ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1991
Seria: INFORMATYKA z. 15 Nr kol. 1044

Małgorzata BACH
Zbigniew SZKARADNIK

GRAFICZNY FORTH DLA MIKROKOMPUTERA COMPAN-8

Streszczenie. Artykuł prezentuje implementację systemu graficznego dla mikrokomputera ComPAN-8. System ten stanowi integralną część translatora języka Forth. Szczególną uwagę poświęcono problemom dołączania driverów oraz realizacji algorytmów wypełniania.

Produkowany w Polsce mikrokomputer ComPAN-8 posiada sporo zalet. W pierwszym rzędzie wymienić tu można dużą, jak na komputer tej klasy, liczącą 64 kB pamięć obrazu oraz ekran o rozdzielczości 640x288. Do tego dochodzi szereg trybów graficznych, semigraficznych i znakowych, z których użytkownik może korzystać oraz sprzętowe wspomaganie takich operacji, jak poziomy czy pionowy scrolling. Niestety istniejące oprogramowanie stosunkowo słabo te możliwości wykorzystuje. Opracowana została wprawdzie biblioteka procedur graficznych [3] , które mogą być wywoływane z programów napisanych przy wykorzystaniu kompilatora języka Fortran F80 (a po napisaniu odpowiednich programów łączących prawdopodobnie także z programów napisanych w innych językach, których kompilatory generują pliki wynikowe w formacie REL firmy Microsoft jak PL/M, Pascal/MT+ czy Supersof C), ale na tym w zasadzie się kończy.

W szczególności brak interpretera któregoś z języków programowania wyposażonego w instrukcje graficzne. Istnienie tego typu narzędzia według rozeznania autorów było bardzo celowe. Postanowiliśmy więc w ramach prac dyplomowych prowadzonych w Instytucie Informatyki Politechniki Śląskiej rozszerzyć interpreter jednego z języków programowania o grafikę [1] . Zdecydowaliśmy się na wykorzystanie w tym celu interpretera języka Forth F83 i rozwinięcie go w kierunku aplikacji graficznych. Pod uwagę brane były także interpretery innych języków programowania (MBasic, Xlisp, EProlog). Basic został odrzucony z powodu braku bezpośredniej możliwości tworzenia w nim podprogramów rekurencyjnych. W przypadku Prologu nie dysponowaliśmy wersją źródłową interpretera. Z kolei Xlisp oznacza się bardzo niewielką szybkością realizacji napisanych w nim programów, a poza tym sam zajmuje sporo miejsca w pamięci operacyjnej. Prostota, z jaką w przypadku Forthu korzystać można z wszystkich zasobów komputera, przesądziła ostatecznie o jego wyborze, aczkolwiek zamierzamy w Instytucie Informatyki wy-

konać również graficzną wersję języka Xlisp. Dodatkowo za wykorzystaniem języka Forth przemawiało doświadczenie jednego z autorów zdobyte przy realizacji podobnego zadania z wykorzystaniem komputera Mera-60 [2] .

Początkowo wydawało się, że adaptacja wspomnianego pakietu na mikrokomputerze ComPAN-8 nie nastręczy wielu trudności i sprowadzi się jedynie do napisania nowych driverów urządzeń zewnętrznych. Rzeczywistość okazała się jednak inna. Wzorcowy pakiet był zrealizowany przy użyciu translatora FIG-Forth. Translator ów był jednak mocno w stosunku do wersji wzorcowej rozszerzony [2] . Wyposażony był między innymi w system plików ze słowami (pojęcie słowa odpowiada w przybliżeniu pojęciu procedury) open filename, i create filename.
ComPAN-8 wyposażony jest wprawdzie w interpreter FIG-Forthu, jednak pozbawiony wszelkich rozszerzeń. Dlatego zdecydowaliśmy się na F83, który jest implementacją nowszego standardu Forth-83 i wyposażony jest między innymi w system plików. Wymagało to zainstalowania tego programu na komputerze ComPAN, co sprowadziło się jedynie do modyfikacji wbudowanego edytora. Następnie przeniesiony został pakiet graficzny i opracowane zostały nowe drivery. Otrzymany w ten sposób pakiet operacji graficznych działał co prawda poprawnie, ale niesłychanie wolno. Spowodowane to było głównie faktem wolnej realizacji operacji mnożenia i dzielenia (procesor Mery-60 posiada rozkazy mnożenia i dzielenia). Przyspieszenie pracy wymagało modyfikacji kodu pakietu polegającej głównie na realizacji mnożenia i dzielenia przez stałą za pomocą przesunięć. Z podstawowego zestawu usunięte zostały niektóre prymitywy (łuk, okrąg, elipsa) w zamian za to dodano inne (jak wypełnianie). W sumie modyfikacje objęły (z wyjątkiem driverów, kt' zostały napisane na nowo) około 30% kodu, w tym około 5% to zmiany zwi zane z przejściem na inny standard języka.

