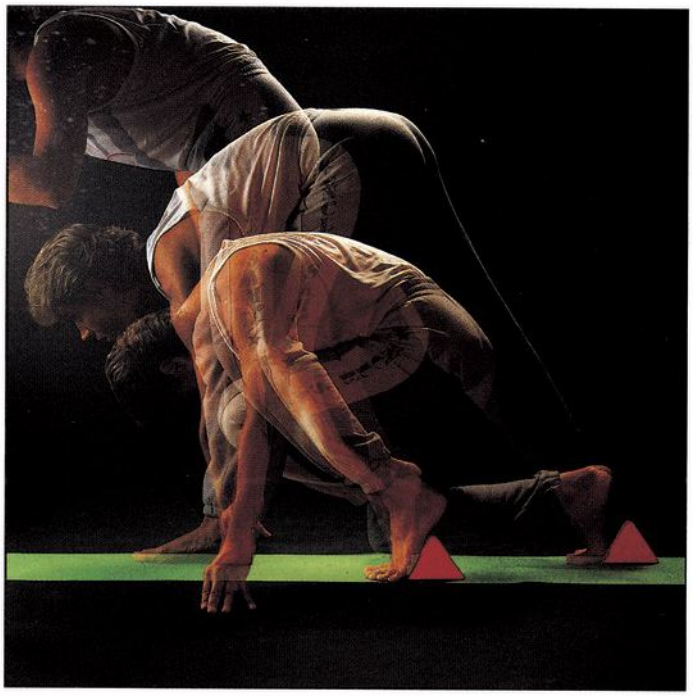
# Sprites ohne Geheimnisse



**Sprites sind für manchen Einsteiger reizvolle, aber schwer zu programmierende Gebilde. Man muß viele Register kennen und mit Binärzahlen arbeiten. Dieser Artikel soll Ihnen helfen, beide Hürden zu meistern.**

In Bild 1 finden Sie einen Fahrplan, der in 13 Stationen den Weg zum funktionierenden Sprite enthält. Diesem Schema werden wir nun folgen und anhand von zwei Sprites Abschnitt für Abschnitt ein Sprite-Programm aufbauen.

**Schritt 1**

Der erste Schritt spielt sich im Kopf ab: Wir fällen eine Entscheidung darüber, ob unser Sprite einfarbig (dafür aber feiner strukturiert) oder mehrfarbig werden soll. Von dieser Entscheidung hängt nämlich Schritt 2 ab. Wir werden zweispurig weiterfahren, indem wir sowohl ein einfarbiges als auch ein »Multicolor«-Sprite wählen.

**Schritt 2**

Hier nehmen wir uns nun ein Blatt kariertes Papier zur Hand und zeichnen darauf ein Rechteck mit 24 Karos horizontaler und 21 Karos vertikaler Ausdehnung. Das wird unser Sprite-Raster. Dann warten wir auf den Kuß der Musen, der uns eine geeignete Sprite-Gestalt eingibt. Sind wir soweit, dann trennen sich die Wege:

a) Das einfarbige Sprite tragen wir in unser Schema ein wie ein Stickmuster: Überall, wo ein Bildpunkt erscheinen soll, machen wir ein Kreuzchen (eine 1). Dann ergibt sich unser erstes Sprite wie Bild 2 zeigt.

b) Etwas komplexer ist das mit dem Multicolorsprite. Hier treten vier Farben auf, die sich durch die Zahlenkombinationen 00, 01, 10 und 11 unterscheiden lassen. 00 besitzt die Hintergrundfarbe (ist also durchsichtig). Am besten benutzt man drei verschiedene Farbstifte (je einen für 01, 10 und 11) und trägt in das Schema mit dem jeweiligen Stift die Zweierkombination ein. Das Ergebnis sieht dann so aus wie in Bild 3.

**Schritt 3**

Nach diesen — mehr schöpferischen — Tätigkeiten kommt nun die Mathematik zu Wort. Wir müssen nämlich nun das, was wir auf dem Papier vor uns haben, in eine dem Computer verständliche Form bringen, also in Zahlen. Sehen Sie sich dazu nochmals Bild 2 und 3 an. Dort wurden zwei senkrechte Linien gezogen, die jede Zeile in drei Achtergruppen aufteilt. Eine solche Gruppe ist der künftige Inhalt eines Bytes. Jedes einzelne Kästchen entspricht einem Bit, und schon sind wir auf diese Weise bei den Binärzahlen gelandet. Wir werden aber hier keine Erklärung über alle Höhen und Tiefen dieser Zahlenart starten, sondern nur feststellen, auf welche Weise wir das Gebilde, das wir vor uns haben, in eine normale Zahl umrechnen können. Das ist ganz einfach. Sehen Sie sich dazu Bild 4 an.

