Pascal uitgediept

Herman Post
MCCM 70

Scanned, ocr'ed and converted to PDF by HansO, 2001

Vlakken kunnen natuurlijk heel saai met één kleur gevuld worden, maar vaak is het vullen met een zogenaamd vulpatroon veel aantrekkelijker.

Vulpatronen

Naar aanleiding van de serie over object georiënteerd tekenen heb ik met mijn collega Jacco Kulman zitten discussiëren over de mogelijkheden—en onmogelijkheden—van berekende Fills. Het ging hierbij over het inkleuren van vlakken in een tekening, maar niet met één kleur, maar met een patroon of met een plaatje dat door de computer berekend is. Dit zou bijvoorbeeld een fractal kunnen zijn, maar ook een willekeurige stippeling of lijnpatroon.

Inkleuren

Voor het inkleuren van een begrensd vlakdeel, of zoals Jacco zegt 'een object' vallen fractals al vlug af, omdat de benodigde rekentijd voor onze computer te lang zou worden. Het inkleuren van ieder vlakje zou al vlug enkele minuten duren en als er dan tientallen vlakjes moeten worden ingekleurd zou de gebruiker erg lang moeten wachten. Dan komen we dus aan bij de berekende vullingen.

Patronen

Hiervoor is de eenvoudigste vulling natuurlijk een uit rechte, horizontale of verticale lijnen opgebouwd patroon. Een eenvoudige FOR-lus kan zo'n vulling opbouwen. Moeilijker wordt het als er ook nog gelijkmatige overgangen in de kleuren moeten zitten. Er moet dan, afhankelijk van het scherm waarin gewerkt wordt, een palet worden aangepast, of kleurnummers moeten zo berekend worden dat de kleurovergangen gelijkmatig zijn.

Listing

Om te demonstreren dat dit redelijk eenvoudig is, heeft Jacco een demonstratieprogramma geschreven dat ik als listing bij dit artikel heb opgenomen. Er wordt gewerkt met een patroon van willekeurige kleuren—de zogenaamde 'ent'-kleuren—waartussen de gemiddelde kleurovergangen worden berekend. Bij het opstarten van het programma wordt eerst gevraagd om de afstand tussen de punten. Deze afstand geeft een vergrotingsfactor aan. In een object georiënteerd programma zou hiermee de vergrotingsfactor traploos in te stellen zijn. Bij deze versie, die voor bitmaps is bedoeld, is de vergroting altijd een tweemacht. U geeft de exponent van de afstand tussen de ent-punten op. Als u dus l opgeeft, is de afstand tussen de ent-punten 2^1=2 pixels. Geeft u 7 op, dan is de afstand 2^7=128 pixels. Tussen de ent-

