Např. po příkazu PLOT 30,20 se tento kurzor nastaví na pozici 30,20. Potom instrukce DRAW 30,20 způsobí vykreslení čáry, jejíž konec bude o 30 bodů dál doprava a o 20 bodů výš. Tzn. že čára skončí v bodě o souřadnicích x=60 (30+30) a y=40 (20+20).

Tato metoda kreslení čar je užitečná, protože nemusíme počítat absolutní pozice každého bodu, ale stačí nám znát relativní velikost posunutí.

Hodnoty v instrukci DRAW mohou být záporné. Záporná hodnota pro x způsobí kreslení čáry směrem doleva a pro y směrem dolů.

Stejně jako u PLOT i u DRAW jsou body čáry kresleny v aktuálně nastavené barvě INK.

Kreslení čáry mezi dvěma určitými body

V mnoha případech chceme ale nakreslit čáru mezi dvěma určitými body s pevně stanovenými souřadnicemi, řekněme x1,y1 a x2,y2.

Prvním krokem musí být nastavení kurzoru na počátek čáry. To provedeme jednoduše - instrukcí PLOT x1,y1. Dalším krokem je zjištění hodnot pro instrukci DRAW. Ty získáme také snadno-jsou to rozdíly odpovídajících si souřadnic obou bodů, tedy x=x2-x1 a y=y2-y1. Nemusíme tyto hodnoty ani nijak počítat, můžeme je vložit rovnou do instrukce DRAW:

```
100  DRAW X2-X1,Y2-Y1
```

Hodnota x bude záporná pokud x2 bude menší než X1 (bod 2 leží vlevo od bodu 1). Hodnota y bude záporná, pokud y2<y1 (bod 2 leží pod bodem 1).

Kreslení pruhů

Nyní už umíme kreslit body a čáry a to v barvách, můžeme se tedy pustit i do složitějších grafických procedur. Začneme jednoduchými pohybujícími se pruhy různých barev.

Princip tohoto kreslení je takovýto: nejprve vybereme dva náhodné body (x1,y1 a x2,y2) a nakreslíme mezi nimi čáru. Dále k bodům přičteme malé číslo a nakreslíme další čáru.

Konstanta přičítaná k hodnotám x a y způsobí, že na obrazovce vidíme jakýsi barevný pruh, pohybující se přes obrazovku.

Při stálém přičítání se může stát, že hodnota x nebo y bude moc velká, že vyjede z obrazovky. Některé počítače toto tolerují,

ale Spectrum při pokusu nakreslit něco mimo obrazovku hned zastaví program a oznámí nám to chybovým hlášením. Proto musíme vždy zkontrolovat, jestli x nepřekročilo 255 nebo je menší než 0 a y nepřekročilo 175 nebo je menší než 0. Pokud se to přihodí, změní se znaménko čísla připočitávaného k té které hodnotě a souřadnice je změněna na správnou hodnotu. To vypadá, jako by řádek jel zpátky přes obrazovku, jako by se odrážel od jejích okrajů.

```
100  REM POHYBUJICI SE CARY
110  FOR N=1 TO 20
120  LET DX1=2-INT (RND*5)
130  LET DX2=2-INT (RND*5)
140  LET DY1=2-INT (RND*5)
150  LET DY2=2-INT (RND*5)
160  LET X1=20+INT (RND*200)
170  LET Y1=20+INT (RND*130)
180  LET X2=20+INT (RND*200)
190  LET Y2=20+INT (RND*130)
200  INK INT (RND*7)
210  PAPER 7
220  CLS
230  FOR K=1 TO 500
240  PLOT X1,Y1
250  DRAW X2-X1,Y2-Y1
260  LET X1=X1+DX1
270  IF X1<=255 AND X1>=0 THEN GOTO 310
280  LET DX1=-DX1
290  LET X1=X1+DX1
300  INK INT (RND*7)
310  LET X2=X2+DX2
320  IF X2<=255 AND X2>=0 THEN GOTO 360
330  LET DX2=-DX2
340  LET X2=X2+DX2
350  INK INT (RND*7)
360  LET Y1=Y1+DY1
370  IF Y1<=175 AND Y1>=0 THEN GOTO 410
380  LET DY1=-DY1
390  LET Y1=Y1+DY1
400  INK INT (RND*7)
410  LET Y2=Y2+DY2
420  IF Y2<=175 AND Y2>=0 THEN GOTO 460
430  LET DY2=-DY2
440  LET Y2=Y2+DY2
450  INK INT (RND*7)
460  NEXT K
470  NEXT N
```

Obr. 3.4.

Aby byl obrázek barevnější, mění se INK po každém "odražení" od okraje obrazovky.

Program dobře demonstruje již několikrát zmíněnou vlastnost Spectra, totiž nutnost mít v znakovém čtverci pouze jednu barvu INK. I tak ale může vytvářet zajímavé abstraktní obrazce.

Tvorba abstraktních obrazců

Jiný druh abstraktního obrazce můžete dostat pomocí následujícího programu. Program vybere náhodný bod na obrazovce a nakreslí od něj radiálně čáry k okraji obrazovky. Změnou pozice bodu a rozteče čar můžeme získat zajímavé obrazce.

```
100  REM ABSTAKTNI OBRAZKY
110  INK 0: PAPER 7
120  FOR K=1 TO 100
130  CLS
140  LET CX=20*INT (RND*200)
150  LET CY=20*INT (RND*130)
160  LET S=2+INT (3*RND)
170  FOR X=0 TO 255 STEP S
180  PLOT CX,CY
190  DRAW X-CX,0-CY
200  PLOT CX,CY
210  DRAW X-CX,175-CY
220  NEXT X
230  FOR Y=0 TO 175 STEP S
240  PLOT CX,CY
250  DRAW 0-CX,Y-CY
260  PLOT CX,CY
270  DRAW 255-CX,Y-CY
280  NEXT Y
290  PAUSE 500
300  NEXT K
```

Obr. 3.5.

Můžeme ale vytvářet barevnější obrazce, užívaje náhodný INK a PAPER. Pokud doplníte do minulého programu následující řádky, stanou se obrázky barevnějšími.

```
152  LET P=INT (RND*8)
154  LET I=INT (RND*8)
156  IF P=I THEN GOTO 154
162  INK I
164  PAPER P
166  BORDER INT (RND*8)
168  CLS
```

Příkaz CLS způsobí přebarvení obrazovky na PAPER p, kde p je náhodné číslo. Čáry budou potom zobrazovány v INK i.

Kreslení přerušovaných čar

Příkaz DRAW kreslí plnou čáru mezi body x1,y1 a x2,y2, ale někdy potřebujeme mezi těmito body nakreslit přerušovanou čáru. Spectrum pro takovouto čáru nemá žádný příkaz, proto musíme využít příkaz PLOT a kreslit jeden bod čáry za druhým.
Program nejprve určí dva náhodné body a potom vypočítá, jak jsou od sebe daleko, tedy kolik bodů je dělí v x-ové a y-ové vzdálenosti.

```
100  REM PRERUSOVANA CARA
110  FOR N=1 TO 20
120  LET X1=INT (RND*255)
130  LET Y1=INT (RND*175)
140  LET X2=INT (RND*255)
150  LET Y2=INT (RND*175)
160  LET XS=1
170  LET YS=1
180  LET XI=1
190  LET YI=1
200  LET DX=X2-X1
210  LET DY=Y2-Y1
220  IF DX<0 THEN LET XS=-1
230  IF DY<0 THEN LET YS=-1
240  LET NX=ABS (DX)
250  LET NY=ABS (DY)
260  IF NX>=NY THEN LET NP=NX: LET YI=NY/NX
270  IF NY>NX THEN LET NP=NY: LET XI=NX/NY
280  PLOT X1,Y1
290  LET S=INT (RND*3)+2
300  FOR J=0 TO NP STEP S
310  LET X=X1+XS*INT (J*XI+.5)
320  LET Y=Y1+YS*INT (J*YI+.5)
330  PLOT X,Y
340  NEXT J
350  PLOT X2,Y2
360  NEXT N
```

Obr. 3.8.

Tento program si samozřejmě můžete upravit pro čáry s různými délkami jednotlivých čárek nebo pro čerchovanou čáru - to už nechám na vás.

Kreslení trojúhelníků

Nejjednodušším útvarem, který můžeme kreslit, je trojúhelník, protože má jen tři strany a tři vrcholy. Pro nakreslení trojúhelníka potřebujeme znát souřadnice jeho tří vrcholů, tedy x1,y1, x2,y2 a x3,y3. Kreslení začne v bodě 1, přes bod 2 do 3 a zpátky do 1.
Tento program nakreslí trojúhelník s vrcholy v náhodně zvolených bodech.

```
100  REM NAHODNE TROJUHELNIKY
110  FOR S=1 TO 20
120  BORDER INT (RND*8)
130  LET P=INT (RND*8)
140  PAPER P: CLS
150  FOR N=1 TO 15
160  REM SOURADNICE NAHODNYCH BODU
170  LET X1=INT (RND*255)
180  LET Y1=INT (RND*175)
190  LET X2=INT (RND*255)
200  LET Y2=INT (RND*175)
210  LET X3=INT (RND*255)
220  LET Y3=INT (RND*175)
230  REM ZVOLENI INKU
240  LET C=INT (RND*8)
250  IF C=P THEN GOTO 240
260  INK C
270  REM KRESLENI TROJUHELNIKU
280  PLOT X1,Y1
290  DRAW X2-X1,Y2-Y1
300  DRAW X3-X2,Y3-Y2
310  DRAW X1-X3,Y1-Y3
320  NEXT N
330  PAUSE 200
340  NEXT S
350  INK 0
```

Obr. 3.10.

Obrazy v zrcadle

Kreslení náhodných trojúhelníků je sice pěkné, ale můžeme dostat atraktivnější obraz, pokud využijeme zrcadlového principu. Tzn. že obrázky jsou kresleny souměrně podle středu obrazovky, která se takto vlastně dělí na čtyři čtvrtiny.

V principu vlastně nakreslíme první trojúhelník do pravé horní čtvrtiny, přičemž souřadnice prvního bodu trojúhelníka x1,y1 přičteme k souřadnícím středu obrazovky. Potom využijeme příkaz DRAW jako v minulém programu. Druhý trojúhelník dostaneme stejně, ale hodnotu y1 odečteme od y-ové souřadnice středu obrazovky. Pro otočení trojúhelníku vzhůru nohama budeme v třech příkazech DRAW odečítat vždy y1-y2 místo y2-y1. Pro levou polovinu obrazu je to podobné, ale nyní odečítáme a přehazujeme i x-ové souřadnice.

Dostatečně to osvětlí následující program, jehož výsledkem je obrázek podobný útvaru v kaleidoskopu:

```
100  REM KALEIDOSKOP
110  FOR S=1 TO 20
120  BORDER INT (RND*8)
130  LET P=INT (RND*8)
140  PAPER P:CLS
150  FOR N=1 TO 20
160  REM URCENI VRCHOLU
170  LET X1=INT (RND*127)
```

```
180  LET Y1=INT (RND*87)
190  LET X2=INT (RND*127)
200  LET Y2=INT (RND*87)
210  LET X3=INT (RND*127)
220  LET Y3=INT (RND*87)
230  REM ZVOLENI INKU
240  LET C=INT (RND*8)
250  IF C=P THEN GOTO 240
260  INK C
270  REM KRESLENI TROJUHELNIKU
280  PLOT 128+X1,88+Y1
290  DRAW X2-X1,Y2-Y1
300  DRAW X3-X2,Y3-Y2
310  DRAW X1-X3,Y1-Y3
320  PLOT 128+X1,88-Y1
330  DRAW X2-X1,Y1-Y2
340  DRAW X3-X2,Y2-Y3
350  DRAW X1-X3,Y3-Y1
360  PLOT 128-X1,88-Y1
370  DRAW X1-X2,Y1-Y2
380  DRAW X2-X3,Y2-Y3
390  DRAW X3-X1,Y3-Y1
400  PLOT 128-X1,88+Y1
410  DRAW X1-X2,Y2-Y1
420  DRAW X2-X3,Y3-Y2
430  DRAW X3-X1,Y1-Y3
440  NEXT N
450  PAUSE 200
460  NEXT S
470  INK 0
```

Obr. 3.11.

Kreslení obdélníků a čtverců

Nyní se naučíme kreslit obrazce se čtyřmi stranami a vrcholy, tedy obdélníky. Známe tedy čtyři vrcholy, v jednom začneme a pokračujeme pomocí DRAW přes ostatní zpět k prvnímu. Následující program kreslí obdélník s body o souřadnicích 40,50 , 140,50 , 140,100 a 40,100.

```
100  REM OBDELNIK SE CTYRMI ZADANYMI BODY
120  LET X1=40
130  LET Y1=50
140  LET X2=140
150  LET Y2=50
160  LET X3=140
170  LET Y3=100
180  LET X4=40
190  LET Y4=100
200  PLOT X1,Y1
210  DRAW X2-X1,Y2-Y1
220  DRAW X3-X2,Y3-Y2
230  DRAW X4-X3,Y4-Y3
240  DRAW X1-X4,Y1-Y4
```

Obr. 3.14.

Využití šířky a výšky

Kreslení obdélníka pomocí čtyř vrcholů není zrovna nejefektivnější, můžeme ale využít jeho šířku a výšku. Potom nám stačí znát souřadnice jednoho bodu a šířka a výška pro vykreslení celého obdélníka.
Následující program kreslí obdélník tímto způsobem, přičemž je zadán levý spodní vrchol. Potom pomocí PLOT přesuneme bodový kurzor do tohoto bodu. První strana půjde směrem doprava, tzn. že y-ová souřadnice se nemění. Další strana jde nahoru (x-ová souřadnice se nemění), další zpět doleva a poslední dolů.

```
100  REM OBDELNIK POMOCI SIRKY A VYSKY
120  LET X=50
130  LET Y=50
140  LET W=100
150  LET H=50
160  REM POCATECNI BOD
170  PLOT X,Y
180  DRAW W,0
190  DRAW 0,H
200  DRAW -W,0
210  DRAW 0,-H
```

Obr. 3.16.

Čtverec je speciálním případem obdélníka, kde výška a šířka jsou stejné. Stačí tedy zadat w=h nebo využívat jenom jednu z těchto proměnných.

Oštření pro okraje obrazovky

Pokud budeme chtít kreslit 100 bodů široký čtverec od sloupce 200, dojde ke kolizi, protože konec čtverce by byl větší než 255. Některé počítače překročení rozsahu obrazovky tolerují, ale Sectrum ne. To je třeba si zapamatovat!

```
100  REM NAHODNE OBDELNIKY S OCHRANOU
110  REM PROTI VYPADNUTI Z OBRAZOVKY
120  FOR S=1 TO 20
130  BORDER INT (RND*8)
140  LET P=INT (RND*8)
150  PAPER P
160  CLS
170  FOR N=1 TO 10
180  REM URCENI PARAMETRU OBDELNIKA
190  LET X=INT (RND*240)
200  LET Y=INT (RND*165)
210  LET W=10+INT (RND*100)
220  LET H=10+INT (RND*100)
230  LET C=INT (RND*8)
240  IF C=P THEN GOTO 230
250  INK C
260  REM KONTROLA SPRAVNOSTI
270  IF X+W>255 THEN W=255-X
```

```
280  IF Y+H>175 THEN H=175-Y
290  REM KRESLENI OBDELNIKA
300  PLOT X,Y
310  DRAW W,0
320  DRAW 0,H
330  DRAW -W,0
340  DRAW 0,-H
350  NEXT N
360  PAUSE 200
370  NEXT S
380  INK 0: PAPER 7
```

Obr. 3.17.

Program vždy zkontroluje, zda náhodně vybraná x-ová souřadnice počátku plus šířka není větší než 255. Pokud ano, zmenší šířku do příslušných mezí. Totéž proběhne pro y-ovou souřadnici a výšku.

Kapitola 4: Kreslící techniky

Takže nyní umíme kreslit čáry a obdélníky, ale většinou budete chtít kreslit i mnohoúhelníky, kruhy a elipsy. Je také potřeba kreslit tyto útvary s určitým úhlem od vodorovné osy. V této kapitole se budeme snažit vyřešit tyto problémy, a začneme kruhem. Máme několik metod pro kreslení kruhu.

Užití příkazu CIRCLE

Nejjednodušší způsob pro nakreslení kruhu je využití příkazu CIRCLE, který má podobu

```
CIRCLE x,y,r
```

kde x,y jsou souřadnice středu a r je poloměr. Pokud chceme nakreslit kruh se středem uprostřed obrazovky a poloměrem 50, stačí napsat: CIRCLE 128,88,50 a objeví se požadovaná kružnice. Funkci příkazu CIRCLE dále osvětlí následující program, kreslící soustředné kružnice se středem uprostřed obrazovky.

```
100  REM KRUHY POMOCI PRIKAZU CIRCLE
110  LET X=128
120  LET Y=88
130  FOR R=10 TO 80 STEP 10
140  CIRCLE X,Y,R
150  NEXT R
```

Obr. 4.1.

Musíte si dávat pozor na to, že celá kružnice musí ležet na obrazovce, ani kousek z ní nesmí "vyjet" - to by způsobilo chybové hlášení.
Pokud chceme tedy vytvořit program pro náhodné kreslení kruhů na obrazovce, musíme jej ošetřit proti tomuto "vyjetí". To se dá udělat např. tak, že nejprve zvolíme poloměr r a potom náhodně souřadnice středu, omezené tímto poloměrem. X-ovou souřadnici středu tedy dostaneme příkazem

```
X=R+INT (RND(*255-2*R))
```

Následující program tedy kreslí na obrazovku náhodně umístěné kružnice, s ohledem na její okraje.

```
100  REM NAHODNE KRUZNICE POMOCI CIRCLE
110  FOR S=1 TO 10
120  CLS
130  REM RAMECEK
140  PLOT 0,0
150  DRAW 255,0
160  DRAW 0,175
170  DRAW -255,0
180  DRAW 0,-175
190  REM KRESLENI KRUZNIC
```

```
200  FOR N=1 TO 15
210  LET R=5+INT (RND*50)
220  LET X=R+INT (RND*(255-2*R))
230  LET Y=R+INT (RND*(175-2*R))
240  CIRCLE X,Y,R
250  NEXT N
260  PAUSE 200
270  NEXT S
```

Obr. 4.3.

Pokud potřebujete kružnici, jejíž část by měla vyjet za okraj obrazovky, je lepší použít některou jinou metodu kreslení kružnic kombinovanou s nekreslením těchto bodů nebo s jejich vykreslením jakoby se přesunuly na opačný okraj orazovky. Proto se nyní podíváme na jiné způsoby kreslení kružnic.

Metoda kvadratické rovnice

První metoda pro kreslení kružnice vychází z matematické definice kružnice.

Představme si nyní kruhovou výseč, kde bod A je středem kruhu, body B a C leží na kružnici na úsečkách ohraničujících výseč a bod D leží na kolmici spuštěné z bodu C na úsečku AB. (Raději si to nakreslete.) Potom vlastně úsečky AB a AC mají délku r, tedy poloměr kružnice. Dále úsečku AD si označíme x a úsečku CD y.

Nakreslete si celý obrázek tak, aby úsečka AB byla vodorovná, tzn. že y-ová souřadnice bodů A a B je stejná.

Dostáváme tedy pravoúhlý trojúhelník ADC, kde AC je přepona, jejíž délka je r, AD=x a CD=y. Potom podle Pythagorovy věty

```
(AC*AC)=(X*X)+(Y*Y)
```

což můžeme zapsat jako

```
r*r=x*x+y*y
```

a z toho odvodíme

```
y=SQR ((r*r) - (x*x))
```

Hodnoty x a y jsou vlastně souřadnice bodu kružnice vzhledem k jejímu středu, označenému souřadnicemi cx a cy.

Pokud tedy chceme nakreslit kružnici, musíme brát hodnoty x z intervalu -r až r a pro každou z nich vypočítat odpovídající y. Ale protože jde o kvadratickou rovnici, dostáváme pro y vždy dva kořeny, jeden kladný a jeden záporný. Potom např. pro kružnici s poloměrem 50 budeme y počítat 100-krát a kreslit 200 bodů.

Následující program kreslí kružnici touto metodou. Má sice pevně zadané souřadnice středu a poloměr, nic vám ale nebrání to změnit podle vlastní potřeby; program je ošetřen proti vyjetí z obrazovky.

```
100  REM KRUZNICE KVADRATICKOU METODOU
110  LET CX=128
```

```
120  LET CY=96
130  LET R=50
140  FOR X=-R TO R
150  LET Y=SQR(R*R-X*X)
160  IF CX+X>255 THEN LET X=255-CX
170  IF CX+X<0 THEN LET X=0-CX
180  IF CY+Y>175 THEN LET Y=175-CY
190  IF CY+Y<0 THEN LET Y=0-CY
200  PLOT CX+X,CY+Y
210  PLOT CX+X,CY-Y
220  NEXT X
```

Obr. 4.6.

Tato rutina ale zvláště pro příliš velké kruhy zabere mnoho času.Je to proto, že funkce SQR na Spectru je příliš pomalá. Je tedy třeba využít nějakou jinou metodu kreselní kruhu.
Při kreslení větších kružnic se dále může stát, že zvláště levá a pravá část kružnice bude mít body daleko od sebe, bude jakoby vytečkovaná. Tomu zabráníme zmenšením kroku ve smyčce x na .5 místo stávající 1.