Sie finden darin ein Byte und die Numerierung der Bits. Überall dort, wo an einer Bit-Position eine 1 steht, merkt man sich die im Bild darunter stehende Zahl. Alle auf diese Weise gemerkten Zahlen addiert man schließlich. Das Ergebnis ist die Dezimalzahl, die uns interessiert. Ein Beispiel soll das noch verdeutlichen. In Bild 3 (dem Multicolorsprite) finden Sie in Zeile 2 als erstes Byte: 10100000. Wenden wir nun unser Bild 4 darauf an:

```
  1  0  1  0 0 0 0 0
128 64 32 16 8 4 2 1
```

Sie sehen, wir müssen addieren:

128 + 32 = 160

Nun wissen Sie, wie es gemacht wird. Üben können Sie das nun bei allen 126 Byte unserer beiden Sprites. Was dabei herauskommt, sehen Sie ebenfalls in Bild 2 und 3 rechts neben dem Raster.

**Schritt 4**

Nachdem wir nun unsere Sprites dem Computer mundgerecht gemacht haben, überlegen wir uns, in welchen Speicherbereich wir sie plazieren. Zwei Bedingungen sind dabei zu beachten:

1) Die Sprites müssen im gleichen 16 K-Bereich liegen wie der Bildschirm. (Im Normalfall also von Speicherstelle 0 bis Speicherstelle 16383.)

2) Die Startadresse der Sprite-Daten muß glatt durch 64 teilbar sein.

| Bit-Nr. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| mal | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

**Bild 4. Bitnummern und ihre Werte. Eine kleine Rechenhilfe.**

Außerdem sollte man natürlich sinnvolle Speicherbereiche auswählen, das heißt solche, die weder der Computer selber braucht (wie die ersten 1023 Byte), noch solche, die für das Basic-Programm benötigt werden. Allerdings gibt es erfreulicherweise Lücken in den ersten 1023 Speicherplätzen, die nicht oder nur recht selten vom Computer benötigt werden. Man kann dort tatsächlich die Daten von vier Sprites ablegen:

704 bis 767 1. Sprite
832 bis 895 2. Sprite
896 bis 959 3. Sprite
960 bis 1023 4. Sprite

Die letzten Bereiche gehören zum Kassettenpuffer. Falls Sie also während eines Sprite-Programmes planen, eine Datasetten-Operation vorzunehmen, müssen Sie dafür sorgen, daß Ihre Sprite-Daten später wieder in den Kassettenpuffer geschrieben werden.

Sollten Sie mehr als vier Sprites benötigen, dann bleibt Ihnen nichts anderes übrig, als dazu den Basic-Speicherraum zu verwenden. Ich lege die Daten in diesem Fall häufig an die obere Grenze, also knapp unter 16383. Dann muß ich diesen Speicherteil vor dem Überschreiben durch das Basic-Programm (und Variable) schützen.

Eine andere Möglichkeit ist es, den Start des Basic-Speichers höher zu legen. All dies soll uns an dieser Stelle aber nicht belasten. Wir gehen hier davon aus, daß wir mit vier Sprite-Mustern auskommen.

Wie in der obigen Aufstellung schon angedeutet, legen wir unsere Sprite-Daten für Sprite 1 (das einfarbige Sprite) nach Speicherstelle 704 und für Sprite 2 nach 832.