punten is het nu gemakkelijk om de gemiddelden te berekenen, omdat de afstand tussen twee punten altijd door twee is te delen.
Stel u geeft als afstand 3 op. De afstand tussen twee ent-punten is dan 2^3=8 pixels. Plaatst u een gemiddelde waarde er tussen in, dan is de afstand tussen het oorspronkelijke punt en het nieuw berekende punt 8 DIV 2 =4 pixels. Bij de volgende berekening is de afstand 4 DIV 2=2 pixels, en de keer daarop is de afstand l pixel, en is de volledige bitmap gesloten met een kleurovergang die keurig van ent-punt tot ent-punt loopt.
Om duidelijk te maken dat de kleur-nummers netjes in elkaar overlopen, wordt nadat het vulpatroon is opgebouwd met behulp van palet wisselingen, een kleurig bewegend geheel opgebouwd. De vertraging die u bij de start van het programma ingeeft wordt hier gebruikt om de afwisseling sneller of langzamer te laten verlopen. Dit programma wordt pas echt leuk als u zelf dingen gaat aanpassen. Probeert u maar eens om de randomfunctie te vervangen door (i MOD 14) +1 of door ((i+j)MOD 14)+1. Ook in de opbouw van de kleuren valt aardig wat te experimenteren, kleurovergangen die bijvoorbeeld door het gehele palet lopen of juist helemaal geen kleurovergangen maar complementaire kleuren tegen elkaar aan. Hoe u leuk een kleurovergang kunt berekenen vindt u terug in MCCM 60 blz 11 K&K bovenaan.

```
PROGRAM Plasma7;

TYPE  Str255 = STRING[255];

VAR actpage      : BYTE ABSOLUTE $FAF6;
    logopr       : BYTE ABSOLUTE $FB02;
    atrbyt       : BYTE ABSOLUTE $F3F2;
    border       : BYTE ABSOLUTE $F3EB;
    ent_dist     : BYTE;
    ch           : CHAR;
    color        : ARRAY[1..43,1..3] OF BYTE;
    ofset        : BYTE;
    vertraging   : INTEGER;

PROCEDURE Screen(mode:byte);
BEGIN
  INLINE($3A/mode/
         $FD/$2A/$F7/$FA/
         $DD/$21/$D1/$00/
         $CD/$1C/$00/
         $FB)
END;

PROCEDURE FastBox(x1, y1, x2, y2 : INTEGER);
BEGIN
  INLINE($AF/$2A/x2/$ED/$5B/x1/$ED/$52/$30/$0A/$CB/$D7/$EB/
         $ED/$5B/x2/$A7/$ED/$52/$E5/$2A/y2/$ED/$5B/y1/$A7/
         $ED/$52/$30/$0A/$CB/$DF/$EB/$ED/$5B/y2/$A7/$ED/$52/
         $D1/$D9/$08/$F3/$3E/$02/$D3/$99/$3E/$8F/$D3/$99/$DB/
         $99/$1F/$38/$F3/$AF/$D3/$99/$3E/$8F/$D3/$99/$0E/$9B/
         $3E/$24/$D3/$99/$3E/$91/$D3/$99/$2A/x1/$ED/$69/$ED/
         $61/$3A/y1/$D3/$9B/$3A/actpage/$D3/$9B/$D9/$0E/$9B/
         $ED/$59/$ED/$51/$ED/$69/$ED/$61/$3A/atrbyt/$D3/$9B/
         $08/$D3/$9B/$3E/$C0/$D3/$9B/$FB)
END;

PROCEDURE ChangeColor(colornr, red, green, blue : Byte);
BEGIN
  INLINE($F3/$3A/colornr/$D3/$99/$3E/$90/$D3/$99/$3A/red/$E6/
         $0F/$07/$07/$07/$07/$47/$3A/blue/$E6/$0F/$B0/$D3/$9A/
         $3A/green/$D3/$9A/$FB)
END;

PROCEDURE PSet(X, Y : INTEGER);
BEGIN
  INLINE($f3/$af/$d3/$99/$3e/$8f/$d3/$99/$0e/$9b/$3e/$24/$d3/