Przyjęte założenia

Pakiet graficzny widziany przez użytkownika powinien realizować co najmniej podstawowe funkcje graficzne. Przyjęliśmy, że będą to:

- zainstalowanie i usunięcie drivera określonego urządzenia zewnętrznego;
- usunięcie rysunku;
- narysowanie punktu o współrzędnych X Y;
- testowanie punktu o współrzędnych X Y;
- przesunięcie do punktu o współrzędnych X Y;
- narysowanie odcinka lub łamanej o określonych parametrach;
- wypełnienie obszaru zadeklarowanego przez kontur lub listę współrzędnych wierzchołków wielokąta.

Podaną listę można oczywiście uzupełniać. Można dodawać kolory, zmieniać wzorce linii i obszarów (wypełnień). Można dodawać nowe prymitywy, jak: łuki, okręgi, prostokąty czy teksty. Można też nie tylko rysować, ale

i usuwać czy też modyfikować już narysowane punkty. Nie było to jednak naszym celem. Zauważmy bowiem, że choć dodawanie tych funkcji jest sprawą prostą (uważamy, że każdy użytkownik Forthu będzie w stanie to zrobić), to od razu rodzi pewne problemy. Nie wszystkie urządzenia mają chociażby zdolność tworzenia rysunków w kolorze. Niemożliwe jest również przykładowo usunięcie raz narysowanej linii za pomocą plottera. Podobnych "problemów" przy dokładniejszej analizie można zresztą wykryć więcej. Dodatkowe trudności sprawiają różne rozdzielczości i formaty rysunków oraz (co jest szczególnie dokuczliwe) różne proporcje pixeli. Tymczasem założeniem naszym było otrzymanie możliwie identycznych rysunków przy użyciu różnego typu urządzeń wyjściowych. Taka filozofia jest zresztą zgodna z ogólną koncepcją Fortha, zakładającą dostarczenie programiście jedynie podstawowych prostych narzędzi.

Kolejnym założeniem było umożliwienie współpracy pakietu z wieloma różnymi urządzeniami wyjściowymi. Opracowany został w tym celu jeden driver monitora, dwa drivery drukarki (Epson FX) oraz driver pisaka (Sharp CE). Przyjęty został również bardzo prosty interfejs driver-pakiet graficzny, co zdaniem autorów powinno ułatwić użytkownikowi tworzenie własnych driverów. Założyliśmy, że każdy driver powinien realizować następujące funkcje:

- narysowanie punktu o współrzędnych X Y;
- testowanie punktu o współrzędnych X Y;
- przesunięcie do punktu o współrzędnych X Y;
- usunięcie rysunku.

Pozostałe funkcje graficzne, takie jak: kreślenie linii czy wypełnianie obszaru powinny być już realizowane przez odpowiednie procedury pakietu graficznego.

Należało także określić reakcje pakietu graficznego w sytuacji przekroczenia efektywnego zakresu rysunku. Dopuszczona została praca w trybie obcinania, zawijania lub przerwanie rysowania (podobnie jak w języku programowania Logo). Właściwie skonstruowane procedury obcinania i zawijania powinny poprawnie pracować również w przypadku zmniejszonego formatu obrazu (okna).