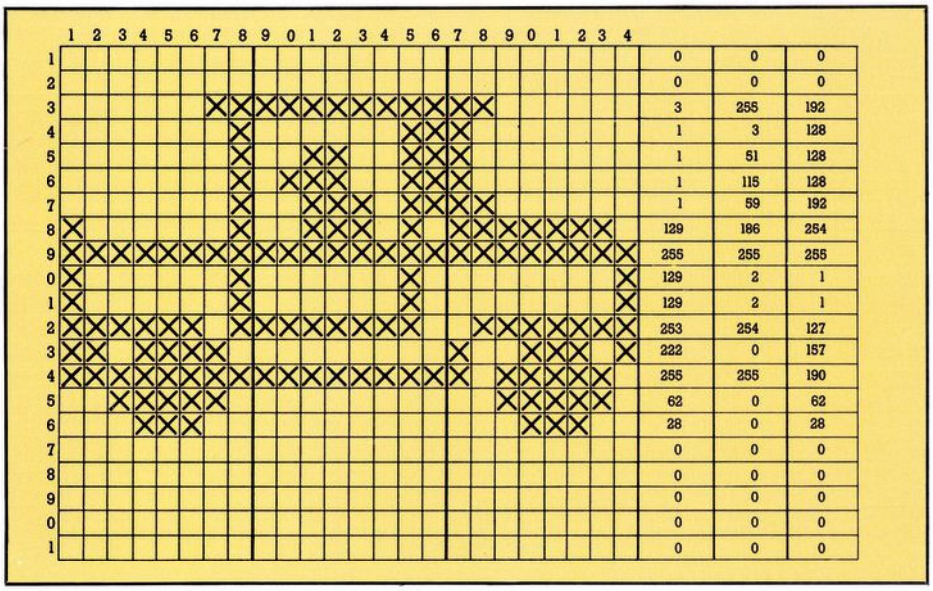



**Bild 2. Entwurf eines einfarbigen Sprite**

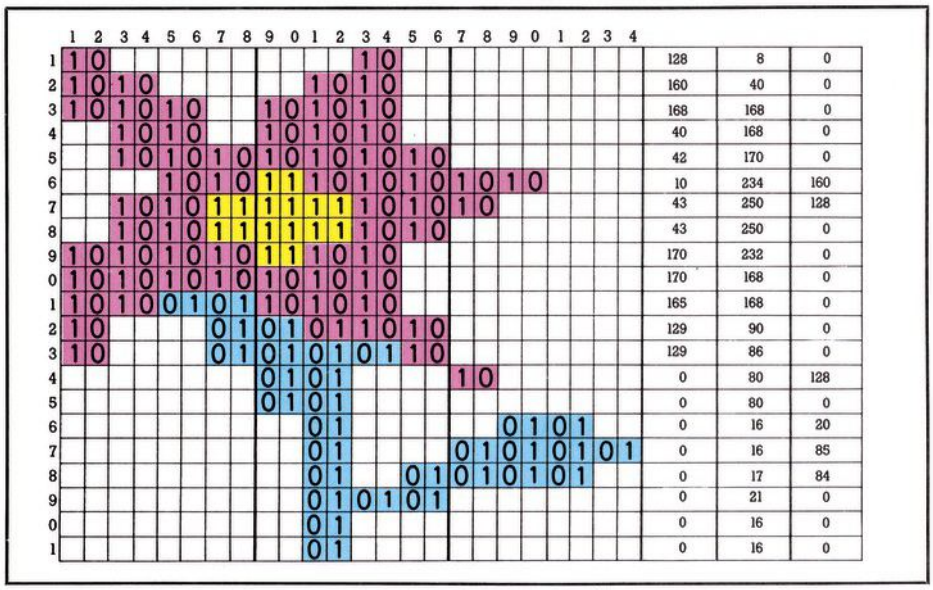



**Bild 3. So kann man einen Multicolor-Sprite aufbauen**

## Nun geht's ans Programmieren:

**Schritt 5**

Wir lesen die gewonnenen Sprite-Daten in den Computerspeicher ein mittels zweier simpler Schleifen:

```
100 FOR I=704 TO 766:READD
    :POKE I, D:NEXT I
110 FOR I=832 TO 894:READD
    :POKE I, D:NEXT I
115 REM ****** DATAS VON
    SPRITE 1 ******
120 DATA 0,0,0,0,0,0,3,255,192,1,3,128
130 DATA 1,51,128,1,115,128,1,59,192
140 DATA 129,186,254,255,255,255,
    129,2,1
150 DATA 129,2,1,253,254,127,222,0,
    157
160 DATA 255,255,190,62,0,62,28,0,28
170 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
175 REM ****** DATAS VON
    SPRITE 2 ******
180 DATA 128,8,0,160,40,0,168,168,0
190 DATA 40,168,0,42,170,0,10,234,160
200 DATA 43,250,128,43,250,0,170,
    232,0
210 DATA 170,168,0,165,168,0,129,90,0
220 DATA 129,86,0,0,80,128,0,80,0
230 DATA 0,16,20,0,16,85,0,17,84,0,
    21,0
240 DATA 0,16,0,0,16,0
```

Damit hätten wir den schlimmsten Teil der ganzen Arbeit hinter uns!

**Schritt 6**

Nun müssen wir unserem Computer mitteilen, an welchen Stellen im Speicher wir die Daten verstaut haben. Dazu gibt es ab Speicherstelle 2040 die sogenannten Sprite-Zeiger. Hier kommt die Startadresse der Daten hinein und zwar eine Kennzahl, die sich so ergibt:

Datenstartadresse/64 = Kennzahl

Für das erste Sprite folgt 704/64 = 11 und für das zweite Sprite 832/64 = 13. Gleichzeitig legt man auf diese Weise eine Sprite-Nummer fest, die von 0 (Speicherstelle 2040) bis 7 (Speicherstelle 2047) gehen kann. Unser Sprite 1 machen wir nun zum Sprite 0 und schreiben die Kennzahl 11 in Speicherstelle 2040. Analog dazu wird Sprite 2 zum Sprite 1 und wir schreiben 13 in Byte 2041:

```
245 REM ****
    SPRITE-ZEIGER ****
250 POKE2040,11:POKE2041,13
```

Immer dann, wenn im folgenden von einer Spritenummer N die Rede ist, meinen wir damit die durch die Spritezeiger festgelegte (also 0 für das einfarbige, 1 für das Multicolorsprite).

