

Mandelbrot

Sadok Habibi Dalin

2015-02-20

1 Uppgiften

Uppgiften går ut på att visualisera mandelbrotmängden genom att implementera en bildgenerator i Erlang.

Uppgiften kan delas upp i mindre deluppgifter:

- Definiera en modul som kan utföra operationer med komplexa tal d.v.s. skapa, addera och kvadrera komplexa tal. Modulen ska också innehålla en funktion som returnerar absolutbeloppet av ett givet komplext tal.
- Definiera en funktion som givet en pixel beräknar ett komplext tal.
- Definiera en funktion som beräknar djupet i mandelbrotmängden givet ett komplext tal och maximala antalet iterationer.
- Definiera en funktion som givet ett djup räknar ut färgen på en pixel.
- Definiera en funktion som beräknar färgen på alla pixlar i ett rektangulärt plan.
- Använd den givna ppm.erl modulen för att skriva en genererad bild till en läsbar fil.
- Beskriv eventuella optimeringar som kan göras för att sänka tidskomplexiteten.

2 Ansats

Lösningar till deluppgifterna givna i uppgiftssektionen:

Definiera en modul som kan utföra operationer med komplexa tal, cmplx.erl:

Funktioner som är definierade i cmplx.erl:

- new(Real, Im). Returnerar ett imaginärt tal på formen {Real, Im}.

- $\text{add}(C1, C2)$. Returnerar $C1+C2 = \{\text{Real1} + \text{Real2}, \text{Im1} + \text{Im2}\}$.
- $\text{sqr}(C)$. Returnerar C^2 på formen $\{\text{Real}^2 - \text{Im}^2, 2 \cdot \text{Real} \cdot \text{Im}\}$.
- $\text{abs}(C)$. Returnerar $|C|$ d.v.s. $\sqrt{\text{Real}^2 + \text{Im}^2}$.

Definiera en funktion som översätter kordinaterna för en given pixel till ett komplext tal, Trans(Width, Height):

Returnerar ett komplext tal på formen $\{X+K^*(\text{Width}-1), Y-K^*(\text{Height}-1)\}$ där X är reella kordinatkomposanten för vänstra sidan av bilden och Y är den imaginära kordinatkomposanten för höjden på bilden.

Definiera en funktion som givet ett komplext tal och maximala antalet iterationer beräknar djupet i mandelbrotmängden, mandelbrot(C, Iterations):

`mandelbrot()` anropar en hjälpfunktion `test(I, Z0, C, Iterations)` där I är antalet iterationer hittils d.v.s. 0 och Z0 är en iterationsvariabel som representerar $z_{n+1} = z_n + c$ d.v.s. 0.

`test()` returnerar I om $|Z0| > 2$. Annars returneras `test(I+1, Z02 + C, C, Iterations)`. Om I == Iterations returneras 0. För vissa punkter så blir $|Z0|$ aldrig större än 2 oavsett antal iterationer. För att undvika att programmet försöker beräkna djupet för dessa punkter i all oändlighet så itereras formeln $z_{n+1} = z_n + c$ endast igenom Iterations gånger.

Definiera en funktion som givet djupet för en punkt och maximala möjliga djupet för en punkt beräknar färgen på en pixel, convert(Depth, MaxDepth):

5 olika färgskalor används för att beräkna färgen på pixlar med olika djup. Färgskalorna genereras m.h.a. decimalerna till $A = \frac{4*Depth}{MaxDepth}$. Beroende på vad A avrundas till för heltalet returneras olika färger på formen $\{R, G, B\}$. Den avrundade variabeln. $Y = 255*(A-\text{trunc}(A))$ används i returtuppeln för att generera färgtoner baserat på en given pixels djup.

Definiera en funktion som beräknar färgen på alla pixlar i ett rektangulärt plan, rows(Width, Height, Trans, Depth, Image):

Width och Height är den genererade bildens dimensioner. Depth är maximala möjliga djupet som en punkt kan ha d.v.s. maximala antalet gånger formeln $z_{n+1} = z_n + c$ kan itereras igenom för en given punkt. Trans är en funktion som översätter en pixels kordinater till ett komplext tal.

`rows()` anropar en funktion `row(Width, IterHeight, Trans, MaxDepth, Row, IterWidth)` Height antal gånger d.v.s. 1 gång för varje rad i bilden som ska genereras. `row()` returnerar en behandlad rad på formen $\{[R, G, B], [R, G, B], \dots\}$ d.v.s. en lista med Width antal tuppler. De behandlade raderna ad-

deras till en resultatlista som returneras efter att `rows()` har itererat genom alla rader.

