Learning to Program with F#
Exercises
Department of Computer Science
University of Copenhagen

Jon Sporring, Martin Elsman, Torben Mogensen, Christina Lioma

October 21, 2022

0.1 Canvas

0.1.1 Teacher's guide

Emne rekursion, grafik og winforms

Sværhedsgrad Middel

0.1.2 Introduction

I det følgende skal vi benytte os af biblioteket `ImgUtil`, som beskrevet i forelæsningerne. Biblioteket `ImgUtil` gør det muligt at tegne punkter og linier på et canvas, at eksportere et canvas til en billedfil (en PNG-fil), samt at vise et canvas på skærmen i en simpel F# applikation. Biblioteket (nærmere bestemt F# modulet `ImgUtil`) er gjort tilgængeligt via en F# DLL kaldet `img_util.dll`. Koden for biblioteket og dokumentation for hvordan DLL'en bygges og benyttes er tilgængelig via github på `https://github.com/diku-dk/img-util-fs`.

0.1.3 Exercise(s)

0.1.3.1: Create an empty canvas of width 400 pixels and height 600 pixels (400×600). Draw a red, green, blue, and yellow line on the left, bottom, right, and top edge, and a black 200×300 rectangle in the middle.

0.1.3.2: Consider points on a circle is give by the coordinate functions

$$x(t) = x_0 + r\cos(t) \tag{1}$$
$$y(t) = y_0 + r\sin(t) \tag{2}$$

whose center is (x_0, y_0), r is its radius, and where $0 \leq t < 2\pi$. Use these equations to make a canvas of size 400×400 with an approximation of a circle centered at $(200, 200)$ and with a radius of 100. The approximation should consists of straight lines connecting $(x(t_i), y(t_i))$ with $(x(t_{i+1}), y(t_{i+1}))$ for $t = [0, 0.1, 0.2, \ldots, 2\pi)$.

0.1.3.3: Make a program, which draws a 20×20 square in the center of a canvas of size 400×400. When the user presses space, or an arrow key, the program should write to the terminal, which key has been pressed.

0.1.3.4: Extend the program in Exercise 3, such that when the arrow keys are pressed, then the square is moved in the direction of the arrow pressed.

0.1.3.5: Vi skal nu benytte biblioteket `ImgUtil` til at tegne Sierpinski-fraktalen, der kan tegnes ved at tegne små firkanter bestemt af et rekursivt mønster. Koden for Sierpinski-trekanten er givet som følger:

```
open ImgUtil

let rec triangle C len (x,y) =
  if len < 25 then setBox blue (x,y) (x+len,y+len) C
```

```
  else let half = len / 2
       do triangle C half (x+half/2,y)
       do triangle C half (x,y+half)
       do triangle C half (x+half,y+half)

do runSimpleApp "Sierpinski" 600 600
                (fun w h ->
                   let C = mk w h
                   in (triangle C 512 (30,30); C))
```

Tilpas funktionen således at trekanten tegnes med røde streger samt således at den kun tegnes 2 rekursionsniveauer ned. **Hint:** dette kan gøres ved at ændre betingelsen `len < 25`. Til at starte med kaldes funktionen `triangle` med `len=512`, på næste niveau kaldes `triangle` med `len=256`, og så fremdeles.

0.1.3.6: I stedet for at benytte funktionen `ImgUtil.runSimpleApp` er det nu meningen at du skal benytte funktionen `ImgUtil.runApp`, som giver mulighed for at din løsning kan styres ved brug af tastaturet. Funktionen `ImgUtil` har følgende type:

```
val runApp : string -> int -> int
          -> (int -> int -> 's -> canvas)
          -> ('s -> Key -> 's option)
          -> 's -> unit
```

De tre første argumenter til `runApp` er vinduets titel (en streng) samt vinduets initielle vidde og højde. Funktionen `runApp` er parametrisk over en brugerdefineret type af tilstande (`'s`). Antag at funktionen kaldes som følger:

```
do runApp title width height draw react init
```

Dette kald vil starte en GUI applikation med titlen `title`, vidden `width` og højden `height`. Funktionen `draw`, som brugeren giver som 4. argument kaldes initielt når applikationen starter og hver gang vinduets størrelse justeres eller ved at funktionen `react` er blevet kaldt efter en tast er trykket ned på tastaturet. Funktionen `draw` modtager også (udover værdier for den aktuelle vidde og højde) en værdi for den brugerdefinerede tilstand, som initielt er sat til værdien `init`. Funktionen skal returnere et canvas, som for eksempel kan konstrueres med funktionen `ImgUtil.mk` og ændres med andre funktioner i `ImgUtil` (f.eks. `setPixel`).

Funktionen `react`, som brugeren giver som 5. argument kaldes hver gang brugeren trykker på en tast. Funktionen tager som argument:

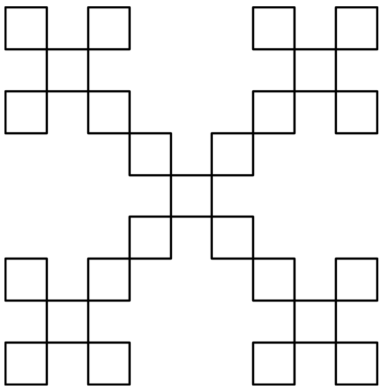
- en værdi svarende til den nuværende tilstand for applikationen, og
- et argument der kan benyttes til at afgøre hvilken tast der blev trykket på.[1]

Funktionen kan nu (eventuelt) ændre på dens tilstand ved at returnere en ændret værdi for tilstanden.

Tilpas applikationen således at dybden af fraktalen kan styres ved brug af piletasterne, repræsenteret ved værdierne `Gdk.Key.u` og `Gdk.Key.d`.

0.1.3.7: Med udgangspunkt i øvelsesopgave 5 skal du i denne opgave implementere en GUI-applikation der kan tegne en version af X-fraktalen som illustreret nedenfor (eventuelt i en dybde større end 2).

[1] Hvis `k` har typen `Gdk.Key` kan betingelsen `k = Gdk.Key.d` benyttes til at afgøre om det var tasten "d" der blev trykket på. Desværre er det ikke muligt med den nuværende version af `ImgUtil` at reagere på tryk på piletasterne.



Bemærk at det ikke er et krav, at dybden på fraktalen skal kunne styres med piletasterne, som det er tilfældet med Sierpinski-fraktalen i øvelsesopgave 6.