**Schritt 7**

Ob wir diesen Schritt jetzt gehen oder später, ist eigentlich gleichgültig. Es dreht sich um das Einschalten

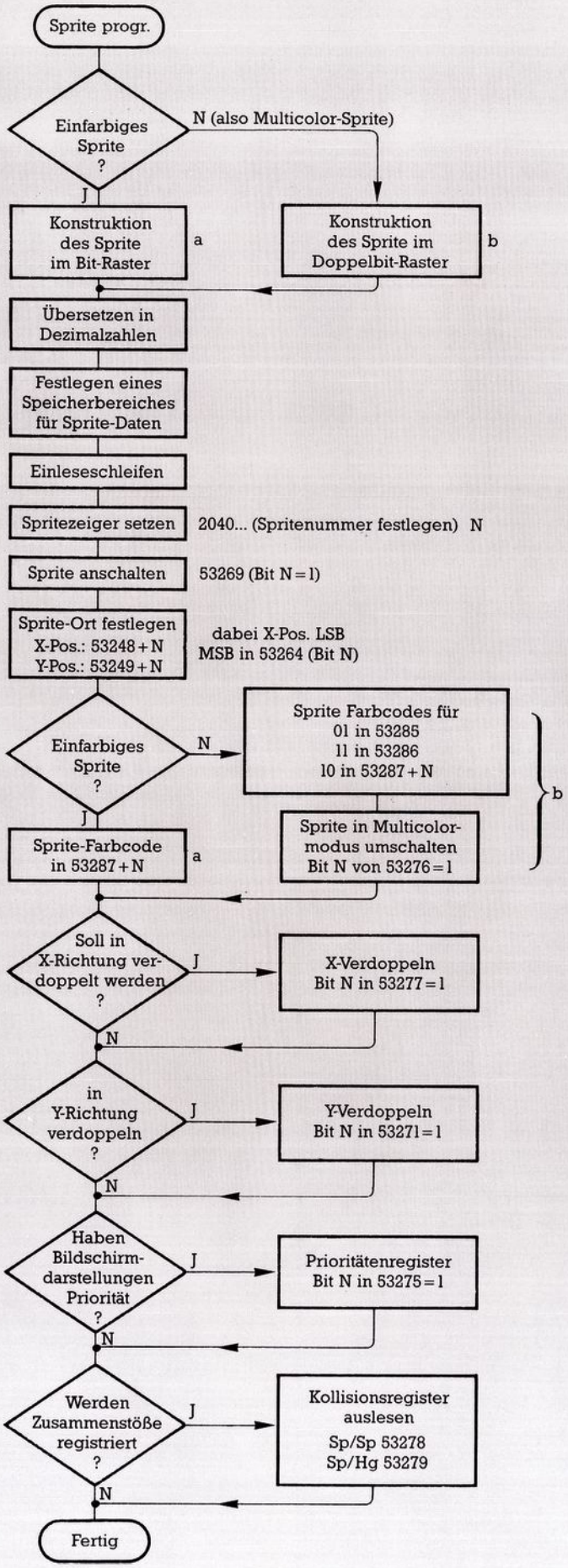



**Bild 1. Fahrplan zum fertigen Sprite**

der Sprites. Dazu bedienen wir uns des Registers in Speicherstelle 53269. Dies ist das erste einer Reihe von Registern, die bei Sprites eine Rolle spielen und nach einem Schema funktionieren: Von den 8 Bits des Registers gehört zu jedem Bit ein Sprite. Also zu Sprite 0 das Bit 0, zu Sprite 1 das Bit 1 und so weiter, allgemein zu Sprite N das Bit N. Jedes Bit mit dem Inhalt 1 bewirkt, daß das dazugehörige Sprite angeschaltet ist, ein Bit-Inhalt 0 sorgt für das Ausschalten des Sprite

Ohne nun auf mathematische Einzelheiten einzugehen, genügt es zu wissen, daß man einzelne Sprites einschaltet mittels:

```
POKE 53269,PEEK(53269)OR(2↑N)
```

und abschaltet mit:

```
POKE 53269,PEEK(53269)AND(255-2↑N)
```

Wir schalten nun unsere beiden Sprites an:

```
255 REM **** EINSCHALTEN ****
260 POKE53269,PEEK(53269)
    OR(2↑0):
    POKE53269,PEEK(53269)
    OR(2↑1)
```

### Schritt 8

Sie werden festgestellt haben, daß kein Sprite auf dem Bildschirm erschienen ist. Wir müssen nämlich noch festlegen, wo das Sprite auftauchen soll, müssen also für jedes Sprite eine X- und eine Y-Position eingeben. Oje, jetzt wird es wieder komplizierter. Sehen Sie sich am besten mal das Bild 5 an. Sie sehen dort ein Koordinatensystem, das X-Werte von 0 bis 511 und Y-Werte von 0 bis 255 zuläßt. Genau das sind die möglichen Sprite-Orte. Dabei zählt die linke obere Ecke des Sprite-Rasters als Anhaltspunkt. Umrahmt ist der Teil der Fläche, der den sichtbaren Bildschirm darstellt. Man kann also Sprites so plazieren, daß sie unsichtbar bleiben, weil sie außerhalb des Bildschirms liegen. Auf dem Bild sind zwei Sprites gezeigt, von denen eins voll sichtbar (mit den Koordinaten 128, 120) und das andere unsichtbar ist (Koordinaten 360, 190). Auch teilweise sichtbare Sprites sind möglich.

