Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

M.J.A.C. Andreoli

Taalprimitiva in *B* voor grafisch editen
Een verkenning

Afdeling Informatica/Algoritmiek & Architectuur Notitie CS-N8509 September

---

Language primitives in B for graphical editing

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

Taalprimitiva in *B* voor grafisch editen

Een verkenning

M.J.A.C. Andreoli
Technische Hogeschool Delft,
Julianalaan 132, 2628 BL Delft, The Netherlands.

In dit rapport worden mogelijkheden onderzocht commando's aan een taal als *B* toe te voegen om "grafisch editen" (interactieve gegevensinvoer via een grafische interface) op een eenvoudige manier mogelijk te maken. 69K34,

1. Inleiding.

In dit rapport probeer ik hulpmiddelen te ontwerpen voor grafisch editen die aansluiten bij de programmeertaal *B*.

Alhoewel het niet de bedoeling is een grafisch pakket te ontwikkelen -ik ga er al van uit dat er in *B* zo'n pakket bestaat- , ontkom ik er niet aan toch enkele zaken te bespreken die hier wel op van invloed kunnen zijn.

In de voorbeeldprogramma's heb ik zoveel mogelijk geprobeerd gebruik te maken van *B*.

Verder is in dit verslag sprake van twee soorten mensen :
- "gebruikers" en
- "programmeurs".

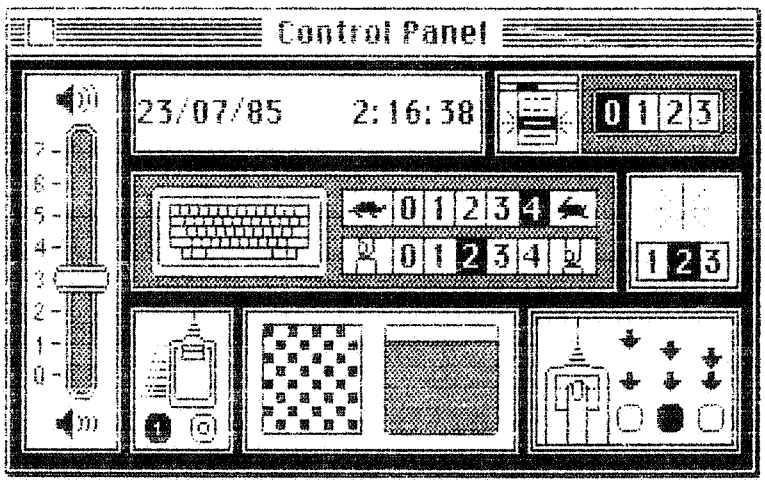
Een programmeur is iemand die een applicatie codeert (dat gebeurt hier in de programmeertaal *B*), vandaar dat de hier besproken hulpmiddelen zoveel mogelijk moeten aansluiten aan *B*.
Een gebruiker ziet alleen de applicatie waarbij hij op één of andere manier invoer moet kunnen geven aan het systeem. Dit moet dan gebeuren op een zo logisch mogelijke wijze (bij voorkeur aansluitend bij het gebruik van andere dagelijkse apparatuur) .

Rapport CS-N8509
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

2. Waarom een grafische editor ?

In het algemeen zijn plaatjes (pictogrammen) eenvoudig te begrijpen.
Bovendien is het vaak eenvoudiger een kleine wijziging in een plaatje aan te geven dan deze wijziging via een toetsenbord in te geven (doorgaans onbegrijpelijke / moeilijke commando's waarbij niet zelden de syntaxis zeer nauw luistert).

Als voorbeeld kan de luidspreker van de Apple Macintosh dienen.
Via een speciaal menu kan een contrôlepaneel gekozen worden met daarop een aantal knoppen en een schuifregelaar (zie figuur 1).
De geluidssterkte kan nu d.m.v. deze schuifregelaar worden ingesteld.



Figuur 1.

Vergelijk dit maar eens met de standaardmanier die op vele andere computers nog steeds geldt : Een (zeer gevaarlijke) poke opdracht, waarbij een gebruiker niet vreemd op hoeft te kijken als de computer daarna totaal onbestuurbaar is geworden omdat hij per ongeluk twee cijfers heeft verwisseld; bovendien moeten voor zò iets simpels al snel één of meer handboeken worden geraadpleegd.

3. Wat wel, wat niet ?

Laten we eerst eens vaststellen wat wel en wat niet de bedoeling is.
Stel het commandodocument bevat de volgende tekst :

PUT 20 IN temperatuur
WRITE temperatuur

dan bevat het uitvoerdocument :

20

Het is nu niet de bedoeling dat 20 in het uitvoerdocument veranderd kan worden (in het commandodocument moet dat natuurlijk wèl mogelijk zijn).
Ook als temperatuur in een grafisch uitvoerdocument gerepresenteerd wordt door bijv. het kwikniveau in een thermometer, dan is het nòg niet de bedoeling dat de temperatuur veranderd kan worden door verplaatsing van het kwikniveau.
Wanneer zou zoiets dan wel mogen, of misschien zelfs wenselijk zijn ?
Het antwoord op deze vraag is in feite simpel :
Als er om invoer van een temperatuur wordt gevraagd, dus :

READ temperatuur

De invoer wordt nu verwacht te komen van het toetsenbord; en wel als een expressie zo nauwkeurig als gewenst is, afgesloten door een druk op de "return" toets.
Als de invoer via een grafisch document (een gecombineerd invoer- / uitvoerdocument) moet lopen, komen er direct een aantal problemen naar voren, zoals : de thermometer moet eerst getekend zijn vòòrdat er een wijziging van temperatuur aangegeven kan worden, hoe zit het met de schaalverdeling, nauwkeurigheid etc.
Bovendien is nog geen rekening gehouden met een ander probleem :
De thermometer kan deel uit maken van een groter geheel (bijv. een vluchtcomputer in een vliegtuig), waarbij er niet alleen een verandering in de temperatuur aangegeven moet kunnen worden, maar evengoed een verandering in hoogte, snelheid etc.; dus een commando als 'READ temperatuur' mag alleen dààr in een programma staan, waar het duidelijk is dat de door de gebruiker ingevoerde waarde bedoeld is voor de temperatuur en niet voor bijv. de hoogte.

De problemen zijn te omvangrijk om ze ineens op te lossen, daarom zullen we beginnen met een eenvoudig probleem en dit langzaamaan uitbreiden tot een model voor een vluchtcomputer.
Tot slot volgt dan nog een beschouwing over de synchronisatie in snelheid tussen mens en machine.

