Pascal uitgediept
De muis onder controle

MSX Computer & Club Magazine nummer 63 - november 1993

Herman Post

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

De Pascalprogrammeur —en die niet alleen— zit vaak met het probleem dat hij in een programma een muis of joystick wil laten gebruiken. Een probleem is echter dat die soms achteraf ingestoken of verwisseld worden. Hier de oplossing en ook een jANSl-tip.
Ik beloofde vorige keer routines die algemeen bruikbaar zijn. Ik wil deze keer een routine behandelen voor het opvragen van de joystick, cursortoetsen en de muis. Op zich natuurlijk erg eenvoudige handelingen omdat in verschillende bibliotheken routines als PAD, STICK en STRIG algemeen aanwezig zijn. Het voordeel van de hier besproken routine is dat er, ongeacht het aantal joysticks/ muizen, altijd een goede waarde terugkomt. Ook het achteraf erbij pluggen van een muis levert geen problemen op. Vervangen van een joystick door een muis of omgekeerd gaat ook goed. Het enige nadeel is dat er niet voorzien is in de aansturing van twee spelers, maar als u deze routine begrijpt, schrijft u dat natuurlijk zelf even. Eerst de problemen die zich voordoen bij het schrijven van deze routine. U hoeft deze niet te kennen om met de routine te werken, maar het geeft een aardig inzicht in de truc die gebruikt wordt om de routine werkend te krijgen.

Ei van Columbus

Als u met een STICK-routine de joystick opvraagt gaat dat altijd goed. De waarde die een stick terug geeft is de waarde die de joystick aanwijst. Bij een muis ligt dat anders. Als u de waarde van een muis opvraagt en er is een joystick aangesloten dan kunt u voor de X- en Y-richting een niet gedefinieerde waarde krijgen. De waarde voor X en Y zijn in dat geval echter altijd gelijk. Hierop is dan ook de routine gebaseerd. We gaan er van uit dat een muis bijna niet exact onder een hoek van 45 graden bewogen kan worden. Gebeurt dat toch dan zouden de X- en Y-waarden gelijk zijn. Als we nu bij het opvragen van de muis een X-Y waarden tegenkomen die precies gelijk zijn, gaan we er vanuit dat deze veroorzaakt worden door een aangesloten joystick en laten we de muis niet reageren, maar vragen de bijbehorende joystick op.
Om de routine sneller te krijgen zijn er nog wat zaken verondersteld. Er wordt van uitgegaan, dat maar één persoon probeert alles te bedienen. Op het moment dat de cursor wordt bewogen worden de joystick's/muizen niet meer bekeken. Dit levert een aardige snelheidswinst op omdat vooral de PAD vrij traag is. Als de cursor niet beweegt/ worden wel de twee joystick poorten bekeken. U kunt dit controleren door twee muizen of twee joysticks aan te sluiten en deze elk een andere richting in te bewegen. Daar staat tegenover dat bijvoorbeeld twee joysticks in dezelfde richting elkaar versterken. De

routine zelf is nogal universeel van opzet. Zoals de routine hier is afgebeeld is hij geschikt voor scherm 5 en 8. Omdat de coördinaten die gebruikt worden liggen in het gebied 0..255. Wilt u dit veranderen voor scherm 6 en 7, dan zult u hier en daar byte7 s moeten vervangen door integers. De waarden waarbinnen alles moet blijven, staan in de constanten 'startx', 'starty', 'eindx' en 'eindy'. Deze vindt u in de subprocedure 'nieuw'. Wilt u bijvoorbeeld een toepassing maken waarbij de grenzen iedere keer anders zijn, dan kunt u ze in de aanroep opnemen en hoeft u alleen de constanten te verwijderen. U moet dan natuurlijk wel bij iedere aanroep de waarden meegeven.
De stappen waarmee de cursor of joystick beweegt staan opgeslagen in de constante array's 'plusx' en 'plusy'. Voor alle negen richtingen (acht richtingen plus stilstaan) staat daar de verschuiving in de x- en y-richting vermeld. Merk op dat hier wel integers gebruikt zijn om ook negatieve waarde op te kunnen slaan. Met een trucje zouden dit ook byte's kunnen zijn, maar dat maakt de routine onnodig moeilijker en trager.
De variabelen 'mx' en 'my' worden gebruikt voor de tijdelijke opslag van de X- en Y-richting. Bij gebruik van cursor of joystick zal dit dus altijd -1/0 of l zijn/ maar bij een muis kan dit natuurlijk veel groter zijn. De subprocedure 'nieuw' telt de gelezen X-Y waarde bij de huidige positie op. Dit ziet er vrij lastig uit maar de berekening zorgt ervoor dat de grenswaarde niet worden overschreven. Dit zou natuurlijk ook in een IF. . THEN. . ELSE-structuur kunnen, maar de hier gekozen berekening is sneller. Als alleen de cursortoetsen en joystick gebruikt zouden worden maakt dat de berekening nog veel eenvoudiger omdat dan niet 'over de grens heen geschoten' kan worden. Let er op dat de variabelen 'mx' en 'my' hier gebruikt worden alsof ze globaal gedefinieerd zijn. Ze worden hier zogenaamd lokaal-globaal gebruikt. Dit wil zeggen dat ze ten opzichte van 'sturing' globaal zijn, maar lokaal ten opzichte van het hoofd-programma. Het voordeel is dat niet bij iedere aanroep van 'nieuw' deze twee variabelen hoeven te worden meegestuurd.
De variabele 'stok' wordt alleen gebruikt om de waarde van de joystick op te vragen. Hierdoor hoeft de joystick per poort maar één keer te worden opgevraagd en werken we verder met die waarde. Zouden we schrijven:

```
mx:=plusx[Stick(l)];
my:=plusy[Stick(l)];
```