Charakterystyka pakietu

Pakiet graficzny wykorzystuje kartezjański oraz biegunowy układ współrzędnych. Wyposażony jest w słowa rysujące punkty, kreślące odcinki oraz zamalowujące powierzchnię. Tak zwaną grafikę żółwia realizują słowa kreślące odcinki określonej długości w przód i w tył oraz słowa wykonujące obrót. W przypadku tego typu grafiki dostępne są podstawowe transformacje rysunku, takie jak: przesunięcie, obrót, skalowanie oraz lustrzane odbicie.

Kreślenie rysunku może być przerwane przez naciśnięcie dowolnego klawisza na klawiaturze. Dostępne są trzy wymienione poprzednio alternatywne tryby obsługi błędów. Najważniejsze słowa pakietu graficznego zebrane zostały w tabeli 1. Formaty rysunków oraz maksymalne rozdzielczości zdeterminowane są przez parametry urządzeń wyjściowych. Są one następujące:

- dla monitora 640 x 240 (okno graficzne) + 4 wiersze (okno znakowe) lub tylko okno graficzne 640 x 288,
- dla drukarki 480 x 360 oraz 960 x 720,
- dla pisaka 960 x 720.

Tabela 1

CLIPON (---)

Ustawienie trybu obsługi błędów z obcinaniem fragmentów rysunku znajdujących się poza obszarem zadeklarowanego obrazu.

FENCEON (----)

Ustawienie trybu obsługi błędów z przerywaniem rysowania po osiągnięciu krawędzi zadeklarowanego obrazu.

WRAPON (---)

Ustawienie trybu obsługi błędów z zawijaniem fragmentów rysunku znajdujących się poza obszarem zadeklarowanego obrazu.

CLEAR (---)

Usunięcie rysunku. W przypadku drukarki polega na wyzerowaniu pliku z mapą bitów.

POINT (x y --- f)

Testowanie punktu o współrzędnych x y. W przypadku, jeżeli jest on ustawiony flaga logiczna f ma wartość true.

PLOT (x y ---)

Przesunięcie (kursora) do punktu o współrzędnych x y.

DOT (x y ---)

Wykreślenie punktu o współrzędnych x y.

DRAW (x y ---)

Wykreślenie odcinka od aktualnego położenia do punktu o współrzędnych x y.

LINE (addr n ---)

Wykreślenie złożonej z n-odcinków łamanej o wierzchołkach w punktach o współrzędnych umieszczonych w tablicy pod adresem addr.

FILL (x y ---)

Zamalowanie figury, wewnątrz której znajduje się punkt o współrzędnych x y.

AREA (addr n ---)

Zamalowanie wielokąta o n-1 wierzchołkach, których współrzędne mieszczą się w tablicy pod adresem addr. Współrzędne pierwszego i n-tego wierzchołka muszą być identyczne.

WIDE (d ---)
Ustawienie współczynnika szerokości rysunku na wartość d.

HIGH (d ---)
Ustawienie współczynnika wysokości rysunku na wartość d.

LARGE (d ---)
Ustawienie współczynnika wielkości rysunku na wartość d.

BREAKOFF (---)
Wyłączenie możliwości przerwania rysowania poprzez naciśnięcie dowolnego klawisza.

BREAKON (---)
Włączenie możliwości przerwania rysowania poprzez naciśnięcie dowolnego klawisza.

MIRROROFF (---)
Wyłączenie lustrzanego odbicia.

MIRRORON (---)
Włączenie lustrzanego odbicia. Lustrzane odbicie uzyskuje się dla figur zdefiniowanych w grafice żółwia przez zamianę obrotu w lewo na obrót w prawo i odwrotnie.

FORWARD (l ---)
Wykreślenie odcinka o długości l w przód.

BACKWARD (l ---)
Wykreślenie odcinka o długości l w tył.

LEFT (α ---)
Obrót o kąt α w lewo.

RIGHT (α ---)
Obrót o kąt α w prawo.

TURN (β ---)
Obrót o kąt β .

MOVE (l ---)
Przesunięcie o odcinek długości l w przód.

POSITION (x y ---)
Ustawienie początku rysowania od współrzędnych x y.