`row(Width, IterHeight, Trans, MaxDepth, Row, IterWidth)` översätter en rad med pixlar till en rad med färger m.h.a. 3 funktioner:

- `Trans(Width, Height): (IterWidth, IterHeight) → C.`
- `mandelbrot(C, Iterations): (C, MaxDepth) → Depth.`
- `convert(Depth, MaxDepth): (Depth, MaxDepth) → {R, G, B}.`

$\{R, G, B\}$ adderas sedan till en resultatlista som returneras efter att `row()` har itererat genom alla pixlar i raden.

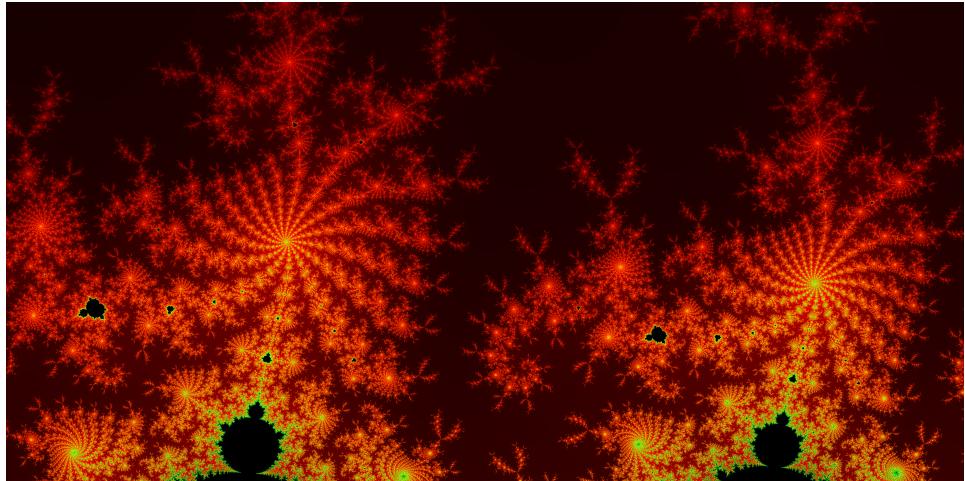
Optimeringar som kan göras för att sänka tidskomplexiteten:

Genom att utnyttja flera processorkärnor kan prestandan förbättras avsevärt. Istället för att behandla alla rader i en bild på 1 process så ”spawnar” jag en process för varje rad. Efter att alla processer som behandlar rader är startade anropas funktionen `rowConcat(Height, Image)`. `rowConcat()` väntar i ett receive-uttryck på behandlade rader på formen $\{\text{RowOrder}, \text{Row}\}$ där `RowOrder` är radens `IterHeight`. När en rad tas emot adderas raden till en resultatlista `NewImage` som sorteras m.h.a. `insertionsort` i stigande `RowOrder` ordning. Sedan returneras `rowConcat(Height-1, NewImage)`. När `Height == 0` returneras en färdig bild.

Alla operationer som involverar konkatinering av listor kan optimeras. Genom att addera nya element från vänster istället för höger så kan tidskomplexiteten sänkas från $O(N^2)$ till $O(1)$.

3 Utvärdering

Nedan är en bild genererad med inställningarna $(\text{Width}, \text{Height}, \text{X}, \text{Y}, \text{X1}, \text{Depth}) = (6000, 3000, -0.1362, 0.8408, -0.132, 2048)$:



Nedan är givet tiden det tar att generera en bild när 1 process respektive Height processer behandlar raderna i bilden:

(Width, Height, X, Y, X1, Depth) = (960, 540, -2.6, 1.2, 1.6, 128)

Metod	Tid
1 process	7.2s
Height processer	4.6s

Tabell 1: Tiden det tar att generera en bild.

(Width, Height, X, Y, X1, Depth) = (1920, 1080, -2.6, 1.2, 1.6, 512)

Metod	Tid
1 process	88s
Height processer	54s

Tabell 2: Tiden det tar att generera en bild.

Att använda Height processer för att behandla raderna minskar körtiden avsevärt. Genom att använda mergesort och sortera 1 gång istället för att använda insertion sort och sortera Height gånger så kan körtiden eventuellt minskas ännu mer.

4 Sammanfattning

Det största problemet som uppstod under utvecklingen av programmet var ett tidsproblem. P.g.a. dåligt optimerad kod så tog det lång tid att generera bilder med hög upplösning och högt djup. Ett minnesproblem uppstod också när bilder med hög upplösning skulle genereras eftersom RAM-minnet tog slut och programmet kraschade.