Goniometrická metoda

Stejně jako z jednotlivých bodů, můžeme nakreslit obraz kružnice pomocí krátkých čárek, vzájemně na sebe navazujících. K tomu ale potřebujeme využít jednoduché goniometrické funkce.
Vezměte si obrázek z kvadratické metody. Zde A je střed kruhu, B a C leží na kružnici a ohraničují výseč (AB je vodorovné) a D je pata kolmice spuštěné z bodu C na úsečku AB. Označme si nyní AD=dx a CD=dy. Dále si zde nakreslete úsečku BC, která bude vlastně částí kreslené kružnice. Dále si konvexní úhel BAC označme th.
Pro nakreslení BC musíme znát souřadnice bodu C. Jsou to vlastně ony dx a dy, vůči bodu A. A zde nastupuje goniometrie.
Pro získání délky dy (CD) využijeme funkci SIN (th). Z definice funkce SIN v pravoúhlém trojúhelníku ADC dostaneme

SIN (th)=CD/AC

a protože AC je vlastně poloměr r a CD=dy, můžeme napsat

SIN (th)=dy/r

No a my potřebujeme znát hodnotu dy, proto upravíme na

dy=r*SIN (th)

Podobně získáme i délku dx, zde ale využijeme funkci COS. Dostaneme tedy vztah

COS (th)=AD/AC , z toho vyplývá dx=r*COS (th)

Pro nakreslení kruhu musíme nejprve umístit kurzor do bodu B, který má souřadnice dx=r a dy=0. To provedeme pomocí PLOT x1,y1, kde x1=cx+r a y1=cy. (cx,cy jsou souřadnice středu kruhu)
Další krok je nalézt souřadnice x2,y2 bodu C. Ty vypočítáme takto:

x2=cx+dx=cx+r*COS (th)
y2=cy+dy=cx+r*COS (th)

a nakreslíme čárku pomocí DRAW x2-x1,y2-y1. Pro další čárku se potom bude x1=x2 a y1=y2 (začátek čárky v konci té minulé). Tato procedura bude probíhat pro hodnoty th od 0 do 360 stupňů.
Jak budeme určovat úhel th? Pro co nejkulatější kruh je třeba udělat co nejvíce kroků. Proto budeme th zvětšovat v závislosti na poloměru kruhu. Tak pro poloměr 60 bude krok 360/60, to je 6 stupňů, proto th vždy zvětšíme o 6 stupňů.
Jenomže Spectrum nepracuje se stupni, ale s radiány. Zde je ale lehká pomoc - 360 stupňů je vlastně 2*PI radiánů, proto velikost kroku dostaneme jako 2*PI/r. (Číslo PI je uloženo v paměti počítače.)
Tento program kreslí náhodné kružnice pomocí právě popsané metody. Program vlastně počítá r souřadnic bodu C.

```
100  REM NAHODNE KRUZNICE
110  REM POMOCI GONIOMETRICKE METODY
120  BORDER 6
130  FOR S=1 TO 10
140  CLS
150  FOR N=1 TO 10
160  LET R=10+INT (RND*40)
170  LET X=INT (RND*255)
180  LET Y=INT (RND*175)
190  GOSUB 500
200  NEXT N
210  PAUSE 100
220  NEXT S
230  STOP
500  REM PODPROGRAM KRESLENI KRUZNICE
510  LET DT=2*PI/R
520  REM URCENI PRVNIHO BODU
530  LET TH=0
540  LET X1=X+INT (R*COS TH)
550  LET Y1=Y+INT (R*SIN TH)
560  REM KONTROLA PRO OKRAJ OBRAZU
570  IF X1>255 THEN LET X1=255
580  IF X1<0 THEN LET X1=0
590  IF Y1>175 THEN LET Y1=175
600  IF Y1<0 THEN LET Y1=0
610  PLOT X1,Y1
620  FOR I=0 TO R
630  LET TH=TH+DT
640  LET X2=X+INT (R*COS TH)
650  LET Y2=Y+INT (R*SIN TH)
660  IF X2>255 THEN LET X2=255
```

```
670  IF X2<0 THEN LET X2=0
680  IF Y2>175 THEN LET Y2=175
690  IF Y1<0 THEN LET Y2=0
700  DRAW X2-X1,Y2-Y1
710  LET X1=X2
720  LET Y1=Y2
730  NEXT I
740  RETURN
```

Obr. 4.8.

Samotná kruznice je kreslena podprogramem od řádku 500, který můžete využít i ve svých programech. Vstupuje se do něj se souřadnicemi středu x,y a poloměrem r. Podprogram také hlídá okraje obrazovky a pokud by kružnie chtěla vyjet a tím způsobit chybové hlášení, je to včas zjištěno a místo té části kruhy, která by byla mimo, se kreslí čára po okraji obrazovky.

Rotační metoda

Jiný způsob kreslení kružnice je počítat souřadnice jejích bodů pomocí otáčení vždy o stejný úhel. Hodnoty x2,y2 počítáme z x1 a y1, pomocí poloměru a konstantního úhlu.
Pokud si představíme kruhovou výseč s jednou stranou vodorovnou a na ní vrchol o souřadnicích x1,y1 a druhý vrchol na druhé straně se souřadnicemi x2,y2; potom tedy y1 vůči středu je nula a x1=r. Můžeme tedy napsat

```
x2=x1*COS th
y2=x1*SIN th
```

kde th je úhel u vrcholu výseče, tak jako v minulém příkladě. Pokud si to nakreslíte, tak vám to asi bude jasnější, já to zde bohužel nemám jak nakreslit.
Nyní uvažujme situaci, kdy jedna strana výseče jde kolmo vzhůru od středu a druhá je od ní doleva o úhel th. Potom x1=0 a y1=r, takže dostaneme

```
x2=-y1*SIN th
y2=y1*COS th
```

Pokud tyto dvě a dvě rovnice zkombinujeme, dostaneme obecné rovnice pro výpočet souřadnic x2,y2 z x1,y1. Tyto jsou:

```
x2=x1*COS th - y1*SIN th
y2=x1*SIN th + y1*COS th
```

Výhodou tohoto výpočtu je to, že úhel th zůstává konstantní, takže hodnoty SIN th a COS th můžeme uložit do proměnných, což zrychlí běh programu (SIN a COS jsou dosti pomalé).
Tento program kreslí náhodné kruhy pomocí této metody.

```
100  REM NAHODNE KRUHY POMOCI
110  REM ROTACNI METODY
120  FOR N=1 TO 15
```

```
130  LET R=10+INT (RND*40)
140  LET X=R+INT (RND*(255-2*R))
150  LET Y=R+INT (RND*(175-2*R))
160  GOSUB 500
170  NEXT N
180  STOP
500  REM PODPROGRAM PRO KRESLENI KRUHU
510  LET TH=2*PI/R
520  LET X1=R
530  LET Y1=0
540  PLOT X+X1,Y+Y1
550  FOR I=1 TO R
560  LET X2=X1*COS TH - Y1*SIN TH
570  LET Y2=X1*SIN TH + Y1*COS TH
580  DRAW X2-X1,Y2-Y1
590  LET X1=X2
600  LET Y1=Y2
610  NEXT I
620  RETURN
```

Obr. 4.12.

Kreslení mnohoúhelníků

Pokud snížíme počet kroků a tím i počet vykreslených čar při kreslení kružnice, dostaneme útvar zvaný polygon neboli mnohoúhelník. Pokud použijeme goniometrickou metodu, prudce se zvýší hodnota, o kterou budeme zvětšovat úhel th. Bude to vlastně 2*PI/8. Tento program kreslí osmiúhelník právě popsaným způsobem:

```
100  REM OSMIUHELNIK
110  LET XC=128
120  LET YC=88
130  LET R=50
140  LET DT=2*PI/8
150  LET TH=0
160  LET X1=XC+R
170  LET Y1=YC
180  PLOT X1,Y1
190  FOR N=1 TO 8
200  LET TH=DT*N
210  LET X2=XC+R*COS TH
220  LET Y2=YC+R*SIN TH
230  DRAW X2-X1,Y2-Y1
240  LET X1=X2
250  LET Y1=Y2
260  NEXT N
```

Obr. 4.13.

Pokud bychom chtěli nakreslit třeba šestiúhelník, byl by počet kroků ve smyčce n 6 a hodnota dt by byla 2*PI/6.
Pro nakreslení obecného polygonu musíme mít proměnnou ns (označující počet stran). Program upravíme pro ns kroků. Hodnota kroku pro zvětšování úhlu th bude dt=2*PI/ns. Abys viděl, jak to funguje, zkus následující program.

```
100  REM SOUSTREDENE POLYGONY
110  LET R=12
120  LET XC=128
130  LET YC=88
140  FOR K=3 TO 9
150  LET NS=K
160  LET DT=2*PI/NS
170  LET X1=XC+R
180  LET Y1=YC
190  PLOT X1,Y1
200  FOR N=1 TO NS
210  LET TH=N*DT
220  LET X2=XC+R*COS TH
230  LET Y2=YC+R*SIN TH
240  DRAW X2-X1,Y2-Y1
250  LET X1=X2
260  LET Y1=Y2
270  NEXT N
280  LET R=R+12
290  NEXT K
```

Obr. 4.14.

Program nakreslí soustředné mnohoúhelníky se společným středem v bodě 128,88 (střed obrazovky). Začne trojúhelníkem a postupně zvětšuje poloměr, až dojde k devítiúhelníku.
Program lze lehce upravit pro tisk nejrůznějších n-úhelníků kdekoliv po obrazovce.

Kreslení mnohoúhelníků pomocí rotace

Stejně jako můžeme kreslit rotační technikou kružnice, můžeme takto kreslit i mnohoúhelníky. Pro osmiúhelník bude úhel th roven 2*PI/8 a osmiúhelník bude vykreslen tímto programem:

```
100  REM OSMIUHELNIK ROTACNI METODOU
110  LET XC=128
120  LET YC=88
130  LET R=50
140  GOSUB 500
150  STOP
500  REM KRESLICI PROCEDURA
510  LET X1=R
520  LET Y1=0
530  LET TH=2*PI/8
540  LET SN=SIN TH
550  LET CN=COS TH
560  PLOT XC+X1,YC+Y1
570  FOR N=1 TO 8
580  LET X2=X1*CN-Y1*SN
590  LET Y2=X1*SN+Y1*CN
600  DRAW X2-X1,Y2-Y1
610  LET X1=X2
620  LET Y1=Y2
630  NEXT N: RETURN
```

Obr. 4.16.

Stejné jako předtím můžeme tuto rutinu změnit pro kreslení k-úhelníka, kde k je počet stran. Potom můžeme napsat program pro kreslení obecného k-úhelníku, který umí i trojúhelník (ale pouze rovnostranný). Tento program kreslí náhodně umístěné k-úhelníky s k od 3 do 12.

```
100  REM NAHODNY POLYGON
110  FOR J=1 TO 20
120  LET R=10+INT (RND*40)
130  LET XC=R+INT (RND*(255-2*R))
140  LET YC=R+INT (RND*(175-2*R))
150  LET K=3+INT (RND*5)
160  GOSUB 500
170  NEXT J
180  STOP
500  LET TH=2*PI/K
510  LET SN=SIN TH
520  LET CN=COS TH
530  LET DX=R
540  LET DY=0
550  PLOT XC+DX,YC+DY
560  FOR N=1 TO K
570  LET XR=DX*CN-DY*SN
580  LET YR=DX*SN+DY*CN
590  DRAW XR-DX,YR-DY
600  LET DX=XR
610  LET DY=YR
620  NEXT N
630  RETURN
```

Obr. 4.17.

Kreslení hvězdic

Procedura kreslící polygony může být lehce upravena pro kreslení hvězdic. Toho dosáhneme tak, že vždy uděláme čáru zs středu do vypočítaného vrcholu polygonu. Tento program kreslí náhodné hvězdice.

```
100  REM HVEZDICE
110  FOR J=1 TO 20
120  LET R=10+INT (RND*40)
130  LET XC=R+INT (RND*(255-2*R))
140  LET YC=R+INT (RND*(175-2*R))
150  LET K=3+INT (RND*5)
160  GOSUB 500
170  NEXT J
180  STOP
500  LET TH=2*PI/K
510  LET SN=SIN TH
520  LET CN=COS TH
530  LET DX=R
540  LET DY=0
550  FOR N=1 TO K
560  LET XR=DX*CN-DY*SN
```

```
570  LET YR=DX*SN+DY*CN
580  PLOT XC,YC
590  DRAW XR,YR
600  LET DX=XR
610  LET DY=YR
620  NEXT N
630  RETURN
```

Obr. 4.18.

Natahování a stlačování útvarů

Při kreslení čtverce, polygonu nebo kruhu ovlivňují výslednou podobu objektu dvě hodnoty, W a R, které používá kreslící rutina. Změnou těchto hodnot můžeme změnit tvar obrázku.

V případě pravoúhelníka jsou zde dvě měřítka, šířka (W) a výška (H). Můžeme tedy objekt natahovat nebo stlačovat, a to buď jenom horizontálně nebo vertikálně nebo v obou směrech.

Můžeme tento nápad rozvinout dvěma proměnnými, které budou určovat měřítko pro x-ové a y-ové souřadnice. Abychom dosáhli uspokojivých výsledků, musíme mít nějaký centrální bod - střed, kolem něhož se obrázek tvoří (u kruhu a polygonu jsme na to zvyklí). Potom těmito speciálními proměnnými násobíme proměnné dx a dy, což jsou vlastně souřadnice bodu vzhledem ke středu objektu. Není vhodné manipulovat s proměnnými cx a cy (střed).

Vezměme si třeba kruh. Následující program ukazuje princip natahování a stlačování. Jsou zde dvě nové proměnné sx a sy, kterými se násobí dx a dy, což ovlivňuje výsledný obrázek.

Nejprve se sy mění od 1 do 0 a sx je rovno jedné (obrázek vlevo). Potom je sy=1 a sx se mění (vpravo). Nakonec jsou postupně měněny obě hodnoty.

```
100  REM STLACOVANI A NATAHOVANI
110  REM NA KRUZNICI
120  LET XC=40
130  LET YC=88
140  LET R=40
145  REM ZAVISLOST NA Y
150  FOR A=1 TO 0 STEP -.2
160  LET SX=1
170  LET SY=A
180  GOSUB 500
190  NEXT A
200  LET XC=210
205  REM ZAVISLOST NA X
210  FOR A=1 TO 0 STEP -.2
220  LET SX=A
230  LET SY=1
240  GOSUB 500
250  NEXT A
260  LET XC=128
265  REM ZAVISLOST NA Y A PAK NA X
270  FOR A=1 TO 0 STEP -.2
280  LET SX=1
290  LET SY=A
```

```
300  GOSUB 500
310  NEXT A
320  FOR A=1 TO 0 STEP -.2
330  LET SX=A
340  LET SY=1
350  GOSUB 500
360  NEXT A
370  STOP
495  REM KRESLICI RUTINA
500  LET DT=2*PI /R
510  LET X1=XC+SX*R
520  LET X2=YC
530  PLOT X1,Y1
540  FOR N=1 TO R
550  LET X2=XC+SX*R*COS (N*DT)
560  LET Y2=YC+SY*R*SIN (N*DT)
570  DRAW X2-X1,Y2-Y1
580  LET X1=X2
590  LET Y1=Y2
600  NEXT N
610  RETURN
```

Obr. 4.19.

Když je sx i sy rovno jedné, nakreslí se kružnice. Pokud sx<1 a sy<=1, nakreslí se elipsa s horizontální hlavní osou. Když je sx>1 a sy>=1, bude mít elipsa hlavní osu vertikálně. Pokud sx a sy budou záporné, obrázek se bude kreslit pozpátku (tzn. že nesymetrické obrázky budou zrcadlově převráceny).

Rotace obrazců

Všechny obrázky, které jsme zatím nakreslili, měly x-ovou osu vodorovnou. Ale někdy potřebujeme i pravoúhelník svírající s vodorovným směrem určitý úhel. Již jsme dokázali otáčet jednotlivé body při kreslení kruhu a polygonu, proč bychom tedy nemohli otáčet celým obrazcem. Budeme tedy otáčet každý vrchol. Vypočítáme jeho pozici relativně vzhledem ke středu otáčení a k horizontální ose.
Pro nalezení pozic dx a dy bodu užijeme:

```
xr=dx*COS (th) - dy*SIN (th)
yr=dx*SIN (th) + dy*COS (th)
```

kde th je úhel otáčení vzhledem k ose x.
Začněme tedy pravoúhelníkem a předpokládejme, že ho budeme otáčet podle levého spodního vrcholu. Pravoúhelník má výšku H a šířku W a budeme ho otáčet o úhel TH.
Začneme bodem o souřadnicích xc,yc, což je vlastně střed otáčení. První čára půjde z tohoto rohu doprava, tzn. že pro souřadnice x2 a y2 budou hodnoty dx=w a dy=0. Musíme ale počítat s otáčením, proto nám pozici bodu ovlivní ještě proměnné xr a yr.
Pro nakreslení první čáry určíme x1=xc a y1=yc. Potom vypočítáme souřadnice konce čáry a to takto

```
x2=xc+yr
y2=yc+yr
```

První čáru potom nakreslíme příkazem DRAW x2-x1,y2-y1.
Pro druhou čáru bude dx stále záviset na W, ale dy už na H. Znovu vypočítáme nové souřadnice a uděláme další čáru s počátkem v konci té staré. Tento proces se analogicky opakuje pro zbývající dvě čáry. Lépe to je vidět v následujícím programu.

```
100  REM OTACENI OBDELNIKA
110  LET XC=128
120  LET YC=40
130  LET W=100
140  LET H=30
150  LET DT=PI/6
160  LET TH=0
170  FOR N=1 TO 6
180  LET X1=XC
190  LET Y1=YC
200  PLOT X1,Y1
210  LET X2=XC+W*COS TH
220  LET Y2=YC+W*SIN TH
230  DRAW X2-X1,Y2-Y1
240  LET X1=X2
250  LET Y1=Y2
260  LET X2=XC+W*COS TH - H*SIN TH
270  LET Y2=YC+W*SIN TH + H*COS TH
280  DRAW X2-X1,Y2-Y1
290  LET X1=X2
300  LET Y1=Y2
310  LET X2=XC-H*SIN TH
320  LET Y2=YC+H*COS TH
330  DRAW X2-X1,Y2-Y1
340  LET X1=X2
350  LET Y1=Y2
360  LET X2=XC
370  LET Y2=YC
380  DRAW X2-X1,Y2-Y1
390  LET TH=TH+DT
400  NEXT N
```

Obr. 4.21.

Pokud zkombinujeme rotaci s natahováním, můžeme dostat zajímavé spirálovité útvary, což dokazuje tento program.

```
100  REM OBRAZKY S ROTACI A NATAHOVANIM
120  LET TH=0
130  LET DT=PI/12
140  LET XC=128
150  LET YC=88
160  FOR W=2 TO 70 STEP 2
170  LET X1=XC
180  LET Y1=YC
185  PLOT X1,Y1
```

```
190  LET DX=W
200  LET DY=0
210  GOSUB 500
220  LET X1=X2
230  LET Y1=Y2
240  LET DX=W
250  LET DY=W
260  GOSUB 500
270  LET X1=X2
280  LET Y1=Y2
290  LET DX=0
300  LET DY=W
310  GOSUB 500
320  LET X1=X2
330  LET Y1=Y2
340  LET DX=0
350  LET DY=0
360  GOSUB 500
370  LET TH=TH+DT
380  NEXT W
390  STOP
500  LET X2=XC+DX*COS TH - DY*SIN TH
510  LET Y2=YC+DX*SIN TH + DY*COS TH
520  DRAW X2-X1,Y2-Y1
530  RETURN
```

Obr. 4.22.

Můžeme aplikovat tento způsob rotace pro každý útvar vytvořený sérií matematických kroků. Pokud ale máme nějaký nepravidelný útvar, musíme postup trošku změnit. Nejjednodušší je vytvořit si tabulku obsahující souřadnice všech vrcholů objektu. Tyto x,y body budou otáčeny kolem určitého středu. To může být střed obrazce nebo i bod úplně mimo něj. Pro nakreslení obrazce vždy vypočítáme souřadnice následujícího vrcholu a uděláme do nich čáru. Ale někdy se může objevit problém, kdy je třeba přesunout pero do některého vrcholu bez nakreslení čáry. To vyřešíme zavedením třetí proměnné L v tabulce, která bude ovlivňovat, jestli se čára kreslit bude (L=1) nebo ne (L=0).

Když teď máme tabulku s hodnotami X,Y a L, můžeme kreslit objekt stejným způsobem jako předešlé, jenom tyto hodnoty budeme načítat z polí. Bod x1,y1 označuje počáteční bod, odkud začneme kreslit a odkud se budou vykreslovat jednotlivé čáry, které půjdou do bodu x2,y2. Tyto hodnoty jsou samozřejmě relativní vzhledem k xc,yc, což je střed otáčení.

Pokud je L rovno 0, tak se čára nekreslí a provede se jen příkaz PLOT, který přenese kurzor do nového bodu. Potom se do proměnných x1,y1 přenesou hodnoty x2,y2 jako začátek další čáry. Tento proces se opakuje, dokud není obrázek kompletní.

Pokud má obrázek více bodů, můžeme jeho kreslení urychlit, a to tak, že pro konstantní otáčecí úhel th můžeme hodnoty SIN th a COS th uložit do proměnných sn a cn (provádění goniometrických funkcí je dosti pomalé).

Tento program rotuje nesymetrický obrázek, můžete si ho samozřejmě upravit pro své vlastní obrazce.

```
100  REM ROTACE NESYMETRICKEHO OBRAZKU
110  DIM X(5)
120  DIM Y(5)
130  DIM L(5)
140  FOR N=1 TO 5
150  READ X(N),Y(N),L(N)
160  NEXT N
170  DATA 20,0,0
180  DATA 60,0,1
190  DATA 60,-15,0
200  DATA 40,0,1
210  DATA 60,15,1
220  LET XC=128
230  LET YC=88
240  LET DT=2*PI/10
250  LET TH=0
260  FOR F=1 TO 10
270  LET SN=SIN TH
280  LET CN=COS TH
290  LET X1=XC
300  LET Y1=YC
310  GOSUB 500
320  LET TH=TH+DT
330  NEXT F
340  STOP
500  PLOT X1,Y1
510  FOR N=1 TO 5
520  LET X2=XC+X(N)*CN-Y(N)*SN
530  LET Y2=YC+X(N)*SN+Y(N)*CN
540  IF L(N)=1 THEN DRAW X2-X1,Y2-Y1
550  PLOT X2,Y2
560  LET X1=X2
570  LET Y1=Y2
580  NEXT N
590  RETURN
```

Obr. 4.23.

