

“Käitumised”

- Käitumised tüüpi `Behavior` a on ajast sõltuvad a tüüpi väärtused; so. (pidevad) funktsioonid ajast `Time` tüüpi a
- Eelnevalt vaatlesime, kuidas animeerida geomeetrilisi kujundeid `Shape`, regioone `Region` ja pilte `Picture`
- Näiteid:

```
dot = shape (ell 0.2 0.2)
```

```
ex1 = reg red (translate (0, time / 2) dot)
```

```
ex2 = reg (lift0 Blue)
```

```
      (translate (sin time, cos time) dot)
```

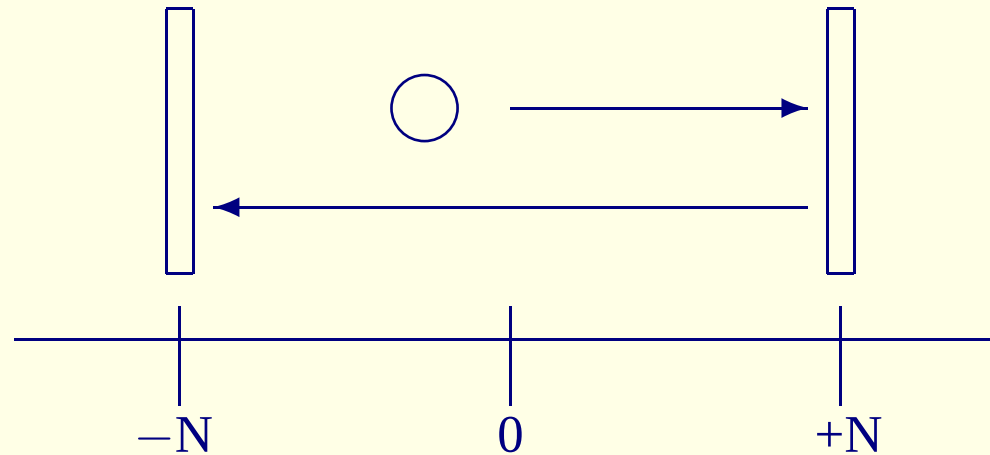
“Käitumised”

- Kuna “käitumised” on tavalised Haskell'i objektid, saame üle nende lihtsasti abstraherida, et moodustada olemasolevatest “käitumistest” uusi:

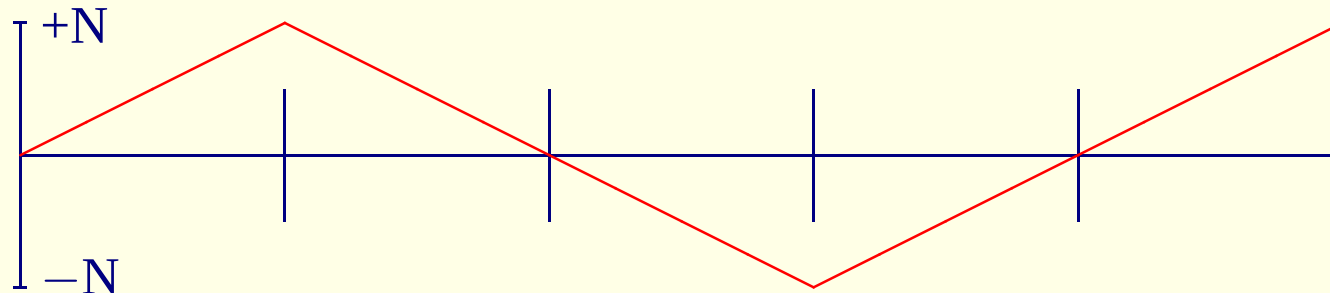
```
wander x y color  
    = reg color (translate (x,y) dot)  
  
ex3 = wander (time/2-3) (sin time) red
```

Näide: põrkuv pall

- Tahame animeerida kahe seina vahel põrkuvat palli:

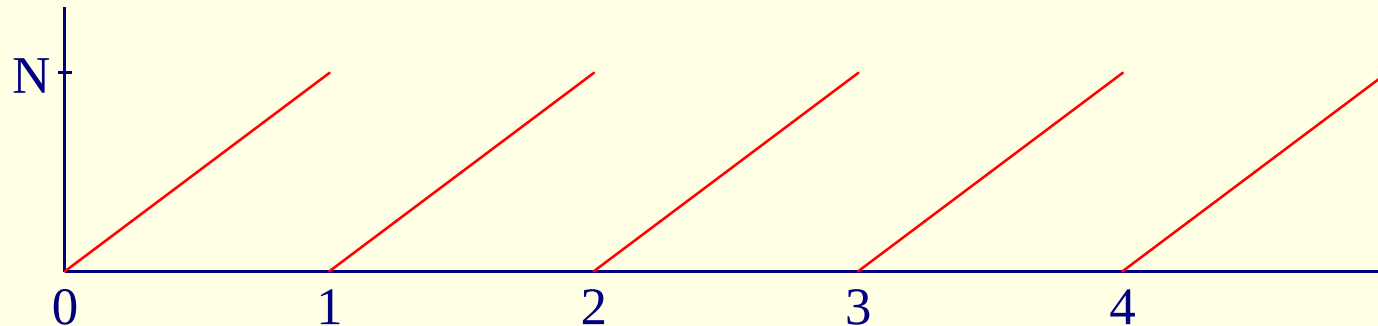


- Y-koordinaat on konstantne, kuid X-koordinaat muutub:

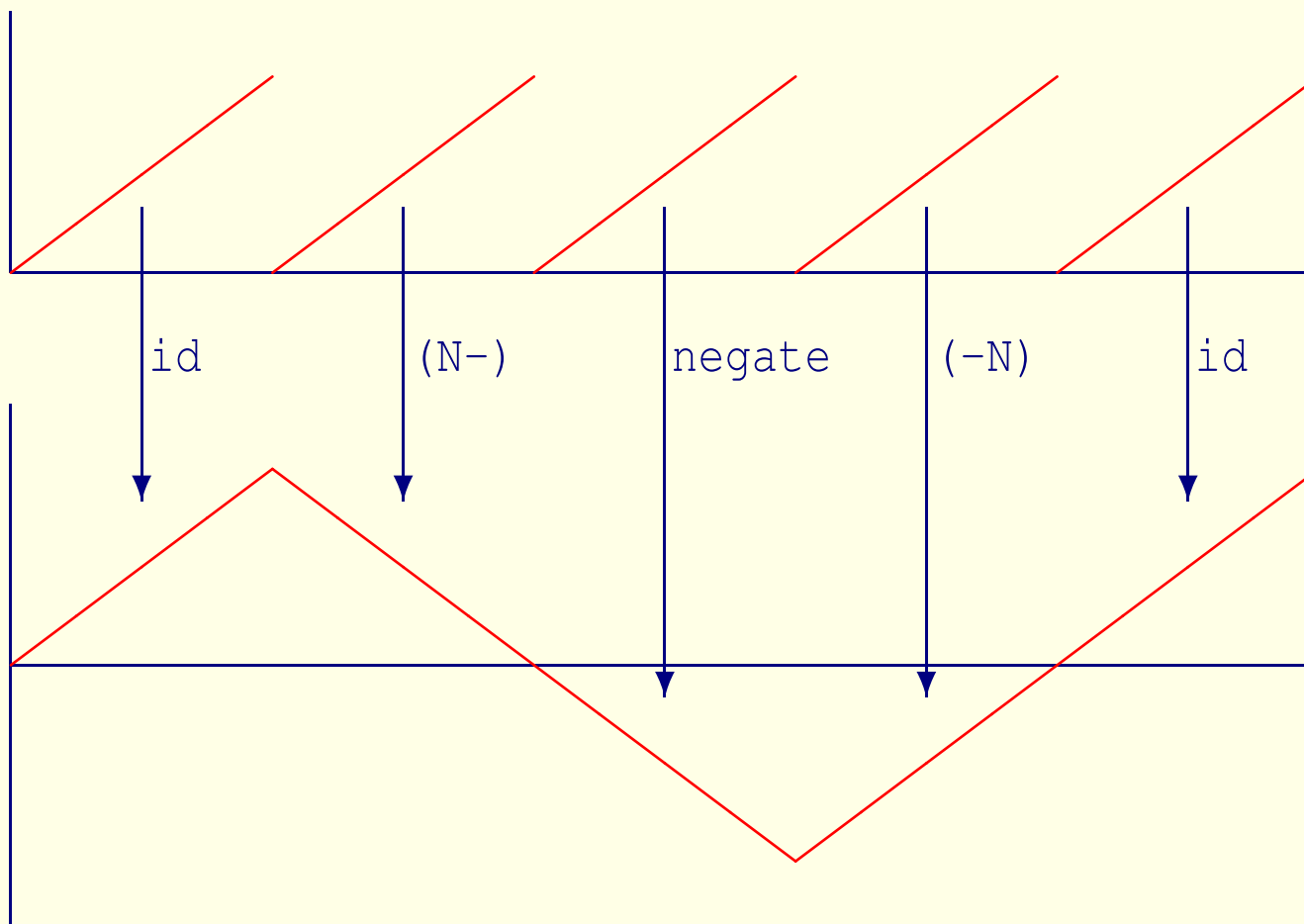


Näide: põrkuv pall

```
modula x y = (period, w)
  where (whole, fract) = properFraction x
        n = whole `mod` y
        period = whole `div` y
        w = (fromInt (toInt n)) + fract
```



Näide: põrkuv pall



Näide: põrkuv pall

```
bounce t = f fraction
  where (period,fraction) = modula t 2
        f = funs !! (period `mod` 4)
        funs = [id, (2.0-), negate, (\x -> x-2.0)]

bball = wander (lift1 bounce time) 0 yellow
```

Reaktiivsed animatsioonid

- FAL (functional animation language) lisab käitumistele reaktiivsuse; so. võime reageerida reaalsaja, aga ka virtuaalsetele, sündmustele; näiteks:
 - klahvivajutus
 - hiire liigutamine
 - riistvara katkestus
 - muutuja omandab mingi etteantud väärtuse (virtuaalsündmus)
 - FAL on lihtsustatud versioon (mittestandardsest) Haskellis teegist Fran
- <http://www.research.microsoft.com/~conal/Fran/>

Reaktiivsed animatsioonid

- Värvid, kujundid ja regioonid:

```
red, blue, yellow, green, white :: Behavior Color
```

```
shape :: Behavior Shape -> Behavior Region
```

```
ell, rec :: Behavior Float -> Behavior Float
                                -> Behavior Region
```

```
translate :: (Behavior Float, Behavior Float)
           -> Behavior Region -> Behavior Region
```


Reaktiivsed animatsioonid

- Aritmeetilised ja loogilised operaatorid:

$(+), (*) :: \text{Num } a \Rightarrow \text{Behavior } a \rightarrow \text{Behavior } a$
 $\rightarrow \text{Behavior } a$

$\text{negate} :: \text{Num } a \Rightarrow \text{Behavior } a \rightarrow \text{Behavior } a$

$(>*), (<*) :: \text{Ord } a \Rightarrow \text{Behavior } a \rightarrow \text{Behavior } a$
 $\rightarrow \text{Behavior Bool}$

$(&\&*), (||*) :: \text{Behavior Bool} \rightarrow \text{Behavior Bool}$
 $\rightarrow \text{Behavior Bool}$

Reaktiivsed animatsioonid

- Pildid:

```
paint :: Behavior Color -> Behavior Region
                                -> Behavior Picture
over  :: Behavior Picture -> Behavior Picture
                                -> Behavior Picture
```

- Sündmused:

```
lbp :: Event ()           -- left button press
key  :: Event Char        -- key press
mm   :: Event Coordinate  -- mouse motion
```