4. Een eenvoudig probleem op de conventionele manier.

Beschouw de volgende applicatie :
De gebruiker moet op één of andere manier door een doolhof.
Hij kan alleen aangeven in welke richting de volgende stap is :
noord, oost, zuid of west.
Het programma kan er als volgt uitzien (de nadruk valt op de mainloop) :

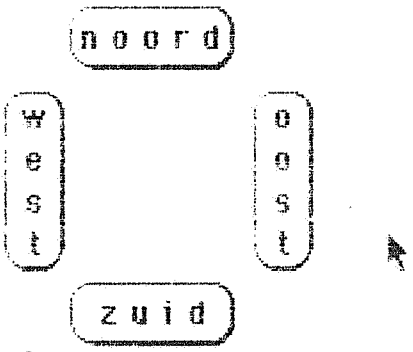
```
HOW'TO DOOLHOF'1 :
   ...
   PUT { "noord" ; "oost" ; "zuid" ; "west" } IN richting
   WHILE uitgang'niet'gevonden :
      READ commando RAW
      SELECT :
         commando in richting :
            BEKIJK'STAP
         ELSE :
            WRITE "Geen geldig commando." /
```

In het subprogramma "BEKIJK'STAP" moet ervoor gezorgd worden dat de juiste acties worden ondernomen (dit is voor ons hier niet van belang) .

Het is niet moeilijk aan dit programma code toe te voegen waardoor de gebruiker ook hoofdletters kan gebruiken of alleen de eerste letter van de gewenste richting in hoeft te toetsen (wèl moet nog door middel van een speciale toets worden aangegeven dat de invoer is beëindigd (de "return" toets)) .
Veel aardiger wordt het wanneer de gebruiker alleen maar de eerste letter van de richting hoeft in te toetsen, of zelfs de richting kan ingeven door gebruik te maken van de pijltoetsen op zijn toetsenbord.
Analoog aan de manier waarop dit in een aantal versies van de taal BASIC gebeurt (het GET-commando), zou er aan *B* een commando toegevoegd kunnen worden : "READ'KEY target" .
Er wordt nu gewacht met de executie van het programma totdat de gebruiker een toets heeft ingedrukt.

5. Het probleem met invoer via een grafisch document.

Laten we nu eens kijken of deze invoer ook mogelijk is via een grafisch document en een positieaangever (muis, joystick, trackerball etc.).
Ik ga er van uit dat het muissymbool altijd zichtbaar is en dat wij ons geen zorgen hoeven te maken over hoe en waar dit symbool wordt afgebeeld.
Ook neem ik aan dat er in *B* de mogelijkheid bestaat om -met behulp van grafische primitieven- figuren te ontwerpen die voor de programmeur verder dienen als "basisfiguren". De programmeur kan bijvoorbeeld een blokje definiëren met de tekst "noord" en verder aan deze figuur refereren zonder zich te bekommeren om de afzonderlijke letters of de lijnen van het blokje.
(de pijl is het muissymbool)



Figuur 2.

De gebruiker kan nu door het bewegen van de muis de punt van de pijl in één van de vier richtingshokjes plaatsen en vervolgens d.m.v. een druk op de knop van de muis te kennen geven dat er een stap in die richting gezet moet worden.

De bijbehorende mainloop van het programma kan er als volgt uitzien :

```
HOW'TO DOOLHOF'2 :
  ...
  PUT { noord(noord'positie) ; oost(oost'positie) ; zuid(zuid'positie) ;
        west(west'positie) } IN richting
  WHILE uitgang'niet'gevonden :
     READ'MOUSE muispositie
     SELECT :
        muispositie inside richting :
           BEKIJK'STAP
        ELSE :
           WRITE "Geen geldig commando." /
```

Nu kan de vraag gesteld worden of het "PUT" commando de samengestelde figuur ook af moet beelden op het scherm; het antwoord hierop is : NEEN.
Bij een normaal "PUT" commando wordt de waarde van de variabele immers ook niet afgebeeld (afgedrukt in een tekst-uitvoerdocument), maar moet dit expliciet gebeuren door een "WRITE" commando; het ligt in de verwachting dat hier iets analoogs aan de hand is, bijvoorbeeld een "DISPLAY" commando om uitvoer in een grafisch-uitvoerdocument zichtbaar te maken.
Het "PUT" commando stelt een nieuwe grafische figuur samen uit de vier eerder gedefinieerde grafische basisfiguren noord, oost, zuid en west waarbij de parameters noord'positie etc. de positie aangeven van de samenstellende primitieven t.o.v. de oorsprong van de nieuwe figuur.
Het type van muispositie, noord'positie etc. is afhankelijk van het coördinatenstelsel waarin gewerkt wordt.
Dit zouden evengoed cartesiaanse als poolcoördinaten kunnen zijn (tweedimensionaal); terwijl voor toepassingen in de CAD/CAM sfeer (driedimensionaal) de ene keer voorkeur bestaat voor cartesiaanse coördinaten (gaten boren in een vlak werkstuk), een andere keer cilinder-coördinaten (draaibank) en er een derde keer misschien zelfs voorkeur bestaat voor bolcoördinaten (robotarm).
Het spreekt voor zich dat in het driedimensionale geval er gewerkt moet worden met een aangepaste "muis" en dat het grafisch pakket de nodige faciliteiten moet bieden ter ondersteuning van driedimensionale toepassingen. Misschien is dit te hoog gegrepen, maar ik wil het niet bij voorbaat uitsluiten.

"READ'MOUSE" onderbreekt de executie van het programma; zodra er op de knop van de muis wordt gedrukt, zullen de coördinaten van de muis worden doorgegeven aan de variabele 'muispositie', de executie van het programma kan verdergaan.

De test "muispositie inside richting" zal true opleveren als in één van de vier samenstellende delen van 'richting' is geklikt. (Op een bepaalde plaats 'klikken' wil zeggen dat de punt van de pijl welke de representatie is van de muis, op die bepaalde plaats wordt gezet en vervolgens éénmaal op de knop van de muis wordt gedrukt.)

6. Een moeilijker probleem.

Bekijk nu het volgende probleem :
Op een contrôlepaneel willen we de temperatuur en de vochtigheid regelen.
Voor de temperatuur kunnen twee commando's gegeven worden :
'hoger' en 'lager'.
De vochtigheid is ook te regelen door twee commando's :
'hoger' en 'lager'.
Eerst wordt de "tekst variant" uitgewerkt, daarna volgt een bespreking van een grafische oplossing.