Die X-Position eines Sprites wird ebenso wie die Y-Position in jeweils eine Speicherstelle gePOKEt. Diese findet man ab 53248, wobei 53248 die X-Position von Sprite 0, 53249 dessen Y-Position enthält, 53250 nimmt dann die X-Koordinate von Sprite 1, 53251 die Y-Koordinate auf und so weiter. Allgemein POKEt man die X-Koordinate für Sprite N in 53248+2*N und die Y-Koordinate in 53249+2*N.

Für die Y-Positionen ist das ohne Probleme. Bei der X-Koordinate tauchen Schwierigkeiten auf: Sobald eine X-Position größer als 255 benötigt wird, kann man sie nicht mehr einPOKEn. In diesem Fall verfährt man so. Man bildet die Differenz X-256 = X1 und POKEt X1 ein. Außerdem muß man nun in einem weiteren Register in 53264 (das wieder für jedes Sprite ein Bit bereithält) das dazugehörige Bit auf 1 setzen.

Für diese Eigenheiten gibt es natürlich eine Menge sehr eleganter Programmier-Möglichkeiten, die alle das Manko haben, reichlich unverständlich zu sein. Begnügen wir uns also mit einfachen IF..THEN-Abfragen. Unser Programm lautet also weiter:

```
265 REM **** SPRITE-
    POSITIONEN ****
270 FOR N=0TO1: PRINT
    "SPRITE"N;:
    INPUT"X,Y= ";X(N),Y(N)
280 POKE 53249+2*N,Y(N)
290 IFX(N)>255THENPOKE53248
    +2*N,X(N)-256
    :POKE53264,PEEK(53264)OR
    (2↑N):GOTO305
300 POKE53248+2*N,X(N):
    POKE53264,PEEK(53264)AND
    (255-2↑N)
305 IF K=1 THEN RETURN
310 NEXT N
```

Dabei verlassen wir uns darauf, daß für X und Y nur sinnvolle Werte eingegeben werden. Wenn das aus irgendeinem Grund fraglich ist, müssen noch weitere IF...THEN-Abfragen dazukommen, die X und Y überprüfen vor Zeile 280. Die Zeile 305 ist vorsorglich für eine spätere Option eingebaut.

**Schritt 9**

Hier geht's um die Farbe unserer Sprites. Wie Sie sich denken können, müssen Multicolor-Sprites anders behandelt werden als einfarbige. Doch beiden gemeinsam ist zunächst folgendes:

Für jedes Sprite existiert ein Farbregister, das bei 53287 anfängt. Unser Sprite 0 hat in dieser Speicherstelle seinen Farbcode stehen, das Sprite 1 (das Multicolor-Sprite) in 53288 und so weiter. Allgemein steht die Farbe von Sprite N in Farbregister 53287+N.

Multicolorsprites bewahren in dieser Speicherstelle die Farbe auf, die zur Ziffernkombination 10 gehört. Die Kombination 01 und 11 werden für alle Multicolor-Sprites gemeinsam festgelegt. Dabei gehört der in 53285 stehende Farbcode zur Kombination 01, der in 53286 zur Kombination 11.

Außerdem muß für die Multicolor-Sprites noch der Mehrfarben-Modus eingeschaltet werden. Dazu existiert wieder ein Register (53276), in dem für jede Spritenummer N ein Bit reserviert ist. Ist dieses Bit gleich 1, dann liegt das dazugehörige Sprite im Mehrfarben-Modus vor. Allgemein schaltet man für Sprite N also den Multicolor-Modus an durch:

POKE 53276,PEEK(53276)OR(2↑N)

Das Rücksetzen geschieht durch Löschen des Bit N:

POKE 53276,PEEK(53276)AND(255-2↑N)

Wir wählen für Sprite 0 die Farbe CYAN, für das Multicolor-Sprite die Zuordnungen:

Kombination 01 = Grün
11 = Gelb
10 = Rot

```
315 REM ****
    SPRITE-FARBEN ****
320 POKE53287,3:REM SPRITE 0
    =CYAN
330 POKE53285,5:REM SPRITE 1
    KENNUNG 01 =GRUEN
340 POKE53286,7:REM
    KENNUNG 11=GELB
350 POKE53288,2:REM
    KENNUNG 10=ROT
360 POKE53276,PEEK
    (53276)OR(2↑1):
    REM MULTICOLORMODUS
    EINGESCHALTET
```