Kapitola 5 Nové znaky a tvary

Zatím jsme při psaní na obrazovku používali pouze normální písmena a znaky a mozaikové symboly. Ale v mnoha případech se stane, že budete potřebovat i nějaké jiné znaky, třeba figurky do her nebo písmena řecké abecedy do matematických programů.
Tyto znaky by se daly kreslit pomocí sekvence příkazů PLOT a DRAW, ale to je velmi nepraktické. Naštěstí Spectrum umožňuje vytváření vlastních znaků, a to dvěma způsoby.

Užití uživatelské grafiky

Pokud si vypíšete na obrazovku všechny znaky užívané Spectrem (pomocí programu z kapitoly 2), jistě si povšimnete za mozaikovými znaky písmen A až U. Může to vypadat tak, že tyto písmena jsou v paměti uložena dvakrát.
Ale ve skutečnosti jde o 20 znaků tzv. uživatelské grafiky. Jejich podoba je uložena v paměti RAM, narozdíl od normálních znaků, takže ji můžeme měnit (ale to jde i u normálních znaků, jak uvidíme později). Tvar těchto uživatelských znaků je uložen na úplném konci paměti RAM.
Stejně jako normální znaky, jsou i uživatelské složeny jeden každý z 8 řádků - bajtů po 8 sloupcích - bitech. Každý znak tedy zabírá v paměti 8 bajtů a změnou jejich hodnoty změníme i tvar uživatelského znaku.

Programování nových znaků

Prvním krokem při tvorbě nového znaku je nakreslit si na čtverečkovaný papír mřížku o rozměrech 8*8 čtverečků a do ní si zakreslit tvar nového znaku.
Tato mřížka vlastně představuje oněch 8 řádků - bajtů, z nichž bude nový znak složen. V každém řádku je 8 pozic, které představují binární číslo (jedničky jsou body a nuly prázdné pozice). K převodu binárního čísla na dekadické slouží jednoduchá metoda, kterou zde nebudu popisovat (viz kterákoliv příručka o počítačích).
Abychom změnili oněch 8 bajtů definovaného znaku, užijeme příkazu

```
POKE adresa,hodnota
```

kde adresa je aktuální adresa toho kterého řádku uživatelského znaku a hodnota je číslo, představující binární tvar onoho řádku.
No a tak vše, co musíme udělat pro nadefinování nového uživatelského znaku, je změnit odpovídajících 8 bajtů v paměti počítače. No a navíc ani nepotřebujeme znát číselnou hodnotu adresy, na kterou máme zapisovat. Pokud chceme předefinovat znak A, stačí pro jeho první řádek napsat

```
POKE USR "a",data
```

kde data je binární hodnota prvního řádku nového znaku. Pro nadefinování druhého řádku užijeme

```
POKE USR "a"+1,data
```

kde data je binární hodnota druhého řádku. No a tak pokračujeme dále pro všech 8 řádků, takže skončíme příkazem

```
POKE USR "a"+7,data
```

kde data je binární hodnota osmého řádku nového znaku. Stejně tak pro jiný uživatelský znak než je A postupujeme úplně stejně, jenom zadáváme např. pro B: POKE USR "b"...

Abychom nový znak dostali na obrazovku, užijeme v příkazu PRINT funkci CHR$ n, kde n je kód uživatelského znaku. A má kód 144, U 164. Nebo můžeme využít grafický mód a písmena A až U, která odpovídají svým uživatelským protějškům.

Při tvorbě nového znaku nemusíme převádět binární číslo na dekadické, můžeme využít funkci BIN nnnnnnnn, kde ona n označují maximálně osmi bitové číslo, které představuje tvar definovaného řádku (jedničky jsou body v barvě INK, 0 jsou mezery, tedy PAPER).

Tento program definuje na znak A siluetu panáčka, využívajíce instrukci BIN.

```
100  REM TVORBA UZIVATELSKEHO ZNAKU POMOCI BIN
120  POKE USR "A",BIN 00011100
130  POKE USR "A"+1,BIN 00011100
140  POKE USR "A"+2,BIN 00001000
150  POKE USR "A"+3,BIN 01111111
160  POKE USR "A"+4,BIN 00001000
170  POKE USR "A"+5,BIN 00011100
180  POKE USR "A"+6,BIN 00100010
190  POKE USR "A"+7,BIN 00100010
200  FOR N=1 TO 20
210  LET R=INT (RND*20)
220  LET C=INT (RND*30)
230  PRINT AT R,C;CHR$ 144
240  NEXT N
```

Obr. 5.3.

Větší obrazce

Někdy ale potřebujeme vytvořit větší obrázek než je 8*8 bodů. Potom jej musíme rozvrhnout do více uživatelských znaků a ty potom vypisovat společně (tak jako jsme dělali panáčka v mozaikové grafice).

Jiná metoda kreslení obrazce je použita v tomto programu.

```
100  REM NAKRESLENI ZNAKU POMOCI POLE
120  DIM D(8,8)
130  FOR B=1 TO 8
```

```
140  FOR A=1 TO 8
150  READ D(A,B)
160  NEXT A
170  NEXT B
180  DATA 0,0,0,1,1,1,0,0
190  DATA 0,0,0,1,1,1,0,0
200  DATA 0,0,0,0,1,0,0,0
210  DATA 0,1,1,1,1,1,1,1
220  DATA 0,0,0,0,1,0,0,0
230  DATA 0,0,0,1,1,1,0,0
240  DATA 0,0,1,0,0,0,1,0
250  DATA 0,0,1,0,0,0,1,0
260  FOR N=1 TO 20
270  LET X=INT (RND*200)
280  LET Y=10+INT (RND*150)
290  FOR B=0 TO 7
300  FOR A=0 TO 7
310  IF D(A+1,B+1)=0 THEN GOTO 330
320  PLOT X+A,Y-B
330  NEXT A
340  NEXT B
350  NEXT N
```

Obr. 5.4.

Zde máme dvourozměrné pole d s osmi řádky a osmi sloupci. Každý prvek pole má hodnotu buď 1 nebo 0, podle toho, jestli v odpovídajícím řádku a sloupci je nebo není bod v barvě INK.
Dvě vnořené smyčky vykreslují znak na zvolenou pozici na obrazovce pomocí příkazů PLOT. Pokud je d(a,b) rovno nule, tak se PLOT neprovede.
Tato technika tvorby tvarů je dosti pomalá, zato ale dokáže jednoduše umístit tvar kamkoliv na obrazovku.

Příkaz POINT

Někdy potřebujeme zjistit, jestli nějaký bod na obrazovce je rozsvícený nebo ne, tedy jestli je v barvě INK nebo PAPER. K tomu slouží příkaz POINT. Potom řádek

```
100  LET P=POINT (X,Y)
```

kde x a y jsou souřadnice kontrolovaného bodu uloží do proměnné p 1, pokud bod x,y je v barvě INK a 0, pokud je PAPER.
Ačkoliv nám POINT prozradí, jestli je bod rozsvícený nebo ne, neřekne nám, jakou má barvu. K tomu slouží instrukce ATTR, která bude podrobněji probrána v kapitole 6.
POINT můžeme samozřejmě použít i v příkazu podmínky IF, potom

```
100  IF POINT (X,Y)=1 THEN GOTO 20
```

skočí na řádek 20, pokud bod x,y bude rozsvícený (podmínka splněna). Stejně tak může podmínkou být POINT (x,y)=0.
Nyní využijeme příkaz POINT pro zajímavé manipulace s bodovými obrazci.

Psaní znaků v souřadnicích jemné grafiky

Nejjednodušší cesta jak vypsat znak na určité místo na obrazovce je užití PRINT AT. Ale tento příkaz má svůj souřadnicový systém s pevně stanovenými řádky a sloupci a pokud chceme znak vytisknout někde mezi tyto pevně stanovené pozice, je nám příkaz PRINT AT na nic. Ale můžeme postupovat jinak.
Každý znak je matice 8*8 bodů v barvách INK nebo PAPER, jak už dávno víme. Nejjednodušší bude vypsat si znak, který chceme psát, na pozici 0,0 a odtud pomocí dvou smyček a příkazu POINT přenést jeho podobu do libovolného místa obrazovky,bez ohledu na souřadnicový systém PRINT ATu. Tento program vypisuje znak A do libovolného místa na obrazovce.

```
100  REM NAKRESLENI ZNAKU NA LIBOVOLNOU POZICI
120  PRINT AT 0,0;"A"
130  FOR N=1 TO 50
140  LET X=8+INT (RND*240)
150  LET Y=8+INT (RND*160)
160  GOSUB 400
170  NEXT N
180  STOP
400  FOR B=0 TO 7
410  FOR A=0 TO 7
420  IF POINT (A,175-B)=0 THEN GOTO 440
430  PLOT X+A,Y-B
440  NEXT A
450  NEXT B
460  RETURN
```

Obr. 5.6.

Znak je tedy nejprve vypsán do levého horního rohu a určuje tak vlastně bodovou matrici, která se bude přepisovat do zvoleného místa. Sloupce (x) budou z intervalu 0 až 7, řádky jsou v jemné grafice číslovány zdola nahoru, proto budeme pro y od 0 do 7 brát řádky 175-y.
Znak se vypisuje s levým horním rohem v pozici x,y. Tohoto způsobu psaní znaků můžeme použít např. při popisování obrázků vytvořených jemnou grafikou.

Rotace znaků

Malou změnou rutiny užívané v minulém programu dosáhneme zajímavých efektů. Např. pokud chcete psát znak vzhůru nohama, stačí v instrukci PLOT zadat y+b a dříve vrchní řádek vypisovaného znaku se stane spodním - znak se tedy otočí vzhůru nohama. Na pozici x,y bude nyní levý spodní roh vypisovaného znaku. Musíte ale dávat pozor, aby y nebylo větší než 175-8, to by způsobilo chybové hlášení.
Pokud chcete napsat znak vzhůru nohama, ale s levým horním rohem v pozici x,y, stačí zadat

```
PLOT x+a,y-8+b
```

Abychom otočili znak zleva doprava, aby vypadal jako v zrcadle, stačí zadat

```
PLOT x+8-a,y-b
```

Nyní si zkusme trošku zaexperimentovat a zadejme

```
PLOT x+b,y+a
```

což způsobí, že řádky vypisované doposud horizontálně se budou vypisovat vertikálně. Abychom znak otočili na druhou stranu, zadáme

```
PLOT x-b,y+a
```

Následující program tiskne znak ve všech čtyřech směrech.

```
100  REM ROTACE ZNAKU POMOCI BODOVEHO KOPIROVANI
120  PRINT AT 0,0;"A"
130  LET X=124
140  LET Y=96
150  GOSUB 400
160  LET X=124
170  LET Y=64
180  GOSUB 500
190  LET X=112
200  LET Y=76
210  GOSUB 600
220  LET X=144
230  LET Y=84
240  GOSUB 700
250  PRINT AT 0,0;" "
260  STOP
390  REM NORMALNI ZNAK
400  FOR B=0 TO 7
410  FOR A=0 TO 7
420  IF POINT (A,175-B)=0 THEN GOTO 440
430  PLOT X+A,Y-B
440  NEXT A
450  NEXT B
460  RETURN
490  REM VZHURU NOHAMA
500  FOR B=0 TO 7
510  FOR A=0 TO 7
520  IF POINT (A,175-B)=0 THEN GOTO 540
530  PLOT X+A,Y+B
540  NEXT A
550  NEXT B
560  RETURN
590  REM OTOCENY DOLEVA
600  FOR B=0 TO 7
610  FOR A=0 TO 7
```

```
620  IF POINT (A,175-B)=0 THEN GOTO 640
630  PLOT X+B,Y+A
640  NEXT A
650  NEXT B
660  RETURN
690  REM OTOCENY DOPRAVA
700  FOR B=0 TO 7
710  FOR A=0 TO 7
720  IF POINT (A,175-B)=0 THEN GOTO 740
730  PLOT X-B,Y-A
740  NEXT A
750  NEXT B
760  RETURN
```

Obr. 5.8.

Větší písmena

Řekněme, že bychom chtěli vytvořit dvojnásobně široký znak. Potom pro dosažení dvojnásobné šířky musíme užít dva příkazy PLOT pro horizontální směr. Samozřejmě musíme také vynechat dva volné body v případě nerozsvíceného bodu.

Podobně budeme postupovat, budem-li chtít dvojnásobnou výšku znaku. Nyní ale musíme užít dva PLOTy pro vertikální směr a vynechat dva body vertikálně.

Pokud tyto dva postupy zkombinujeme, můžeme produkovat dvojnásobné symboly. Po další úpravě můžeme psát tak velkými písmeny, jako potřebujeme. Musíme si ale zajistit, aby zvětšený znak nepřesahoval přes okraj obrazovky.

Následující program demonstruje zvětšování znaku v obou směrech. Tato technika může být velmi užitečná při psaní nadpisů.

```
100  REM ZVETSENE ZNAKY
110  LET CY=165
120  FOR V=1 TO 5: REM SIRKA
140  LET CX=10: LET CY=CY-8*V
150  FOR H=1 TO 5: REM VYSKA
160  PRINT AT 0,0;"$"
170  LET CX=CX+10*H
180  GOSUB 500
190  NEXT H
200  NEXT V
210  PRINT AT 0,0;" "
220  STOP
500  FOR Y=0 TO 7
510  FOR X=0 TO 7
520  IF POINT (X,175-Y)=0 THEN GOTO 570
530  FOR K=0 TO V-1
540  FOR J=0 TO H-1
550  PLOT CX+H*X+J,CY-V*Y+K
560  NEXT J: NEXT K
570  NEXT X
580  NEXT Y
590  RETURN
```

Obr. 5.10.

Skloněné znaky

Někdy chceme psát znaky na obrazovku skloněné, tzn. svírající určitý úhel s vertikální osou. Tohoto efektu dosáhneme odečtením určité měnící se hodnoty od souřadnice x právě tištěného řádku. Touto hodnotou je výška h, nebo lépe INT (h/2).
Pokud budeme tuto hodnotu přičítat, bude znak skloněn doprava (v případě odečítání doleva). Tento program ukazuje podrobněji, jak celé sklánění probíhá.

```
100  REM SKLONENE ZNAKY
110  LET CY=165
115  REM SIRKA
120  FOR V=1 TO 5
130  LET CX=10
140  LET CY=CY-8*V
145  REM VYSKA
150  FOR H=1 TO 5
160  PRINT AT 0,0;"$"
170  LET CX=CX+10*H
180  GOSUB 500
190  NEXT H
200  NEXT V
210  PRINT AT 0,0;" "
220  STOP
500  FOR Y=0 TO 7
510  FOR X=0 TO 7
520  IF POINT (X,175-Y)=0 THEN GOTO 590
530  FOR K=0 TO V-1
540  FOR J=0 TO H-1
550  PLOT CX+H  *X+J-Y*H/2,CY-V*Y+K
570  NEXT J
580  NEXT K
590  NEXT X
600  NEXT Y
610  RETURN
```

Obr. 5.12.

Dále můžete experimentovat s přičítáním (odečítáním) k y-ové souřadnici, což způsobí odklon od horizontální osy. Nebo můžete zkombinovat obě operace, potom se budou znaky psát pod úhlem 45 stupňů. Jenomže znaky budou trošku pokrouceny, ale vylepšení tohoto postupu už nechám na vás.

Kapitola 6: Více o barvách

Zatím jsme v této knize užívali pouze příkazy INK a PAPER pro změnu barev - INK pro barvu bodů a PAPER pro barvu pozadí. Na Spectru ale existuje ještě několik dalších příkazů pro ovládání barev na obrazovce a zjistíte, že poněkud chudá nabídka osmi barev se dá trošku rozšířit.

Jasnější a blikající barvy

Tyto užíváme v případě, že chceme na nějakou informaci na obrazovce upozornit nebo k indikaci prováděného děje. Jiným příkladem může být blikající kurzor Spectra.

Jedním ze způsobů, jak učinit část obrazu výraznější, je užití příkazu BRIGHT. Tento příkaz následuje pouze číslo 1 nebo 0, kde 1 zapíná jasnější zobrazování a 0 jej vypíná. BRIGHT 1 zjasní vždy pozadí znaku, tedy PAPER. Po přepnutí na BRIGHT 1 se obraz nemění, příkaz působí až v místě psaní.

Jiným prostředkem je užití instrukce FLASH, která má zase pouze jeden parametr (1 nebo 0) jako BRIGHT. FLASH 1 zapíná blikání, FLASH 0 jej vypíná. Blikání je vlastně přehazování barev mezi body v INKu a PAPERu. Stejně jako u BRIGHTu i zde příkaz nechává původní stav obrazu nezměněn a působí až při psaní. Po vypnutí blikání nebo jasu samozřejmě blikající nebo jasné znaky zůstanou nezměněny.

Tento program názorně ukazuje funkci příkazů FLASH a BRIGHT.

```
100  REM BLIKAJICI A JASNE ZNAKY
110  FOR K=1 TO 4
120  PRINT AT K,0;" ";
130  FOR J=0 TO 7
140  INK J
150  PRINT CHR$ (136+J);
160  PRINT CHR$ (136+J);
170  PRINT CHR$ (136+J);
180  NEXT J
190  NEXT K
200  FOR K=5 TO 8
210  BRIGHT 1
220  PRINT AT K,0;" ";
230  FOR J=0 TO 7
240  INK J
250  PRINT CHR$ (136+J);
260  PRINT CHR$ (136+J);
270  PRINT CHR$ (136+J);
280  NEXT J
290  NEXT K
300  BRIGHT 0
310  FOR K=9 TO 12
320  FLASH 1
330  PRINT AT K,0;" ";
340  FOR J=0 TO 7
```

```
350  INK J
360  PRINT CHR$ (136+J);
370  PRINT CHR$ (136+J);
380  PRINT CHR$ (136+J);
390  NEXT J
400  NEXT K
410  FLASH 0
420  INK 0
430  PRINT AT 2,26;"NORMAL"
440  PRINT AT 6,26;"BRIGHT"
450  PRINT AT 10,26;"FLASH"
```

Obr. 6.1.

Plnění útvarů barvou

Dalším způsobem jak zvýraznit nějaký útvar na obrazovce je neponechat ho jenom ohraničený čárami, ale vyplnit ho celý uvnitř, takže celá plocha útvaru má barvu INK. Toto je ale výhodné jen pro větší útvary.
Začněme např. nejjednodušším tvarem - obdélníkem. Abychom jej vyplnili, budeme dělat horizontální čáry jednu těsně vedle druhé od dolní po vrchní stranu obdélníka.

```
100  REM VYBARVENI OBDELNIKA
110  FOR N=1 TO 20
120  CLS
130  INK INT (RND*7)
140  LET W=15+INT (RND*100)
150  LET H=10+INT (RND*60)
160  LET X=INT (RND*(255-W))
170  LET Y=INT (RND*(175-H))
180  PLOT X,Y
190  DRAW W,0
200  DRAW 0,H
210  DRAW -W,0
220  DRAW 0,-H
230  IF H>W THEN GOTO 290
235  REM VYPLNENI HORIZONTALNIMI RADKY
240  FOR J=1 TO H
250  PLOT X,Y+J
260  DRAW W,0
270  NEXT J
280  GOTO 330
285  REM VYPLNENI VERTIKALNIMI RADKY
290  FOR J=1 TO W
300  PLOT X+J,Y
310  DRAW 0,H
320  NEXT J
330  PAUSE 50
340  NEXT N
```

Obr. 6.2.

Vezměme např. horizontální řádky. Vždy je udělán jeden bod a z něj vedena čára dlouhá jako šířka obdélníka. Další bod se udělá o jednu pozici výše a z něj zase čára. Toto probíhá ve smyčce

určené výškou obdélníka.

Jsou dva způsoby, jak vybarvit kruh. První metoda nakreslí nejprve hraniční kružnici a potom všechny kružnice ve stejném středě s poloměrem menším než je poloměr r hlavní kružnice. Názorně to předvádí tento program.

```
100  REM PLNENI KRUHU POMOCI SOUSTREDNYCH KRUZNIC
120  FOR N=1 TO 25
130  LET R=INT (RND*50)
140  LET X=R+INT (RND*(255-2*R))
150  LET Y=R+INT (RND*(175-2*R))
160  INK INT (RND*7): CLS
170  CIRCLE X,Y,R
180  FOR J=0 TO R
190  CIRCLE X,Y,J
200  NEXT J
210  PAUSE 50
220  NEXT N
```

Obr. 6.3.

Druhým způsobem je plnění kruhu pomocí radiálních úseček, vycházejících ze středu kruhu a posunujících se po malých krocích (úhlech) po obvodu kruhu, dokud nezaplní celý kruh. Názorně to předvádí tento program.

```
100  REM PLNENI KRUHU POMOCI RADIALNICH USECEK
120  FOR N=1 TO 25
130  LET R=INT (RND*50)
140  LET X=R+INT (RND*(255-2*R))
150  LET Y=R+INT (RND*(175-2*R))
160  INK INT (RND*7):CLS
170  CIRCLE X,Y,R
180  LET DT=PI/(R*4)
190  FOR J=0 TO R*8
200  PLOT X,Y
210  DRAW R*COS (J*DT),R*SIN (J*DT)
220  NEXT J
230  PAUSE 50
240  NEXT N
```

Obr. 6.4.

Pro vyplnění mnohoúhelníka je nejjednodušší udělat sérii soustředných mnohoúhelníků s poloměrem menším než je poloměr plněného (poloměry jdoucí po jednom kroku, tak jako první příklad plnění kruhu).