Het programma kan er als volgt uitzien :

```
HOW'TO KLIMAAT'I :
   ...
   PUT { "hoger" ; "lager" } IN opdrachten
   READ commando RAW
   SELECT :
      commando in opdrachten :
         ONDERNEEM'ACTIE
      ELSE :
         WRITE "Geen geldig commando." /
```

Nu ontstaan er problemen : Voor welk apparaat is het commando bedoeld ?
Dit kan opgelost worden door bijvoorbeeld 'hoger' en 'lager' bij de thermostaat te vervangen door de commando's 'warmer' resp. 'kouder' ; er moet dus voor gezorgd worden dat een commando maar op één manier geïnterpreteerd kan worden, waardoor het aantal commando's behoorlijk toe kan nemen (niet erg gebruikersvriendelijk en bovendien lastig voor de programmeur).
Een andere oplossing kan zijn het opgeven van een prefix, hetzij als apart commando, hetzij als onderdeel van het commando zelf. Deze methode heeft als nadeel dat er toch weer extra werk gedaan moet worden door de gebruiker, bovendien zal er een behoorlijke uitbreiding van code moeten plaatsvinden om alles goed te laten lopen.

7. De grafische aanpak.

Figuur 3.

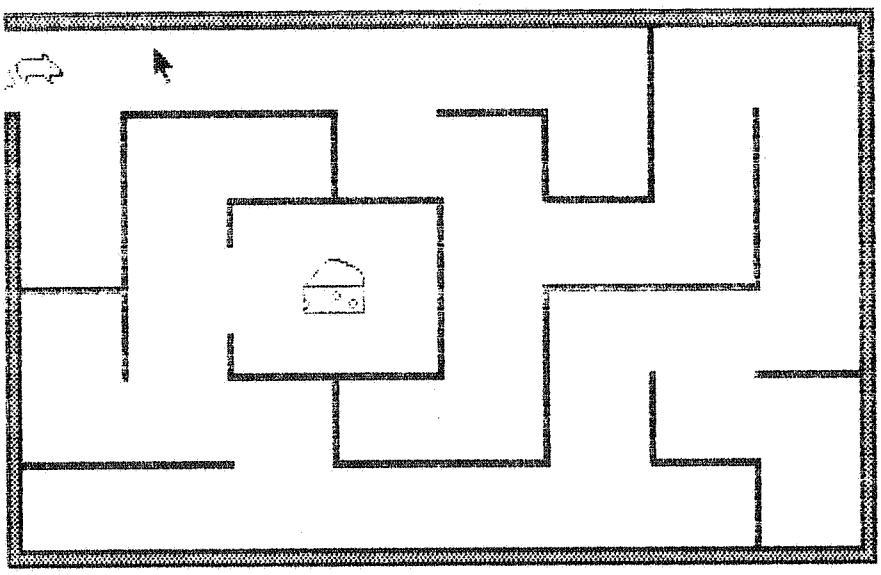
Bij deze oplossing is het overigens zeer eenvoudig (en bovendien het meest logisch) al in de hoofdloop van het programma een onderscheid te maken tussen de temperatuur en de vochtigheidsgraad regeling.

```
HOW'TO KLIMAAT'2 :
   ...
   PUT { hoger( temp'hoger'pos) ; lager( temp'lager'pos) } IN temperatuur
   DISPLAY temperatuur
   PUT { hoger( vocht'hoger'pos) ; lager( vocht'lager'pos) } IN vochtigheid
   DISPLAY vochtigheid
   READ'MOUSE muispositie
   SELECT :
      muispositie inside temperatuur :
         WIJZIG'TEMPERATUUR
      muispositie inside vochtigheid :
         WIJZIG'VOCHTIGHEID
      ELSE :
         WRITE "Geen geldig commando." /
```

Het is duidelijk dat dit programma zeer eenvoudig is uit te breiden met andere instrumenten zoals een hoogtemeter, kompas etc.
Bij het programma "KLIMAAT'1" daarentegen zal uitbreiding moeilijker en foutgevoeliger zijn, ondanks het feit dat er een eenvoudig te begrijpen programmeertaal is gebruikt.

8. Het veranderen in een plaatje.

Wat is er aan de hand als we geen gebruik willen maken van de 'knoppen' maar bijvoorbeeld direct het kwikniveau in de thermometer willen veranderen ?
Laten we eerst weer aan de hand van een doolhof bekijken wat er allemaal geregeld moet worden en welke problemen daarbij optreden :



Figuur 4.

Om verwarring te voorkomen : de pijl in de figuur is de representatie van het "stuurkastje" (de muis, joystick) .
In de onderstaande bespreking wordt met "muisje" het getekende diertje bedoeld dat door het doolhof moet worden geloodst.

Het eerste probleem :
Hoe groot zijn de stappen die het muisje kan zetten ?

Is de stapgrootte :
- één punt (voor een quasi-continue beweging)
- in de ordegrootte van het muisje
- een sprong vanaf de huidige plaats tot dààr waar de neus van het muisje de pijl raakt
  (natuurlijk alleen als er geen muur tussen de pijl en het muisje staat!)

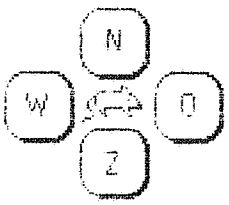
De eerste mogelijkheid is bij deze applicatie zeker niet gewenst (in dit kleine doolhof zijn dan toch al tegen de duizend stappen nodig!) ; als echter de temperatuur ingesteld moet worden op een thermostaat, is het juist wèl wenselijk dat er nauwkeurig gewerkt kan worden, dus deze mogelijkheid kan niet zomaar overboord gezet worden.

De tweede mogelijkheid is heel redelijk, maar waar en hoe moet deze stapgrootte gedefiniëerd worden? Misschien is het mogelijk twee soorten graphics te hebben : de ene soort kan overal op het scherm staan, de andere soort kan alleen op bepaalde roosterpunten verschijnen (bijv. de oorsprong van deze figuren mag alleen coördinaten hebben die een veelvoud van tien zijn).

