Bachelor-Praktikum: Rechnerarchitektur

Tricorn-Fraktal

Shutao Shen Zhiyi Pan Yujie Zhang

Department of Informatics

January 28, 2020

Contents

Outline

Markroanalyse

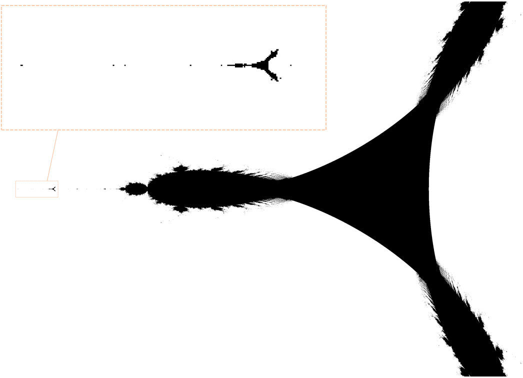
Tricorn Fraktal

1.ähnlich wie Mandelbrot Menge 2.Selbstähnlichket

Formel

$$z_{i+1} = \bar{z}_i^2 + c (i \geq 0) \quad z_0 = 0 \tag{1}$$

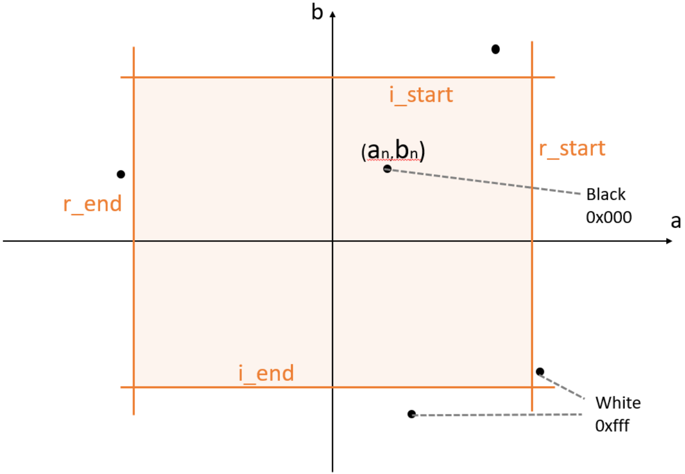
$$c = a + bi \quad a \in [-2; 1] \text{ und } b \in [-1; 1] \tag{2}$$



Markroanalyse

void multicorn

void multicorn(float r_start, float r_end, float i_start, float i_end, float res, unsigned char *img)



Markroanalyse

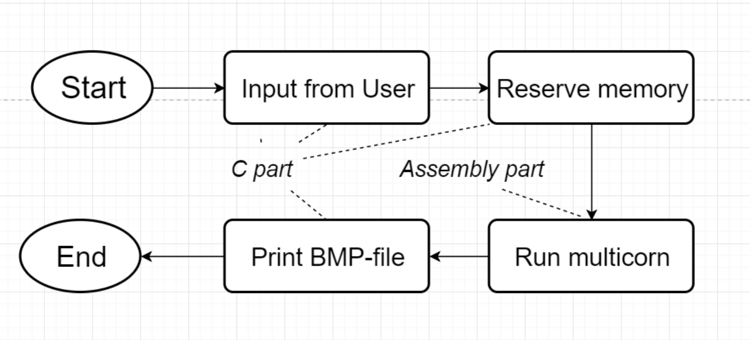
void multicorn

void multicorn
numA ← getColumn
numB ← getRow
for numB ≥ 0
b ← b + res
for numA ≥ 0
a ← a + res
numIter ← getIter
for numIter ≥ 0
calculateParallel
isINForNAN
testBoundary
writeColor

1.a,b :float
->xmm-Reg
2.Bearbeitung mit Menge
-> SIMD

Markroanalyse

ganzes Verfahren



Outline

2 Implementierung Detail

- Besonderer Wert
- Rundungsfehler
- For-schleife statt while-schleife
- SIMD Randfall
- Anzahl von Iteration
- Ausgabe: BMP

Implementierung

Besonderer Wert

NAN(Keine Zahl)