CURSOROFF (---)
Wyłączenie kursora graficznego (tylko monitor).

CURSORON (---)
Włączenie kursora graficznego (tylko monitor)

HOME (---)
Ustawienie (kursora) na środku obrazu.

FRAME (---)
Wykreślenie ramki ograniczającej zadeklarowany obszar rysunku.

MONITOR (---)
Przydzielenie monitora jako urządzenia wyjściowego.

PRINTER (---)
Przydzielenie drukarki jako urządzenia wyjściowego.

PLOTTER (---)
Przydzielenie plottera jako urządzenia wyjściowego.

PRINT (---)
Wyprowadzenie rysunku na drukarkę po skończeniu rysowania.

X (--- addr)
Zmienna zawierająca aktualną współrzędną X.

Y (--- addr)
Zmienna zawierająca aktualną współrzędną Y.

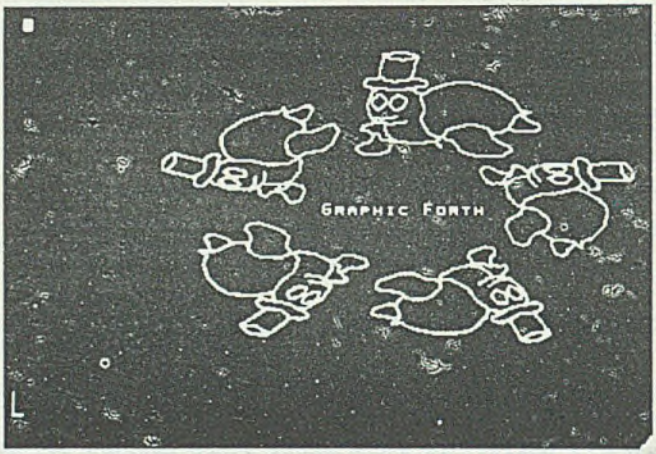
XMIN (--- addr)
Zmienna zawierająca minimalną możliwą wartość współrzędnej X.

YMIN (--- addr)
Zmienna zawierająca minimalną możliwą wartość współrzędnej Y.

XMAX (--- addr)
Zmienna zawierająca maksymalną możliwą wartość współrzędnej X.

YMAX (--- addr)
Zmienna zawierająca maksymalną możliwą wartość współrzędnej Y.

Skalowanie, a więc wydłużanie, rozszerzanie i powiększanie zrealizowane zostało za pomocą mnożenia składowych rysowanego wektora przez odpowiednie współczynniki skali. Współczynniki te mogą być zmieniane dynamicznie przy użyciu słów high, wide i large. Współczynniki skali są liczbami podwójnej długości z kropką dziesiętną. Pod uwagę brane są cztery pierwsze cyfry znaczące. Złożenie wydłużenia i obrotu ilustruje fotografia 1.



Fotografia 1

Istnieje możliwość deklarowania rysunków (a właściwie okien) o zmniejszonej rozdzielczości. Odbywa się to poprzez określenie minimalnych i maksymalnych współrzędnych dla rysunku. Przykładowo sekwencja słów:

```
monitor
100 xmin ! 200 xmax !
100 ymin ! 200 ymin !
clipon
```

spowoduje zadeklarowanie okna o rozdzielczości 100x100, którego dolny lewy róg leży w punkcie o współrzędnych 100 100. Okno zostanie zadeklarowane na monitorze, zaś fragmenty rysunku leżące na zewnątrz okna będą obcinane.

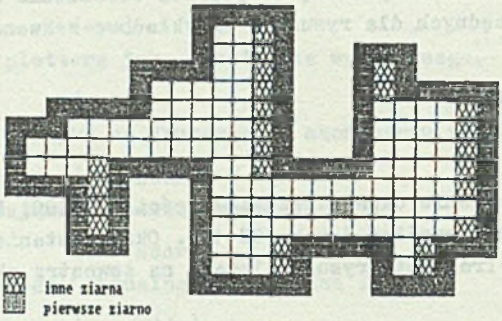
Zastosowane algorytmy

Algorytmy zastosowane do realizacji podstawowych funkcji w większości są albo bardzo proste, albo powszechnie znane jak na przykład zastosowany do kreślenia linii algorytm Bresenhama. Wyjątek stanowią algorytmy wypełniania powierzchni, które dzięki właściwościom języka Forth mogły być efektywnie i elegancko zrealizowane. Z tego względu zostaną one tu przedstawione.

