Werkexemplaar

Funktionele programmering van een "(beeldscherm)schildpad"

Maarten M Fokkinga, 3 mei 1985

We geven een SASL programma zodat een gebruiker vanachter de terminal de cursor op het beeldscherm als een schildpad kan besturen middels het intikken van commando's zoals V (doe een stap Voorwaarts), R (draai naar rechts), L (draai naar links) enzovoorts.

* * *

In het alleszins lezenswaardige boek Principles of Functional Programming [Glaser et al 1984] staat het principe beschreven hoe de cursor op het beeldscherm als een schildpad bestuurd kan worden met eenvoudige commando's zoals V (stap Voorwaarts), R (draai Rechtsom), L (draai Linksom) enzovoorts. Het programma dat ik presenteer is daarop geïnspireerd; het is iets anders en in meer detail geformuleerd en bovendien iets uitgebreider (maar wel korter) dan Glaser's programma.

Ten aanzien van het beeldscherm gaan we uit van de volgende aannamen. Het beeldscherm is een apparaat met een ingangskanaal waarover besturingscharacters gezonden kunnen worden die dan een verandering

van het getoonde beeld tot gevolg hebben. Met name veronderstellen we dat de volgende effecten bereikt kunnen worden.

North: cursor gaat een positie omhoog,
East, South, West: analoog,
Clear: beeldscherm wordt schoon, cursor staat linksboven,
Print x: character x wordt getoond op de cursorpositie.

We veronderstellen dat de beeldscherm besturingscharacter(tjes) met bovenstaande effecten in het SASL programma de voorgedefinieerde namen N, E, S, W, C, P x (met character-denotatie x) hebben, dat de breedte en hoogte van het beeldscherm, gemeten in cursorposities, Width en Height zijn, en dat Screen (de UNIX path-name voor) het ingangskanaal van het beeldscherm is.

Het begrip Schildpad definiëren we als volgt. De positie van een schildpad is op het scherm zichtbaar als de positie waar de cursor staat. Voorts heeft een schildpad een al of niet zichtbare interne toestand, te weten de richting waarin hij staat (noord, oost, zuid, west), en het character dat hij "in zich" draagt (en waaruit een spoor op het scherm gevormd wordt wanneer de schildpad stappen voorwaarts doet). Initieel is de positie het mid-

den van het scherm, is de richting noordwaarts en is het character een sterretje. De schildpad op het scherm wordt vanaf de terminal bestuurd door het intikken van schildpadcommando's. Tot nader orde beschouwen we alleen de volgende commando's.

R: draai een kwartslag Rechtsom,
L: draai een kwartslag Linksom,
V: doe een stap Voorwaarts,
T: doe een stap Terug (en laat een spatie als spoor achter),
Wc: Wijzig het interne character in c.

Spaties en regelovergangen mogen naar believe tussen deze commando's worden geplaatst en hebben geen effect. Let er wel op dat er tussen de twee characters van het wijzig-commando ~~g~~ niet zo maar een spatie gezet mag worden. Als de schildpad geïnstalleerd is, kan de gebruiker vanachter de terminal de schildpad een vierkantje laten door-lopen door

V V R V V R V V R V V

in te tikken. Als dit direkt na installatie gebeurt zal er een spoor van sterretjes achterblijven.

De probleemstelling luidt nu: schrijf een funktioneel programma dat de cursor als een schildpad laat besturen vanaf de terminal.

Onze oplossing is als volgt. We schrijven een funktie Turtle die een lijst van schildpadcommando's omzet in een lijst van beeldschermcommando's. Door het SASL commando

Turtle interactive

wordt de lijst van daarna ingetikte characters (tot en zonder eof) aan Turtle onderworpen. Door in de definitie van Turtle de resultaatlijst te voorzien van het achtervoegsel to Screen wordt die resultaatlijst naar het beeldscherm gestuurd, en wel character voor character zo gauw ze beschikbaar komen (vanwege lazy evaluation). De interne toestand van de schildpad, of beter: alleen ~~het~~ character en richting, geven we funktioneel vorm als extra parameters van een hulpversie t van Turtle, vgl. [Folkinga 1985, Paragrafen 3, 8 en 11].

