

9 Zeitabhängige Modelle

Programme bearbeiten oft Daten, die sich mit der Zeit ändern:

- Zeitbomben



- Stand der Sonne



- Stand des Aktienmarktes



- Zusammensetzung der Bundesregierung

- Zeitabhängige Komponente der Programmausführung: *Zustand*
- Programm bewirkt *Transformation des Zustands*
- Ggf beeinflusst von Benutzereingaben
- Bsp: Lauf der Sonne über den Himmel

9.1 Das Teachpack `image.ss`

- Prozeduren zur Erzeugung von Bildern
- `rectangle` : `nat nat mode color -> image`
 - Breite und Höhe des Rechtecks
 - `mode` ist entweder `"solid"` oder `"outline"` (ein String)
 - `color` ist Name einer Farbe.
Z.B.: `"red"`, `"blue"`, `"yellow"`, `"black"`, `"white"` oder `"gray"`
 - Ein Wert der Sorte `image` wird direkt in der REPL angezeigt.

9.1.1 Weitere Bilderzeuger

- `circle : nat mode color -> image`
 - Radius des Kreises
- `ellipse : nat nat mode color -> image`
- `triangle : nat mode color -> image`
gleichseitiges Dreieck
- `line : nat nat rel rel rel rel color -> image`
Aufruf `(line w h x1 y1 x2 y2 c)` liefert
 - Bild der Größe `w h`
 - Darin eine Linie von `(x1, y1)` nach `(x2, y2)`
Koordinatenursprung `(0,0)` ist oben links

9.1.2 Kombination von Bildern

- `overlay` : `image image h-place v-place -> image`
 - legt das zweite Bild auf das erste
 - `h-place` ist die Sorte für
 - ; Eine horizontale Positionsangabe ist eins der folgenden
 - ; - eine Zahl (Abstand vom linken Rand)
 - ; - "left", "right" oder "center"
 - `v-place` ist die Sorte für
 - ; Eine vertikale Positionsangabe ist eins der folgenden
 - ; - eine Zahl (Abstand vom oberen Rand)
 - ; - "top", "bottom" oder "center"
 - Das Ergebnisbild umfasst beide Argumentbilder.

9.1.3 Weitere Kombinationen von Bildern

- `above` : `image image h-mode -> image`
- `beside` : `image image v-mode -> image`
- `clip` : `image nat nat nat nat -> image`
 - `(clip img x y w h)` schneidet ein Teilrechteck aus
 - `(x,y)` ist linke obere Ecke
 - `(w,h)` sind Breite und Höhe des Teilrechtecks
- `pad` : `image nat nat nat nat -> image`
 - `(pad img l r t b)` fügt links, rechts, oben und unten Pixel an (`l`, `r`, `t` bzw `b`)
- `image-width` : `image -> nat`
- `image-height` : `image -> nat`
- Externe Bilder können per Menü als Werte definiert werden.

9.2 Modelle und Ansichten

- Sichtbare Darstellungen für innere Größen

Uhr	Uhrzeit
Thermometer	Temperatur
Hygrometer	Luftfeuchtigkeit
Kontoauszug, Kontostand	Transaktionsgeschichte

- Bezeichnungen:

Ansicht sichtbare Darstellung (*view*)

Modell innere Größe (*model*)

- zeitveränderliches Modell: *Zustandsmodell*
- zeitveränderliche Größe: *Zustand*

- Verwende die Aufteilung in Modell und Ansicht um Programme zu strukturieren!

9.2.1 Beispiel: Anzeige der Uhrzeit durch Sonnenstand

Die Sonne bewegt sich in einem Halbkreis über einen rechteckigen Himmel.

Die Bewegung von Osten nach Westen dauert immer zwölf Stunden.

Das heißt, das **Modell** ist eine Zahl zwischen 0 und 12.

Die folgenden Definitionen realisieren eine **Ansicht** dieses Modells.

Die klare Trennung zwischen Modell und Ansicht ist ein wichtiges Entwurfsprinzip für die Programmierung interaktiver Benutzerschnittstellen.

9.2.2 Realisierung der Sonnenstandsanzeige

```
; Bild der Sonne
; sun : image
(define sun )

; Breite des Himmels
; sky-width : number
(define sky-width 500)

; Höhe des Himmels
; sky-height : number
(define sky-height (/ sky-width 2))

; Bild der Himmels
; sky : image
(define sky
  (rectangle sky-width sky-height "solid" "lightblue"))
```

9.2.3 Sonnenstandsanzeige, Fortsetzung

```
; Drehradius der Sonne
; sky-radius : number
(define sky-radius 200)

; Bild mit der Sonne vor dem Himmel zu einer bestimmten Zeit erzeugen
; sky-with-sun : number -> image
(define sky-with-sun
  (lambda (t)
    (let* ((angle (/ (* pi t) 12))
           (sun-x (- (+ (/ sky-width 2) (* sky-radius (cos angle)))
                     (/ (image-width sun) 2)))
           (sun-y (- sky-height
                     (* sky-radius (sin angle))
                     (image-height sun))))
      (overlay sky sun sun-x sun-y))))
```

9.3 Bewegung

- Ziel: Anzeige der Bewegung der Sonne am Himmel als Animation
- Ersetze Teachpack `image.ss` durch `world.ss`
- (Alle Definitionen aus `image.ss` sind weiterhin verfügbar.)
- `world.ss` realisiert *eventbasierte Programmierung*