Rozdílné odstíny barev

Pokud užíváme příkazy INK a PAPER, máme k dispozici pouze osm barev. Jejich počet ale můžeme zvýšit užitím příkazu BRIGHT, který způsobí zjasnění barev. Tento BRIGHT užitý pro znak s barvou INKu červenou způsobí barvu růžovou.

Můžeme tedy vlastně produkovat spoustu nových barev mícháním základních barev na obrazovce. Pokud nakreslíme červenou čáru

těsně vedle žluté čary, dostaneme zdánlivě oranžovou, protože obě čáry zdánlivě splynou v jedinou.

Pokud nakreslíme čtverec a barevně ho vyplníme, přičemž některé řádky uděláme jinou barvou než ostatní, dostaneme jednoduchý příklad míchání barev, jak je ukázáno v tomto programu.

```
100  REM MICHANI BAREV HORIZONTALNIMI CARAMI
120  FOR P=7 TO 0 STEP -1
130  FOR I=0 TO 7
140  PAPER 7
150  CLS
160  INK 0
170  PAPER P
180  PLOT 87,47
190  DRAW 82,0
200  DRAW 0,82
210  DRAW -82,0
220  DRAW 0,-82
230  INK I
240  FOR J=0 TO 77 STEP 2
250  PLOT 88,50+J
260  DRAW 79,0
270  PLOT 88,49+J
280  DRAW PAPER P; INVERSE 1; 79,0
290  NEXT J
300  PAUSE 50
310  NEXT I
320  NEXT P
```

Obr. 6.5.

Stejně tak můžeme barvy míchat pomocí vertikálních čar, viz následující program.

```
100  REM MICHANI BAREV POMOCI VERTIKALNICH CAR
120  FOR P=7 TO 0 STEP -1
130  FOR I=0 TO 7
140  PAPER 7
150  CLS
160  INK 0
170  PAPER P
180  PLOT 87,47
190  DRAW 82,0
200  DRAW 0,82
210  DRAW -82,0
220  DRAW 0,-82
230  INK I
240  FOR J=0 TO 77 STEP 2
250  PLOT 90+J,48
260  DRAW 0,79
270  PLOT 89+J,48
280  DRAW PAPER P; INVERSE 1; 0,77
290  NEXT J
300  PAUSE 50
310  NEXT I
320  NEXT P
```

Obr. 6.6.

Ale ještě lepšího dojmu dosáhneme smícháním obou postupů, tj. mícháním barev pomocí horizontálních a vertikálních čar.

```
100  REM MICHANI BAREV POMOCI KRIZICICH SE CAR
120  FOR P=7 TO 0 STEP -1
130  FOR I=0 TO 7
140  PAPER 7
150  CLS
160  INK 0
170  PAPER P
180  PLOT 87,47
190  DRAW 82,0
200  DRAW 0,82
210  DRAW -82,0
220  DRAW 0,-82
230  INK I
240  FOR J=0 TO 79 STEP 2
250  PLOT 88,49+J
260  DRAW 78,0
270  NEXT J
280  FOR J=0 TO 77 STEP 2
290  PLOT PAPER P; INVERSE 1;89+J,49
300  DRAW PAPER P; INVERSE 1;0,77
310  NEXT J
320  PAUSE 50
330  NEXT I
340  NEXT P
```

Obr. 6.7.

Ale tento postup je moc těžkopádný a nepraktický. Lepší je vytvořit si speciální znak (pomocí uživatelské grafiky) ve tvaru šachovnice a tento pak zobrazovat s různými INK a PAPER. Můžeme takto snadno namíchat nejrůznější odstíny. Tento program využívá tuto techniku a ukazuje vám celou škálu odstínů.

```
100  REM MICHANI BAREV POMOCI SPECIALNIHO ZNAKU
120  FOR N=0 TO 7 STEP 2
130  POKE USR "A"+N,170
140  POKE USR "A"+N+1,85
150  NEXT N
160  FOR P=7 TO 0 STEP -1
170  PAPER P
180  FOR I=0 TO 7
190  INK I
200  FOR B=0 TO 1
210  BRIGHT B
220  FOR R=5 TO 15
230  FOR C=10 TO 20
250  PRINT AT R,C;CHR$ 144;
260  NEXT C: NEXT R
270  PAUSE 50
280  NEXT B: BRIGHT 0
290  NEXT I
300  NEXT P
```

Obr. 6.9.

Inverse Video

Normálně se body tisknou v barvě INK a pozadí v barvě PAPER. Užitím INVERSE 1 se body tisknou barvou PAPER a pozadí barvou INK. To můžeme využít pro zvýraznění nějakého slova nebo nadpisu. Dalo by se s tím zařídit i blikání textu, na to ale slouží příkaz FLASH.

Užití příkazu OVER

Dalším důležitým příkazem pro ovládání grafiky Spectra je OVER. Stejně jako FLASH nebo INVERSE má i on pouze dva stavy - Ø nebo 1.
Pokud přepneme na OVER 1 tak nově vypisované znaky nemažou ty doposud vypsané, ale dva znaky na jedné tiskové pozici (nebo dva body jemné grafiky) se vzájemně překrývají.
OVER 1 funguje jako logická funkce exclusive OR. To znamená, že nový znak vypisovaný přes nějaký jiný znak se vypíše podle určitých pravidel. Pokud byl původní bod rozsvícený a nově psaný bod by měl být zhasnutý, tak bod zůstane rozsvícený. Stejně tak naopak - pokud byl bod zhaslý a teď se má rozsvítit, tak se rozsvítí. Pokud jsou oba body zároveň zhaslé nebo zároveň rozsvícené, tak výsledný bod bude zhaslý. Pokud bereme rozsvícený bod jako bod v barvě INK a zhaslý bod jako PAPER, potom můžeme sestavit takovouto tabulku:

Starý bod	Nový bod	Výsledek
PAPER	PAPER	PAPER
INK	PAPER	INK
PAPER	INK	INK
INK	INK	PAPER

Tedy efektem OVER 1 je, že pokud napíšeme dva znaky přes sebe, tak body na stejných pozicích zhasnou (přejdou do barvy PAPER) a body na různých pozicích zůstanou rozsvícené (v barvě INK). Je samozřejmé, že pokud byl dříve vytištěný znak v jiných barvách než nový, tak se tyto barvy změní na barvy nového znaku.
Jedním způsobem využití OVER 1 je psát určitý stejný text dvakrát po sobě na stejné místo. Po prvním napsání se znaky vytisknou normálně, ale po druhém se všechny body v barvě INK přepnou do PAPER, tím pádem nám vlastně zůstane místo znaku jen mezera. Jinými slovy můžeme takto snadno smazat text.
Můžeme ale přes sebe psát i rozdílné znaky a potom můžeme dostat různé nové znaky a tvary bez jejich složitého vytváření.

Vytvoření kreslícího programu

Zatím jsme kreslili čáry, obdélníky a polygony pouze tak, že jsme vypočítali pozice jednotlivých bodů a potom jsme je pospojovali pomocí PLOT a DRAW. V mnoha případech ale budete chtít nakreslit obrázek pouze jakoby perem, kterým by jste pohybovali po obrazovce a spouštěli ho a zvedali a ono by psalo. To se dá lehce zařídit pomocí jednoduchého kreslícího programu, jehož princip si nejprve vysvětlíme.

Pohyb kurzoru budeme zajišťovat pomocí kláves s šipkami. Pero může být zvedlé nebo sundané a v závislosti na tom zanechává čáru nebo ne. Pokud je pero zvedlé, tak se pouze pohybuje a nekreslí. Pero budeme ovládat klávesami U(up - nahoru) a D(down - dolů). Po spuštění programu bude pero na pozici 0,0, tzn. v pravém dolním rohu.

Malým problémem je, že pokud je pero zvedlé, nemáme možnost vidět jeho pozici na obrazovce. Program to řeší tak, že bod na pozici kurzoru přece jen vytiskne. Potom, když perem pohneme, se bod smaže a vykreslí na nové kurzorové pozici, takže dále víme o jeho poloze. Pokud je bod na pozici kurzoru již rozsvícen, tak program to pomocí příkazu POINT pozná a zapamatuje si to. Když se potom pero přesune na novou pozici, je obnoven původní stav bodu.

```
100  REM KRESLICI PROGRAM
110  PRINT "KLAVESY SIPEK  POHYB PERA"
120  PRINT "D  SPOUSTI PERO DOLU"
130  PRINT "U  ZVEDA PERO"
140  PRINT "E  MAZE CARU"
150  PRINT "CISLA 0 AZ 6 NASTAVUJI BARVU"
160  PRINT "C,V,B,N  DAVAJI DIAGONALNI CARU"
170  PRINT "STISKNI TLACITKO"
180  PAUSE 0
185  REM INICIALIZACE
190  LET X=0: LET Y=0: LET X1=0: LET Y1=0
200  LET S=0: LET P=0: INK 0: PAPER 7
210  LET E=0: CLS: PLOT X,Y
220  IF INKEY$="" THEN GOTO 220
230  LET A$=INKEY$
240  IF A$="Q" OR A$="q" THEN STOP
250  IF A$="U" OR A$="u" THEN LET P=0: GOTO 270
260  IF A$="D" OR A$="d" THEN LET P=1: LET E=0
270  IF A$="E" OR A$="e" THEN LET E=1
280  IF A$=CHR$ 8 THEN LET X=X-1: GOTO 360
290  IF A$=CHR$ 9 THEN LET X=X+1: GOTO 360
300  IF A$=CHR$ 10 THEN LET Y=Y-1: GOTO 360
310  IF A$=CHR$ 11 THEN LET Y=Y+1: GOTO 360
320  IF A$="C" OR A$="c" THEN LET X=X-1: LET Y=Y+1
330  IF A$="V" OR A$="v" THEN LET X=X-1: LET Y=Y-1
340  IF A$="B" OR A$="b" THEN LET X=X+1: LET Y=Y-1
350  IF A$="N" OR A$="n" THEN LET X=X+1: LET Y=Y+1
360  IF X>255 THEN LET X=X-256
370  IF X<0 THEN LET X=X+256
380  IF Y>163 THEN LET Y=Y-164
```

```
390  IF Y<0 THEN LET Y=Y+164
400  LET C=(CODE A$)-48
410  IF C>6 OR C<0 THEN GOTO 430
420  INK C
430  IF P=1 OR S=1 THEN GOTO 450
440  PLOT OVER 1;X1,Y1: OVER 0
450  LET S=POINT (X,Y)
460  IF E=1 THEN INVERSE 1
470  PLOT X,Y
480  LET X1=X:LET Y1=Y
490  PRINT AT 0,0;"X= ";X;"  Y= ";Y;"        "
500  INVERSE 0
510  GOTO 220
```

Obr. 6.10.

Zmáčknutá tlačítka zjišťuje program pomocí příkazu INKEY$. Čeká, dokud není nic zmáčknuto a potom uloží zmáčklý znak do proměnné a$. Pak následují rozhodovací smyčky, které podle obsahu a$ provádějí určité činnosti. Pro pohyb bodu se používají šipky, tedy klávesy 5,6,7,8 splou s CAPS SHIFTEM. Program kontroluje kód pro posun šipek, proto musí být zmáčklá jak číslice, tak CAPS SHIFT.

Dalším vylepšením programu je klávesa E, která maže bod na pozici kurzoru. To využijete při opravování chyb. Dále klávesy C,V,B a N slouží pro posun kurzoru po diagonálách. C je krok vlevo nahoru, V vlevo dolu, N vpravo nahoru no a B vpravo dolu.

Atributy barev

Spectrum ukládá do jedné paměti jak textové znaky (ve formě bodové informace), tak body jemné grafiky. Ale informace o barvě ukládá zase do jiné části paměti.

Barevná informace není samostatná pro každý bod (to by zabralo příliš mnoho paměti), ale vždy pro čtvereček 8*8 bodů. V toto čtverečku může být vždy jen jedna barva INK, jeden PAPER, pro celý čtvereček může být zaplý nebo vyplý FLASH a BRIGHT. Tyto čtverečky se nazývají atributy.

Každý atribut je vlastně 1 bajt, tedy 8 bitů. Tyto bity mají následující funkce:

Číslo bitu	Číselná hodnota	Barva atributu
0	1	modrá INK
1	2	červená INK
2	4	zelená INK
3	8	modrá PAPER
4	16	červená PAPER
5	32	zelená PAPER
6	64	BRIGHT
7	128	FLASH

Nejnižší tři bity (bráno zprava doleva) určují barvu INKu v atributu. Výsledná barva je získána mícháním tří základních - modré, červené a zelené. Pokud bude z těchto tři bitů pouze bit

číslo 1 (druhý zprava) ve stavu logické 1, bude barva INKu v dotyčném atributu červená. Pokud budou v jedničce bity 1 a 2, bude INK žlutý (červená + zelená) apod.
Další tři bity určují barvu PAPERu pro dotyčný atribut. Jsou zde zase tytéž tři základní barvy a platí zde totéž míchání jako u INKu.
Bit číslo 6 určuje BRIGHT aributu. Pokud je nastaven (tedy ve stavu log. jedničky), je atribut ve stavu BRIGHT 1. Pokud je v nule, je BRIGHT atributu 0. Stejně tak pro bit 7, který určuje FLASH atributu.

Kontrola atributů

Pokud chceme zjistit, jak jsou nastaveny barvy určitého čtverečku (atributy), můžeme k tomu užít příkaz

LET A=ATTR (R,C)

kde r a c jsou číslo řádku a sloupce kontrolovaného atributu (souřadnice stejné jako u PRINT AT). Výsledkem bude hodnota bajtu pro atribut na souřadnicích r,c uložená v proměnné A.
Na 48k Spectru jsou atributy uloženy mezi adresami 22528 a 23295. Jsou tam uloženy tím způsobem, že atribut čtverečku na pozici 0,0 je uložen na adrese 22528. Atribut vedle něj na řádku je uložen o adresu výše a tak dále - v paměti jsou tedy uloženy řádky jeden za druhým. Abychom dostali adresu atributu na řádku r a sloupci c, užijeme

Adresa paměti = 22528+(32*r)+c

Pokud cheme znát hodnotu tohoto atributu, užijeme funkci PEEK adresa paměti. Můžeme tuto hodnotu také změnit pomocí příkazu POKE.
Často chceme znát barvu nějakého samostatného bodu na obrazovce. Potom můžeme užít funkci ATTR v takovémto tvaru:

LET A=ATTR (21-Y/8,X/8)

Kde x,y jsou souřadnice bodu (sloupec, řádek). Potom pro sloupec atributu stačí podělit x osmi. Pro řádek je to trošku složitější, protože v jemné grafice se řádky počítají odspodu nahoru. Proto musíme hodnotu y/8 odečíst od 21. (Pozn. Povšimněte si, že u souřadnic jemné grafiky je první sloupec a potom až řádek, kdežto u souřadnic znakových je tomu naopak.)
Nyní tedy můžeme snadno zjistit stav každého bitu jednoho atributu, nastavením proměnných "flag" (praporek, ukazatel) do stavu 1 nebo 0. Abychom zjistili hodnoty pro jednotlivé flagy, musíme hodnotu atributu porovnávat s číselnými hodnotami z tabulky ze strany 57. Pokud bude hodnota atributu menší než 128, bude FLASH roven 0. Když ne, je FLASH 1 a od hodnoty atributu odečteme 128. No a tak stále stejně až do osmého flagu. Názorně to osvětlí následující podprogram.

```
500  REM ROZKODOVANI BAREV ATRIBUTU
510  LET A=ATTR (R,C)
515  REM FLASH
520  IF A<128 THEN LET FL=0: GOTO 540
530  LET FL=1: LET A=A-128
535  REM BRIGHT
540  IF A<64 THEN LET BR=0:GOTO 560
550  LET BR=1: LET A=A-64
555  REM BITY PAPERU
560  IF A<32 THEN LET PG=0: GOTO 580: REM GREEN
570  LET PG=1: LET A=A-32
580  IF A<16 THEN LET PR=0: GOTO 600: REM RED
590  LET PR=1: LET A=A-16
600  IF A<8 THEN LET PB=0: GOTO 620: REM BLUE
610  LET PB=1: LET A=A-8
615  REM BITY INKU
620  IF A<4 THEN LET IG=0: GOTO 640
630  LET IG=1: LET A=A-4
640  IF A<2 THEN LET IR=0: GOTO 660
650  LET IR=1: LET A=A-2
660  LET IB=A
670  RETURN
```

Obr. 6.12.

Kapitola 7: Grafy a diagramy

Při užívání počítače pro různé výpočty a operace dostaneme často spoustu výsledků a hodnot. Můžeme je vypsat jako sloupec čísel nebo i do tabulky, ale to není příliš užitečné pro čtení závislostí nebo znázornění postupu.

Lepší metodou je uspořádat si výsledky nebo hodnoty na obrazovce do grafu nebo diagramu, ze kterých už mnohem snadněji něco vyčteme. Může to být třeba jenom skupina bodů různě vzdálených od nějaké základní čáry, ukazující rozdíly hodnot jednotlivých výsledků.

Nebo můžeme vytvořit sloupcový ukazatel. Jednoduchým příkladem tohoto je ukazatel teploty.

Rtuťový teploměr

Začneme nejprve s teploměrem vykresleným pomocí mozaikové grafiky.

Ve skutečném teploměru ukazuje výška rtuťového sloupce naměřenou teplotu. Rtuťový sloupec nahradíme nakreslení jednoduchého vertikálního sloupce, jehož výška bude úměrná zadané teplotě. Obal teplměru nahradíme obdélníkem rozdílné barvy okolo "rtuťového" sloupce. Výška obalu musí být dostatečná pro nejvyšší možnou ukazatelnou teplotu a musí se vlézt do rozměrů obrazovky.

Aby měl náš teploměr smysl, musí mít nějaké měřítko, podle kterého zjistíme ukazovanou hodnotu. Na obyčejném teploměru je stupnice natištěna na jeho obalu, my si ji uděláme vedle obalu. Znaky mínus (-) budou reprezentovat značky označující hodnoty teploty. U některých z nich uvedeme i číselnou hodnotu,uvádějící odpovídající teplotu. V našem případě stačí pouze nejnižší a nejvyšší teplota. Teplota se mění s výškou rtuťového sloupce, proto i na našem teploměru bude jeho vršek ukazovat na odpovídající hodnotu teploty.

Řekněme, že chceme rozsah od 0 do 100 stupňů Celsia. Jak již víme, mozaiková gragika má pro své "body" 44 řádků. Přiměřená výška rtuťového sloupce bude 20 mozaikových řádků. Každý řádek bude tedy reprezentovat 5 stupňů. Nyní už můžeme nakreslit obal teploměru. Jeho spodek dosataneme nakreslením mozaikových znaků s kódy 129,131 a 130 zhruba uprostřed řádku 20 na obrazovce. Potom pomocí smyčky nakreslíme od řádku 19 do 9 tělo tepoměru. A nakonec uděláme jeho vršek - na řádku 8. Při vykreslování těla udělame samozřejmě i stupnici (znaky -). Potom označíme mezní teploty a tělo teploměru je hotovo.

Pro vykreslení rtuťového sloupce musíme teplotu t podělit 5, protože každý dílek stupnice je 5 stupňů (viz výše). Ale nejprve musíme k t přičíst 5, protože 0 stupňů je prvním dílkem stupnice (jinak by se nám teplota ukazovala o 5 stupňů nižší). Tento podíl uložíme do proměnné y. Dále probíhá smyčka od 1 do INT (y/2) - každá pozice PRINT ATu reprezentuje dva řádky mozaikové grafiky. Tato smyčka zaplní skoro celý sloupec určující zobrazovanou teplotu. Pokud ale INT (y/2)<>y/2, musí se

zobrazit ještě jeden mozaikový řádek, protože znakový řádek reprezentuje vlastně 10 stupňů.
Program fungující podle těchto zákonitostí je tento:

```
100  REM TEPLOMER MOZAIKOVOU GRAFIKOU
110  CLS
120  INK 0: PAPER 7
130  LET X0=118 :LET Y0=16
140  REM KRESLENI OBALU TEPLOMERU
150  PRINT AT 20,15;CHR$ 129; CHR$ 131; CHR$ 130
160  FOR N=1 TO 11
170  PRINT AT 20-N,14;"-";CHR$ 133; CHR$ 128; CHR$ 138
180  NEXT N
190  PRINT AT 8,15; CHR$ 132; CHR$ 140; CHR$ 136
200  REM STUPNICE
210  PRINT AT 19,13;"0"
220  PRINT AT 9,11;"100"
230  PRINT AT 13,11;"C"
240  PRINT AT 14,10;"DEG"
250  REM 100 RUZNYCH TEPLOT
260  FOR K=1 TO 100
270  LET T=INT (RND*100)
280  INK 1
290  PRINT AT 1,1;"TEPLOTA = ";T;" 'C"
300  GOSUB 500
310  PAUSE 200
320  NEXT K
330  STOP
500  INVERSE 1
510  REM VYMAZANI PREDESLE TEPLOTY
520  FOR N=1 TO 11
530  PRINT AT 20-N,16;CHR$ 143
540  NEXT N
550  INVERSE 0
560  REM NAKRESLENI NOVE HODNOTY
570  INK 2
580  LET Y=INT ((T+5)/5+0.5)
590  FOR N=1 TO INT (Y/2)
600  PRINT AT 20-N,16;CHR$ 143
610  NEXT N
620  IF INT (Y/2)=Y/2 THEN RETURN
630  LET Y=INT (Y/2)
640  PRINT AT 19-Y,16; CHR$ 140
650  RETURN
```

Obr. 7.1.

Náhodně zvolená teplota se vypíše na nultém řadku obrazovky a zároveň se nakreslí na zobrazeném teploměru. Před každou novou hodnotou se smaže starý rtuťový sloupec pomocí INVERSE 1, která přepíše sloupec do barvy pozadí, tedy PAPERu. Rtuť se kreslí pomocí červeného inkoustu.
Samozřejmě, že takovýto ukazatel můžete uplatnit pro jinou fyzikální veličinu nebo pro nějaké jiné hodnoty, to už záleží na vás. Můžete také udělat ukazatel horizontální.