dan werkt alles wel, maar kunnen er kleine fouten ontstaan als de joystick precies tussen deze twee statements wordt losgelaten. Nu komen we bij het opvragen van de muis. Hier wordt wel de meest smerige truc uit de routine gebruikt. Zoals al eerder vermeld is de PAD-routine nogal traag. Om echter de waarde van de muis in poort één op te vragen vermeldt het handboek de volgende handeling:
1:PAD(12)initialiseren. 2: PAD(13) lees X-richting. 3: PAD(14) lees Y-richting. Na het lezen van de documentatie van de sub-ROM blijkt, dat al bij het initiali-seren de volledige muis wordt opgevraagd. De X- en Y-waarden worden ingelezen en opgeslagen in twee systeem variabelen. Deze systeemvariabe-len staan in het gebied, dat al direct toegankelijk is voor Pascal. We kunnen deze waarden dus ook uitlezen zonder PAD (13) en PAD (14) te gebruiken. Dit gebeurt met de beide variabelen 'xsave' en 'ysave', originele ASCII-bena-ming. Door deze variabelen op een absoluut adres te zetten/ wijzen ze meteen de juiste variabelen aan. Blijft nog het probleem over dat de waarde die daar

staat loopt van O...255 en voor de muis willen we -l 26... 127 krijgen. Dit zou niet zo moeilijk zijn als dit kon door 'xsave-126'. Het dient echter zo omgerekend te worden dat O... 127 blijft 0...127 en 128...255 wordt -126...0. Dit krijgen we voor elkaar met de regel:

IF xsave>127 THEN mx:=xsave-256 ELSE mx:-xsave;

Blijft nog over de regel:

mx:=PAD(12+poort SHL 2);

Hier wordt berekend of PAD (12) of PAD (16) wordt gebruikt. SHL staat voor 'Shift Horizontal Leff, dus schuift alle bits naar links. Dit is een snelle manier om te vermenigvuldigen met inte-gers of byte's. SHL x schuift alle bits x positiefs naar links. Dit komt neer op vermenigvuldigen met twee^x. Ofwel SHL l = * 2, SHL 2 = *4, SHL 3= *8,SHL 4 = *'16, enz. Wel is hier de waarschuwing op zijn plaats, dat bits, die niet meer in de byte of in de integer passen, zonder meer verloren zijn.

Nu een opmerking over het uitlezen van de vuurknoppen. Op een muis of joystick kunnen twee vuurknoppen aanwezig zijn. Op het toetsenbord is echter alleen de spatie als vuurknop aanwezig. Om dat probleem op te vangen gebruik ik—net als Konami—de 'N' als tweede vuurknop. Het uitlezen van de toetsen met READ (KBD, k) werkt uitstekend echter..... de standaard functie KEYPRESSED werkt niet goed. Dit komt door een bug in Turbo Pascal. Dezelfde bug die er voor zorgt dat 'CTRL-C niet goed functioneert. Ik heb hier niet direct een eenvoudige oplossing voor. Bent u in staat om deze bug te omzeilen laat het dan even weten.

De manier die ik hier gebruik om de vuurknoppen uit te lezen is in het gebruik echter zeer handig. Niet meer overal op de afzonderlijke knoppen testen/ en besluit u halverwege uw programma andere toetsen te gebruiken/ dan hoeft u niet alles af te zoeken naar de plaats waar ze gedefinieerd staan. Op de disk bij dit magazine staat nog een tweede versie van deze routine. Deze is volledig als muis/joystick-test geschreven en u kunt hiermee precies zien hoe alles op elkaar reageert. Beweeg de muis en de cursor hiervoor wel gelijktijdig.

