Programmering og Problemløsning
Datalogisk Institut, Københavns Universitet
Uge(r)seddel 6 – gruppeopgave

Revideret 5/10 af Torben Mogensen

Deadline 20. oktober

I denne periode skal I arbejde grupper. Formålet er at arbejde med endelige træer.

Vi arbejder i opgaverne med en træstruktur til at beskrive geometriske figurer med farver. For at gøre det muligt at afprøve jeres opgaver, udleveres et simpelt bibliotek `makeBMP.dll`, der kan lave bitmapfiler. I dette bibliotek findes kun en funktion:

```
makeBMP : string -> int -> int -> (int*int -> int*int*int) -> unit
```

Det første argument er navnet på den ønskede bitmapfil (uden extension). Det andet argument er bredden af billedet i antal pixel, det tredje element er højden af billedet i antal pixel og det sidste argument er en funktion, der afbilder koordinater i billedet til farver. En farve er en tripel af tre tal mellem 0 og 255 (begge inklusive), der beskriver hhv. den røde, grønne og blå del af farven.

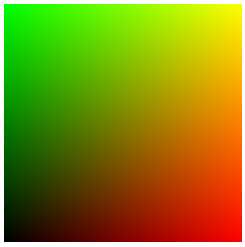
Koordinaterne starter med $(0, 0)$ i nederste venstre hjørne og $(w-1, h-1)$ i øverste højre hjørne, hvis bredde og højde er hhv. w og h. For eksempel vil en programfil `testBMP.fsx` med indholdet

```
makeBMP.makeBMP "test" 256 256 (fun (x,y) -> (x,y,0))
```

kunne oversættes med kommandoen

```
fsharpc testBMP.fsx -r makeBMP.dll -o testBMP.exe
```

til et program, der kan køres med kommandoen `mono testBMP.exe` og lave en billedfil med navnet `test.bmp`, der indeholder følgende billede:



Bemærk, at alle programmer, der bruger `makeBMP` skal oversættes med `-r makeBMP.dll` som en del af kommandoen. Tilsvarende skal man bruge `fsharpi -r makeBMP.dll` for at bruge `makeBMP` i det interaktive system.

NB! Nogle har haft problemer med at bruge biblioteket `makeBMP.dll` ved først at oversætte med `fsharpc -r makeBMP.dll` og siden `mono` på den oversatte fil. Hvis du får det problem, så prøv at køre `fsharpi -r makeBMP.dll testBMP.fsx` eller lignende for dine egne programmer. **Update 7/10:** En anden løsning er at finde filen `FSharp.Core.dll` og kopiere den til den folder, hvorfra Mono køres. Så kan Mono finde dette bibliotek.

Bemærk endvidere, at LaTeX ikke kan inkludere BMP-filer med `includegraphics`. Men man kan bruge diverse billedprogrammer (IrfanView, osv.) til at konvertere BMP til PNG.

I Linux kan man konvertere billeder med kommandoen `convert`. Hvis vi har billedfilen `test.bmp`, vil kommandoen `convert test.bmp test.png` lave en tilsvarende billedfil `test.png` i PNG-formatet (som LaTeX *kan* håndtere).

Vi arbejder i opgaverne med følgende datastruktur:

```
type point = int * int // (x, y)
type colour = int * int * int  // (red, green, blue), 0..255

type figure = Circle of point * int * colour
              // center, radius^2, colour
            | Rectangle of point * point * colour
              // bottom-left, top-right, colour
            | Mix of figure * figure
              // combine figures with average colours at overlap
```

Bemærk, at vi angiver kvadratet på radius af cirklen i stedet for blot radius.

For eksempel kan man lave følgende funktion til at finde farven af en figur i et punkt (forudsat, at punktet ligger i figuren):

```
// finds colour of figure at point
let rec colourAt (x,y) = function
  | Circle ((cx,cy), r2, col) ->
      if (x-cx)*(x-cx)+(y-cy)*(y-cy) <= r2
      then Some col else None
  | Rectangle ((x0,y0), (x1,y1), col) ->
      if x0<=x && x <= x1 && y0 <= y && y <= y1
      then Some col else None
  | Mix (f1, f2) ->
      match (colourAt (x,y) f1, colourAt (x,y) f2) with
      | (None, c) -> c
      | (c, None) -> c
      | (Some (r1,g1,b1), Some (r2,g2,b2)) ->
          Some ((r1+r2)/2, (g1+g2)/2, (b1+b2)/2)
```

Opgaverne i denne uge er delt i øve- og afleveringsopgaver.

Øveopgaverne er:

Ø6.1. Lav en figur af typen `figure`, der består af en rød cirkel med centrum i (50,50) og radius $\sqrt{2000}$ samt en blå rektangel med hjørnerne (40,40) og (90,110).