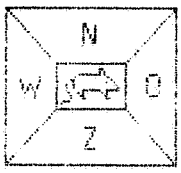
De laatste mogelijkheid van de drie lijkt erg aantrekkelijk, maar hoe moet dan gecontroleerd worden dat de weg tussen het muisje en de pijl geen muur bevat? Het is onredelijk hiervoor een apart commando of een aparte functie in te voeren; dit kan gecontroleerd worden door het muisje in een loop telkens een klein stapje te laten doen en dan te controleren of er geen muur wordt gesneden (moet door de programmeur gecodeerd worden!). Dit laatste is ook nodig bij de vorige oplossing als de stapgrootte groter is dan de afmetingen van het voorwerp.

Wat de beste oplossing is voor dit probleem is niet duidelijk, het is afhankelijk van de soort applicatie en het lijkt vooralsnog het beste dit over te laten aan de programmeur.

Volgend probleem :
Wat gebeurt er als er bijv. in zuidoostelijke richting wordt geklikt ?
Dit probleem is eigenlijk zeer simpel op te lossen :
Op één of andere manier moet de programmeur toch achter de bedoelde richting zien te komen, waarom dan niet gelijk in de programmeertaal de mogelijkheid van "onzichtbare" graphics ingebouwd ?



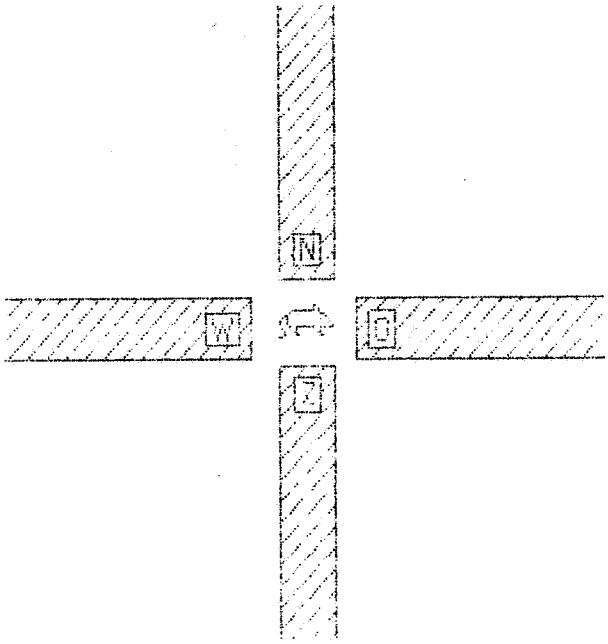
figuur 5a.



figuur 5b.

Door deze knoppen met bijvoorbeeld het "PUT" commando aan het muisje te verbinden, kunnen de knoppen op zeer eenvoudige wijze met het muisje mee bewegen door het doolhof; terwijl door het "DISPLAY" commando alleen te laten werken op het muisje (i.p.v. op de totale figuur) de knoppen onzichtbaar blijven.
De programmeur kan zelf bepalen wat voor soort knoppen bij zijn applicatie de voorkeur verdienen, zodat klikken in zuidoostelijke richting geen resultaat geeft (figuur 5a) of een stap aangeeft in zuidelijke of oostelijke richting (figuur 5b).

Eventueel is zelfs nog een bijzonder soort knop mogelijk :



Figuur 5c.

Knoppen die doorlopen tot de randen van het scherm (of het grafisch venster).
Dit is juist voor die applicatie waar het mogelijk moet zijn dat het muisje grote sprongen kan maken.

9. De vluchtcomputer en de "traagheid" van de gebruiker.

Keren we nu terug naar onze vluchtcomputer :
Bij het begin van een vlucht kan bijvoorbeeld het "READ'MOUSE" commando worden gebruikt om een aantal gegevens in te voeren, tijdens de vlucht is het zonde om de computer te laten "hangen" bij een "READ'MOUSE" om eventuele veranderingen in deze gegevens mogelijk te maken.
Het is beter een "POLL'MOUSE target" te gebruiken, een procedure die de huidige muispositie doorgeeft aan 'target' en de executie van het programma niet onnodig ophoudt. Er wordt dus niet gewacht met het doorgeven van de muispositie tot de knop op de muis wordt ingedrukt.

Het programma (tijdens de vlucht) kan er dan als volgt uitzien :

```
HOW'TO VLIEG :
   ...
   WHILE onderweg :
      POLL'MOUSE muispositie
      SELECT :
         muispositie inside temperatuur :
            WIJZIG'TEMPERATUUR
         muispositie inside vochtigheid :
            WIJZIG'VOCHTIGHEID
         muispositie inside andere'meter
            WIJZIG'IETS'ANDERS
         ELSE :
            DOE'NIETS
      CONTROLEER'EN'STUUR'BIJ
      DOE'ANDERE'TAKEN
```

Als de muis niet in één van de "knoppen" staat, is er niets aan de hand en moet de werkelijke waarde van de temperatuur, vochtigheid etc. via een meetinstrument aan de computer worden doorgegeven, waarna de computer deze waarden kan vergelijken met de opgegeven waarden en eventueel maatregelen treffen. Dit gebeurt allemaal in het deel "CONTROLEER'EN' STUUR'BIJ".

Dan moeten de delen "WIJZIG'... " nader bekeken worden.
De eerste oplossing die ons invalt is de volgende :

```
HOW'TO WIJZIG'TEMPERATUUR'1 :
   READ'MOUSE muispositie
   BEREKEN'NIEUWE'TEMPERATUUR
```

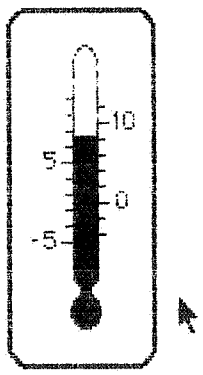
Een "READ'MOUSE" is nodig om de gebruiker de tijd te geven de nieuwe waarde in te stellen.

Aan deze oplossing kleven nog een aantal nadelen :
- de computer ligt door de "READ'MOUSE" lam totdat er op de knop van de muis wordt gedrukt, als het lang duurt voordat de gebruiker de nieuwe waarde heeft ingevoerd, kan het vliegtuig inmiddels al wel zijn neergestort.
- de muis kan per ongeluk in de thermometer terecht zijn gekomen, de gebruiker moet echter een nieuwe temperatuur opgeven om hier weer uit te komen; vooral als de gebruiker dit niet in de gaten heeft, kan het dus lang duren voordat dit gebeurt.