Binärdarstellung:

x[11111111]$_{(exp)}$xxxxxxxxxxxxxxxxxxxxxxx

$$„\text{UCOMISS}“: „\text{UNORDERED}(ZF, PF, CF \leftarrow 111)“$$

INF(Unendlichkeit)

Binärdarstellung

x[11111111]$_{(exp)}$00000000000000000000000

$$|x| > float.max(0x7f7fffff)$$

Der Rundungsfehler[1]

Dezimal	Gespeicherter Wert	Hexadezimal
0.1	0.100000001490116119384765625	0x3dcccccd
0.5	0.5	0x3f000000
0.15	0.1500000059604644775390625	0x3e19999a

$(0.15 - 0.1) == (0.1 - 0.05)$ // false
$|(0.15 - 0.1) - (0.1 - 0.05)| < \mathit{Genauigkeit}$ // true

Implementierung

For-schleife statt while-schleife

Die Abtastung

Pixels: schrittweise ,,res“ im Raume [-2,1][-1,1] abtasten.

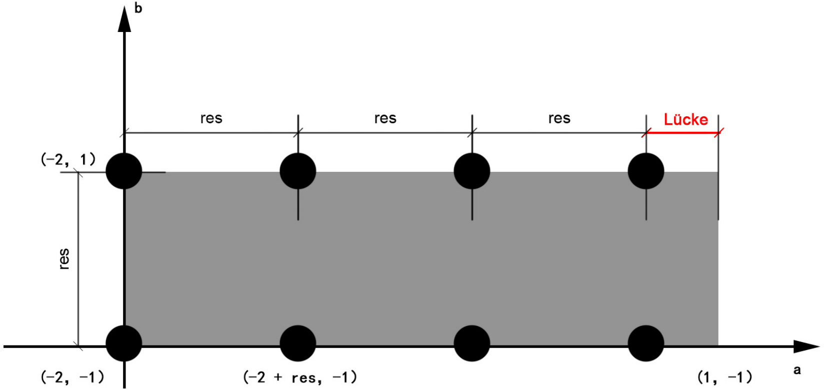


Figure: Abtastung, *a*, *b*

Implementierung

For-schleife statt while-schleife

While-schleife

$|float_a - a_{end}| < Genauigkeit$

Lücke oder Rundungsfehler?

Beispiel

$a_{end} = 2, float_a = 1.999953...,$

wobei res= 0.0001, Genauigkeit=?

Implementierung

For-schleife statt while-schleife

For-schleife

$a_{number} = \lfloor float_x \rfloor + 1$

wir berechnen $a_{number} = x$ durch folgende Formel:

$$\text{Sei } e = \lceil x \rceil - x, x \Leftarrow \begin{cases} round(x) + 1, & \text{wenn } e < 0.01 \\ \lfloor x \rfloor + 1, & \text{sonst} \end{cases} \tag{3}$$

Die Genauigkeit ist einfach zu bestimmen(fest), z.B. $p = 0.01$

Beispiel

Randfall:

$float_x = 3.99999, e = 0.00001, a_{number} = round(3.99999) + 1 = 5$

Implementierung

SIMD Randfall

RCX als Zähler

RCX $\geq$ 4: gültige Daten $=$ 4

rcx>4				rcx>4				rcx=3			
-2.0	-1.7	-1.4	-1.1	-0.8	-0.5	-0.2	0.1	0.4	0.7	1.0	
-2.0	-1.7	-1.4	-1.1	-0.8	-0.5	-0.2	0.1	0.4	0.7	1.0	
-2.0	-1.7	-1.4	-1.1	-0.8	-0.5	-0.2	0.1	0.4	0.7	1.0	
-2.0	-1.7	-1.4	-1.1	-0.8	-0.5	-0.2	0.1	0.4	0.7	1.0	
-2.0	-1.7	-1.4	-1.1	-0.8	-0.5	-0.2	0.1	0.4	0.7	1.0	