Reaktiivsed animatsioonid

- Lihtne reaktiivsus:

`(->>) :: Event a -> b -> Event b`

`untilB :: Behavior a -> Event (Behavior a)
-> Behavior a`

- Näide: reaktiivne värv

`color1 :: Behavior Color
color1 = red `untilB` (lbp ->> blue)`

Reaktiivsed animatsioonid

- Näide: värvi muutev pall

```
circ :: Behavior Region
```

```
circ = translate (cos time, sin time) (ell 0.2 0.2)
```

```
ball1 :: Behavior Picture
```

```
ball1 = paint color1 circ
```

- Näide: rekursiivne reaktiivsus

```
color1r = red `untilB` lbp ->>
```

```
    blue `untilB` lbp ->>
```

```
    color1r
```

Reaktiivsed animatsioonid

- Sündmuste valik:

```
(.|.) :: Event a -> Event a -> Event a
```

- Näited:

```
color2 = red `untilB` ((lbp ->> blue) .|.
                      (key ->> yellow))
```

```
color2r = red `untilB` colorEvent
  where colorEvent
    = (lbp ->> blue    `untilB` colorEvent) .|.
      (key ->> yellow `untilB` colorEvent)
```

Reaktiivsed animatsioonid

- Sündmuste rekursiivne töötlus:

```
switch :: Behavior a -> Event (Behavior a)
      -> Behavior a
```

- Näide:

```
color2h = red `switch` ((lbp ->> blue) .|.
                        (key ->> yellow))
```

- Sündmuste teisendamine:

```
withElem_ :: Event a -> [b] -> Event b
```

- Näide:

```
color1h = red `switch`
          (lbp `withElem_` cycle [blue,red])
```

Reaktiivsed animatsioonid

- Sündmuste teisendamine:

```
(=>>) :: Event a -> (a->b) -> Event b
```

- Näide:

```
color3 = white `switch` (key =>> \c ->
    case c of 'R' -> red
              'B' -> blue
              'Y' -> yellow
              _   -> white )
```

Reaktiivsed animatsioonid

- Käitumiste “säilitamine”

```
snapshot :: Event a -> Behavior b -> Event (a,b)
```

- Näide:

```
color4 = white `switch`  
      ((key `snapshot` color4) ==>> \(c,old) ->  
       case c of 'R' -> red  
                 'B' -> blue  
                 'Y' -> yellow  
                 _   -> lift0 old)
```


Reaktiivsed animatsioonid

- Virtuaalsed sündmused:

```
while, when :: Behavior Bool -> Event ()
```

- Näide:

```
color5 = red `untilB` (when (time >* 5) ->> blue)
```

- Arvkäitumiste integreerimine:

```
integral :: Behavior Float -> Behavior Float
```

- NB! Saab kasutada kiiruse põhjal asukoha ning kiirenduse põhjal kiiruse arvutamiseks:

```
s,v :: Behavior Float
```

```
s = s0 + integral v
```

```
v = v0 + integral f
```

Reaktiivsed animatsioonid

- Näide: “põrandale” pörkuv pall

```
ball2 = paint red (translate (x,y) (ell 0.2 0.2))
  where g = -4
        x = -3 + integral 0.5
        y = 1.5 + integral v
        v = integral g `switch`
            (hit `snapshot_` v ==>> \v' ->
              lift0 (-v') + integral g)
        hit = when (y <* -1.5)
```

Reaktiivsed animatsioonid

- Sündmuste akumulereerimine

```
stepAccum :: a -> Event (a->a) -> Behavior a
```

- Näide: seinte vahel pörkuv pall

```
wander x y color  
    = paint color (translate (x,y) (ell 0.2 0.2))
```

```
bball = wander x 0 yellow  
    where xvel = 1 `stepAccum` (hit ->> negate)  
          x = integral xvel  
          left = x <* -2.0 &&* xvel <* 0  
          right = x >* 2.0 &&* xvel >* 0  
          hit = when (left ||* right)
```