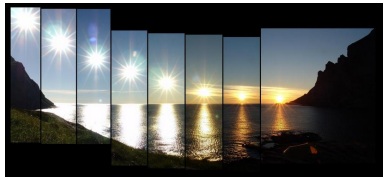
7 Zeitabhängige Modelle

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: Flug eines Vogels über den Himmel

7.1 Teachpack `image.ss`

- Prozeduren zur Erzeugung von Bildern

```
(: rectangle (natural natural mode image-color -> image))
```

  - Breite und Höhe des Rechtecks
  - `mode` ist `(one-of "solid" "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.

7.1.1 Weitere Bilderzeuger

`(: circle (natural mode image-color -> image))`

- Radius des Kreises

`(: ellipse (natural natural mode image-color -> image))`

- Breite und Höhe der Ellipse

`(: triangle (natural mode image-color -> image))`

- gleichseitiges Dreieck

```
(: line (natural natural number number number number image-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

`(: text (string natural image-color -> image))`

- Aufruf `(text s f c)` erzeugt ein Bild mit Text `s`, wobei die Buchstaben die Größe `f` haben

7.1.2 Kombination von Bildern

```
(: overlay (image image h-place v-place -> image))
```

- legt das zweite Bild auf das erste
- `h-place` ist der Vertrag für horizontale Positionsangaben

```
(define h-place
  (contract (mixed integer ; Abstand vom linken Rand
                   (one-of "left" "right" "center")))
```

- `v-place` ist der Vertrag für vertikale Positionsangaben

```
(define v-place
  (contract (mixed integer ; Abstand vom oberen Rand
                   (one-of "top" "bottom" "center")))
```

- Das Ergebnisbild umfasst beide Argumentbilder.

7.1.3 Weitere Kombinationen von Bildern

```
(: above (image image h-mode -> image))
```

```
(: beside (image image v-mode -> image))
```

```
(: clip (image natural natural natural natural -> 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 natural natural natural natural -> image))
```

- `(pad img l r t b)` fügt links, rechts, oben und unten Pixel an (l, r, t bzw b)

```
(: image-width (image -> natural))
```

```
(: image-height (image -> natural))
```

- Externe Bilder können per Menü als Werte definiert werden.

7.2 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!

7.2.1 Beispiel: Animation eines Geiers

Geier fliegt von links nach rechts durch das Bild.

Ein Flug von links nach rechts dauert immer 60 Zeiteinheiten.

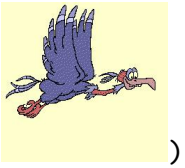
Das heißt, das **Modell** ist eine Zahl zwischen 1 und 60.

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

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

7.2.2 Realisierung des Geierfluges

```
; Graphische Repräsentation eines Geiers
(: geier-image image)
```



```
(define geier-image )

; Höhe des Himmels
(: sky-width natural)
(define sky-width 600)

; Breite des Himmels
(: sky-height natural)
(define sky-height 250)

; Darstellung des Himmels
(: sky image)
(define sky
  (rectangle sky-width sky-height "solid" "lightblue"))

; Zeichnen des Geiers in Abhängigkeit von der Position
(: draw-geier (natural -> image))
(define draw-geier
  (lambda (position)
    (overlay sky geier-image (* 10 position) 0)))
```

7.3 Bewegung

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

7.3.1 Prozeduren im Teachpack `world.ss`

`(: big-bang (natural natural number world -> (one-of #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) -> (one-of #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) -> (one-of #t)))`
`(on-tick-event transform-model)` registriert im System eine Prozedur `transform-model`, die bei jeden Uhrtick aufgerufen wird und das Modell auf den nächsten Stand bringt

7.3.2 Verwendung von `big-bang` und Co

- Position des Geiers wird durch eine natürlich Zahl bestimmt. Diese Zahl modelliert dann auch die Welt.

```
; Die Position am linken Bildschirmrand
(: initial-geier-position natural)
(define initial-geier-position 1)

; Die Position am rechten Bildschirmrand
(: last-geier-position natural)
(define last-geier-position 60)
```

- Erzeuge eine Simulation von passender Größe mit Zeitschritten der Länge 0.1 Sekunden beginnend beim Modellzustand 1:

```
(big-bang sky-width sky-height 0.1 initial-geier-position)
```

- Registriere eine bilderzeugende Prozedur:
  `draw-geier` hat einen passenden Vertrag `(natural -> image)`

```
(on-redraw draw-geier)
```

- Registriere eine Prozedur, die bei jedem Uhrtick aufgerufen wird

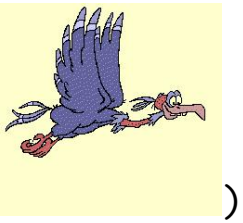
```
; Berechnung der nächsten Welt
(: next-geier-position (natural -> natural))
(define next-geier-position
  (lambda (geier-position)
    (if (= last-geier-position geier-position)
        initial-geier-position
        (+ 1 geier-position))))

(on-tick-event next-geier-position)
```

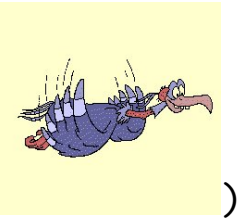
7.4 Mehr Animation

- Flügelschlag des Geiers soll ebenfalls animiert werden.
- Benutze verschiedene Version des Geiers und zeige diese schnell nacheinander an:

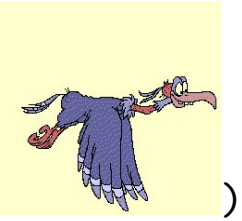
```
(: geier-oben image)
```



```
(define geier-oben )
(: geier-mitte image)
```



```
(define geier-mitte )
(: geier-unten image)
```



```
(define geier-unten )
```

7.4.1 Verfeinertes Modell

- Das Modell (`world`) besteht jetzt nicht mehr nur aus der Position des Geiers sondern auch aus der Position der Flügel:
  - `"unten"`: Flügel sind unten
  - `"unten->oben"`: Flügel gehen von unten nach oben
  - `"oben"`: Flügel sind oben
  - `"oben->unten"`: Flügel gehen von oben nach unten

```
(define flügel-position (contract (one-of "unten" "unten->oben"
                                          "oben" "oben->unten")))
```

- Das Modell wird jetzt als ein Rekord repräsentiert:

```
(define-record-procedures world
  make-world world?
  (world-flügel-position world-geier-position))
(: make-world (flügel-position natural -> world))
```

- (Kompletter Code: siehe Demo oder Datei `20091117-geier.scm`)

7.4.2 Noch ein anderer Event

```
(: on-key-event ((world string -> world) -> (one-of #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.
- Rückgabewert von `process-key` ist die möglicherweise geänderte Welt.

- *Beispiel:* Anhalten der Animation bei Drücken von `s`
  - Erweitere Modell um Information ob Animation gestoppt:

```
(define-record-procedures world make-world world?
  (world-flügel-position world-geier-position world-stopped?))
(: make-world (flügel-position natural boolean -> world))
```

  - Programmiere Key Handler:

```
; Stoppen/starten der Animation
(: toggle-stop (world string -> world))
(define toggle-stop
  (lambda (world key)
    (cond
      ((string=? key "s")
       (make-world (world-flügel-position world)
                   (world-geier-position world)
                   (not (world-stopped? world))))
      (else world))))
```

  - Registriere Key Handler: `(on-key-event toggle-stop)`
  - Berücksichtige `world-stopped?` innerhalb von `next-world`

7.5 Zusammenfassung

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