         $99/$3e/$91/$d3/$99/$2a/X/$ed/$69/$ed/$61/$3a/Y/$d3/
         $9b/$3a/ActPage/$d3/$9b/$d3/$9b/$d3/$9b/$d3/$9b/$d3/
         $9b/$3a/Atrbyt/$d3/$9b/$af/$d3/$9b/$3a/Logopr/$e6/
         $0f/$f6/$50/$d3/$9b/$fb)
END;

FUNCTION Point(x,y : INTEGER) : BYTE;
VAR color : BYTE;
BEGIN
  INLINE($F3/$0E/$9B/$3E/$20/$D3/$99/$3E/$91/$D3/$99/$2A/x/
         $ED/$69/$ED/$61/$3A/y/$D3/$9B/$3A/actpage/$D3/$9B/
         $3E/$2D/$D3/$99/$3E/$91/$D3/$99/$AF/$D3/$9B/$3E/$40/
         $D3/$9B/$3E/$07/$D3/$99/$3E/$8F/$D3/$99/$DB/$99/$32/
         color/$AF/$D3/$99/$3E/$8F/$D3/$99/$FB);
```

```
  Point:=color
END;

PROCEDURE SetColors;

  PROCEDURE ChColor(c,r,g,b : BYTE);
  BEGIN
    color[c,1]:=r;
    color[c,2]:=g;
    color[c,3]:=b;
    color[28+c,1]:=r;
    color[28+c,2]:=g;
    color[28+c,3]:=b;
    color[30-c,1]:=r;
    color[30-c,2]:=g;
    color[30-c,3]:=b;
    ChangeColor(c,r,g,b);
  END;

BEGIN
  ChColor(1,7,7,0);
  ChColor(2,6,6,1);
  ChColor(3,5,5,2);
  ChColor(4,4,4,3);
  ChColor(5,3,3,4);
  ChColor(6,2,2,5);
  ChColor(7,1,1,6);
  ChColor(8,0,0,7);


  ChColor(9,1,1,6);
  ChColor(10,2,2,5);
  ChColor(11,3,3,4);
  ChColor(12,4,4,3);
  ChColor(13,5,5,2);
  ChColor(14,6,6,1);
  ChColor(15,7,7,0);
  ofset:=0;
END;

PROCEDURE RotateColors;
VAR i,j : BYTE;
BEGIN
  FOR i:=1 TO 15 DO
  BEGIN
    j:=i+ofset;
    ChangeColor(i,color[j,1],color[j,2],color[j,3]);
  END;
  ofset:=SUCC(ofset);
  IF ofset=29 THEN ofset:=1;
END;

PROCEDURE ZetEnten;
VAR i,j,s : INTEGER;
BEGIN
  s:=1 SHL ent_dist;
  j:=0;
  REPEAT
    i:=0;
    REPEAT
      i:=i+s;
```

```
      atrbyt:=RANDOM(14)+1;
      PSet(i,j);
    UNTIL i>255;
    j:=j+s;
  UNTIL j>211;
END;

PROCEDURE ZetGemiddelden;
VAR i,j,s : INTEGER;
    p1,p2,p3,p4 : BYTE;
    g1,g2,g3,g4 : BYTE;
BEGIN
  s:=1 SHL ent_dist;
  REPEAT
    j:=0;
    REPEAT
      i:=0;
      p1:=Point(i,j);
      p4:=Point(i,j+s);
      REPEAT
        g4:=(p1+p4) SHR 1;
        p2:=POINT(i+s,j);
        g1:=(p1+p2) SHR 1;
        atrbyt:=g1;
        PSet(i+s SHR 1,j);
        atrbyt:=g4;
        PSet(i,j+s SHR 1);
        p3:=POINT(i+s,j+s);
        atrbyt:=(p1+p2+p3+p4) SHR 2;
        PSet(i+s SHR 1,j+s SHR 1);
        i:=i+s;
        p1:=p2;
        p4:=p3;
      UNTIL i>255;
      j:=j+s;
    UNTIL j>211;
    s:=s SHR 1;
  UNTIL s=1;
END;

BEGIN
  ClrScr;
  WRITE('Geef de ent-afstand (1-7)   : ');
  READLN(ent_dist);
  WRITE('Geef de vertraging (0-32767): ');
  READLN(vertraging);
  border:=0;
  Screen(5);
  AtrByt:=0;
  FastBox(1,1,257,212);
  FastBox(0,0,1,1);
  SetColors;
  ZetEnten;
  ZetGemiddelden;
  ch:='*';
  REPEAT
    RotateColors;
    DELAY(vertraging);
    IF KEYPRESSED THEN READ(KBD,ch);
  UNTIL ch<>'*';
  ChangeColor(4,0,0,7);
```

```
  ChangeColor(15,7,7,7);
  Screen(0);
END.
```