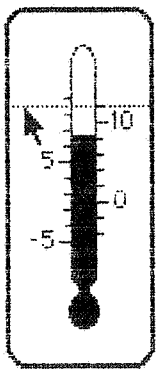
Mijn voorstel is om een predicaat "button" in te voeren, welke de waarde 'true' oplevert als de knop van de muis is ingedrukt en de waarde 'false' als dit niet zo is.
Het programmadeel kan dan als volgt herschreven worden :

```
HOW'TO WIJZIG'TEMPERATUUR'2 :
   WHILE 1 = 1 :
      POLL'MOUSE muispositie
      SELECT :
         button :
            BEREKEN'NIEUWE'TEMPERATUUR
            QUIT
         muispositie inside thermometer :
            CONTROLEER'EN'STUUR'BIJ
         ELSE :
            QUIT
```

Juist omdat de computer "door blijft lopen" is er nog iets leuks mogelijk : Het instellen van de temperatuur op zo'n thermometer kan wel eens lastig zijn; door nu een hulplijntje te tekenen kan invoer van gegevens voor de gebruiker veel eenvoudiger worden gemaakt :



Figuur 6a.



Figuur 6b.

Zodra de muis het "thermometergebied" binnengaat, wordt er een hulplijn getekend.
Dit kan als volgt geprogrammeerd worden :

```
HOW'TO WIJZIG'TEMPERATUUR'3 :
   WHILE 1 = 1 :
      POLL'MOUSE muispositie
      SELECT :
         button :
            BEREKEN'NIEUWE'TEMPERATUUR
            QUIT
         muispositie inside thermometer :
            TEKEN'HULPLIJN
            CONTROLEER'EN'STUUR'BIJ
         ELSE :
            QUIT
```

Zolang de pijl zich in het thermometergebied bevindt, zullen eventuele andere taken niet uitgevoerd worden. Dit is misschien niet de bedoeling, zodat "DOE'ANDERE'TAKEN" ook uitgevoerd moet worden in het selectiedeel "muispositie inside thermometer".
Al met al wordt het er niet mooier op, het programma "VLIEG" kan beter als volgt herschreven worden :

```
HOW'TO VLIEG'ANDERS :
   ...
   WHILE onderweg :
      POLL'MOUSE muispositie
      SELECT :
         muispositie inside temperatuur :
            TEKEN'HULPLIJN
            IF button :
               BEREKEN'NIEUWE'TEMPERATUUR
         muispositie inside vochtigheid :
            TEKEN'ANDERE'HULPLIJN
            IF button :
               BEREKEN'NIEUWE'VOCHTIGHEID
         muispositie inside andere'meter
            DOE'IETS'ANDERS
         ELSE :
            DOE'NIETS
      CONTROLEER'EN'STUUR'BIJ
      DOE'ANDERE'TAKEN
```

10. Bij een "trage" machine: een interruptmechanisme voor kliks.

Op bovenstaande wijze is de "traagheid" van de gebruiker redelijk bevredigend opgevangen, maar hoe zit het nu met de traagheid van de computer ?
Immers de delen "CONTROLEER'EN'STUUR'BIJ" en "DOE'ANDERE'TAKEN" kunnen veel tijd in beslag nemen, terwijl de gebruiker intussen op een knop heeft gedrukt die geen "hulplijn" ondersteuning nodig heeft (bijvoorbeeld de aan/uit knop van een radio).
Een druk op de knop van de muis zou bijvoorbeeld een interrupt kunnen genereren, waarna deze "muisklik" door een interrupt-afhandelings-mechanisme wordt afgehandeld. Helaas bevinden interrupts en de bijbehorende mechanismen zich dicht op de onderliggende machine, zodat deze mogelijkheid direct afvalt, tenzij...

Is er een mogelijkheid interrupts op een hoger niveau te brengen ?
Bijvoorbeeld door aan het begin van een programma de volgende commando's op te nemen :

```
WHEN button :
   KNOP'AFHANDELING
```

In eerste instantie zal alleen op een bepaalde plaats in de computer moeten worden vastgelegd (een vlag) dat er een "button interrupt" kan komen.
De werkelijke code van "KNOP'AFHANDELING" zal pas worden uitgevoerd op het moment dat er op de knop van de muis wordt gedrukt.
Een eigenschap van interrupts is dat ze regelmatig op de meest ongelegen momenten komen, net wanneer de computer aan een tijdcritische taak bezig is; het is daarom verstandig ook commando's te hebben als :

"BUTTON'INTERRUPT'OFF"
en
"BUTTON'INTERRUPT'ON".

Of misschien zijn alleen algemene commando's als

"INTERRUPTS'OFF"
en
"INTERRUPTS'ON"

al voldoende voor de doelgroep *B* programmeurs.
Als er twee versies van *B* komen, zal ik zo'n interrupt mogelijkheid alleen in willen bouwen in de versie voor professionele programmeurs.

11. Bij een "trage" machine: een buffermechanisme voor kliks.

Een andere mogelijkheid is het bufferen van buttonacties (dit zal overigens ook moeten gebeuren bij de interruptvariant als de interrupts disabled zijn).
Bij dit bufferen moet dan ook de positie van de muis op het moment van klikken worden opgeslagen, er mag niet geëist worden dat de muis op het moment van afhandeling op precies dezelfde plaats staat als tijdens het klikken.

Bij de interruptvariant zal afhandeling plaatsvinden zodra interrups weer worden toegelaten, dit blijft doorgaan tot er geen interrupts meer in de wachtrij staan.

Bij de niet-interruptvariant zal een buttonactie net zo lang in de wachtrij blijven staan, tot er een test "button" plaatsvindt, waarbij dan de eerste buttonactie uit de wachtrij wordt behandeld. Eventuele meerdere buttonacties blijven in de wachtrij.

In het bovenstaande is alleen gesproken over het bufferen van een klik; maar een andere veelgebruikte muistechniek is "dragging".
Draggen is de knop van de muis op een bepaald punt indrukken, de muis verplaatsen (met de knop ingedrukt) en op een ander punt de knop loslaten.