Figure: 1. Iteration

Implementierung

SIMD Randfall

RCX als Zähler

RCX $\geq$ 4: gültige Daten $=$ 4

rcx>4				rcx>4				rcx=3		
-2.0	-1.7	-1.4	-1.1	-0.8	-0.5	-0.2	0.1	0.4	0.7	1.0
-2.0	-1.7	-1.4	-1.1	-0.8	-0.5	-0.2	0.1	0.4	0.7	1.0
-2.0	-1.7	-1.4	-1.1	-0.8	-0.5	-0.2	0.1	0.4	0.7	1.0
-2.0	-1.7	-1.4	-1.1	-0.8	-0.5	-0.2	0.1	0.4	0.7	1.0
-2.0	-1.7	-1.4	-1.1	-0.8	-0.5	-0.2	0.1	0.4	0.7	1.0

Figure: 2. Iteration

Implementierung

SIMD Randfall

RCX als Zähler

RCX $<$ 4: gültige Daten $=$ *RCX*

Rechnen mit Padding-Daten, RCX als Zähler

rcx>4				rcx>4				rcx=3			
-2.0	-1.7	-1.4	-1.1	-0.8	-0.5	-0.2	0.1	0.4	0.7	1.0	X
-2.0	-1.7	-1.4	-1.1	-0.8	-0.5	-0.2	0.1	0.4	0.7	1.0	
-2.0	-1.7	-1.4	-1.1	-0.8	-0.5	-0.2	0.1	0.4	0.7	1.0	
-2.0	-1.7	-1.4	-1.1	-0.8	-0.5	-0.2	0.1	0.4	0.7	1.0	
-2.0	-1.7	-1.4	-1.1	-0.8	-0.5	-0.2	0.1	0.4	0.7	1.0	

Figure: 3. Iteration

Implementierung

Anzahl von Iteration

Statistisches Verfahren

Standard: 500 Iterationen gebildeten Bild

Algorithmus:

```
for res: 0.001 to 1, step: 0.001:
        Iteration-number erhöht sich;
loop-bis kontinuierlich fuenf-mal 99.9%ige Ähnlichkeit
```

Statistisches Verfahren

```
Arguments:
r_start:-2.000000, r_end:2.000000, i_start:-2.000000, i_end:2.000000
res: from 0.001000 to 1, step:0.001000, total:1000
  res:0.001000 => min_Iteration_num:219
  res:0.002000 => min_Iteration_num:216
  res:0.003000 => min_Iteration_num:220
             .............
  res:0.998000 => min_Iteration_num:5
  res:0.999000 => min_Iteration_num:6
  res:1.000000 => min_Iteration_num:2
```

Figure: Statistisches Ergebnis

Implementierung

Anzahl von Iteration

Statistisches Verfahren

Es gibt Ausreißer in den Daten

TRIMMEAN(array, 0.1) in Excel: Daten von 5% bis 95%

0,5	221	241	221	213	225	237	238
7,5	235	206	214	212	217	210	248
210	216	215	208	213	217	200	230
242	232	210	217	205	243	218	850
209	222	235	213	211	217	245	821

Figure: TRIMMEAN

Implementierung

Ausgabe: BMP

BMP

BMP: header(54) | option(0) | [Blue(1),Green(1),Red(1)].....
Jede Zeile: vielfaches von 4 Bytes, sonst padding 0.

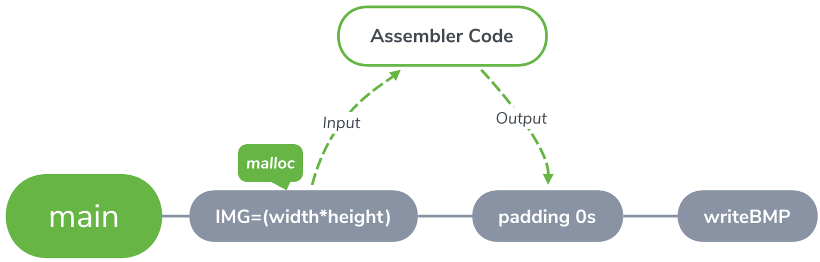


Figure: Ausgabe-BMP

Outline

3 Weiterführung
- Genauigkeit
- Performanzanalyse
- Dokumentation der Implementierung
- Zusammenfassung

Weiterführung

Genauigkeit

```
n ←1
while n ≤ 99
    do r_start = -2^n
    r_end = 2^n
    i_start = -2^n
    i_end = 2^n
    res ← 0.001
    while res ≤ 1
        do bild_generieren
        bild_vergleichen
        res_genauigkeitsrate_rechen
        res ←res+0.001
    end
    grenze_ genauigkeitsrate ← average(res_genauigkeitsrate)
    n ←n+1
end
genauigkeitsrate ← average(grenze_genauigkeitsrate)
```