9.3.1 Prozeduren im Teachpack `world.ss`

- `big-bang` : `nat nat number world -> #t`
(`big-bang w h step init`) erzeugt eine Animation bestehend aus
 - einer Ansicht der Breite `w` und der Höhe `h`
 - Uhrticks, die im Abstand von `step` Sekunden erzeugt werden
 - `init` als initiales Modell
- `on-redraw` : `(world -> image) -> #t`
(`on-redraw generate-view-from-world`) registriert im System eine Prozedur `generate-view-from-world`, die aus einem Modell eine Ansicht generiert, falls das notwendig sein sollte
- `on-tick-event` : `(world -> world) -> #t`
(`on-tick-event transform-model`) registriert im System eine Prozedur `transform-model`, die bei jedem Uhrtick aufgerufen wird und das Modell auf den nächsten Stand bringt
- `end-of-time` : `string -> world`
falls `transform-model` als Ergebnis (`end-of-time "..."`) liefert, so wird die Animation beendet

9.3.2 Verwendung von `big-bang` und `Co`

- Erzeuge eine Simulation von passender Größe mit Zeitschritten der Länge 0.1 Sekunden beginnend beim Modellzustand 0:

```
(big-bang sky-width sky-height 0.1 0)
```

- Registriere eine bilderzeugende Prozedur: `sky-with-sun` hat einen passenden Vertrag

```
(on-redraw sky-with-sun)
```

- Registriere eine Prozedur, die bei jedem Uhrtick aufgerufen wird

```
; bei jedem Zeitschritt schreitet der Sonnenstand um 0.1 voran
```

```
; next-time : number -> number
```

```
(define next-time
```

```
  (lambda (t)
```

```
    (+ t 0.1)))
```

```
(on-tick-event next-time)
```

9.3.3 Abbruch der Animation

- Das Ende der Animation ist erreicht, wenn die Zeit 12 erreicht ist

```
; Transformation des Modells in einem Zeitschritt
```

```
; next-time : number -> number
```

```
(define next-time
```

```
  (lambda (t)
```

```
    (if (< t 12)
```

```
        (end-of-time "Der Himmel bricht zusammen!")
```

```
        (+ t 0.1))))
```

9.4 Tag und Nacht

; Bild vom Mond

; moon : image

(define moon )

; Bild des Nachthimmels

; night-sky : image

(define sky

(rectangle sky-width sky-height "solid" "black"))

9.4.1 Nachthimmel mit Mond

- Das Zeichnen des Mondes vor dem Nachthimmel geht genau wie das Zeichnen der Sonne vor dem Taghimmel

⇒ Abstrahiere von den Objekten und vom Himmel!

; Bild mit einem Objekt vor einem Himmel zu einer bestimmten Zeit erzeugen

; `sky-with-object` : `image image -> (number -> image)`

```
(define sky-with-object
  (lambda (sky object)
    (lambda (t)
      (let* ((angle (/ (* pi t) 12))
             (sun-x (- (+ (/ sky-width 2) (* sky-radius (cos angle)))
                       (/ (image-width object) 2)))
             (sun-y (- sky-height
                       (* sky-radius (sin angle))
                       (image-height object))))
        (overlay sky object sun-x sun-y))))))
```

9.4.2 Zeichnen des Tag- und Nachthimmels

; Zeichnen des Taghimmels mit der Sonne

; `sky-with-sun` : `number -> image`

```
(define sky-with-sun  
  (sky-with-object sky sun))
```

; Zeichnen des Nachthimmels mit dem Mond

; `sky-with-moon` : `number -> image`

```
(define sky-with-moon  
  (sky-with-object night-sky moon))
```

9.4.3 Verfeinertes Modell

- Es muss zwischen Tag und Nacht unterschieden werden.

```
; Eine sky-world ist ein Wert
; (make-sky-world d t)
; wobei d : boolean anzeigt ob Tag (#t) oder Nacht (#f) ist
; und t : number die Tageszeit angibt mit (<= 0 t 12)
(define-record-procedures sky-world
  make-sky-world sky-world?
  (sky-world-day? sky-world-time))
```

- Entsprechend muss die on-tick-event Prozedur angepasst werden

```
; Weltzustand nach dem Zeitschritt
; next-sky-world : sky-world -> sky-world
(define next-sky-world
  (lambda (sw)
    (let ((d (sky-world-day? sw))
          (t (+ (sky-world-time sw) 0.1)))
      (if (<= t 12)
          (make-sky-world d t)
          (make-sky-world (not d) 0)))))
```

- Zu guter Letzt muss eine on-redraw Prozedur für sky-worlds erstellt werden

```
; transformiere Modell nach Ansicht
; sky-world-image : sky-world -> image
(define sky-world-image
  (lambda (sw)
    (let ((t (sky-world-time sw)))
      (if (sky-world-day? sw)
          (sky-with-sun t)
          (sky-with-moon t))))))
```

- Die erweiterte Simulation wird gesteuert durch

```
(big-bang sky-width sky-height 0.1 (make-sky-world #t 0))
(on-redraw sky-world-image)
(on-tick-event next-sky-world)
```

9.4.4 Noch ein anderer Event

- `on-key-event` : `(world string -> world) -> #t`
- Nach Registrierung von `(on-key-event process-key)` wird bei jedem Tastendruck die Prozedur `process-key` aufgerufen.
- Das `string` Argument ist (gewöhnlich) ein String der Länge 1, der das entsprechende Zeichen enthält.
- Beispiel: Tag- und Nachtwechsel beim Drücken von x

```
(on-key-event
  (lambda (sw str)
    (cond
      ((string=? str "x")
       (make-sky-world (not (sky-world-day? sw))
                       (sky-world-time sw)))
      (else
       sw))))
```

9.5 Zusammenfassung

- Behandlung von Zustand
- Trennung von Modell und Ansicht
- Bilder als Werte
- Eventbasierte Programmierung
- Interaktive Animation