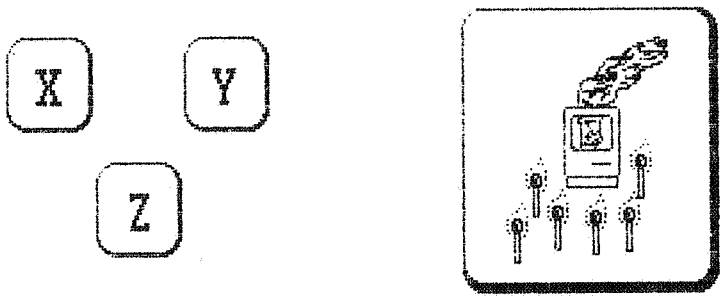
Als op deze wijze een rechte lijn kan worden getekend, is het voldoende om alleen het begin- en eindpunt te bufferen; voor een willekeurige kromme is het noodzakelijk dat ook alle tussenliggende punten opgeslagen moeten worden.
Dit laatste lijkt mij een beetje teveel van het goede, het is beter om alleen een klik eventueel te bufferen en draggen alleen toe te laten onder directe contrôle van de computer.

12. Een addertje onder het gras bij gebufferde kliks.

Met het afhandelen van gebufferde buttonacties kan er nog iets fout gaan, beschouw hiervoor het volgende voorbeeld :

Figuur 7a.

Stel dat er gekozen kan worden uit de bediening van een benzinepomp en een radio door op de gewenste knop te drukken.
Nadat de keuze is gemaakt verschijnt er een commandoscherm met voor dàt apparaat relevante knoppen (voor de radio bijvoorbeeld een aan/uit knop, een schuifregelaar voor de geluidssterkte, een afstemknop etc. ; voor de benzinepomp een aantal knoppen voor de bediening (X, Y en Z) plus een brandmeldingsknop, zie figuur 7b.) .



Figuur 7b.

De gebruiker drukt op de knop voor bediening van de benzinepomp.
Het duurt echter even (zeg een kleine tien seconden) voordat de computer zover is dat de benzinepomp bestuurd kan worden (een eventueel lopende taak moet netjes worden afgebroken, de benodigde software voor de pompbediening moet geladen worden) .
Eventuele meerdere buttonacties worden in een wachtrij opgenomen.
Tijdens deze wachttijd klikt de gebruiker op de knop voor de radio; deze actie wordt gebufferd.
Op het moment dat de "radioklik" afgehandeld kan worden is er een ander commandoscherm actief. De positie van deze klik is op dezelfde plaats als die van het brandalarm, zodat deze klik wordt geïnterpreteerd als een brandmelding, wat in dit geval niet de bedoeling is.

Voor dit probleem zijn de volgende oplossingen mogelijk :

- De klik wordt geacht afkomstig te zijn van het huidige (active) commandoscherm; in ons voorbeeld wordt er dus een brandmelding gegeven.

- Tijdens een wisseling van commandoscherm wordt de buffer geleegd; de klik wordt volledig genegeerd.

- Bij het bufferen van een klik wordt niet alleen de plaats van klikken vastgelegd, maar ook in welk commandoscherm dit heeft plaatsgevonden. Zodra dat scherm weer actief wordt kan de klik worden afgehandeld.

Geen van deze drie oplossingen is écht bevredigend, immers

- in het eerste geval gebeurt er iets wat de gebruiker totaal niet verwacht (hij heeft toch op de radio geklikt ?)

- de tweede oplossing is zeer verwarrend voor een gebruiker; als er geen schermwisseling plaatsvindt kan hij wèl snel achter elkaar op de verschillende knoppen drukken (deze worden dan gebufferd en afgehandeld zoals verwacht) terwijl dit bij een schermwisseling ineens niet meer kan

- de laatste van de voorgestelde oplossingen kan ook verwarrend zijn ; nadat de bediening van de benzinepomp is afgesloten denkt de gebruiker terug te komen in het commandoscherm zoals figuur 7a. Zodra dit scherm actief wordt kan de radioklik worden afgehandeld en (na figuur 7a in een flits op het beeldscherm te hebben gezien) wordt de gebruiker geacht de radio te gaan bedienen. Bovendien is het niet ondenkbaar dat tijdens de pompbediening de gebruiker is afgelost door een collega die niet weet dat er inmiddels ook op de radio is geklikt.

Uitgaande van de tweede oplossing lijkt het volgende voorstel mij wel acceptabel :
Definiëer twee soorten knoppen: de ene soort heeft geen wisseling van commandoscherm tot gevolg, terwijl de andere soort juist wèl een wisseling van commandoscherm tot gevolg heeft en maak dit op een of andere manier duidelijk aan de gebuiker.
Als de programmeur de beschikking heeft over een "CLEAR'BUTTON" commando kan hij, zodra het nieuwe commandoscherm is opgebouwd, m.b.v. dit commando de gebufferde kliks verwijderen.

13. Twee soorten knoppen: verwarrend voor de gebruiker.

Om dit voor een gebruiker begrijpbaar te maken zijn nog wat andere maatregelen nodig :

Er kan gekozen worden de ene soort knop rechthoekig af te beelden op het beeldscherm en de andere soort cirkelvormig, of

als er op een knop met commandoschermwisseling wordt gedrukt deze knop "invers" af te beelden op het beeldscherm en deze afbeelding zo te laten totdat de buttonactie is afgehandeld of

zodra er op zo'n bijzondere knop wordt gedrukt kunnen alle knoppen "gedimd" worden, dit is de manier die Apple ook toepast op de Macintosh bij applicaties als MacWrite. Wanneer de gebruiker vanuit MacWrite een Write document wilt openen, is er een knop zichtbaar om de huidige diskette uit de diskdrive te werpen (eject); deze knop is zwart afgebeeld als zich in de drive een diskette bevindt, maar de knop is grijs (gedimd) als de drive leeg is (in het laatste geval kan de ejectknop niet ingedrukt worden).

Er moet dus voor gezorgd worden dat de gebruiker ziet dat er iets met deze knop aan de hand is.

Nog een (klein) addertje :
Als er gewerkt wordt met een zeer gevoelige muis, zal de muispositie op het moment van indrukken van de knop verschillen van de positie bij het loslaten van de knop : een klik wordt dus gezien als een drag.
Voor een klik moet dus gekozen worden voor één van beide mogelijkheden.
Bij veel programmatuur op de Macintosh is het op dit moment zo, dat als op een "beeldschermknop" wordt gedrukt deze knop invers wordt weergegeven; mocht de gebruiker (misschien juist door deze nadruk) zien dat hij de verkeerde knop heeft, kan hij (als hij de knop op de muis nog steeds heeft ingedrukt) door draggen buiten de beeldschermknop zijn keuze niet door laten gaan.
Voor de argeloze gebruiker is het dus verstandig om van een klik alleen de "eindpositie" als muispositie te gebruiken (dit kan ook bij gebufferde kliks).