r_start=-2, r_end=2, i_start=-2, i_end=2, res=0.001
r_start=-2, r_end=2, i_start=-2, i_end=2, res=0.002
r_start=-2, r_end=2, i_start=-2, i_end=2, res=0.003
....
r_start=-4, r_end=4, i_start=-4, i_end=4, res=0.001
....
r_start=-2^99, r_end=2^99, i_start=-2^99, i_end=2^99, res=1

Figure: Algorithmus zur Berechnung der Genauigkeit

Weiterführung

Genauigkeit

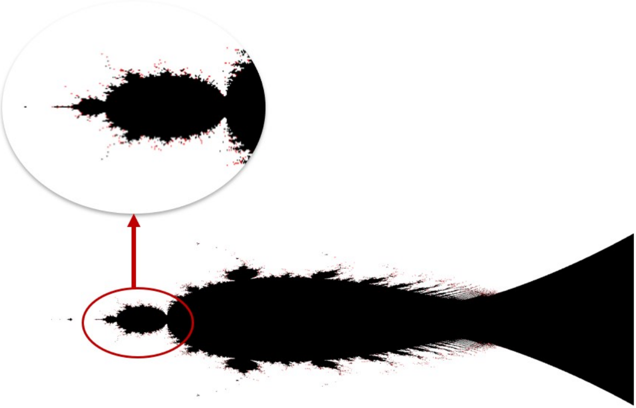


Figure: Fehleranalyse

Weiterführung

Performanzanalyse

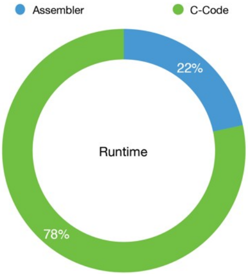


Figure: Vergleich von der Laufzeit zwischen C-Code und Assembler

Weiterführung

Performanzanalyse

Assembler - Code

```
subps xmm10 , xmm14
addps xmm14 , xmm13
mulps xmm10 , xmm14
```

Disassembly - Code

```
addss xmm1 , DWORD PTR [ rbp -0 x34 ]
movss xmm0 , DWORD PTR [ rbp -0 x38 ]
subss xmm0 , DWORD PTR [ rbp -0 x34 ]
mulss xmm0 , xmm1
```

Weiterführung

Dokumentation

Benutzer-Dokumentation

- -a, –r start
- -b, –r end
- -c, –i start
- -d, –i end
- -r, –res
- -o, –output location
- -h, –help

Entwickler-Dokumentation

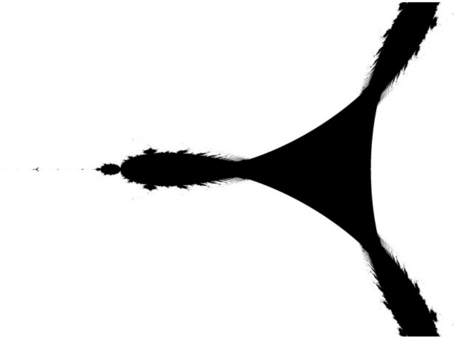
- all
- example
- testIterationNum
- testCorrection
- testPerformance
- clean
- buildWithClmp

Weiterführung

Zusammenfassung

- Die Eigenschaft vom Tricorn-Fraktal
- Fließkommazahlen
- BMP-Datei
- SIMD

Vielen Dank für Ihre Aufmerksamkeit!



Wikipedia.
Ieee 754 — wikipedia, die freie enzyklopädie, 2020.
[Online; Stand 26. Januar 2020].