Geomeetriliste kujundite joonistamine

- Geomeetriliste kujundite andmetüüp:

```
data Shape = Rectangle Side Side
           | Ellipse Radius Radius
           | RtTriangle Side Side
           | Polygon [Vertex]
     deriving Show

type Radius = Float
type Side   = Float
type Vertex = (Float,Float)
```


Geomeetriliste kujundite joonistamine

- `Rectangle`, `Ellipse`, `RtTriangle` ei sõltu paigutusest, aga `Polygon` asukoht on definitsiooniga määratud.
- Asukohast sõltumatuted kujundite aknasse joonistamisel paigutame nad akna keskele; so. kujundite joonistamisel on koordinaatsüsteemi algpunkt akna keskel.
- Samal ajal, graafika-teegis kasutatava koordinaadistiku algpunkt on akna vasak ülemine nurk.
- Kujundite esitamisel kasutatakse pikkusühikuna tolli; graafika-teegis aga pikseleid.
- Vaja defineerida teisendusfunktsioonid ühest koordinaatsüsteemist teise!

Geomeetriliste kujundite joonistamine

- Ühikute teisendamine

```
inchToPixel  :: Float -> Int
inchToPixel x = round (100*x)

pixelToInch  :: Int -> Float
pixelToInch n = intToFloat n / 100

intToFloat   :: Int -> Float
intToFloat  n = fromInteger (toInteger n)
```


Geomeetriliste kujundite joonistamine

- Koordinaatide teisendamine

```
xWin, yWin :: Int
xWin = 600
yWin = 500

xWin2, yWin2 :: Int
xWin2 = xWin `div` 2
yWin2 = yWin `div` 2
```

Geomeetriliste kujundite joonistamine

- Koordinaatide teisendamine (järg)

```
trans :: Vertex -> Point
trans (x,y) = ( xWin2 + inchToPixel x,
                yWin2 - inchToPixel y )

transList :: [Vertex] -> [Point]
transList []     = []
transList (p:ps) = trans p : transList ps
```


Geomeetriliste kujundite joonistamine

- Kujundite teisendamine graafikaks

```
shapeToGraphic :: Shape -> Graphic

shapeToGraphic (Rectangle s1 s2)
  = let s12 = s1/2
        s22 = s2/2
    in polygon
         (transList [(-s12,-s22),(-s12,s22),
                     (s12,s22),(s12,-s22)])
```

Geomeetriliste kujundite joonistamine

- Kujundite teisendamine graafikaks (järg)

```
shapeToGraphic (Ellipse r1 r2)
  = ellipse (trans (-r1,-r2)) (trans (r1,r2))
shapeToGraphic (RtTriangle s1 s2)
  = polygon (transList [(0,0),(s1,0),(0,s2)])
shapeToGraphic (Polygon pts)
  = polygon (transList pts)
```


Geomeetriliste kujundite joonistamine

```
sh1,sh2,sh3,sh4 :: Shape

sh1 = Rectangle 3 2
sh2 = Ellipse 1 1.5
sh3 = RtTriangle 3 2
sh4 = Polygon [(-2.5,2.5), (-1.5,2.0),
               (-1.1,0.2), (-1.7,-1.0),
               (-3.0,0)]
```

Geomeetriliste kujundite joonistamine

```
main0
  = runGraphics $
    do w <- openWindow "Drawing Shapes" (xWin,yWin)
       drawInWindow w (withColor Red
                              (shapeToGraphic sh1))
       drawInWindow w (withColor Blue
                              (shapeToGraphic sh2))
       spaceClose w
```


Geomeetriliste kujundite joonistamine

```
type ColoredShapes = [(Color,Shape)]

shs :: ColoredShapes
shs  = [(Red,sh1),    (Blue,sh2),
        (Yellow,sh3), (Magenta,sh4)]

drawShapes :: Window -> ColoredShapes -> IO ()
drawShapes w [] = return ()
drawShapes w ((c,s):cs)
  = do drawInWindow w (withColor c (shapeToGraphic s))
       drawShapes w cs
```

Geomeetriliste kujundite joonistamine

```
main1
  = runGraphics $
    do w <- openWindow "Drawing Shapes" (xWin,yWin)
       drawShapes w shs
       spaceClose w
```


Geomeetriliste kujundite joonistamine

```
conCircles = map circle [2.4,2.1 .. 0.3]

coloredCircles =
  zip [Black, Blue, Green, Cyan, Red,
            Magenta, Yellow, White]
      conCircles

main2
  = runGraphics $
    do w <- openWindow "Bull's Eye" (xWin,yWin)
       drawShapes w coloredCircles
       spaceClose w
```

Kõrgemat-järku funktsioonid

- Listi iga elemendi korrutamine kahega

```
doubleList :: [Int] -> [Int]
doubleList []     = []
doubleList (x:xs) = (2*x) : doubleList xs
```

- Stringi kõikide tähtede muutmine suurteks

```
upperString :: String -> String
upperString []     = []
upperString (c:cs) = toUpper c : upperString cs
```

- Mõlemal juhul on rekursiooniskeem täpselt sama!

Kõrgemat-järku funktsioonid

- Funktsioon `map`

$$\texttt{map}\ f\ [x_1, x_2, \ldots, x_n] \quad \Longrightarrow \quad [f\,x_1, f\,x_2, \ldots, f\,x_n]$$

- Definitsioon:

```
map          :: (a -> b) -> [a] -> [b]
map f []      = []
map f (x:xs) = f x : map f xs
```

- Näited:

```
doubleList  = map (\x -> 2*x)
upperString = map toUpper
transList   = map trans
```

Kõrgemat-järku funktsioonid

- Listi elementide summa

```
sum :: [Int] -> Int
sum []     = 0
sum (x:xs) = x + sum xs
```

- Listide listi konkateneerimine

```
concat :: [[a]] -> [a]
concat []       = []
concat (xs:xss) = xs ++ concat xss
```

- Jällegi on rekursiooniskeemid samad!

Kõrgemat-järku funktsioonid

- Funktsioon `foldr`

$$\begin{aligned} &\texttt{foldr}\ (\oplus)\ e\ \ (x_1 : (x_2 : (x_3 : \texttt{[]}))) \\ &\quad\Longrightarrow\ (x_1 \oplus (x_2 \oplus (x_3 \oplus e))) \end{aligned}$$

- Definitsioon:

```
foldr :: (a -> b -> b) -> b -> [a] -> b
foldr f b []     = b
foldr f b (x:xs) = f x (foldr f b xs)
```

Kõrgemat-järku funktsioonid

- `foldr` näiteid:

```
sum    = foldr (+) 0
concat = foldr (++) []
and    = foldr (&&) True
or     = foldr (||) False
length = foldr (\_ n -> 1+n) 0
map f  = foldr (\x ys -> f x : ys) []
```


Kõrgemat-järku funktsioonid

- Funktsioon `foldl`

$$\texttt{foldl}\ (\oplus)\ e\ (x_1\ :\ (x_2\ :\ (x_3\ :\ \texttt{[]}))) \Longrightarrow (((e \oplus x_1) \oplus x_2) \oplus x_3)$$

- Definitsioon:

```
foldl :: (a -> b -> a) -> a -> [b] -> a
foldl f a []     = a
foldl f a (x:xs) = foldl f (f a x) xs
```

Kõrgemat-järku funktsioonid

- `foldl` näiteid:

```
sum     = foldl (+) 0
product = foldl (*) 1
length  = foldl (\ a _ -> 1+a) 0
reverse = foldl (\ a x -> x:a) []
decimal = foldl (\ a x -> 10*a+x) 0
```