Zadanie wypełniania można zdefiniować jako:

- zamalowanie obszaru, którego kontur (niekoniecznie regularny) jest dany;
- zamalowanie wnętrza wielokąta (niekoniecznie wypukłego), którego współrzędne wierzchołków są dane.

Omówimy najpierw pierwszy z algorytmów realizowany przez słowo fill (listing 1). Algorytm ten zakłada istnienie w pamięci obrazu zamkniętego konturu obszaru oraz znajomość współrzędnych punktu leżącego w jego wnętrzu. Współrzędne tego punktu, zwanego ziarnem, przesyła się na stos. W kolejnym kroku pobiera się współrzędne ziarna ze stosu, rysuje się w podanym miejscu punkt, po czym wykreśla się w lewo i w prawo odcinki łączące dany punkt z konturem. Podczas wykonywania tej czynności wyznacza się współrzędne skrajnych punktów konturu Xleft i Xright. Następnie wywołuje się słowo (scan), które testuje linię leżącą powyżej i linię leżącą poniżej wykreślonego odcinka ograniczonego przez Xleft i Xright. Testowanie to polega na wyznaczaniu nowych ziaren, których współrzędne zapisywane są na stosie. Opisane czynności powtarza się aż do momentu opróżnienia stosu. Ilustruje to rysunek 1.

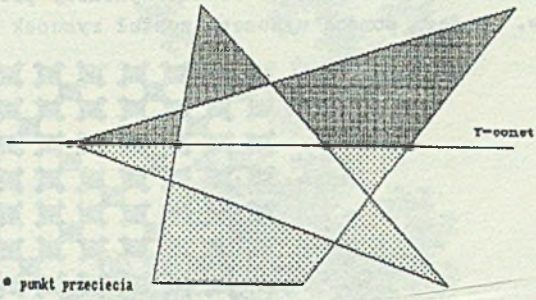


Rys. 1. Ilustracja działania słowa FILL
Fig. 1. Principles of working of the word FILL

Drugi z algorytmów wypełniania, realizowany przez słowo area (listing 2), wymaga określenia współrzędnych wierzchołków zamalowywanego wielokąta. Nie wymaga on istnienia mapy bitów obrazu, dlatego tego typu wypełnianie realizują poprzez wbudowane komendy niektóre plottery. Wypełniany kontur powinien być zamknięty (współrzędne pierwszego i ostatniego wierzchołka muszą być identyczne). Z kolei kontur jest wykreślany za pomocą słowa "line". W następnym kroku wyznacza się minimalną wartość współrzędnej Y, przeszukując w tym celu współrzędne wierzchołków. Wykonują to słowa miny i maxy. Teraz dla każdej linii Y, należącej do wyznaczonego przedziału, wyznacza się współrzędne X punktów przecięć ze wszystkimi odcinkami należącymi do konturu. Można udowodnić, że jeśli kontur jest zamknięty, to liczba jego przecięć z prostą Y = const jest parzysta. Żeby taki kontur zamalować, należy więc wyznaczone punkty odpowiednio połączyć. Ilustruje to rysunek 2.

Problem w tym przypadku polega na tym, że współrzędne X punktów przecięcia nie są wyznaczane po kolei. Wobec tego przed rozpoczęciem operacji łączenia punktów należy otrzymane współrzędne posortować. Ponieważ z kolei nie wiemy z góry, ile będzie punktów przecięcia, ich współrzędne X zapamiętywane są na stosie. Aby algorytm ten ulepszyć, przechowuję kolejno wyznaczane współrzędne X wykorzystując stos danych, przy czym za każdym razem w momencie dopisywania do stosu nowej liczby stos jest sortowany. Sortowanie stosu wykonuje słowo sort. Aby proces sortowania stosu usprawnić, wykorzystaliśmy stos powrotów. Samo sortowanie zrealizowane jest przez wstawianie. Polega na zdjęciu ze stosu danych wszystkich współrzędnych mniejszych od współrzędnej wstawianej. Te zdejmowane ze stosu współrzędne są przechowywane czasowo na stosie powrotów. Następnie wpisuje się na stos wstawianą współrzędną i przepisuje ze stosu powrotów na stos wszystkie