Hier is de SASL definitie van Turtle.

```
Turtle L = (L₀ ++ t c₀ d₀ L) to Screen
where L₀ = [C]++ {E| i←1..Width div 2} ++ {S| i←1..Height div 2}
           --clear screen and place cursor in the middle
      c₀ = %*    --initial character held in the turtle
      d₀ = 0     --initial direction of the turtle; 0,1,2,3 staan voor N,E,S,W
      t c d [] = []
      t c d (x:L) =
        x=%R → t c ((d+1) mod 4) L
        x=%L → t c ((d-1) mod 4) L
        x=%V → [P c, [N,E,S,W] (d+1)] ++ t c d L
        x=%T → [P sp, [N,E,S,W] ((d+2) mod 4)] ++ t c d L
        x=%W → t c' d' L'   where c':L' = L
        x=sp | x=nl | x=bal → t c d L
        [P %?] ++ t c d L   --show ? otherwise
```

We breiden nu de verzameling schildpadcommando's uit met iteratie- en "onthoud"-commando's. Op pedagogische gronden, vgl [van Thienen 1985], kiezen we een bijna structuurloze syntaxis voor deze commando's: het einde van een regel, dus het character nl, betekent zonodig samengestelde commando's af. Hier zijn een paar voorbeelden

```
4 V V R nl      = "doe 4 maal: V V R nl"
B3 V V R nl     = "berg op in locatie 3: V V R nl"
H3              = "haal op en voer uit wat onder locatie 3 is opgeborgen"
4 H3 nl         = "doe viermaal: H3 nl"
```


De aanpassing van Turtle om herhalingscommando's aan te kunnen luidt als volgt. Voeg in de definitie van t een extra clause

```
digit x → t c d ({ y | i←1..digitval x; y←L' }++L'')
          where [L',L''] = splits L
```

toe. Oefening voor de lezer: definieer de functie splits zó dat splits L = [L', L''] met L' ++ L'' = L én L' bevat één nl en wel achteraan. Tweede oefening voor de lezer: laat niet alleen cijfers toe als herhalingsgetallen maar ook meercijferige getallen.

Om de "onthoud"-commando's te kunnen verwerken breiden we de interne toestand uit met een geheugen, nml. een gegevensbank van paren [n, L] met n een cijfer en L een characterlijst, zie ook [Folkeringa 1985, Paragraaf 11]. Dus voeg overal een (eerste) extra parameter b toe aan t met $b_0 = []$ en breid de clauses van t uit met

```
x=%B → t (berg^b n L') c d L''   where [n:L', L''] = splits L
x=%H → t b c d ((haal n b) ++ L')   where n:L' = L
```

en definieer voorts

```
berg b n L = [n, L]: {[n',L'] | [n',L'] ← b; n' ≠ n}
haal n b   = hd {L' | [n',L'] ← b; n' = n}
```

Het zal nu duidelijk zijn dat je met groot gemak nog veel meer schildpadcommando's kunt invoeren. We komen dan meer op het terrein van ~~een~~ het ontwerpen van een (pädagogisch verantwoorde ?) schildpadtaal, dan op het terrein van uitdagende problemen voor funktioneel programmeren. Dat zullen we dus maar achterwege laten.

Tenslotte merken we nog op dat de definitie van Turtle nog ietwat gestructureerder en modulairder kan worden opgezet. Met name verdient het aanbeveling om het ontleden van de ingetikte characterlijst, en het omzetten van zo'n ontleede lijst in een lijst van beeldschermcommando's gescheiden te programmeren. Stel dat dat gebeurt in funkties TurtleProper en ParseInput, dan luidt de definitie van Turtle zelf

```
Turtle = TurtleProper . ParseInput
```

(of: Turtle L = TurtleProper (ParseInput L) to Screen).

Literatuur

Folkinga M.M: Functioneel Programmeren in een vogelvlucht. INF-85-4, 1985.

Glaser, H., Hankin, C., Till, D.: Principles of Functional Programming. Prentice-Hall, Englewood Cliffs, N.J., 1984.

van Thienen, H: ....