**Schritt 10**

Wenn Sie bis hierher unser schrittweise entwickeltes Programm eingegeben und gestartet haben, sehen Sie nun auf Ihrem Bildschirm zwei nette Sprites. Im folgenden werden wir nun noch einige Besonderheiten der Sprite-Programmierung kennenlernen, die wir aber im Text nur noch mit den allgemeinen Befehlen bearbeiten. Ein beigefügtes Beispielprogramm, das bis hierher (außer den Zeilen bis 100) mit unserem bisher entwickelten identisch ist, enthält die konkreten Befehle. Um das weitere Vorgehen deutlich zu machen, sind dort zunächst in den Zeilen 370 bis 400 drei weitere Sprites (mit denselben Daten) eingerichtet worden.

Schritt 10 stellt uns vor die Frage, ob wir unser Sprite in X-Richtung doppelt so groß darstellen wollen. Zu diesem Zweck gibt es dann wieder ein Register (jedes Bit ein Sprite), das Register 53277. Die X-Vergrößerung ist eingeschaltet, wenn für Sprite N das Bit N auf 1 gesetzt wurde mittels:

POKE53277,PEEK(53277)OR(2↑N)

Das Zurückschalten findet dann wieder statt mit ...AND(255-2↑N). Im Programm wurde Sprite 2 derart vergrößert.

**Schritt 11**

Dasselbe wie eben kann auch in der Y-Richtung geschehen. Dazu dient uns das Register 53271. Anschalten dieser Y-Verdoppelung also durch:

POKE53271,PEEK(53271)OR(2↑N)

und Abschalten wie oben schon gezeigt mittels ...AND(255-2↑N).

Im Programm ist Sprite 3 so vergrößert worden. Sprite 4 ist — auch das funktioniert — sowohl in X- als auch in Y-Richtung gedehnt worden.

**Schritt 12**

Hier geht's um die Vorfahrt. Wenn sich zwei Sprites überdecken, bleibt immer dasjenige »vorne«, das die niedrigere Nummer besitzt. Also verdeckt Sprite 0 alle anderen. Man hat darauf — außer durch geeignete Auswahl der Spritenummer ganz zu Beginn — keine Einflußmöglichkeit.

Anders verhält sich das bei der Vorfahrt von Sprites und Bildschirmzeichen. Die kann mittels Register 53275 geregelt werden. Wieder gehört zu jedem Sprite ein Bit. Ist dieses gleich 0, erscheint das Sprite vor den Bildschirmzeichen, ist es gleich 1, versteckt es sich dahinter. Im Beispielprogramm wurde in den Zeilen 450 und 460 geregelt, daß sich alle Autos (Sprites 0,2 und 4) hinter den Bildschirmzeichen aufhalten.

**Schritt 13**

Unser Computer paßt auch auf Zusammenstöße auf. Zwei Register (wieder Bit N für Sprite N) sind zuständig:

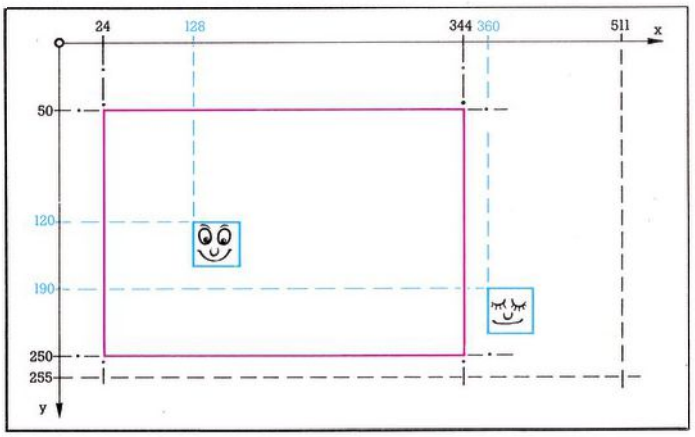



**Bild 5. In diesem Feld können sich Sprites aufhalten. Rot umrandet, der sichtbare Bildschirm.**

Kollisionen Sprite mit Sprite (53278), Kollisionen Sprite mit Bildschirmzeichen (53279). Für jedes Sprite, das in eine Kollision verwickelt wurde, wird das entsprechende Bit auf 1 gesetzt. Wenn also Sprite 0 mit Bildschirmzeichen zusammenstößt, findet man in Register 53279 den Wert 1. Die Berechnung des zu erwartenden Wertes kann mit dem Bild 4 geschehen, wobei für mehrere Sprites die Zahlen zu addieren sind. Im Beispielprogramm wird in den Zeilen 530 und 540 das Sprite/Sprite-Kollisionsregister 53278 auf Zusammenstöße von Sprite 0 mit Blumen (Sprite 1 und Sprite 3) abgefragt. Bei einer Kollision werden im Register 53278 die Werte 3 oder 9 erzeugt.