poprzednio tam umieszczone współrzędne. Po wyznaczeniu i posortowaniu wszystkich punktów przecięć słowo drawscan łączy odcinkami punkty, których współrzędne X przechowywane są na stosie, zamalowując w ten sposób wielokąt.



Rys. 2. Ilustracja działania słowa AREA
Fig. 2. Principles of working of the word AREA

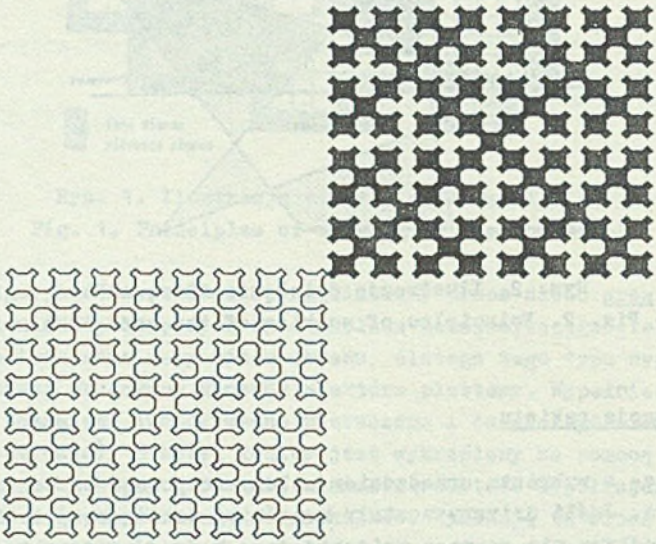
Eksploatacja pakietu

Współpracę z wybranym urządzeniem należy rozpocząć od zainstalowania jego drivera. Jeśli drivery zostały wcześniej załadowane, to instalacja urządzenia odbywa się poprzez wpisanie jego nazwy, np. monitor, plotter, printer (definicje poszczególnych słów zawiera listing 3). Mapa bitów rysunku zarówno w przypadku drukarki, jak i w przypadku pisaka (jeśli będzie wykorzystywany do zamalowywania powierzchni) przechowywana jest w pliku dyskowym. Szybką wymianę danych między tym plikiem a pamięcią operacyjną zapewnia istniejący w języku FORTH mechanizm pamięci wirtualnej. Wielkość tego pliku powinna wynosić 90 kB dla drukarki pracującej w trybie wysokiej rozdzielczości oraz dla pisaka lub 24 kB dla trybu niskiej rozdzielczości. Plik taki powinien być na początku wyzerowany słowem clear. Aby zwiększyć szybkość rysowania w celu przechowywania pliku z mapą bitów najlepiej używać RAM-dysku. W przypadku drukarki rysunek po zakończeniu rysowania powinien być wyprowadzony na papier za pomocą słowa print. Adaptacja drivera dla innej drukarki (bez zmiany maksymalnych rozdzielczości) wymaga modyfikacji tylko tego słowa.

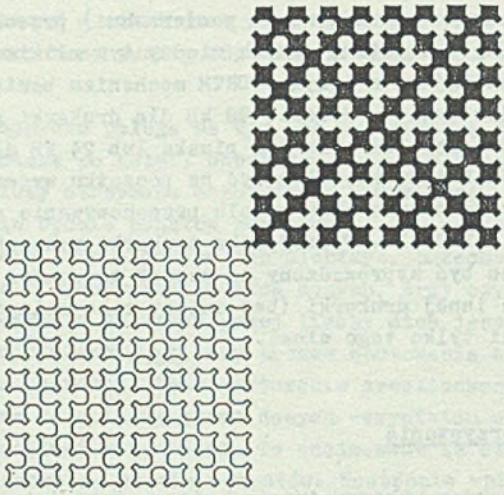
Przykłady wykorzystania

Przykładowy program wykorzystujący opisany pakiet graficzny przedstawiony jest na listingu 4. Jest to słowo kreślące rekurencyjne krzywe Sier-