Vylepšený teploměr

Největší nevýhodou mozaikového teploměru je jeho nepřesnost - každý dílek představuje 5 stupňů Celsia. Pokud užijeme jemnou grafiku, tak tento problém odpadne - můžeme krelit po stupních, ale objeví se jiné. Je jednoduché nakreslit obal teploměru pomocí jemné grafiky, ale při jeho popisování se musíme řídit souřadnicemi znakovými, které jsou dosti odlišné. Stejně tak musíme opatrně zacházet s barvami, protože ty jsou také závislé na znakových čtvercích.

Obal teploměru je nakreslen jednoduše pomocí příkazů DRAW. Stejně tak nakreslíme i stupnici, a pro zjednodušení nakreslíme značku pro 0 samostatně. Popis stupnice se provede pomoc PRINT AT.

Nakreslení rtuťového sloupce se provede jako nakreslení obdélníka o výšce t. Poměr sloupce k teplotě je tedy 1:1, nemusíme proto nic přepočítávat ani kontrolovat jako u mozaikového teploměru. Sloupec nakreslíme jako šest vertikálních čar těsně vedle sebe. Stará hodnota se maže stejným obdélníkem, ale kresleným při INVERSE 1.

```
100  REM TEPLOMER V JEMNE GRAFICE
110  CLS
120  INK 0: PAPER 7
130  LET X0=118: LET Y0=16
140  REM NAKRESLENI OBALU
150  PLOT X0,Y0
160  DRAW 10,0
170  DRAW 0,108
180  DRAW -10,0
190  DRAW 0,-108
200  REM STUPNICE
210  PLOT X0,Y0+4
220  DRAW -3,0
230  DRAW 3,0
240  FOR N=1 TO 10
250  DRAW 0,10
260  DRAW -3,0
270  DRAW 3,0
280  NEXT N
290  PRINT AT 19,13;"0"
300  PRINT AT 6,11;"100"
310  PRINT AT 13,12;"C"
320  PRINT AT 14,11;"DEG"
330  REM 100 TEPLOT
340  FOR K=1 TO 100
350  LET T=INT (RND*100)
360  INK 1
370  PRINT AT 1,1;"TEPLOTA = ";"T;"'C  "
380  GOSUB 500
390  PAUSE 200
400  NEXT K
410  STOP
500  INVERSE 1
```

```
510  REM VYMAZANI STARE TEPLOTY
520  PLOT X0+2,Y0+1
530  LET Y=104
540  FOR N=1 TO 6
550  DRAW 0,Y: DRAW 1,0
560  LET Y=-Y
570  NEXT N
580  DRAW 0,Y
590  INVERSE 0
600  REM NOVA TEPLOTA
610  INK 2
620  PLOT X0+2,Y0+4
630  FOR N=1 TO 6
640  DRAW 0,T: DRAW 1,0
650  LET T=-T
660  NEXT N
670  DRAW 0,T
680  RETURN
```

Můžete program snadno změnit pro kreslení horizontálně - stačí popřehazovat hodnoty u PLOT a DRAW (případně některé upravit) a změnit souřadnice u PRINT AT.

Program si náhodně zvolí teplotu a tu zobrazí. Je ale možné připojit ke Spectru nějaký speciální interface a pomocí něj skuteně měřit určitou hodnotu - počítač by potom fungoval jako elektronický teploměr.

Sloupcové diagramy

I když je teploměr užitečná a zajímavá aplikace, mnohem užitečnější by bylo, kdybychom mohli na obrazovce vidět teploty z nějakého časového období. Můžeme například chtít zobrazit najednou teploty naměřené kazdý den týdne v poledne. Takováto věc se dá realizovat pomocí diagramu. Na levé straně bude stupnice jako na teploměru a vedle ní bude řada sedmi sloupců, indikujících teplotu v ten který den v poledne. Pro větší názornost necháme mezi jednotlivými sloupci mezery. Takovýto diagram se potom nazývá sloupcový.

Sloupcové diagramy se používají hlavně tam, kde chceme sledovat nějaký trend nebo závislost určitého jevu, třeba v obchodě zisk v jednotlivých měsících roku nebo třeba počet návštěvníků něčeho v určitých obdobích. Potom je velmi jednoduché z takovéhoto diagramu vyčíst určité závislosti.

Další výhodou sloupcového diagramu je to, že pokud sloupec přesáhne (nebo nedosáhne) určité hodnoty, může se přebytek vybarvit jinou barvou, což může upozornit na nějakou nebezpečnou nebo důležitou situaci. Případně se může jinou barvou vybarvit celý sloupec.

Mozaiková grafika může být užita pro kreslení sloupcového grafu a dostaneme celkem solidní výsledek.

Následující program kreslí sloupcový graf pomocí mozaikové grafiky.

```
100  REM JEDNODUCHY SLOUPCOVY DIAGRAM
110  REM POMOCI MOZAIKOVE GRAFIKY
120  BORDER 3
130  INK 0: PAPER 7
140  DIM D$(7,2): DIM T(7)
150  REM NACTENI DAT
160  FOR N=1 TO 7
170  READ D$(N),T(N)
180  NEXT N
190  DATA "PO",60,"UT",65,"ST",80
200  DATA "CT",55,"PA",65
210  DATA "SO",70,"NE",65
220  REM KRESLENI STUPNICE
230  FOR N=1 TO 22
240  PRINT AT 19,7+N;CHR$ 131
250  NEXT N
260  FOR N=1 TO 11
270  PRINT AT 19-N,7;"-";CHR$ 138
280  NEXT N
290  FOR N=1 TO 7
300  PRINT AT 20,7+3*N;D$(N);" ";
310  NEXT N
320  PRINT AT 18,6;"0"
330  PRINT AT 8,4;"100"
340  PRINT AT 12,5;"F"
350  PRINT AT 13,4;"DEG"
360  FOR J=1 TO 7
370  GOSUB 500
380  NEXT J
390  PRINT AT 2,10;"DENNI TEPLOTY"
400  STOP
500  INK 2
510  LET Y=INT ((T(J)+5)/5+.5)
520  FOR N=1 TO INT (Y/2)
530  PRINT AT 19-N,7+3*J;CHR$ 143;CHR$ 143;
540  NEXT N
550  IF INT (Y/2)=Y/2 THEN RETURN
560  LET Y=INT (Y/2)
570  PRINT AT 19-Y,7+3*J;CHR$ 140;CHR$ 140
580  RETURN
```

Obr. 7.5.

Program kreslí každý sloupec zvlášť stejným postupem jako u teploměru. Hodnoty pro jednotlivé dny jsou načítány z dat, což může být jednoduše změněno. Můžete hodnoty zadávat pomocí INPUT nebo je změnit v příkaze DATA. Změněním nadpisu a popisu můžete program snadno upravit pro znázornění jiných veličin.

Sloupcový graf jemnou grafikou

Následující program dělá totéž co předešlý, ale využívá přitom jemnou grafiku.

```
100  REM SLOUPCOVY DIAGRAM JEMNOU GRAFIKOU
110  CLS
120  BORDER 3
130  DIM D$(7,2): DIM T(7)
140  FOR N=1 TO 7
150  READ D$(N),T(N)
160  NEXT N
170  DATA "PO",15,"UT",18,"ST",25
180  DATA "CT",12,"PA",17,"SO",20,"NE",18
190  INK 0
200  LET X0=48: LET Y0=20
210  PLOT X0,Y0
220  DRAW 168,0
230  PLOT X0,Y0
240  FOR N=1 TO 6
250  DRAW 0,20
260  DRAW -3,0
270  DRAW 3,0
280  NEXT N
290  PRINT AT 20,7;;
300  FOR J=1 TO 7
310  PRINT D$(J);" ";
320  NEXT J
330  PRINT AT 19,2;"0"
340  PRINT AT 4,2;"30"
350  PRINT AT 11,2;"C"
360  PRINT AT 12,1;"DEG"
370  INK 2
380  PLOT X0,Y0
390  DRAW 4,0
400  FOR K=1 TO 7
410  DRAW 8,0
420  LET Y=T(K)*4
430  FOR N=1 TO 4
440  DRAW 0,Y
450  DRAW 1,0
460  DRAW 0,-Y
470  DRAW 1,0
480  NEXT N
490  DRAW 8,0
500  NEXT K
510  INK 1
520  PRINT AT 2,6;"DENNI TEPLOTA"
530  STOP
```

Obr. 7.7.

Zde se hodnoty teploty kreslí podobně jako u teploměru. Jiným řešením by ale bylo nakreslení sloupce z malých krátkých horizontálních čáreček. Tento postup je sice také funkční, ale velmi pomalý.

I zde jsme museli dávat pozor při míchání jemné grafiky a textů, zvláště kvůli barvám.

Dvojitý sloupcový graf

Pokud chceme v jednom grafu zobrazit dvě rozdílné veličiny, musíme je nějak odlišit. Nejjednodušší je kreslit každou z nich jinou barvou. Nebo můžeme některé kreslit jako prázdné obdélníky s rozdílnou barvou okraje. Takovéto grafy mohou totiž ukazovat více závislostí najednou a i vztahy mezi nimi (třeba výdaje a příjmy domácnosti).

Následující program kreslí graf nejnižších a nejvyšších teplot v jednotlivých dnech týdne pomocí jemné grafiky.

```
100  REM DVOJUTY SLOUPCOVY GRAF
110  CLS
120  BORDER 3
130  DIM D$(7,2)
140  DIM L(7)
150  DIM H(7)
170  FOR N=1 TO 7
180  READ D$(N),L(N),H(N)
190  NEXT N
200  DATA "PO",7,15,"UT",10,18,"ST",15,25,"CT",5,12
210  DATA "PA",2,17,"SO",7,20,"NE",15,18
240  INK 0
250  LET X0=48: LET Y0=20
260  PLOT X0,Y0
270  DRAW 168,0
280  PLOT X0,Y0
290  FOR N=1 TO 6
300  DRAW 0,20
310  DRAW -3,0
320  DRAW 3,0
330  NEXT N
340  PRINT AT 20,7;;
350  FOR J=1 TO 7
360  PRINT D$(J)," ";
370  NEXT J
380  PRINT AT 19,4;"0"
390  PRINT AT 4,3;"30"
400  PRINT AT 11,2;"C"
410  PRINT AT 12,1;"DEG"
420  REM KRESLENI SLOUPCU
430  PLOT X0+8,Y0
440  FOR K=1 TO 7
450  INK 5
460  LET Y=L(K)*4
470  FOR N=1 TO 4
480  DRAW 0,Y
490  DRAW 1,0
500  DRAW 0,-Y
510  DRAW 1,0
520  NEXT N
530  DRAW 4,0
540  INK 2
550  LET Y=H(K)*4
```

```
560  FOR N=1 TO 4
570  DRAW 0,Y
580  DRAW 1,0
590  DRAW 0,-Y
600  DRAW 1,0
610  NEXT N
620  DRAW 4,0
630  NEXT K
640  REM LEGENDA
650  INK 1
660  PRINT AT 1,6;"DENNI TEPLOTY"
670  INK 5
680  PRINT AT 3,6;"NIZKE ";CHR$ 143;CHR$ 143
690  INK 2
700  PRINT AT 3,17;"VYSOKE ";CHR$ 143;CHR$ 143
710  STOP
```

Obr. 7.9.

Jeden druh sloupců je kreselen červeně a jeden modře, aby jste je mohli odlišit. Aby jste věděli, který je který, k tomu slouží legenda pod nadpisem.

Vědecké grafy

Ačkoliv sloupcové grafy jsou užitečné třeba v obchodě nebo v jednoduché statistice, pokud budeme mít nějak matematické nebo vědecké výpočty, obdržíme často množství výsledků, jejichž zaznamenání do sloupcového grafu by bylo nepraktické nebo i nemožné.

Většinou tyto situace řešíme tak, že horizontální (x-ová) osa obsahuje výchozí hodnoty výpočtu a vertikální (y-ová) výsledky. Do grafu potom nanášíme pouze jednotlivé body, jako by vrcholky sloupců. Pro lepší viditelnost můžeme body nahradit malými křížky, trojúhelníčky apod.

Ve sloupcovém grafu byly hodnoty pouze kladné, ale ve vědeckém může být jak x, tak y záporné nebo kladné. Potom kladná část osy x půjde od počátku doprava, záporná doleva. Stejně tak kladná část osy y půjde od počátku nahoru, záporná dolu (pokud si to ovšem nezvolíte jinak, ale tato kombinace je nejužívanější). Někdy třeba potřebujeme jen jednu čtvrtinu nebo polovinu takovéhoto souřadnicového systému, potom můžeme vzít jenom ji a mít ji příslušně zvětšenou, protože získáme prostor nevyužívaných kvadrátů.

Jako příklad takovéhoto grafu si uvedeme graf funkce y=SIN(x) pro x z intervalu -10 až 10. Hodnota SIN(x) bude vždy z intervalu -1 až 1, což nám vymezuje měřítko osy y. Abychom mohli y-ovou souřadnici zobrazit v přijatelném rozmezí, musíme ji násobit nějakou konstantou. Zvolíme si třeba 60, takže každou vypočítanou hodnotu y budeme před nakreslením bodu násobit 60 neboli proměnnou ys, nastavenou na začátku programu. Graf bude mít tedy 60-ti bodovou kladnou a 60-ti bodovou zápornou osu y. Pro x zvolíme nějaký krok, řekněme 10. Potom celková délka osy x bude 200 bodů a vzdálenost celých hodnot x bude oněch 10 (proměnná xs). No a změnou proměnných xs a ys můžete snadno

změnit rozměry grafu.

Prvním krokem je nakreslení souřadnicových os a měřítek. To se provede pomocí příkazů DRAW a dvou smyček. Nejprve umístíme grafický kurzor doprostřed grafu (do počátku) pomocí PLOT 128,88 což je zároveň i střed obrazovky. Dále nakreslíme kladnou část osy x, pomocí krátkých horizontálních čar (dlouhých xs bodů), na jejichž koncích jsou kratičké vertikální čárky (stupnice). Po nakreslení kladné části se vrátí do počátku a kreslí zápornou. Osa y se kreslí podobně. Osy se vykreslují podprogramem, i když by to šlo i bez něj. Posledním krokem je vypsání popisu os.

Když už máme osy, nakreslíme samotný graf. Zde bereme úhel x (v radiánech) z intervalu -10 až 10 a pozici bodu dostaneme pomocí jednoduchého výpočtu:

xp=x0+(xs*x)

kde x0 je x-ová souřadnice středu os a xp souřadnice kresleného bodu. Díky tomuto můžeme pouhou změnou x0 změnit pozici grafu.

Druhou souřadnici bodu dostaneme ze vztahu

yp=y0+ys*SIN (x)

kde y0 je y-ová souřadnice středu os a yp y-ová souřadnice bodu který chceme nakreslit. To provedeme příkazem PLOT xp,yp.

Následující program kreslí graf funkce y=SIN (x) tak, jak jsme si právě popsali. Barvy INKu a PAPERu jsou zde pouze černá a bílá, ale to se dá snadno změnit.

```
100  REM GRAF SINU
110  BORDER 5
120  CLS
130  LET XS=10
140  LET YS=60
150  LET X0=128
160  LET Y0=92
170  REM KRESLENI OS
180  GOSUB 500
190  REM KRESLENI BODU GRAFU
200  FOR X=-10 TO 10 STEP 0.1
210  LET Y=SIN X
220  LET XP=X0+INT (XS*X)
230  LET YP=Y0+INT (YS*Y)
240  PLOT XP,YP
250  NEXT X
260  REM POPIS
270  PRINT AT 20,11;"Y = SIN (X)";
280  STOP
500  LET X=XS
510  REM OSA X
520  FOR K=1 TO 2
530  PLOT X0,Y0
540  FOR J=1 TO 10
550  DRAW X,0
560  DRAW 0,-3
```

```
570  DRAW 0,3
580  NEXT J
590  LET X=-X
600  NEXT K
610  LET Y=YS/10
620  REM OSA Y
630  FOR K=1 TO 2
640  PLOT X0,Y0
650  FOR J=1 TO 10
660  DRAW 0,Y
670  DRAW -3,0
680  DRAW 3,0
690  NEXT J
700  LET Y=-Y
710  NEXT K
720  PRINT AT 10,0;"-10"
730  PRINT AT 10,29;"+10"
740  PRINT AT 1,15;"+1"
750  PRINT AT 19,15;"-1"
760  RETURN
```

Obr. 7.12.

Můžeme samozřejě do jednoho grafu nanést dvě nebo i více křivek různých barev, ale pokud se budou střetávat v jedné znakové pozici, tak se body dříve nakreslené křivky v tomto místě přepnou do nové barvy. To by ale nemělo činit větší obtíže, neboť ve většině případů je takovýchto míst málo.

Spojování bodů grafu

Abychom u grafu funkce SIN z minulého příkladu dostali pěkný obrázek, potřebovali bychom mnohem více bodů s menším rozdílem mezi nimi. Ale ty by se nám v daném intervalu pro x nevlezly na obrazovku. Proto jsou jednotlivé body grafu osamoceny.

Někdy můžeme chtít znát hodnotu y pro x, které jsme nepočítali a nezobrazovali v grafu. To se dělá pomocí tzv. interpolace a můžeme dostat aproximační hodnotu pro kterýkoliv bod křivky.

Nejjednodušší technikou interpolace je spojení sousedních bodů křivky přímou čarou. To se nazývá lineární interpolace. Můžeme v praxi spojovat body již při kreslení grafu. Musíme ale dávat pozor, protože pokud budeme spojovat pouze málo bodů, může dojít k nepřesnostem.

Pro spojení sousedních bodů užijeme příkazu DRAW, který nakreslí čáru z posledního bodu křivky o souřadnicích x1,y1 do nově vypočítaného bodu x2,y2. Hodnoty x1,y1 se potom změní na x2,y2. V příkazu DRAW uvedem rozdíl mezi x2,x1 a y2,y1. Příkazem PLOT musíme nastavit grafický kurzor na první bod grafu, do prvních souřadnic x1,y1. Tento program kreslí graf funkce y=COSx pro stejné podmínky jako předešlý (y=SINx), ale spojuje body navzájem.

```
100  REM GRAF FUNKCE COS SE SPOJENYMI BODY
110  BORDER 2
120  CLS
130  LET XS=10
140  LET YS=60
150  LET X0=128
160  LET Y0=92
170  REM NAKRESLENI OS
180  GOSUB 500
190  REM NAKRESLENI GRAFU
200  LET X1=X0+XS*-10
210  LET Y1=Y0+INT (YS*COS -10)
220  PLOT X1,Y1
230  FOR X=-10 TO 10 STEP .3
240  LET X2=X0+INT (XS*X)
250  LET Y2=Y0+INT (YS*COS X)
260  DRAW X2-X1,Y2-Y1
270  LET X1=X2
280  LET Y1=Y2
290  NEXT X
300  REM POPIS
310  PRINT AT 20,12;"Y = COS (X)"
320  STOP
500  LET X=XS
510  REM OSA X
520  FOR K=1 TO 2
530  PLOT X0,Y0
540  FOR J=1 TO 10
550  DRAW X,0
560  DRAW 0,-3
570  DRAW 0,3
580  NEXT J
590  LET X=-X
600  NEXT K
610  LET Y=YS/10
620  REM OSA Y
630  FOR K=1 TO 2
640  PLOT X0,Y0
650  FOR J=1 TO 10
660  DRAW 0,Y
670  DRAW -3,0
680  DRAW 3,0
690  NEXT J
700  LET Y=-Y
710  NEXT K
720  PRINT AT 10,0;"-10"
730  PRINT AT 10,29;"+10"
740  PRINT AT 1,15;"+1"
750  PRINT AT 19,15;"-1"
760  RETURN
```

Obr. 7.13.

Ciferník

Někdy se může hodit užití ciferníkového zobrazení (tedy jako ručičkové hodiny). Typickým příkladem může být třeba letový simulátor nebo hodiny.

Základní ciferník se skládá z kruhu nebo mnohoúhelníku a jedné nebo dvou ručiček. Ručičky se pohybují dokola kolem středu ciferníku. Ten může být také vybarvený, aby se odlišil od ostatního pozadí. Stupnice je nakreslena okolo ciferníku.

Tento typ zobrazení použijeme tam, kde ani tak nepotřebujeme přesnost a rychlost zobrazení, ale tam, kde chceme rychle zjistit okamžitou hodnotu. Vezměte si třeba hodinky - digitální display je sice rychlý, ale čas mnohem lépe zjistíte z analogového, tedy z ciferníku.

Při kreslení ciferníku musíme nakreslit kružnici, což provedeme příkazem CIRCLE. Pro nakreslení stupnice musíme nejprve nakreslit kružnici s větším poloměrem o stejném středu a potom ji ocejchovat. Pro každou značku potřebujeme dva úhly. S1 je úhel vnitřního konce značky a S2 je úhel jejího vnějšího konce. Tyto úhly se vypočítají otáčením a potom se podle S1 nakreslí krátká čárka. Druhý úhel se využije pro určení pozice popisného znaku.

Zajímavou vlastností ručičky je to, že se otáčí po směru hodinových ručiček při zvětšování veličiny. Opak tedy dostaneme normální rovnicí pro otáčení. Pozici ručičky dostaneme snadno pomocí modifikované rovnice otáčení. Úhel rotace je vždy v určitém poměru k celému ciferníkovému úhlu. Pokud užijeme celý kruh, tedy 360 stupňů, vypočítáme úhel TH takto:

TH=2*PI*X/FS

kde X je hodnota kroku, FS je úhel ciferníku a TH je úhel otáčení. Pro otočení do normálního směru musíme změnit znaménko souřadnice y.