14. Nogmaals: de traagheid van de gebruiker.

Dan moet ik nog even terugkomen op de traagheid van de gebruiker; in de voorbeeldprogramma's ben ik er van uit gegaan dat wanneer een gebruiker op de knop van de muis drukt, hij dit zolàng doet dat de selectie "button" minstens eenmaal true oplevert maar dat hij dit aan de andere kant zokòrt doet, dat deze selectie hoogstens eenmaal true oplevert.

Als voorbeeld hetvolgende programma :

```
HOW'TO IS'GEEN'KLIK :
   WHILE 1 = 1 :
      SELECT :
         button :
            DOE'KLIKACTIE
         ELSE :
            DOE'IETS'ANDERS
```

Als het deel "DOE'IETS'ANDERS" veel executietijd vraagt, kan het zijn dat een klik gemist wordt (tenzij er een buffermechanisme is).
Als "DOE'KLIKACTIE" zeer snel wordt uitgevoerd, is het niet ondenkbaar dat wanneer even later de loop weer wordt doorlopen de gebruiker nog steeds de knop van de muis heeft ingedrukt wat resulteert in het nogmaals uitvoeren van "DOE'KLIKACTIE".

Er zou een aparte functie "click" ingevoerd kunnen worden, maar het lijkt ook mogelijk dit probleem aan de programmeur over te laten :

```
HOW'TO IS'WEL'KLIK :
   WHILE 1 = 1 :
      SELECT :
         button :
            WHILE button :
               DOE'NIETS
            DOE'KLIKACTIE
         ELSE :
            DOE'IETS'ANDERS
```

Voor de directe afhandeling van een klik werkt dit wel, maar de gebufferde klik gaat finaal de mist in.
De beste oplossing lijkt dan toch maar een functie "click" in te voeren, welke dan alleen werkt op gebufferde kliks (directe kliks kunnen als in bovenstaand voorbeeld worden verwerkt, een voorbeeldprogramma met beide mogelijkheden volgt later).

```
HOW'TO DOE'GEBUFFERDE'KLIK :
   SELECT :
      click :
         DOE'KLIKACTIE
      ELSE :
         DOE'IETS'ANDERS
```

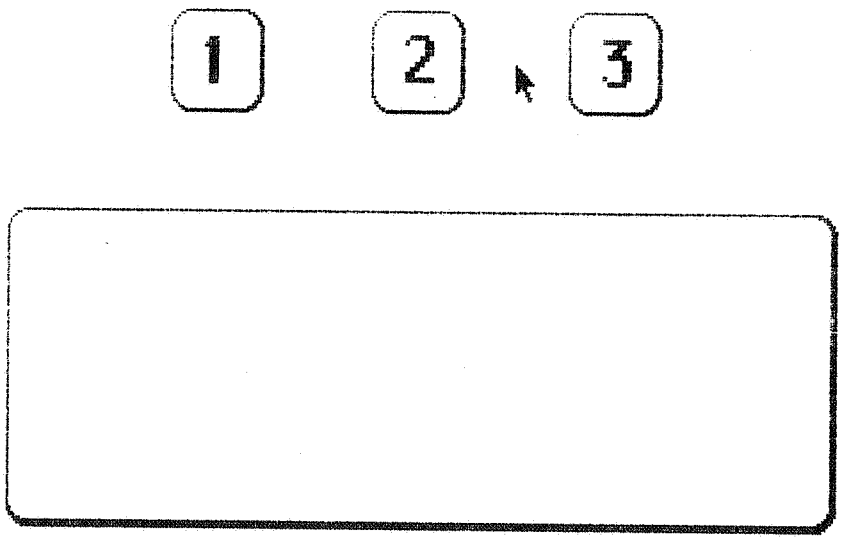
Om de positie van de muis te verkrijgen tijdens de klik zou er dan een functie "POLL'CLICK" gebruikt moeten worden om verwarring met de functie "POLL'MOUSE" te vermijden (hierdoor blijft het dus mogelijk regelmatig te controleren of de muis soms in een "alarmgebied" staat, zonder eerst alle gebufferde kliks af te werken (N.B. er moet dan natuurlijk niet geklikt worden in dit alarmgebied)).

De functie "click" levert "false" op als er geen kliks in de wachtrij staan en "true" als er wel kliks zijn gebufferd. In het laatste geval moet de eerste klik uit de wachtrij worden verwijdert; omdat pas nà deze functie er een functie "POLL'CLICK" kan worden uitgevoerd ("POLL'CLICK" op een lege wachtrij zal ongewenste resultaten leveren), zal de muispositie van deze verwijderde klik op een speciale plaats in de computer moeten worden opgeslagen, waarna deze waarde met een "POLL'CLICK" kan worden opgehaald.

15. Een voorbeeld.

Een voorbeeld met zowel directe als gebufferde kliks en bovendien een vorm van "dragging" :
Op het commandoscherm staat een drietal knoppen afgebeeld waarop geklikt kan worden én er is een tekenvel aanwezig waarop getekend kan worden. Dit tekenen gebeurt met een "potlood" ; de punt van het potlood wordt door het indrukken van de muisknop op het papier geplaatst, door nu (met de knop ingedrukt) de muis te bewegen kan een lijn op het papier worden getekend (punt voor punt, dus als de computer traag is mag de muis niet te snel bewogen worden) .
Zoals al eerder opgemerkt kan alleen een klik gebufferd worden, het tekenen kan alleen onder direct toezicht van de computer.

Eerst worden de gebufferde kliks afgehandeld, dan wordt gekeken of de knop van de muis (op dàt moment) is ingedrukt. Als dit laatste het geval is, kan het zijn dat er getekend wordt of dat er een directe klik plaatsvindt.



Figuur 8.

```
HOW'TO BESTUUR'COMMANDOSCHERM :
   ...
   WHILE 1 = 1 :
      SELECT :
         click :
            POLL'CLICK muispositie
            KLIK
         button :
            POLL'MOUSE muispositie
            SELECT :
               muispositie inside tekenvel :
                  WHILE button :
                     POLL'MOUSE muispositie
                     TEKEN'PUNT muispositie
               ELSE :
                  WHILE button :
                     DOE'NIETS
                  POLL'MOUSE muispositie
                  KLIK
         ELSE :
            DOE'IETS'ANDERS
KLIK :
   SELECT :
      muispositie inside knop'1 :
         DOE'ACTIE'1
      muispositie inside knop'2 :
         DOE'ACTIE'2
      muispositie inside knop'3 :
         DOE'ACTIE'3
      ELSE :
         WRITE "Geen geldig commando." /
```

Merk op dat een "drag" van buiten het tekenvel naar binnen het tekenvel geen effect heeft.
"Draggen" van binnen het tekenvel naar buiten moet opgelost worden binnen "TEKEN'PUNT".