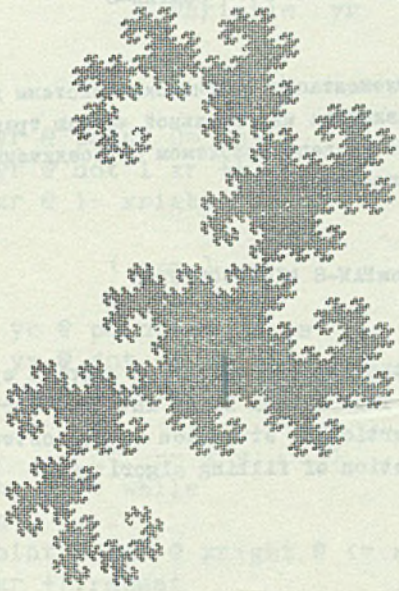
pińskiego. Za pomocą jednego słowa można wyprowadzać identyczne krzywe na dowolne urządzenia graficzne. Ilustrują to rysunki 3a i 3b. Przedstawione na nich krzywe poddane zostały dodatkowym transformacjom. Inną krzywą rekurencyjną, zwaną krzywą smoczą, przedstawia rysunek 4 (listing 5). Są to oczywiście jedynie przykłady, których można podać znacznie więcej. Dodatkowo jako przykład zastosowania pakietu został wykonany program do sporządzania rysunków. Za jego pomocą wykonany został rysunek z fotografii 1.



Rys. 3a. Krzywe Sierpińskiego (drukarka Epson FX)
Fig. 3a. Curves of Sierpiński (Epson printer)



Rys. 3b. Krzywe Sierpińskiego (ploter Sharp CE)
Fig. 3b. Curves of Sierpiński (Sharp plotter)



Rys. 4. Fraktal zwany DRAGONEM
Fig. 4. Fractal called DRAGON

LITERATURA:

[1] M. GRUSZKA. Opracowanie pakietu operacji graficznych dla mikrokomputera ComPAN- (praca dyplomowa), Instytut Informatyki Politechniki Śląskiej, Gliwice, 1988.

[2] Z. SZKARADNIK, H. BIESIADA, W. PETRYKOWSKI, Opis języka Forth-60, Gliwice, 1984.

[3] Dokumentacja użytkowa biblioteki procedur graficznych PLOT, Zakłady Urządzeń Komputerowych Mera-Elzab, Zabrze, 1986.

Recenzent: Dr Jerzy Karczmarczuk

Wpłynęło do Redakcji 31.08.1988.

ГРАФИЧЕСКИЙ FORTH ДЛЯ МИКРОКОМПЬЮТЕРА COMPAN-8

Р е з ю м е

В статье показана имплементация графической системы для микрокомпьютера Com PAN-8. Система эта является интегральной частью транслятора языка Forth. Особенное внимание уделено проблемом присоединения драйвов а также реализации алгоритмов заполнения.

GRAPHIC FORTH FOR THE ComPAN-8 MICROCOMPUTER

S u m m a r y

This paper presents the implementation of graphic system for the ComPAN-8 microcomputer. This system is an integral part of the Forth language translator. The particular attention is dedicated to problem of drivers linking and realization of filling algorithms.