Někdy může ciferník mít úhel 90, 180 nebo 270 stupňů. V tomto případě nahradíme hodnotu 2*PI v rovnici odpovídajícím úhlem v radiánech. Tento úhel nalezneme lehce pomoci vzorce

RAD=DEG*PI/180

Normálně se předpokládá, že hodnota 0 bude vpravo (jako ve tři hodiny na ciferníku hodinek). Ale někdy ji můžeme chtít nahoře (jako ve 12). Potom se musí k úhlu TH přičíst 90 stupňů (PI/2) před výpočtem hodnot x a y (souřadnice konce ručičky). V ukázkovém programu se užívá SIN pro výpočet x a COS pro výpočet y, což dá ten samý výsledek jako přičtení oněch 90 stupňů.

Kreslení značek stupnice je podobné kreslení ručičky s tím rozdílem, že jejich začátek leží na kružnici s větším poloměrem než má ciferník a že pro každou značku počítáme dva úhly pro dvě pozice - jeden pro začátek samotné čárečky a jeden pro písmeno (nebo číslici) popisu. Po nalezení středu znaku ho přepíšeme pomocí podprogramu pro přepis znaku na pozici jemné grafiky.

Vždy před každým novým zobrazením ručičky musíme smazat tu starou, tzn. přepsat ji do barvy, jíž je vybarven ciferník nebo

do barvy pozadí (PAPER).
Následující ukázkový program kreslí jednoduchý ciferník s jednou ručičkou. Pozice 0 je nahoře a ciferník má úhel 270 stupňů.

```
100  REM JEDNODUCHY CIFERNIK
110  LET XC=128
120  LET YC=88
130  LET R=40
140  LET S1=R+5
150  LET S2=R+10
160  LET ST=R+20
170  REM KRESLENI CIFERNIKU
180  CIRCLE XC,YC,R
190  REM CEJCHOVANI
200  LET DT=1.5*PI/S1
210  LET TH=0
220  LET X1=0
230  LET Y1=S1
240  PLOT XC+X1,YC+Y1
250  FOR N=1 TO S1
260  LET TH=TH+DT
270  LET X2=S1*SIN TH
280  LET Y2=S1*COS TH
290  DRAW X2-X1,Y2-Y1
300  LET X1=X2
310  LET Y1=Y2
320  NEXT N
330  LET DT=1.5*PI/6
340  LET TH=0
400  FOR N=0 TO 6
410  LET TH=N*DT
420  LET X1=S1*SIN TH
430  LET X2=S2*SIN TH
440  LET Y1=S1*COS TH
450  LET Y2=S2*COS TH
460  PLOT XC+X1,YC+Y1
470  DRAW X2-X1,Y2-Y1
480  LET XT=XC+ST*SIN TH
490  LET YT=YC+ST*COS TH
500  GOSUB 1000
510  NEXT N
520  REM RUCICKA
530  LET P1=0
540  LET FS=6
550  LET RP=R-5
560  FOR K=1 TO 20
570  FOR P=0 TO 6 STEP .2
580  GOSUB 700
590  NEXT P
600  FOR P=6 TO 0 STEP -.2
610  GOSUB 700
620  NEXT P
630  NEXT K
```

```
640  STOP
690  REM PODPROGRAM PRO POSUN RUCICKY
700  LET TH=1.5*PI*P1/FS
710  REM VYMAZANI STARE RUCICKY
720  INVERSE 1
730  PLOT XC,YC
740  DRAW RP*SIN TH,RP*COS TH
750  INVERSE 0
760  LET TH=1.5*PI*P/FS
770  REM NOVA RUCICKA
790  PLOT XC,YC
800  DRAW RP*SIN TH,RP*COS TH
810  LET P1=P
820  RETURN
990  REM TEXTOVY SYMBOL NA POZICI JEMNE GRAFIKY
1000 PRINT AT 0,0;STR$ (N)
1010 REM COPY SYMBOL
1020 FOR J=0 TO 7
1030 FOR I=0 TO 7
1040 IF POINT (I,175-J)=0 THEN GOTO 1060
1050 PLOT XT+I-4,YT-J+4
1060 NEXT I
1070 NEXT J
1080 PRINT AT 0,0;" "
1090 RETURN
```

Obr. 7.15.

Pokud chceme dvě ručičky, musíme užít dvakrát tutéž rutinu pro jiný úhel nebo jinou ryhlost otáčení. Můžeme mít samozřejmě i více ručiček. Pokud bude mít každá jiný tvar (délku), musíme mít více rutin pro jejich tisk.

Kruhový diagram

Dosti atraktivní a užívanou formou grafického znázornění nějaké skutečnosti je tzv. kruhový diagram. Ten se užívá pro znázornění nějakého poměru (nebo poměrů), hlavně v obchodě nebo ve statistice.

Jak už název napovídá, kruhový diagram se skládá z kruhu rozdêleného na několik částí (kruhových výsečí), z nichž každá znázorňuje jednu položku. Velikost úhlů každé položky je přímo úměrná její velikosti vůči ostatním položkám.

Pro nakreslení kruhového diagramu musíme tedy nakreslit několik kruhových výsečí. Nejjednodušší bude nakreslit celý kruh (příkazem CIRCLE) a potom z jeho středu vést čáry k okraji, kde úhly mezi jednotlivými čarami budou úměrné poměru položek. Budeme-li brát poměr v procentech, potom pro souřadnice konce čar (vlastně poloměrů) dostaneme rovnice

```
XR=DX*COS (TH)-DY*SIN (TH)
YR=DX*SIN (TH)+DY*COS (TH)
TH=P*2*PI/100
```

kde P je procentuální podíl kresleného úhlu. DX a DY jsou

souřadnice konce posledně nakreslené úsečky. Souřadnice konce nové úsečky jsou XR a YR. Po nakreslení nové úsečky se do proměnných DX a DY uloží obsah XR a YR.

Jednoduchý program kreslící kruhový diagram rozdělený na 5 částí je uveden dále. Číslování jednotlivých částí se dost dobře nedá provést pomocí PRINT AT, protože je pravděpodobné, že bychom se netrefili doprostřed. Proto se využívá rutina pro tisk znaků na pozici jemné grafiky (viz minulý program nebo kapitola 5.)

Legenda označující, co který sektor znamená, už může být vypsána obyčejně pomocí PRINT AT.

```
100  REM JEDNODUCHY KRUHOVY DIAGRAM
110  DIM S(5)
120  REM NACTENI DAT
130  FOR N=1 TO 5
140  READ S(N)
150  NEXT N
160  DATA 20,15,25,30,10
170  REM KRESLENI KRUHU
180  LET XC=180
190  LET YC=88
200  LET R=50
210  CIRCLE XC,YC,R
220  REM SEKTORY
230  LET DX=R
240  LET DY=0
250  LET TH=0
260  FOR N=1 TO 5
270  LET XR=DX*COS TH-DY*SIN TH
280  LET YR=DX*SIN TH+DY*COS TH
290  REM KRESLENI CARY
300  PLOT XC,YC
310  DRAW XR,YR
320  LET DX=XR
330  LET DY=YR
340  REM NAPSANI CISLA SEKTORU
350  LET TH=PI*S(N)/100
360  LET XR=DX*COS TH-DY*SIN TH
370  LET YR=DX*SIN TH+DY*COS TH
380  LET XT=XC+INT (.7*XR)
390  LET YT=YC+INT (.7*YR)
400  LET A$=STR$ (N)
410  GOSUB 600
420  LET DX=XR
430  LET DY=YR
440  NEXT N
450  REM POPIS
460  PRINT AT 1,1;"PRIKLAD KRUHOVEHO DIAGRAMU"
470  PRINT AT 4,1;"1 = MASO"
480  PRINT AT 6,1;"2 = RYBY"
490  PRINT AT 8,1;"3 = OBILNINY"
500  PRINT AT 10,1;"4 = OVOCE"
510  PRINT AT 12,1;"5 = ZELENINA"
```

```
520  STOP
600  PRINT AT 0,0;A$
610  FOR J=0 TO 7
620  FOR I=0 TO 7
630  IF POINT (I,175-J)=0 THEN GOTO 650
640  PLOT XT+I-4,YT-J+4
650  NEXT I
660  NEXT J
670  PRINT AT 0,0;" "
680  RETURN
```

Obr. 7.18.

Poznámka : číslování programů /obrázků/ je provedeno přesně dle originálu knihy.

Kapitola 8: Svět pohybu

V mnoha počítačových hrách potřebujeme pohybovat určitým objektem. Jak asi víte, počítač 50krát za sekundu zobrazuje televizní obraz, což naše oko není schopno zachytit. Pokud tedy budeme rychle a po malých krocích měnit pozici určitého znaku nebo obrázku, dostaneme dojem pohybu (to je také princip kreslených filmů). Musíme vždy smazat starý obrázek a na novou pozici nakreslit nový. Vzdálenost mezi těmito dvěma obrázky nesmí být velká, protože by pohyb vypadal trhaně.

Pro větší přesvědčivost pohyby může pohybující se předmět měnit tvar. Vezměme si třeba jdoucího panáčka. Pokud by se pohyboval stále stejně, vypadalo by to jako by se klouzal po ledě. Proto musíme měnit pozici nohou a případně i rukou nebo celého těla. Pro takovýto pohyb potom můžeme dostat více obrázků, které budeme pravidelně střídavě zobrazovat.

Pohybující se míček

Často ve hrách využijeme pohybující se míček. Symbol míče vždy zobrazíme na novou pozici; ze staré pozice ho vymažeme jednoduše přepsáním mezerou.

Začněme tedy hledat postup pro pohybování míčkem po obrazovce. Pro zaqatek si vybereme tu nejjednodušší podobu míčku - užijeme znak s ASCII kódem 143 (černý čtverec). Pozici míčku znázorníme dvěma souřadnicemi - y (řádek) a x (sloupec). Míček umístíme do středu obrazovky (x=15, y=10) pomocí

```
100  PRINT AT Y,X;CHR$ 143
```

Pokud chceme míčkem pohybovat horizontálně po obrazovce, je to jednoduché. Abychom získali jeho novou pozici přičteme k x jedničku pro pohyb doprava nebo od x jedničku odečteme pro pohyb doleva. Tak dostaneme novou hodnotu x, nazvěme ji xn. Tu využijeme v příkazu PRINT AT pro zobrazení nového míčku. Můžete se ptát, proč jsme nezměnili rovnou proměnnou x. Tu ale musíme využít pro smazání starého míčku (přepsáním mezerou). Před dalšími pohyby potom uložíme do x hodnotu xn.

Pohyb vertikálně je podobný, jenomže měníme souřadnici y. Pro pohyb nahoru odečítáme, pro pohyb dolů přičítáme. Zase užijeme proměnnou yn pro novou pozici a míček vymažeme přepsáním mezerou.

Pro pohyb šikmo musíme měnit obě souřadnice, to už je trošku složitější. Pro pohyb vpravo nahoru změníme x+1 a y-1. Pro pohyb vpravo dolů změníme x+1 a y+1. Vlevo nahoru bude x-1 a y-1. No a pohyb vlevo dolu bude x-1 a y+1. Nejlepší je si nakreslit schéma pohybu do všech osmi směrů; nezapomeňte že x je sloupec a y řádek. Pro pohyb šikmo musíte nejprve změnit obě souřadnice a potom až zobrazit nový míček (nebo cokoliv jiného).

Pro rychlejší pohyb můžeme souřadnice x a y měnit o více než jednu jednotku. To se dá vyřešit zavedením proměnných dx a dy, které reprezentují hodnotu přičtenou (nebo odečtenou) k x a k y

při každém kroku. Výhoda tohoto způsobu je i to, že můžeme snadno obrátit směry pohybu, pokud zadáme dx=-dx a dy=-dy.

Odrážení od okrajů obrazovky

Pokud zkoušíte pohybovat míčkem po obrazovce, tak se objeví určitý problém. Řekněme, že jím pohybujeme horizontálně doprava. Vše je v pořádku až do okamžiku, kdy dojedeme na pravý okraj obrazovky a hodnota x=31. Při dalším kroku doprava dojde k zastavení programu a vypsání chybového hlášení. To znamená, že hodnota x nesmí přesáhnout 31. Stejně tak nesmí být menší než 0. No a y také nesmí být větší než 21 nebo menší než 0. Musíme tedy zjistit, kdy se míček octne na okraji obrazovky a v tom okamžiku změnit jeho směr. Bude to vypadat, jako by se ode zdi odrazil pod stejným úhlem, jako do ní narazil.

Je celkem jednoduché tento problém vyřešit. Po vypočítání nových souřadnic míčku (xn,yn) zkontrolujeme, zda některá z nich dosáhla kritické hodnoty (okraje). Pokud ano, pak pro příslušný směr změníme znaménko proměnné dx nebo dy, kterou přičítáme k příslušné souřadnici (pokud bude xn=31 nebo xn=0, bude dx=-dx). Tento postup bude perfektně fungovat, pokud dx a dy je jedna (nebo -1). Při větších hodnotách musíme kontrolovat ještě před přičtením. Celý postup odrážení ukazuje názorně tento program.

```
100  REM JEDNODUSE SE POHYBUJICI MICEK
110  BORDER 1
120  PAPER 4
130  CLS
140  LET X=15
150  LET Y=10
160  LET DX=1
170  LET DY=1
180  INK 7
190  PRINT AT Y,X;CHR$ 143
200  REM POHYBOVA SMYCKA
210  LET XN=X+DX
220  LET YN=Y+DY
230  IF XN=0 OR XN=31 THEN LET DX=-DX
240  IF YN=0 OR YN=21 THEN LET DY=-DY
250  PRINT AT Y,X;" "
260  PRINT AT YN,XN;CHR$ 143
270  LET X=XN: LET Y=YN
280  GOTO 210
```

Obr. 8.2.

Míček se "rozjede" ze středu obrazovky a pokračuje šikmo stále dále, přičemž se odráží od okrajů obrazovky.

Odraz od pálky

Ve hrách jako je SQUASH nebo BREAKOUT se míček odráží také od pálky ovládané hráčem. Pozici pálky si označíme bx a by.

Nyní tedy do podmínek kontrolujících, zda míček není na okraji

musíme zahrnout i podmínku, zda nová pozice míčku (xn,yn) není totožná s pozicí pálky. Pokud jsou si tyto souřadnice rovny musíme změnit znaménko buď u dy (pokud je pálka nahoře nebo dole) nebo u dx (pokud je pálka na boku obrazovky). Zároveň se může změnit skóre.

Ale pálku většinou netvoří jeden znak, ale více. Vezměme si třeba tříznakovou pálku umístěnou horizontálně dole na obrazovce. Potom musíme novou pozici míčku kontrolovat se třemi souřadnicemi pálky. Souřadnice pálky by bude vždy stejná, budeme mít tři rozdílné x-ové souřadnice (bx, bx+1, bx+2) - bx bude souřadnice znaku pálky, který je nejvíce vlevo.

Často se ale míček odráží v závislosti na tom, který znak pálky zasáhne. Pokud se odráží od středu pálky (xn=bX+1), změní se dx na 0 a dy=-dy. To způsobí, že míček poletí kolmo vzhůru. Pokud je zasažen některý z okrajů pálky, tak se mění znaménko dx i dy. Většina her tohoto typu nepřipouští, aby se míček odrazil od řádku, v němž se pohybuje pálka, bez odrazu od pálky. Pokud se tedy netrefíme, ztrácíme jeden míček. Netrefení se zjistíme snadno - pro předešlou pálku se y nesmí rovnat 21 a když se rovná, došlo k netrefení.

Pohybování pálkou

Nejjednodušeji můžeme pálkou pohybovat pomocí dvou tlačítek, jedno znamená pohyb doleva a jedno doprava. Zvolme třeba Z a X.

Nyní už můžeme vytvořit jednoduchoučkou hru, ve které získáváme body odrážením míčku od pálky. Máme pět míčků, každý nový míček začíná ze středu obrazovky.

```
100  REM JEDNODUCHY SQUASH
110  BORDER 1
120  LET HS=0
130  LET BX=15: LET BY=20
140  LET BX1=BX: LET BY1=BY
150  CLS
160  LET NB=0
170  LET X=15: LET Y=9
180  LET SC=0
190  LET B$=CHR$ 143+CHR$ 143+CHR$ 143
200  PRINT AT 21,0;"ZMACKNI MEZERU PRO PODANI        ";
210  LET A$=INKEY$: IF A$<>" " THEN GOTO 210
220  LET X=15: LET Y=9
230  PRINT AT Y,X;CHR$ 143
240  LET DX=INT (RND*3)-1: LET DY=-1
250  PRINT AT 21,0;"SKORE = ";SC;
260  PRINT "    REKORD = ";HS;"     ";
270  LET M$=INKEY$
280  IF M$="X" OR M$="x" THEN LET BX1=BX+1
290  IF M$="Z" OR M$="z" THEN LET BX1=BX-1
300  IF BX1<0 THEN LET BX1=0
310  IF BX1+2>31 THEN LET BX1=29
320  PRINT AT BY,BX;"   "
330  PRINT AT BY1,BX1;B$
```

```
340  LET BX=BX1: LET BY=BY1
350  LET XN=X+DX: LET YN=Y+DY
360  IF XN=0 OR XN=31 THEN LET DX=-DX
370  IF YN=0 THEN LET DY=-DY
380  IF YN=BY THEN GOTO 450
390  PRINT AT Y,X;" "
400  PRINT AT YN,XN;CHR$ 143
410  LET X=XN: LET Y=YN
420  GOTO 250
450  IF XN=BX THEN GOTO 550
460  IF XN=BX+1 THEN GOTO 550
470  IF XN=BX+2 THEN GOTO 550
480  LET NB=NB+1
490  IF NB=5 THEN GOTO 1000
500  PRINT AT Y,X;" "
510  GOTO 200
550  LET DX=-DX : LET DY=-1
560  IF XN=BX AND BX<>0 THEN LET DX=-1
570  IF XN=BX+2 AND BX<>29 THEN LET DX=1
580  LET SC=SC+1
590  GOTO 250
990  REM KONEC HRY
1000 BEEP 1,12
1010 PRINT AT 0,0;"KONEC HRY"
1020 IF HS>SC THEN GOTO 1050
1030 PRINT AT 2,0;FLASH 1;"*NOVY REKORD*"
1040 LET HS=SC
1050 INPUT "DALSI HRA (A/N)",G$
1060 IF G$="A" OR G$="a" THEN GOTO 130
1070 STOP
```

Obr. 8.3.

Animace pomocí jemné grafiky

Velkou nevýhodou právě popsaného způsobu pohybu je to, že míček (nebo cokoli jiného) se pohybuje po příliš velkých skocích, pohyb působí trhaně. Pokud využijeme jemnou grafiku, bude výsledný dojem lepší.

V souřadnicovém systému jemné grafiky dojde k malé změně. Souřšdnice y bude mít hodnotu 0 na nejspodnějším řádku obrazovky a pokud ji budeme zvětšovat, bude se bod pohybovat nahoru (a pokud zmenšovat, tak dolů). Tzn. že pro pohyb vpravo nahoru změníme x=x+1 a y=y+1; pro pohyb vpravo dolů x=x+1 a y=y-1. Pro pohyb vlevo dolů změníme x=x-1 a y=y-1; pro pohyb vlevo nahoru bude x=x-1 a y=y+1. Znovu doporučuji si všech osm směrů jednoduše načrtnout. V jemné grafice je jeden krok roven jednomu bodu, což již dává realistický dojem plynulého pohybu. Tento program ukazuje bod pohybující se po obrazovce a odrážející se od jejích okrajů.

```
100  REM JEDNODUCHY POHYB V JEMNE GRAFICE
110  BORDER 6
120  LET X=128
130  LET Y=88
```

```
140  LET DX=2
150  LET DY=2
170  PLOT X,Y
180  LET XN=X+DX
190  LET YN=Y+DY
200  IF XN>0 AND YN<255 THEN GOTO 220
210  LET DX=-DX: LET XN=XN+DX
220  IF YN>0 AND YN<175 THEN GOTO 240
230  LET DY=-DY: LET YN=YN+DX
240  PLOT OVER 1;X,Y: REM VYMAZANI STAREHO BODU
250  PLOT OVER 1;XN,YN: REM NOVY BOD
260  LET X=XN
270  LET Y=YN
280  GOTO 180
```

Obr. 8.5.

Pohybování větším objekty

Zatím jsme pohybovali jenom samostatným bodem, ale většinou budeme chtít pohybovat s něčím větším, třeba s létajícím talířem jak demonstruje tento program.

```
100  REM LETAJICI TALIR V JEMNE GRAFICE
110  PLOT 0,0
120  DRAW 255,0
130  DRAW 0,175
140  DRAW -255,0
150  DRAW 0,-175
160  LET X=128: LET Y=88
170  LET X1=X: LET Y1=Y
180  LET DX=2: LET DY=DX
190  OVER 1
200  GOSUB 500
210  FOR N=1 TO 500
220  LET X1=X+DX: LET Y1=Y+DY
230  IF X1+DX<6 OR X1+DX>248 THEN LET DX=-DX
240  IF Y1+DY<6 OR Y1+DY>168 THEN LET DY=-DY
250  GOSUB 500
260  LET X=X1: LET Y=Y1
270  GOSUB 500: NEXT N: STOP
500  PLOT X,Y-2: DRAW 3,0:DRAW 2,2
510  DRAW -2,2: DRAW -1,0: DRAW -2,2
520  DRAW -2,-2: DRAW -1,0: DRAW -2,-2
530  DRAW 2,-2: DRAW 3,0: RETURN
```

Obr. 8.6.

Létající talíř je kreslen pomocí příkazu PLOT a série příkazů DRAW a to vždy na pozici x,y. Program hned na začátku nastaví OVER 1 a proto smazání talíře provedeme jeho opětovným vykreslením do stejných souřadnic. Potom se x,y změní na nově vypočítané souřadnice x1,y1 a vykreslí se nový talíř.