Das Auslesen dieser beiden Register ist ziemlich radikal: Nach dem PEEK-Kommando sind sie gelöscht. Deshalb sollte man die ausgelesenen Werte in einer Variablen speichern (Zeile 530:A). Außerdem kann man die Register durch PEEKen vor der ersten Kollisionsabfrage leeren, was wir in Zeile 460 machen.

Unser Beispielprogramm (Listing 1) ist so aufgebaut, daß wir mit den Cursortasten das Sprite 0 steuern können. Jedesmal, wenn dabei eine Blume überfahren wird, ändert sich die Autofarbe. Um das Programm kurz und übersichtlich zu gestalten, finden Sie keine Sicherung gegen das Heraussteuern aus dem zulässigen Sprite-Koordinaten-Bereich. Durch Eingabe von ← kann das Programm beendet werden.

Damit wissen Sie alles, was Sie zur Programmierung von Sprites brauchen. Probieren Sie mal ein wenig durch Veränderungen am Beispielprogramm herum. Falls Sie nun Blut geleckt haben, können Sie Ihr Wissen noch vertiefen durch die nachfolgend genannte Literatur:

— »Reise durch das Wunderland der Grafik«, 64'er, Ausgabe 8 (1984) Seite 142 ff. (Erscheint in diesen Tagen auch als Buch im Verlag Markt & Technik).

— H. Kunz hat etwas über das schnelle Bewegen von Sprites geschrieben im 64'er, Ausgabe 4 (1984) Seite 70 ff.

— Eine nette Anwendung stammt von H. Grigat in Happy-Computer, Ausgabe 11 (1983), Seite 99 ff.

— Einen Überblick geben Schneider und Ebert in den Bänden 1 und 3 des Commodore 64-Buches, erschienen im Verlag Markt & Technik (1984).

— Ebenfalls zu empfehlen ist S. Krutes »Grafik und Musik auf dem Commodore 64«, welches ebenfalls bei Markt & Technik erschien.

— Für alle, die es ganz genau wissen wollen (bis in die weiten Ebenen der Elektronik), empfehle ich schließlich noch den Aufsatz von P. Dornier »Sprites ohne Esoterik«, 64'er-Ausgabe 11 (1984), Seite 74 ff.