16. Alternatief voor "click" : "mouse'up" & "mouse'down" .

Bij een "click" hoort één muispositie. De ene keer is het belangrijk waar de knop van de muis is ingedrukt, een andere keer is juist de plaats waar de knop is losgelaten belangrijk.
Als voor de mogelijkheid wordt gekozen de "click" te splitsen in "mouse'up" en "mouse'down" kan de programmeur eenvoudig kiezen welk van de twee de voorkeur verdient.
Bovendien blijkt het nu ook mogelijk "dragging" te bufferen, zij het dat de "drag" geacht wordt verlopen te zijn via een rechte lijn tussen het punt waar de knop van de muis werd ingedrukt en het punt waar de knop werd losgelaten.
Dit kan als volgt geprogrammeerd worden, waarbij zich de volgende situaties kunnen voordoen :

a. zowel begin- als eindpunt zijn gebufferd
b. alleen het beginpunt is gebufferd, de "drag" is nog niet afgerond
c. er is niets gebufferd, de "drag" moet nog beginnen

```
HOW'TO DRAG'STRAIGHT :
   SELECT :
      mouse'down :      \ geval a. of b.
         POLL'MOUSE beginpositie
         SELECT :
            mouse'up : \ geval a.
               POLL'MOUSE eindpositie
            ELSE :      \ geval b.
               WHILE button :
                  GET'MOUSE eindpositie
      ELSE :            \ geval c.
         WHILE not button :
            GET'MOUSE beginpositie
         WHILE button :
            GET'MOUSE eindpositie
```

Stel nu dat vlak hiervòòr een "mouse'down" is afgehandeld en dat de erbij behorende "mouse'up" nog in de buffer staat (maar we zijn daarin niet geïnteresseerd), terwijl ook inmiddels de "drag" in de buffer staat.
De buffer bevat dus achtereenvolgens : "mouse'up" "mouse'down" "mouse'up"
De "mouse'up" als bedoeld in bovenstaand programma is echter de tweede "mouse'up" in de buffer, tenzij de afhandeling van "mouse'down" òòk de ervoor staande "mouse'up" verwijdert.
Dit zal waarschijnlijk geen bezwaren opleveren, immers als men in eerste instantie alleen geïnteresseerd is in een "mouse'down" en er bijvoorbeeld na tien maal pas een "mouse'up" nodig is, zal dit ongetwijfeld de "mouse'up" zijn nà de tiende "mouse'down" en niet de eerste "mouse'up" in de buffer.
Omgekeerd zal bij een "mouse'up" de ervòòr staande "mouse'down" ook verwijdert kunnen worden.
Als men om en om geïnteresseerd is in "mouse'down" en "mouse'up" is er niets aan de hand.

17. Conclusie.

Uit dit verslag blijkt dat met toevoeging van slechts enkele functies en procedures op vrij eenvoudige wijze te werken valt met een "grafische editor".
Het grootste probleem blijkt te zijn de synchronisatie in snelheid tussen de gebruiker en de machine.

Samengevat zijn deze functies en procedures de volgende :

- een procedure "POLL'MOUSE" voor het doorgeven van de huidige muispositie (of eventueel ander "aanwijs" apparaat);

- een procedure "POLL'CLICK" voor het doorgeven van de muispositie van een "mouse'down" of een "mouse'up" die op het moment van de klik niet direct afgehandeld kon worden (een gebufferde klik);

- een procedure "CLEAR'BUTTON" voor het leegmaken van de buffer (handig bij een schermwisseling), het lijkt mij niet nodig om dit commando te splitsen in "CLEAR'MOUSE'DOWN" en "CLEAR'MOUSE'UP". De praktijk moet uitwijzen of de onderstelling gerechtvaardigd is;

- een predicaat "button" welke de waarde "true" aflevert als de knop van de muis is ingedrukt en "false" als dit niet zo is;

- een predicaat "mouse'up" dat "true" oplevert als er een nog niet afgehandelde "mouse'up" in de buffer staat;

- een predicaat "mouse'down" dat "true" oplevert als er een nog niet afgehandelde "mouse'down" in de buffer staat;

- een predicaat "inside", welke de waarde "true" aflevert als de opgegeven positie (meestal muispositie) zich in het aangegeven grafisch basisfiguur bevindt en "false" als dit niet het geval is.

18. Mogelijke uitbreiding.

Iets wat ik nog niet besproken heb, maar wat zeker interessant is, is het volgende (misschien hoort zoiets bij het operating system van een *B* systeem, of bij een grafisch pakket (inclusief grafische editor)):

Bij het werken op de Apple Macintosh viel het mij op dat het muissymbool niet constant van vorm is (in de menubalk is het een pijl, in tekst is het een "I-beam" etc. (vooral MacPaint is een prachtig voorbeeld)).
Ik kan mij indenken dat het wenselijk is dat bijvoorbeeld bij het wandelen door een doolhof met "onzichtbare" knoppen het muissymbool een pijl naar boven is als de muis naar het noord-commando wijst, een pijl naar rechts voor het oost-commando etc.
Aan ieder door de programmeur samengesteld basisfiguur zou hij via een "CHANGE'MOUSE" commando een ander muissymbool aan die "knop" kunnen koppelen. (Zo'n ander muissymbool kan een basisfiguur zijn, opgebouwd uit de beschikbare grafische primitieven.)

Dit is dan een verandering afhankelijk van de positie van de muis, het is echter ook interessant de mogelijkheid te bestuderen een verandering te laten plaatsvinden afhankelijk van de "tijd".

Stel dat de gebruiker op een knop heeft gedrukt welke een wisseling van commandoscherm tot gevolg heeft.
Enkele pagina's terug is voorgesteld zo'n knop een speciale vorm te geven of anders te laten reageren op een klik (pagina 19); het is ook mogelijk het muissymbool te veranderen als zo'n bijzondere knop is ingedrukt, het muissymbool zou dan veranderd kunnen worden in een polshorloge (ook bekend bij gebruikers van de Apple Macintosh).