Stejně jako při pohybu míčku nebo bodu i zde musíme kontrolovat vyjetí z obrazovky a odrážet talíř od jejích okrajů. Talíř se pohybuje pomocí proměnných dx a dy, které přičítáme k souřadnicím x a y. Pokud x1 nebo y1 dosáhne kraje obrazovky,

změní se znaménko buď u dx nebo u dy. Všimněte si, že program nekontroluje přímo okraj obrazovky, musíme totiž brát v úvahu výšku a šířku talíře (x,y jsou souřadnice jeho středu).
Jenomže i když program funguje pěkně, je příliš pomalý. Procedura by musela být napsána ve strojovém kódu, aby byla dostatečně rychlá. Proto většina her je napsána ve strojovém kódu.

Animace pomocí definovaných znaků

Existuje ještě jeden způsob, jak provést animaci pomocí PRINT AT a přitom docílit solidního dojmu. Při něm se využívjí definované znaky uživatelské grafiky.
Řekněme, že chceme vytvořit malý diamant, který se bude pohybovat horizontálně zleva doprava přes obrazovku. Musíme si navrhnout tvay znaků (viz kap. 5) a to tak, že diamant bude nejprve úplně vlevo ve dvou znakových čtvercích a postupně se bude vždy po dvou bodech posunovat doprava. Dostaneme tak 5 obrázků, tedy 10 znaků (ale dva z nich jsou mezery). Program to ukazuje v praxi.

```
100  REM POHYBUJICI SE DIAMANT POMOCI UZIVATELSKE GRAFIKY
120  GOSUB 500
130  FOR M=1 TO 20
140  FOR C=1 TO 20
150  PRINT AT 10,C;CHR$ 144;CHR$ 128
160  PRINT AT 10,C;CHR$ 145;CHR$ 148
170  PRINT AT 10,C;CHR$ 146;CHR$ 149
180  PRINT AT 10,C;CHR$ 147;CHR$ 150
190  PRINT AT 10,C;CHR$ 128;CHR$ 144
200  NEXT C
210  NEXT M
220  STOP
500  POKE USR "A",BIN 00001000
510  POKE USR "A"+1,BIN 00010100
520  POKE USR "A"+2,BIN 00100010
530  POKE USR "A"+3,BIN 01000001
540  POKE USR "A"+4,BIN 00100010
550  POKE USR "A"+5,BIN 00010100
560  POKE USR "A"+6,BIN 00001000
565  POKE USR "A"+7,0
570  POKE USR "B",BIN 00000010
580  POKE USR "B"+1,BIN 00000101
590  POKE USR "B"+2,BIN 00001000
600  POKE USR "B"+3,BIN 00010000
610  POKE USR "B"+4,BIN 00001000
620  POKE USR "B"+5,BIN 00000101
630  POKE USR "B"+6,BIN 00000010
640  POKE USR "B"+7,0
650  POKE USR "C",BIN 00000000
660  POKE USR "C"+1,BIN 00000001
670  POKE USR "C"+2,BIN 00000010
680  POKE USR "C"+3,BIN 00000100
```

```
700  POKE USR "C"+4,BIN 00000010
710  POKE USR "C"+5,BIN 00000001
720  POKE USR "C"+6,BIN 00000000
730  POKE USR "C"+7,0
740  POKE USR "D",0
750  POKE USR "D"+1,0
760  POKE USR "D"+2,0
770  POKE USR "D"+3,1
780  POKE USR "D"+4,0
790  POKE USR "D"+5,0
800  POKE USR "D"+6,0
810  POKE USR "D"+7,0
820  POKE USR "E",0
830  POKE USR "E"+1,0
840  POKE USR "E"+2,BIN 10000000
850  POKE USR "E"+3,BIN 01000000
860  POKE USR "E"+4,BIN 10000000
870  POKE USR "E"+5,0
880  POKE USR "E"+6,0
890  POKE USR "E"+7,0
900  POKE USR "F",BIN 10000000
910  POKE USR "F"+1,BIN 01000000
920  POKE USR "F"+2,BIN 00100000
930  POKE USR "F"+3,BIN 00010000
940  POKE USR "F"+4,BIN 00100000
950  POKE USR "F"+5,BIN 01000000
960  POKE USR "F"+6,BIN 10000000
970  POKE USR "F"+7,0
980  POKE USR "G",BIN 00100000
990  POKE USR "G"+1,BIN 01010000
1000 POKE USR "G"+2,BIN 10001000
1010 POKE USR "G"+3,BIN 00000100
1020 POKE USR "G"+4,BIN 10001000
1030 POKE USR "G"+5,BIN 01010000
1040 POKE USR "G"+6,BIN 00100000
1050 POKE USR "G"+7,0
1060 RETURN
```

Obr. 8.8.

Srážka s jiným předmětem

V mnoha hrách potřebujeme zjistit, zda vyslaná střela nebo bomba trefila cíl nebo zda naše raketka není zasažena. To se dá zjistit pomocí porovnání souřadnic střely a tělesa před pohybem střely. Pokud budou nové souřadnice střely stejné jako souřadnice něčeho, co máme zasáhnout, pak tedy musíme vymazat střelu a na místě zasaženého předmětu zobrazit explozi a posléze i tu vymazat. Jinak se střela pohybuje notmálně dál.

Pokud ale máme mnoho zasažitelných předmětů, je tento postup zdlouhavý a pomalý. Lepší je využít funkci POINT, která zjistí, zda v bodě, do nějž se přesune střela, něco je. Pokud bude výsledek POINTu 1, potom střela něco zasáhla a je třeba se podle toho zachovat, jinak se střela může volně posunout.

Nebo využijete funkci ATTR, která zjistí barevný kód na další

```
650  POKE USR "B"+7,BIN 10010000
660  POKE USR "C",BIN 01111100
670  POKE USR "C"+1,BIN 01000001
680  POKE USR "C"+2,BIN 01011100
690  POKE USR "C"+3,BIN 01000000
700  POKE USR "C"+4,BIN 00111111
710  POKE USR "C"+5,BIN 00100100
720  POKE USR "C"+6,BIN 00011000
730  POKE USR "C"+7,BIN 00100100
740  POKE USR "D",BIN 01111100
750  POKE USR "D"+1,BIN 10000010
760  POKE USR "D"+2,BIN 00111010
770  POKE USR "D"+3,BIN 00000010
780  POKE USR "D"+4,BIN 11111100
790  POKE USR "D"+5,BIN 00100100
800  POKE USR "D"+6,BIN 00011000
810  POKE USR "D"+7,BIN 00100100
820  RETURN
```

Obr. 8.9.

Pro kráčejícího muže je třeba větší úroveň a komfort. Proto si znaky třeba uložte do řetězce a načítejte je z něj. Také můžete pro větší realističnost udělat více než jenom dva rozdílné tvary, pokuste se zachytit více momentů pohybu. Zde ale můžete narazit také na problém paměťového prostoru. Pokud se ale bude předmět pohybovat rychle, potom klidně stačí dva rozdílné tvary, protože to v té rychlosti nepostřehnete.

Animace obrázků vyžaduje nejprve jejich studium v reálném životě. Potom je teprve můžete zobrazit na počítači a hýbat s nimi.

Kapitola 9: Využití prostorové grafiky

Obrázky a grafy, které jsme doposud kreslili, využívaly pouze dvouosý dvourozměrný zobrazovací systém s horizontální osou x a vertikální osou y. Ale v reálném světě existuje ještě třetí rozměr - pokud si x a y představíte jako souřadnice bodu na papíře, potom třetí rozměr (určený osou z) je výška bodu nad papírem.

Jelikož máme k disopozici pouze plochou obrazovku, musíme si tříosý systém pro ni upravit. Nejlépe vyjdeme s osami umístěnými tak, že z jde kolmo nahoru a osy x a y svírají úhel 30 stupňů s horizontální osou (kterou ovšem nevyužíváme). Vypadá to asi takto:

```
Y  Z  X
 \ ! /
  \!/
   Ø
```

ale úhel mezi osami je samozřejmě mnohem větší (jinak to nejde přiblížit). Nyní tedy pohyb po ose x je vpravo nahoru, po ose y vlevo nahoru a po ose z kolmo nahoru (pro kladné souřadnice).

Tříosý graf

Jedním z typů lehce zobrazitelných obrázků v třírozměrném systému je graf. Prvním krokem pro nakreslení grafu je vybrat si počátek, tedy bod o souřadnicích (Ø,Ø,Ø). Vůči němu budou počítány souřadnice ostatních bodů. Dále si zkusíme nakreslit síť bodů, jejichž souřadnice Z je nulová.

Pro nakreslení osy X pod určitým úhlem potřebujem pro y-ovou souřadnici poloviční krok než pro x-ovou, proto pro body této osy (kde y a z je Ø) platí:

```
X1=CX+X
Y1=CY+X/2
```

kde CX a CY jsou souřadnice počátku.

Pokud jsou souřadnice x a z obě nulové, dostaneme osu y. Ta jde doleva nahoru. Poněvadž jde doleva od počátku, znamená to, že x-ová souřadnice bodu osy musí být menší než CX. Pro úhel 30 stupňů je postup stejný jako u osy x, jenom x se zmenšuje. Tedy:

```
X1=CX-Y
Y1=CY+Y/2
```

Pro libovolný bod se zetovou souřadnicí nulovou sloučíme tyto dva vztahy a dostaneme tyto vzorce pro souřadnice bodu:

```
X1=CX+X-Y
Y1=CY+X/2+Y/2
```

Tento program využívá tyto rovnice a kreslí síť čar rovnoběžných s osami x nebo y s výškou 0.

```
100  REM 3-OSY GRAF PRO Z=0
110  CLS
120  LET CX=116: LET CY=20: REM POCATEK
130  LET XM=96: LET YM=80: REM MAXIMALNI HODNOTY
140  LET XS=12: LET YS=8: REM KROKY
145  REM OSY X
150  FOR Y=0 TO YM STEP YS
160  LET X1=-Y
170  LET X2=XM-Y
180  LET Y1=Y/2
190  LET Y2=(XM+Y)/2
200  PLOT INT (CX+X1),INT (CY+Y1)
210  DRAW INT (X2-X1),INT (Y2-Y1)
220  NEXT Y
225  REM OSY Y
230  FOR X=0 TO XM STEP XS
240  LET X1=X
250  LET X2=X-YM
260  LET Y1=X/2
270  LET Y2=(X+YM)/2
280  PLOT INT (CX+X1),INT (CY+Y1)
290  PLOT INT (X2-X1),INT (Y2-Y1)
300  NEXT X
310  PLOT CX,CY
320  DRAW INT (XM+XS),INT ((XM+XS)/2)
330  PLOT CX,CY
340  DRAW -INT (YM+YS),INT ((YM+YS)/2)
350  PRINT AT 13,3;"Y"
360  PRINT AT 12,28;"X"
370  PRINT AT 20,14;"0"
```

Obr. 9.2.

Z-ová hodnota se připočítává vertikálně, proto stačí změnit pouze hodnotu Y. Pokud chceme nakreslit čáru, jejíž výška reprezentuje hodnotu z, musíme vypočítat souřadnice pro vrchol čáry. Hodnota X bude stejná jako X1 a pro novou hodnotu Y se přičte hodnota z k Y, potom pro souřadnice X2,Y2 dostaneme:

$$X2=X1=CX+X-Y$$
$$Y2=Y1+Z=CY+X/2+Y/2+Z$$

Čáru potom dostaneme jako spojnici bodů X1,Y1 a X2,Y2. Do bodu X1,Y1 se dostaneme pomocí PLOT, čáru nakreslíme příkazem DRAW. Řekněme např. že chceme zobrazit graf funkce Z=(X^2)/3+(Y^2)/2. Pro zobrazení hodnoty z nakreslíme vždy čáru z bodu X,Y o velikosti úměrné vypočítané hodnotě.

```
100  REM JEDNODUCHÝ TŘÍDIMENZNÍ GRAF
110  CLS
120  LET CX=116: LET CY=20: REM POCATEK
130  LET XM=96: LET YM=80: REM MAX. HODNOTY
140  LET XS=16: LET YS=10: REM KROKY
```

```
150  DIM Z(7,9)
160  FOR Y=1 TO 9
170  FOR X=1 TO 7
180  LET X1=X-1: LET Y1=Y-1
190  LET Z(X,Y)=X1*X1/3+Y1*Y1/2
200  LET X2=INT (XS*X1-YS*Y1)
210  LET Y2=INT ((XS*X1+YS*Y1)/2)
220  PLOT CX+X2,CY+Y2
230  DRAW 0,Z(X,Y)
240  NEXT X
250  NEXT Y
260  PLOT CX,CY
270  DRAW INT (XM+XS),INT ((XM+XS)/2)
280  PLOT CX,CY
290  DRAW INT -(YM+YS),INT ((YM+YS)/2)
300  PRINT AT 13,2;"Y"
310  PRINT AT 12,29;"X"
320  PRINT AT 20,14;"0"
330  STOP
```

Obr. 9.4.

Aby na obrázku jednotlivé čáry nesplývaly, musí být rozdílný krok pro x a pro y.

Produkce širokého grafu

Zatím jsme pro znázornění hodnoty z v třírozměrném grafu používali pouze tenkou čáru. Můžeme vytvořit atraktivnější obrázek pomocí protažení čáry ve směru rovnoběžném s osou x. To se zařídí lehce - nakreslí se ještě jedna čára se stejnou výškou z ale s trochu posunutými souřadnicemi počátku (čáry) a obě čáry se spojí (dole a nahoře). Prostor ohraničený těmito čarami vyplníme barvou INK. Následující program bere tytéž hodnoty z jako předešlý a zobrazuje je pomocí těchto pásků.

```
100  REM TRIDIMENZNI GRAF S PASKOVYM ZOBRAZENIM
110  REM Z=(4+2*SIN (X/20))*(Y/10+1)
120  LET CX=128: LET CY=16
130  LET DX=10: LET DY=5
140  GOSUB 400
150  FOR X=100 TO 0 STEP -20
160  FOR Y=90 TO 0 STEP -15
170  LET Z=INT ((4+2*SIN (X/20))*(Y/10+1))
180  GOSUB 500
190  NEXT Y
200  NEXT X
210  STOP
390  REM VYKRESLENI OS
400  FOR X=0 TO 100 STEP 20
410  PLOT CX+X,CY+X/2
420  DRAW -100,50
430  NEXT X
440  FOR Y=0 TO 90 STEP 15
450  PLOT CX-Y,CY+Y/2
```

```
460  DRAW 110,55
470  NEXT Y
480  RETURN
500  REM PODPROGRAM KRESLICI PASKY GRAFU
510  FOR N=0 TO Z-1
520  PLOT CX+X-Y,CY+X/2+Y/2+N
530  DRAW DX,DY
540  NEXT N
550  INVERSE 1
560  PLOT CX+X-Y,CY+X/2+Y/2
570  DRAW DX,DY
580  DRAW 0,Z
590  DRAW -DX,-DY
600  DRAW 0,-Z
610  INVERSE 0
620  RETURN
```

Obr. 9.6.

Pásky se kreslí s kratšími hranami rovnoběžnými s osou x a s delšími jdoucími kolmo nahoru. Všimněte si, že hodnota x a y se zmenšuje, takže nejprve se kreslí pásky vzadu. Každý pásek je po nakreslení obtažen čárou s INVERSE 1, takže se vzájemně nepřekrývají.

Tvorba kubických grafů

Nyní doplníme graf ještě o pásku rovnoběžnou s osou y, takže vlastně po menší úpravě dostaneme kvádr, který svou výškou znázorňuje hodnotu z. Pro lepší dojem můžeme jednu stěnu kvádru vybarvit. Názorně to předvádí tento program.

```
100  REM TRIDIMENZNI KUBICKY GRAF
110  REM Z=(4+2*COS (X/20))*(Y/10+1)
120  LET CX=128: LET CY=16
130  LET DX1=10: LET DY1=5
140  LET DX2=8: LET DY2=4
150  REM NAKRESLENI OS
160  GOSUB 400
170  FOR X=100 TO 0 STEP -20
180  FOR Y=90 TO 0 STEP -15
190  LET Z=INT ((4+2*COS (X/20))*(Y/10+1))
200  GOSUB 500
210  NEXT Y
220  NEXT X
230  STOP
390  REM PODPROGRAM KRESLICI OSY
400  FOR X=0 TO 100 STEP 20
410  PLOT CX+X,CY+X/2
420  DRAW -100,50
430  NEXT X
440  FOR Y=0 TO 90 STEP 15
450  PLOT CX-Y,CY+Y/2
460  DRAW 110,55
470  NEXT Y
```

```
480  RETURN
500  REM PODPROGRAM KRESLICI KVADRY
510  FOR N=Ø TO Z-1
520  PLOT CX+X-Y,CY+X/2+Y/2+N
530  DRAW DX1,DY1
540  NEXT N
545  REM VYMAZANI PROSTORU STENY, KTERA BUDE PRAZDNA
550  INVERSE 1
560  FOR N=Ø TO Z-1
570  PLOT CX+X-Y,CY+X/2+Y/2+N
580  DRAW -DX2,DY2
590  NEXT N
600  INVERSE Ø
605  REM NEVYBARVENA STENA
610  PLOT CX+X-Y,CY+X/2+Y/2
620  DRAW -DX2,DY2
630  DRAW Ø,Z
640  DRAW DX2,-DY2
650  DRAW Ø,-Z
660  REM VYMAZANI VRCHOLU KVADRU
670  INVERSE 1
680  FOR N=Ø TO DX2-1
690  PLOT CX+X-Y-N,CY+Z+(X+Y+N)/2
700  DRAW DX1,DY1
710  NEXT N
720  INVERSE Ø
725  REM NAKRESLENI VRCHOLU KVADRU
730  PLOT CX+X-Y,CY+X/2+Y/2+Z
740  DRAW DX1,DY1
750  DRAW -DX2,DY2
760  DRAW -DX1,-DY1
770  DRAW DX2,-DY2
780  RETURN
```

Obr. 9.8.

Program nakreslí jednu stěnu kvádru v barvě INK. Potom při INVERSE 1 vyprázdní prostor pro druhou stěnu a tu nakreslí pomocí čar při INVERSE Ø. Nakonec se vyprázdní vrchol pro třetí stěnu (tu horní, vršek) a ta se nakreslí také pomocí čar.

Kubický dvourozměrný graf

Jednoduchý dvourozměrný graf můžeme také nakreslit pomocí kvádrů. V tomto případě budeme mít osu X pod úhlem 30 stupňů a nejprve nakreslíme pozadí grafu. Potom nakreslíme kvádry pomocí stejné techniky jako předtím. Názorně to ukazuje následující program.

```
100  REM DVOUOSY GRAF S KVADRY
110  REM Y=X*X/100
120  LET CX=64: LET CY=16
130  LET AX=6: LET AY=3
140  LET BX=8: LET BY=4
150  LET LX=100: LET LY=100
```

```
160  REM NAKRESLENI POZADI
170  GOSUB 400
180  REM NAKRESLENI GRAFU
190  FOR X=100 TO 0 STEP -15
200  LET Y=X*X/100
210  GOSUB 500
220  NEXT X
230  STOP
400  PLOT CX-BX,CY+BY
410  DRAW LX,LX/2
420  DRAW 0,LY
430  DRAW -LX,-LX/2
440  DRAW 0,-LY
450  RETURN
500  REM PODPROGRAM KRESLICI KVADRY
510  FOR N=0 TO Y-1
520  PLOT CX+X,CY+X/2+N
530  DRAW AX,AY
540  NEXT N
550  INVERSE 1
560  FOR N=0 TO Y-1
570  PLOT CX+X,CY+X/2+N
580  DRAW -BX,BY
590  NEXT N
600  INVERSE 0
610  PLOT CX+X,CY+X/2
620  DRAW -BX,BY
630  DRAW 0,Y
640  DRAW BX,-BY
650  DRAW 0,-Y
660  INVERSE 1
670  FOR N=0 TO BX-1
680  PLOT CX+X-N,CY+Y+X/2+N/2
690  DRAW AX,AY
700  NEXT N
710  INVERSE 0
720  PLOT CX+X,CY+X/2+Y
730  DRAW AX,AY
740  DRAW -BX,BY
750  DRAW -AX,-AY
760  DRAW BX,-BY
770  RETURN
```

Obr. 9.10.

Mapa povrchu

Pokud pospojujeme vrcholky čar znázorňujících výšku z (přičemž tyto čáry nebudeme kreslit), dostaneme realistický tvar, znázorňující rovinnou mapu prostoru ohraničeného osami x a y. Spojíme všechny body se souřadnicí y=0. Potom spojíme všechny body s hodnotou y o něco větší a tak stále dále, až se dostaneme na maximální hodnotu y. Totéž provedeme pro body se stejnými souřadnicemi x a obdržíme požadovanou "mapu" různě zprohýbané roviny. Můžete různé vysoké oblasti odlišit barvou jako na

skutečné mapě.
Výsledný dojem bude tím lepší, čím více bodů budete spojovat. Ale zase to nesmíte přehnat, protože pak by byl obrázek příliš přeplněn a nebylo by z něj možno nic vyčíst. Musíte prostě trošku experimentovat, než dosáhnete toho správného dojmu.

Třírozměrné kruhy

Dosti zajímavou tématikou třírozměrného zobrazování je experimentování s kruhy a s jejich částmi, z něhož nám mohou vyjít zajímavé obrazce.
Tento program ukazuje jeden takovýto experiment.

```
100  REM 3D KRUHOVY GRAF
110  REM NASTAVENI PARAMETRU
120  LET K=PI/2000
130  LET M=1/SQR (2)
140  DEF FN A(Z)=10*COS (K*(X*X+Y*Y))
150  REM NAKRESLENI OBRAZKU
160  FOR X=-100 TO 100
170  LET Y1=5*INT (SQR (10000-X*X)/5)
180  FOR Y=Y1 TO -Y1 STEP -5
190  LET Z=FN A(SQR (X*X+Y*Y))-M*Y
200  IF Y=Y1 THEN GOTO 220
210  IF Z<Z1 THEN GOTO 240
220  PLOT 128+X,80+INT (Z/2)
230  LET Z1=Z
240  NEXT Y
250  NEXT X
```

Obr. 9.12.