Samen met BASIC

Rest mij nog één laatste opmerking. Vlak voor het ter perse gaan werd mij gevraagd of het niet mogelijk was om deze routine zo te compileren dat hij ook in BASIC als hybride routine te gebruiken is. Helaas is dit niet mogelijk. Pascal maakt namelijk gebruik van routines in z'n standaard bibliotheek. Ook de logische operaties en rekenkundige bewerkingen zitten in deze bibliotheek. Bij het compileren komt deze bibliotheek automatisch vanaf adres $0100 te staan waar u het startadres ook neerzet. Vanuit BASIC is dit niet te laden, en zijn dus ook geen Pascal-hybride routines te schrijven. Jammer, maar misschien is er een machinetaalexpert die hiervoor wel een oplossing weet.

```
PROGRAM MUIS;

CONST uit_aan : ARRAY[FALSE..TRUE] OF STRING[3] = ('UIT','AAN');
VAR posx,posy : BYTE;

FUNCTION Escape : BOOLEAN;
VAR k : CHAR;
BEGIN
  k:=' ';
  IF KEYPRESSED THEN READ(KBD,k);
  Escape:=(k=#27)
END;

FUNCTION Stick(id : BYTE) : BYTE;
BEGIN
  INLINE($3A/id/$FD/$2A/$C0/$FC/$DD/$21/
         $D5/$00/$CD/$1C/$00/$32/id/$FB);
  Stick:=id
END;

FUNCTION Strig(id : BYTE) : BOOLEAN;
BEGIN
  INLINE($3A/id/$FD/$2A/$C0/$FC/$DD/$21/
         $D8/$00/$CD/$1C/$00/$32/id/$FB);
  Strig:=(id<>0)
END;

FUNCTION Knop1 : BOOLEAN;
BEGIN
  Knop1:=Strig(0) OR Strig(1) OR Strig(2)
END;

FUNCTION Knop2 : BOOLEAN;
VAR k : CHAR;
BEGIN
  k:=' ';
  IF KEYPRESSED THEN READ(KBD,k);
  Knop2:=(UPCASE(k)='N') OR Strig(3) OR Strig(4)
END;

FUNCTION Pad(id : BYTE) : BYTE;
VAR a : BYTE;
BEGIN
  INLINE($FD/$2A/$C0/$FC/
    $DD/$21/$DB/$00/ $3A/id/
    $CD/$1C/$00/ $32/id);
  Pad:=id
END;

PROCEDURE Besturing(VAR x,y : BYTE);

CONST plusx : ARRAY [0..8] OF INTEGER = (0,0,1,1,1,0,-1,-1,-1);
      plusy : ARRAY [0..8] OF INTEGER = (0,-1,-1,0,1,1,1,0,-1);
VAR xsave : BYTE ABSOLUTE $FAFE;
    ysave : BYTE ABSOLUTE $FB00;
    mx,my : INTEGER;
```

```
    stok,poort : BYTE;

  PROCEDURE nieuw;
  CONST startx : BYTE=0;
        starty : BYTE=0;
        eindx  : BYTE=255;
        eindy  : BYTE=211;
  BEGIN
    x:=x+mx-ORD(((x+mx)<startx) OR ((x+mx)>eindx))*mx;
    y:=y+my-ORD(((y+my)<starty) OR ((y+my)>eindy))*my
  END;

BEGIN
  stok:=Stick(0);
  IF stok<>0 THEN
  BEGIN
    mx:=plusx[stok];
    my:=plusy[stok];
    nieuw
  END ELSE
  FOR poort:=0 TO 1 DO
  BEGIN
    mx:=Pad(12+poort SHL 2);
    IF xsave<>ysave THEN
    BEGIN
      IF xsave>127 THEN mx:=xsave-256 ELSE mx:=xsave;
      IF ysave>127 THEN my:=ysave-256 ELSE my:=ysave;
      nieuw
    END
    ELSE
    BEGIN
      stok:=stick(poort+1);
      mx:=plusx[stok];
      my:=plusy[stok];
      nieuw
    END
  END
END;

BEGIN
  CLRSCR;
  posx:=100;
  posy:=100;
  WRITELN('positie cursor (x,y) : ');
  WRITELN('vuurknop 1 :');
  WRITELN('vuurknop 2 :');
  WRITELN;
  WRITELN('ESCAPE voor einde.');
  REPEAT
    GOTOXY(24,1);WRITE(posx:4,posy:4);
    GOTOXY(14,2);WRITE(uit_aan[Knop1]);
    GOTOXY(14,3);WRITE(uit_aan[Knop2]);
    Besturing(posx,posy)
  UNTIL Escape;
  CLRSCR
END.
```