Ø6.2. Brug `makeBMP` og `colourAt` til at lave en funktion

`makePicture : string -> figure -> int -> int -> unit`

sådan at kaldet `makePicture` *`filnavn figur b h`* laver en billedfil ved navn *`filnavn`*`.bmp` med et billede af *`figur`* med bredde *`b`* og højde *`h`*.

På punkter, der ingen farve har (jvf. `colourAt`), skal farven være grå (som defineres med RGB-værdien (128,128,128)).

Du kan bruge denne funktion til at afprøve dine opgaver.

Ø6.3. Lav med `makePicture` en billedfil med navnet `63.bmp` og størrelse 100×150 (bredde 100, højde 150), der viser figuren fra opgave Ø6.1.

Ø6.4. Lav en funktion `checkFigure : figure -> bool`, der undersøger, om en figur er korrekt: At kvadratradiusen i cirkler er ikke.negativ, at nederste venstre hjørne i en rektangel faktisk er nedenunder og til venstre for det øverste højre hjørne (bredde og højde kan dog godt være 0), og at farvekompenterne ligger mellem 0 og 255.

Ø6.5. Lav en funktion `move : figure -> int * int -> figure`, der givet en figur og en vektor flytter figuren langs vektoren.

Ø6.6. Lav en funktion `boundingBox : figure -> point * point`, der givet en figur finder hjørnerne (bund-venstre og top-højre) for den mindste akserette rektangel, der indeholder hele figuren. **Vink:** Man kan runde et kommatal x op til det nærmeste heltal med `int (System.Math.Ceiling` x`)`.

Ø6.7. Lav en funktion `scale : figure * real -> figure`, der skalerer en figur med en skala angivet som et positivt kommatal. Alle værdier i den nye figur skal afrundes til nærmeste heltal. NB! Husk at cirkler angiver kvadratet på radius!. Hvis skalaen er ≤ 0.0, skal en passende fejlmeddelse gives. **Vink:** Man kan runde et kommatal x af til det nærmeste heltal med `int (System.Math.Round` x`)`.

Ø6.8. Lav en funktion `flipVertical : figure * int -> figure`, der givet en figur og et x, spejler figuren i den lodrette akse omkring x. NB! Pas på hjørnerne af rektangler.

Afleveringsopgaven er:

Vi udvider typen `figure` til en ny type `figure'` med følgende definition:

```
type figure' = Circle of point * int * colour
               // center, radius^2, colour
             | Rectangle of point * point * colour
               // bottom-left, top-right, colour
             | Mix of figure' * figure'
               // combine figures with average colours at overlap
             | Ellipse of point * point * int * colour
               // Two focal points, great axis, colour
```

Som tilføjer ellipser givet ved to brændpunkter og en storakse. Et punkt er i en ellipse, hvis *summen* af afstandende mellem punktet og de to brændpunkter er mindre end eller lig med storaksen. Se evt. `https://da.wikipedia.org/wiki/Ellipse_%28geometri%29`.

A6.1. Lav en figur `twoEllipses : figure'`, som består af en blå ellipse med brændpunkter i (10,10) og (80,80) og storakse 110 samt en gul ellipse med brændpunkter i (20,60) og (40,15) og storakse 70.

A6.2. Lav funktioner `colourAt'`, `makePicture'`, `checkFigure'`, `move'` og `scale'` svarende til de tilsvarende funktioner for `figure`.

Bemærk, at i en korrekt ellipse skal storaksen være mindst lige så stor som afstanden mellem brændpunkterne.

Afleveringsopgaven skal afleveres som både LaTeX, den genererede PDF, samt en fsx fil med løsningen for hver delopgave, navngivet efter opgaven (f.eks. `A6-1.fsx`), som kan oversættes med fsharpc, og hvis resultat kan køres med mono. Det hele samles i en zip-fil med sædvanlig navnekonvention (se tidligere ugesedler).

God fornøjelse

Ugens nød 3

Vi vil i udvalgte uger stille særligt udfordrende og sjove opgaver, som interesserede kan løse. Det er helt frivilligt at lave disse opgaver, som vi kalder "Ugens nød", men der vil blive givet en mindre præmie til den bedste løsning, der afleveres i Absalon.

Denne uges opgave omhandler billeder. Opgaven går i sin enkelhed ud på at bruge funktionen `makeBMP` til at lave det flotteste billede på 512×512 pixels. Instruktorer og undervisere vil fungere som dommere og udkåre det flotteste billede.

Den eneste begrænsning er, at programmet højest må være på 20 linjer på hver maksimalt 80 tegn. Blanke linjer og kommentarlinjer tæller ikke med.

Du skal aflevere et billede, det program, der blev brugt til at lave billedet, og en rapport, der beskriver, hvordan billedet er lavet.

Programmet skal kunne oversættes med kommandoen
`fsharpc -r makeBMP.dll` *`filnavn`*`.fsx`
og kunne køre i mono, eller køre med kommandoen
`fsharpi -r makeBMP.dll` *`filnavn`*`.fsx`
Dit program må ikke bruge andre imperative features end `makeBMP`-funktionen.