Listing 1.

```
variable  xleft                variable  xright
variable  xr                   variable  yr

: right>                  ( --- )
  1 xr +! begin
            xr @ yr @ point 0= while
            xr @ yr @ dot 1 xr +!
          repeat xr @ 1- xright ! ;

: left>                   ( --- )
  -1 xr +! begin
            xr @ yr @ point 0= while
            xr @ yr @ dot -1 xr +!
          repeat xr @ 1+ xleft ! ;

: (scan)                  ( --- x y ... )
  begin xr @ xright @ <= while
    0 flag ! begin
      xr @ yr @ point 0= xr @ xright @ <= and while
      1 flag ! 1 xr +! repeat
    flag @ if
      xr @ yr @ point 0= xr @ xright @ = and if
        xr @ yr @ else xr @ 1- yr @ then then
    xr @ begin
      xr @ yr @ point xr @ xright @ < and while
      1 xr +! repeat
    xr @ = if 1 xr +! then
  repeat ;

variable stack

: fill                    ( x y --- )
  sp@ 4 + stack !
  begin
    yr ! xr !
    xr @ yr @ dot
    xr @ right> xr ! left>
    xleft @ xr ! 1 yr +! (scan)
    xleft @ xr ! -2 yr +! (scan)
sp@ stack @ = until ;
```

Listing 2.

```
: points          ( yn xn ... y1 x1 n --- )
  create 0 do , , loop does> ;

: .x              ( adr n --- x [n] ) 2* 2* + @ ;

: .y              ( adr n --- y [n] ) 2* 2* 2+ + @ ;

: line            ( adr n --- )
  swap dup dup 0 .x swap 0 .y plot
  swap 1 do
         dup dup i .x swap i .y draw
       loop drop ;

variable pts   variable vert

: sort            ( xn ... x1 xi --- xn . xi . x1 )
  xmin @ begin sp@ stack @  <> while
                                 2dup < if swap then >r repeat
  begin dup xmin @ <> while r> repeat drop ;

: maxy            ( --- maxy )
  ymin @ vert @ 0 do pts @ i .y max loop ;

: miny            ( --- miny )
  ymax @ vert @ 0 do pts @ i .y min loop ;

: drawscan         ( ys --- )
  >r begin sp@ stack @ <> while r@ plot r@ draw repeat r> drop ;

: area            ( adr n --- )
  2dup line vert ! pts !
  sp@ stack ! maxy 1+ miny do
                 vert @ 1- 0 do
                   pts @ i .y pts @ i 1+ .y
                   2dup min j < rot rot 2dup max j >= rot rot
                   <> and and if
                     j pts @ i .y -
                     pts @ i 1+ .x pts @ i .x -
                     pts @ i 1+ .y pts @ i .y -
                     */mod swap drop pts @ i .x + sort
                   then
                 loop i drawscan
               loop ;
```

Listing 3.

```
: monitor                ( --- )
  0 xmin ! 639 xmax !
  0 ymin ! 239 ymax !
  ['] mt-clear is clear
  ['] mt-point is point
  ['] mt-plot  is plot
  ['] mt-dot   is dot
  ['] (draw)   is draw
  clipon ;

: printer                ( --- )
  0 xmin ! 959 xmax !
  0 ymin ! 719 ymax !
  ['] pr-clear is clear
  ['] pr-point is point
  ['] pr-plot  is plot
  ['] pr-dot   is dot
  ['] (draw)   is draw
  clipon ;

: plotter                ( --- )
  0 xmin ! 959 xmax !
  0 ymin ! 719 ymax !
  ['] pl-clear is clear
  ['] pl-point is point
  ['] pl-plot  is plot
  ['] pl-dot   is dot
  ['] pl-draw  is draw
  10 plemit
  27 plemit ascii b plemit
  clipon ;
```

Listing 4

```
( Sierpinski curves )

variable edge   variable level   variable diagonal

: s                 ( level --- )
  dup 0= if drop else
          dup 1- recursive s 45 right diagonal @ forward 45 righ
          dup 1- recursive s 90 left edge @ forward 90 left
          dup 1- recursive s 45 right diagonal @ forward 45 righ
          1- recursive s then ;

: Sierpinski          ( edge level --- )
  level ! edge !
  edge @ 1000 m* 1414 m/mod swap drop diagonal !
  4 0 do level @ s 45 right diagonal @ forward 45 right loop ;
```

Listing 5.

```
( Dragon )

variable step   3 step !

: dragon              ( sign level --- )
  dup 0= if
           2drop step @ forward
         else
           over 45 * turn
           1 over 1-
           recursive dragon
           over -90 * turn
           -1 over 1-
            recursive dragon
            drop 45 * turn
          then ;
```