Změnou funkcí můžete dosáhnout jiné zajímavé obrazce, ale tato metoda je velmi zdlouhavá, protože se musí vypočítat souřadnice každého bodu.

Perspektivní kreslení

Všechny trojrozměrné obrázky, které jsme doposud kreslili, měly jednu nevýhodu, nevypadaly přesvědčivě. Tento způsob kreslení se nazývá izometrický a jednoduše řečeno spočívá v tom, že vertikální výška (z) je po celé ploše ve stejném měřítku, tzn. že i dvě úsečky, z nichž jedna je vpředu a jedna vzadu a které jsou kolmé k rovině xy jsou na našem obrázku stejně vysoké.
Abychom dostali iluzI skutečně prostorového pohledu, musíme použít tzv. perspektivu. Tuto metodu malíři mnoho století propracovávali a my teď můžeme říci, že čím je objekt vzdálenější od pozorovatele, tím se mu jeví menší. Pokud toto využijeme při kreslení obrázků, dostaneme nyní mnohem realističtější pohled než dříve.
Aby jste si perspektivu lépe objasnili, tak si představte, že stojíte na cestě jdoucí od vás přímo bez zákrutu až k horizontu. Ačkoliv krajnice jsou po celé délce cesty rovnoběžné a tudíž stále stejně vzdálené, vám se bude zdát jako by se stále

přibližovaly až na úrovni horizontu splynou. To je způsobeno optickým klamem, který musíme přenést do svých obrázků.

Nejprve si musíme zvolit nějaký souřadnicový systém, v němž můžeme vypočítat pozici bodů ve skutečnosti a převést ji na souřadnice bodu na obrazovce. Zvolíme si tedy osu x jdoucí zleva doprava jako obvykle a osu z také v obvyklém vertikálním směru, tedy jdoucí zdola nahoru. No a zbylou osu y je nejlepší umístit ve směru pohledu.

Pokud by jste se na silnici dívali s očima umístěnýma na jejím povrchu, tak by jste viděli pouze horizontálně jdoucí úsečku ležící na ose x, protože všechny ostatní body silnice by splynuly do této úsečky. Pro své pozorování musíme tedy mít oči umístěné nad silnicí (nebo nad čímkoliv jiným).

Tento program kreslí perspektivní pohled na silnici i s dělícím pruhem a se stromy okolo ní.

```
100  REM SILNICE SE STROMY
105  REM KRESLENI OBLOHY
110  PAPER 5: CLS
115  REM NAKRESLENI POVRCHU
130  PAPER 6
140  FOR Y=10 TO 21
150  FOR X=0 TO 31
160  PRINT AT Y,X;" "
170  NEXT X: NEXT Y
175  REM NAKRESLENI SILNICE
180  FOR X=-128 TO 127
190  PLOT INK 0;128,96
200  DRAW INK 0;X,-96
210  NEXT X
220  LET SX=1024: LET SY=960
225  REM NAKRESLENI DELICICH CAREK
230  FOR Y=100 TO 2000 STEP 100
240  LET Y1=INT (SY*10/(Y+50))
250  LET Y2=INT (SY*10/Y)
260  LET X=INT (SX*1/(Y+50))
270  FOR K=128-X TO 128+X
280  PLOT PAPER 7; INVERSE 1;K,96-Y1
290  DRAW PAPER 7; INVERSE 1;0,Y1-Y2
300  NEXT K
310  LET X1=INT (SX*1/Y)
320  FOR K=0 TO X1-X
330  PLOT PAPER 7; INVERSE 1;128-X,96-Y1
340  DRAW PAPER 7; INVERSE 1;-K,Y1-Y2
350  PLOT PAPER 7; INVERSE 1;128+X,96-Y1
360  DRAW PAPER 7; INVERSE 1;K,Y1-Y2
370  NEXT K: NEXT Y: INVERSE 0
375  REM KRESLENI HORIZONTU
380  PLOT INK 0;0,96
390  DRAW INK 0;255,0
395  REM  KRESLENI STROMU
400  LET TR=5: LET TT=20: LET TW=3
410  FOR Y=150 TO 1200 STEP 150
420  LET Y1=INT (SY*10/Y)
```

```
430  LET Y2=INT (SY*(10-TR)/Y)
440  LET Y3=INT (SY*(10-TT)/Y)
450  LET X=INT (SX*12.5/Y)
460  LET X1=INT (SX*TW/Y)
470  LET X2=128-X: GOSUB 1000
480  LET X2=128+X: GOSUB 1000
490  NEXT Y
500  STOP
1000 PLOT INK 0;X2,96-Y1
1010 DRAW INK 0;0,Y2
1020 FOR K=-X1 TO X1
1030 PLOT INK 0;X2,96-Y3
1040 DRAW INK 0;K,Y3-Y2
1070 NEXT K: OVER 0
1080 RETURN
```

Obr. 9.15.

Objasnění jednotlivých výpočtů v programu je dost složité a ani mně se ho nepodařilo dostatečně pochopit - omluvám se, zde musíte prostudovat odbornější literaturu.

Kapitola 10: Vytvoření vlastního znakového souboru

V 5. kapitole jste se naučili používat a definovat znaky uživatelské grafiky. Jenomže při praktickém programování brzy zjistíte, že jejich počet je značně malý a že byste potřebovali znaků mnohem více.

Na Spectru máte možnost přeprogramování celého souboru znaků - počínaje mezerou (ASCII kód 32) a konče copyrightem (c v kroužku, ASCII kód 127), tedy dohromady 96 znaků. Ba co více - můžete tento znakový soubor měnit v průběhu programu a to každou chvíli (omezí vás pouze rozsah volné paměti).

Jak je vám již známo, každý znak se skládá z 8 bajtů po 8 bitech, dohromady tedy 64 bitů (stejně jako u uživatelské grafiky). Stačí tedy přepsat odpovídajících 8 bajtů v paměti počítače a dostaneme novou podobu znaku. V každém bajtu nuly reprezentují nerozsvícené body a jednIčky rozsvícené body. Mozaika pro všech 96 znaků se normálně nachází v paměti ROM od adresy 15616.

Nyní si jistě říkáte, že pokud je originální mozaika uložená v paměti ROM, tak se nedá přepsat. Jenomže my můžeme počítači přikázat, aby podobu znaků načítal od jiné adresy. V systémové proměnné CHAR$ je uložena adresa začátku úseku paměti, z níž čte počítač tvary znaků, ovšem tato adresa je zde uložena zmenšená o 256. Systémová proměnná CHAR$ se nachází na adresách 23606 a 23607. Aby sis mohl vytvořit vlastní "generátor znaků", musíš pro nové tvary vytvořit místo v paměti (snížením RAMTOPu), vepsat vzorce na svá místa a nakonec změnit obsah CHAR$.

```
100  REM GENERATOR ZNAKOVEHO SOUBORU
110  LET RAMTOP=PEEK 23730+256*PEEK 23731
120  CLEAR RAMTOP-768
130  LET NOVY=PEEK 23730+256*PEEK 23731+1
140  FOR I=0 TO 767: POKE NOVY+I,PEEK (15616+I): NEXT I
150  PRINT NOVY: LET NOVY=NOVY-256
160  POKE 23606,NOVY-256*INT (NOVY/256): POKE 23607,INT  (NOVY/
     256)
```

Na řádku 110 počítač zjistí stávající adresu RAMTOPu (ze systémové proměnné RAMTOP - adresy 23730 a 23731) a na řádku 120 ji sníží o 768 bajtů. Potom znovu vypočítá adresu nového RAMTOPu a připočte k ní 1 a dostane první adresu nového znakového souboru (proměnná NOVY). Potom se na řádku 140 přepíší do paměti RAM od adresy NOVY originální tvary znaků. Na řádku 150 se zobrazí proměnná NOVY a potom se zmenší o 256 a zapíše se do systémové proměnné CHAR$.

Po spuštění programu se na chvíli nic neděje (to se přepisují tvary znaků), potom se na obrazovce objeví číslo (hodnota proměnné NOVY - někam si jej poznačte) a vzápětí se program zastaví. Nyní vše vypadá normálně, ale vy už na obrazovce vidíte tvary znaků načítané z paměti RAM. Pro jejich modifikaci můžete užít třeba tento program.

```
100  REM GENERATOR VLASTNICH ZNAKU
110  INPUT "ZNAK ";A$
120  LET AC=CODE A$
130  LET AD=PEEK 23606+256*PEEK 23607+256+8*(AC-32)
140  FOR I=AD TO AD+7
150  INPUT "RADEK ";(I-AD+1);" ? ";R
160  POKE I,R
170  PRINT AT 10,16;A$
180  NEXT I
```

Při použití NEW není nic ztraceno - naše tvary jsou chráněny RAMTOPem. Stačí jen obnovit adresu v systémové proměnné CHAR$ (podle řádku 160 prvního programu, ale nezapomeňte hodnotu NOVY (kterou máte zapsanou na papírku) snížit o 256.

Znakový soubor se ale smaže po resetu nebo vypnutí počítače. Můžete si ho ale nahrát na kazetu pomocí příkazu

```
SAVE "nazev" CODE X,768
```

kde X je číslo poznamenané na lístečku.

Pro využití souboru z magnetofonu ve vašem programu můžete využít třeba takovýto řádek.

```
1  CLEAR X-1: LOAD "" CODE: POKE 23606,X-256*INT (X/256): POKE
   23607,INT ((X-256)/256)
```

s tímtéž významem pro x.

Můžete tedy používat v jednom programu více odlišných druhů písma a nebo si vytvořit celou škálu postaviček do her.

Kapitola 11: Uložení obrazu v paměti

Pro práci s grafikou ve strojovém kódu ale i v Basicu byste měli alespoň zhruba vědět, jak je obrázek uložen v paměti.
Obrazovková paměť je složená ze dvou částí - paměti kresby a paměti atributů. Paměť kresby to jsou vlastně jednotlivé body, buď zhaslé nebo rozsvícené a uložené vždy po osmi bodech - bitech v jednom bajtu. Tzn. máme 196 bodových řádků (musíme připočítat i dva dialogové řádky, které jsou sice v Basicu těžko přístupné, ale počítač si je musí zapamatovat) a 256 bodových sloupců. Po vynásobení 196*256=49152 bodů, tedy děleno osmi 6144 bajtů. Stejně tak paměť atributů (viz kap.6) má rozměr 24 řádků krát 32 sloupců, což je 768 bajtů.
Dohromady tedy 6144+768=6912 bajtů, uložených od adresy 16384. Obrazovku tedy můžeme nahrát buď příkazem SAVE "jméno" SCREEN$ nebo ekvivalentním SAVE "jméno" CODE 16384,6912 (přičemž změnou těchto dvou číslic na jiné hodnoty můžeme nahrát jen část obrazovky).
Nejprve si povězme něco podrobněji o paměti kresby.

Paměť kresby

Každý bajt paměti kresby obsahuje 8 bitů - bodů. Obrazovka je tedy vlastně rozdělena na jakési 8 bodů široké sloupce. Tyto sloupce se dále rozdělí zase po 8 bajtech a dostaneme naše známé rozdělení obrazovky na 32*24 čtverců (znakových pozic - zobrazují se do nich znaky příkazy PRINT). Tyto čtverečky mají souřadnicový systém s počátkem v levém horním rohu obrazovky. Souřadnicový systém jemné grafiky jak už víte má počátek souřadnic v levém dolním rohu. Pro tento náš popis budeme ale brát jako počátek onen pravý horní roh. Takže budeme mít 256 bodových sloupců (bráno zleva doprava) a 196 bodových řádků (bráno shora dolů). Pozici každého bodu na obrazovce teď můžeme určit pomocí jeho souřadnic vůči počátku nebo jako souřadnice v znakovém čtverci, přičemž musíme samozřejmě určit polohu onoho čtverce.

První bajt paměti kresby je na adrese 16384. Můžete tedy pomocí instrukce POKE změnit jeho obsah - na obrazovce se vlevo nahoře objeví binární tvar zapsaného čísla (při POKE 16384,x pro x=1 bude rozsvícen bod v 7. sloupci nultého bodového řádku (jedna binárně je 00000001), pro x=128 (10000000 BIN) se rozsvítí bod na pozici 0,0 apod).
Adresa 16384 je vlastně nultý bodový řádek nultého znakového čtverce. Stejně tak adresa 16385 je nultý bodový řádek prvního znakového čtverce (tedy čtverce AT 0,1) a tak dále až po adresu 16415, což je nultý bodový řádek 31. znakového čtverce. Máme tedy hotový celý nejvrchnější bodový nultý řádek.
Adresa 16416 nás ale nezavede na 1. bodový řádek nultého znakového čtverce, jak by jste možná čekali, nýbrž přeskočí na 8 bodový řádek, tedy na nultý bodový řádek znakového čtverce AT 1,0. Dalších 31 bajtů je znovu celý tento bodový řádek a

potom se zase přeskočí 7 řádků a stále dál...
Ale ne až do konce obrazu. Obrazovka je totiž navíc rozdělena na třetiny vždy po osmi znakových řádcích. Proto po 32*8 bajech se dostaneme na adresu 16639 (16384+32*8-1). To je poslední nultý bajt posledního znakového čtverce (AT 7,31) první třetiny. Další adresa (16640) nás zavede na 1. bodový řádek znakového čtverce AT 0,0 a tak dále osmkrát vždy po 32 bajtech zaplníme osm bodových řádků, přičemž po každém jich sedm přeskočíme. No a tak stále dále, až zaplníme celou první třetinu. V každém znakovém čtverci je 8 bodových řádků, proto poslední bajt 1. třetiny má adresu

16384 + 8 * 8 * 32 = 16384 + 2048 = 18431 (musíme odečíst 1)

Druhá třetina je organizována úplně stejně, tedy na adrese 18432 je nultý bdový řádek znakového čtverce AT 8,0 atd. Stejně tak i třetí třetina.
Možná vám to zatím připadá složité, ale vždyť vy to důvěrně znáte - vzpomeňte si, jak se vykresluje obrázek při nahrávání a okamžitě pochopíte způsob uložení paměti kresby v paměti.
Můžeme tedy zapsat i do oblasti editační zóny, je ale třeba si pamatovat že příkaz INPUT nebo nějaké hlášení počítače obsah editační zóny změní.
Poslední bajt paměti kresby má tedy adresu:

16384 + 3 * 8 * 32 * 8 = 16384 + 6144 = 22527 (zase odečteme 1)

kde 3 je počet tretin, 8 počet znakových řádků v jedné třetině, 32 počet znakových čtverců v jednom řádku a 8 počet bodových řádků v jednom znakovém čtverci. Jedničku odečítáme proto, že musíme počítat s adresou 16384 (správně bychom tedy měli přičítat k adrese 16383).
Pokud se zamyslíte nad takovýmto uspřádáním paměti, musí vám připadat strašně komplikované, protože uvažujete v dekadické soustavě. Ale počítač má raději binární a tak si číslo 16384 převedeme do binární formy.

16484 = 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Je to číslo pěkné, takové kulaté a jednoduché, že. Pokud si nyní zapíšeme binárně adrsy začátků všech tří třetin, dostaneme

0. třetina
0. bodový řádek 16384 = 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0. znakový řádek ---

1. třetina
0. bodový řádek 18432 = 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0. znakový řádek ----

2. třetina
0. bodový řádek 20480 = 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0. znakový řádek ---

Všimněte si, že se změnou čísla třetiny se mění obsah pouze dvou označených bitů, které tak vlastně označují číslo třetiny. Pokud bychom stejným způsobem rozebírali třeba čísla bodových řádků v jednotlivých znakových řádcích nebo čísla sloupců, dostali bychom podobný výsledek - každý údaj je ovládán pouze skupinkou bitů a jeho změna ovlivní pouze tyto bity a žádné jiné. Můžeme potom napsat takovouto tabulku :

```
: 15 14 13 : 12 11 : 10 9 8 : 7 6 5 : 4 3 2 1 0 :
:          :       :       :       :           :
: 0  1  0  : 0  0  :  0 0 0 : 0 0 0 : 0 0 0 0 0 :
:          :       :       :       :           :
:    A     :   B   :    C  :   D   :     E     :
```

kde

A nemění se, určuje adresu obrazovky

B určení třetiny (bráno shora)

C určení bodového řádku (počítáno shora v jednom znakovém řádku)

D určení znakového řádku (počítáno shora v jedné třetině)

E určení sloupce, tedy znakového čtverce na řádku; bráno zleva

Čísla 15 až 0 označují jednotlivé bity ve dvou bajtech adresy.

Díky takovémuto systému je velmi snadné určit pozici kteréhokoliv bodu na obrazovce a jeho umístění v paměti.

Paměť atributů

Paměť atributů je paměť, v níž jsou uloženy barvy - nalistujte si kapitolu 6 a tam se dozvíte, jak jsou jednotlivé atributy tvořeny. Pro připomenutí - paměť atributů začíná na adrese 22528 a má délku 768 bajtů. Každý bajt obsahuje informaci o barvě jednoho znakového čtverce, přičemž tyto informace jsou řazeny za sebou postupně - jeden řádek za druhým.
Pokud bychom rozebrali jednotlivé bity adres paměti atributů, dostali bychom takovéto schéma:

```
: 15 14 13 12 11 10 : 9 8 :: 7 6 5 : 4 3 2 1 0 :
:                   :     ::       :           :
: 0  1  0  1  1  0  : 0 0 :: 0 0 0 : 0 0 0 0 0 :
:                   :     ::       :           :
:       A           :  B  ::   C   :     D     :
                    :     ::       :
                    :    BC        :
```

kde

A se nemění, označuje umístění paměti atributů v paměti

B je určení třetiny, počítáno shora

BC je číslo znakového řádku, bráno shora (0-23)

C je číslo řádku ve třetině, bráno shora

D je určení znakoveho čtverce (sloupce), bráno zleva

Pokud víte i toto, máte již zhruba jasno, jak je uložen obraz v paměti Spectra a jak máte hledat adresy pro jednotlivé body a jejich barvy.

PŘÍLOHA A - další programy uvedené v originále.
=========

```
100 REM All round perspective
110 REM view of a block
120 DIM v(50,3): DIM e(100): DIM l(100)
130 LET sx=10: LET sy=10
140 READ nv
150 FOR p=1 TO nv
160 READ v(p,1),v(p,2),v(p,3)
170 NEXTp
180 READ ne
190 FOR j=1 TO ne
200 READ e(j),l(j)
210 NEXT j
230 LET d=80: LET p=22: REM Set viewer position
240 Let r=0: LET h=45: REM Main program loop
250 CLS: PRINT "Heading= ";h
260 PRINT "Pitch= ";p: PRINT "Roll= ";r
280 LET h=h*PI/180: LET p=p*PI/180
300 LET r=r*PI/180
310 GO SUB 1000: REM Calculate multipliers
320 LET xv=-d*cp*sh
330 LET yv=-d*cp*ch: LET zv=-d*sp
360 LET x1=0: LET y1=0: REM Project image on screen
370 FOR j=1 TO ne
380 LET n=e(j): LET x=v(n,1): LET y=v(n,2)
400 LET z=v(n,3)
410 GO SUB 1200
415 REM Check if line to be drawn
420 IF l(j)=0 THEN GO TO 450
425 REM Draw line
430 PLOT x1,y1
440 DRAW xs-x1,ys-y1
445 REM Update cursor position
450 LET x1=xs: LET y1=ys
460 NEXT j
480 INPUT "Another view /y/n/";a$
490 IF a$<>"y" THEN STOP
500 INPUT "Heading /deg/=";h
510 INPUT "Pitch /deg/=";p
```

```
 520 INPUT "Roll /deg/=";r
 530 GO TO 25Ø
 540 STOP
1000 REM Multiplier Factors
1010 LET ch=COS h: LET sh=SIN h: LET cp=COS p: LET sp=SIN p
1030 LET cr=COS r: LET sr=SIN r: LET m1=ch*cr-sh*sp*sr
1050 LET m2=-sh*cr-ch*sp*sr: LET m3=cp*sr
1070 LET m4=sh*cp: LET m5=ch*cp:LET m6=sp
1100 LET m7=ch*sr-sh*sp*cr: LET m8=-sh*sr-ch*sp*cr
1120 LET m9=cp*cr
1130 RETURN
1200 REM Move viewer position
1210 LET x=x-xv: LET y=y-yv: LET z=z-zv
1220 REM Rotate view
1230 LET x3=m1*x+m2*y+m3*z: LET y3=m4*x+m5*y+m6*z
1250 LET z3=m7*x+m8*y+m9*z
1260 REM Calculate x,y position o screen
1270 LET xs=128+INT (sx*d*x3/y3)
1280 LET ys=8Ø+INT (sy*d*z3/y3)
1290 RETURN
2000 REM Number of points
2010 DATA 12: REM x,y,z coordinates
2030 DATA 5,-3,4
2040 DATA -5,-3,4
2050 DATA -5,-3,-4
2060 DATA 5,-3,-4
2070 DATA 5,3,4
2080 DATA -5,3,4
2090 DATA -5,3,-4
2100 DATA 5,3,-4
2110 DATA 4,-3,3
2120 DATA -4,-3,-3
2130 DATA 4,-3,-3
2140 DATA -4,-3,3
2200 REM Number of edges
2210 DATA 2Ø: REM Edge and line data
2230 DATA 1,Ø,2,1,3,1,4,1
2240 DATA 1,1,5,1,6,1,7,1
2250 DATA 8,1,5,1,2,Ø,6,1                    Obr, 9.17.
2260 DATA 3,Ø,7,1,4,Ø,8,1
2270 DATA 9,Ø,1Ø,1,11,Ø,12,1
```

---===xxx===---

Zde naše vydání knihy Grafika Spectra končí. Zvukovou část jsme vynechali. Zájemci o zvukové možnosti Spectra nechť si raději přečtou jiné knihy, např. "40 NEJ... PUTIN", kterou si můžete od nás rovněž koupit /zatím/ - nabízíme ji jako sborník č. 18.

Natištěno pro Klub VT Karolinka.