(Heimo Ponnath/ah)

```
1 REM ********************************    <132>
2 REM *                              *    <051>
3 REM *  BEISPIELPROGRAMM ZU SPRITES  *    <014>
4 REM *                              *    <053>
5 REM * HEIMO PONNATH   HAMBURG  1985 *    <130>
6 REM *                              *    <055>
7 REM ********************************    <138>
8 REM                                      <070>
45 REM **** BILDSCHIRMFARBEN ****          <156>
50 POKE 53280,0:POKE 53281,0:POKE 646,14   <072>
95 REM **** EINLESEN SPRITE-DATEN ****     <152>
100 FOR I=704 TO 766:READ D:POKE I,D:NEXT
    I                                      <135>
110 FOR I=832 TO 894:READ D:POKE I,D:NEXT
    I                                      <111>
115 REM ****** DATAS VON SPRITE 1 ******   <199>
120 DATA 0,0,0,0,0,0,3,255,192,1,3,128     <222>
130 DATA 1,51,128,1,115,128,1,59,192       <058>
140 DATA 129,186,254,255,255,255,129,2,1   <195>
150 DATA 129,2,1,253,254,127,222,0,157     <225>
160 DATA 255,255,190,62,0,62,28,0,28       <070>
170 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0     <028>
175 REM ****** DATAS VON SPRITE 2 ******   <132>
180 DATA 128,8,0,160,40,0,168,168,0        <118>
190 DATA 40,168,0,42,170,0,10,234,160      <185>
200 DATA 43,250,128,43,250,0,170,232,0     <164>
210 DATA 170,168,0,165,168,0,129,90,0      <174>
220 DATA 129,86,0,0,80,128,0,80,0          <143>
230 DATA 0,16,20,0,16,85,0,17,84,0,21,0    <087>
240 DATA 0,16,0,0,16,0                     <175>
245 REM **** SPRITE-ZEIGER ****            <142>
250 POKE 2040,11:POKE 2041,13              <243>
255 REM **** EINSCHALTEN ****              <151>
260 POKE 53269,PEEK(53269)OR(2↑0):POKE 532
    69,PEEK(53269)OR(2↑1)                  <045>
265 REM **** SPRITE-POSITIONEN ****        <079>
270 FOR N=0 TO 1:PRINT"SPRITE"N;:INPUT"X,Y
    =";X(N),Y(N)                           <120>
280 POKE 53249+2*N,Y(N)                    <188>
290 IF X(N)>255 THEN POKE 53248+2*N,X(N)-2
    56:POKE 53264,PEEK(53264)OR(2↑N):GOTO
    305                                    <102>
300 POKE 53248+2*N,X(N):POKE 53264,PEEK(53
    264)AND(255-2↑N)                       <136>
305 IF K=1 THEN RETURN                     <060>
310 NEXT N                                 <180>
315 REM **** SPRITE-FARBEN ****            <174>
320 POKE 53287,3:REM SPRITE 0 =CYAN        <053>
330 POKE 53285,5:REM SPRITE 1 KENNUNG 01=G
    RUEN                                   <058>
340 POKE 53286,7:REM KENNUNG 11=GELB       <167>
350 POKE 53288,2:REM KENNUNG 10=ROT        <228>
360 POKE 53276,PEEK(53276)OR(2↑1):REM MULT
    ICOLORMODUS EINGESCHALTET              <208>
365 REM *** SPRITE-GROESSEN ***            <095>
367 REM +++++ 3 WEITERE SPRITES +++++      <067>
370 POKE 2042,11:POKE 2043,13:POKE 2044,11
    :REM SPRITEZEIGER AUF VORHANDENE DATEN <149>
380 POKE 53269,PEEK(53269)OR(2↑2):POKE 532
    69,PEEK(53269)OR(2↑3)                  <245>
385 POKE 53269,PEEK(53269)OR(2↑4):REM ANSC
    HALTEN                                 <008>
390 POKE 53248+2*2,100:POKE 53249+2*2,100:
    POKE 53248+2*3,80:POKE 53249+2*3,200   <018>
395 POKE 53248+2*4,150:POKE 53249+2*4,150:
    POKE 53291,4                           <224>
400 POKE 53289,1:POKE 53290,6:POKE 53276,P
    EEK(53276)OR(2↑3):REM POSITIONEN+FARBE
    N                                      <170>
405 REM +++++ DIESE VERGROESSERN +++++     <236>
410 POKE 53277,PEEK(53277)OR(2↑2):REM SPRI
    TE 2 IN X-RICHTUNG VERDOPPELN          <077>
420 POKE 53271,PEEK(53271)OR(2↑3):REM SPRI
    TE 3 IN Y-RICHTUNG VERDOPPELN          <245>
430 POKE 53271,PEEK(53271)OR(2↑4):POKE 532
    77,PEEK(53277)OR(2↑4):REM SPRITE 4 X+Y <241>
435 REM **** VORFAHRT-REGELUNG *****       <079>
440 POKE 53275,PEEK(53275)OR(2↑0):POKE 532
    75,PEEK(53275)OR(2↑2)                  <131>
450 POKE 53275,PEEK(53275)OR(2↑4):REM ALLE
     AUTOS HINTER BILDSCHIRMZEICHEN        <245>
455 REM *** KOLLISIONEN VORBEREITEN ***    <193>
460 A=PEEK(53278):A=PEEK(53279)            <243>
465 REM *** STEUERN VON SPRITE 0 ***       <151>
470 PRINT CHR$(147):N=0                    <058>
480 GET A$:IF A$=""THEN 480                <197>
490 IF A$=CHR$(17)THEN POKE 53249,PEEK(532
    49)+3:REM ABWAERTS                     <252>
500 IF A$=CHR$(145)THEN POKE 53249,PEEK(53
    249)-3:REM AUFWAERTS                   <144>
510 IF A$=CHR$(157)THEN X(N)=X(N)-3:K=1:GO
    SUB 290:K=0:REM LINKS                  <011>
520 IF A$=CHR$(29)THEN X(N)=X(N)+3:K=1:GOS
    UB 290:K=0:REM RECHTS                  <199>
525 REM *** SPRITE/SPRITE-KOLLISION ***    <022>
530 A=PEEK(53278)                          <144>
540 IF A=3 OR A=9 THEN GOSUB 600:REM KOLLI
    SION MIT BLUMEN ?                      <091>
545 REM *** ENDE ABFRAGESCHLEIFE ***       <070>
550 IF A$="←"THEN END:REM PROGRAMMENDE     <130>
560 GOTO 480:REM ERNEUTE ABFRAGE           <050>
565 REM *** KOLLISIONSFOLGE ***            <011>
600 F=PEEK(53287)+1:IF F>255 THEN F=0:REM
    FARBCODE SPRITE 0 ERHOEHEN             <089>
610 POKE 53287,F:RETURN:REM IN FARBREGISTE
    R VON SPRITE 0                         <086>
```

**Listing 1. So können Sprites komplett programmiert werden.**

**Beachten Sie die Eingabehinweise zum neuen Checksummer auf Seite 53.**