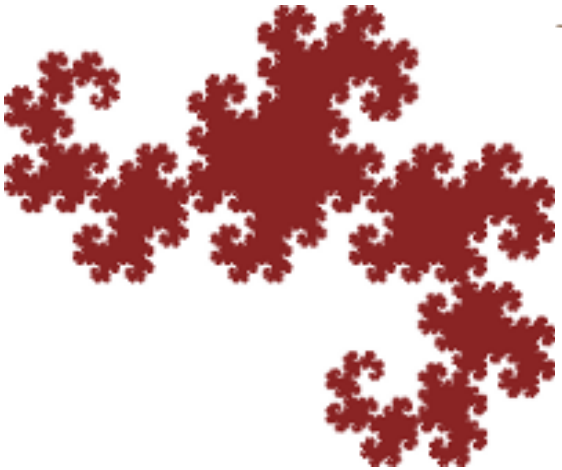
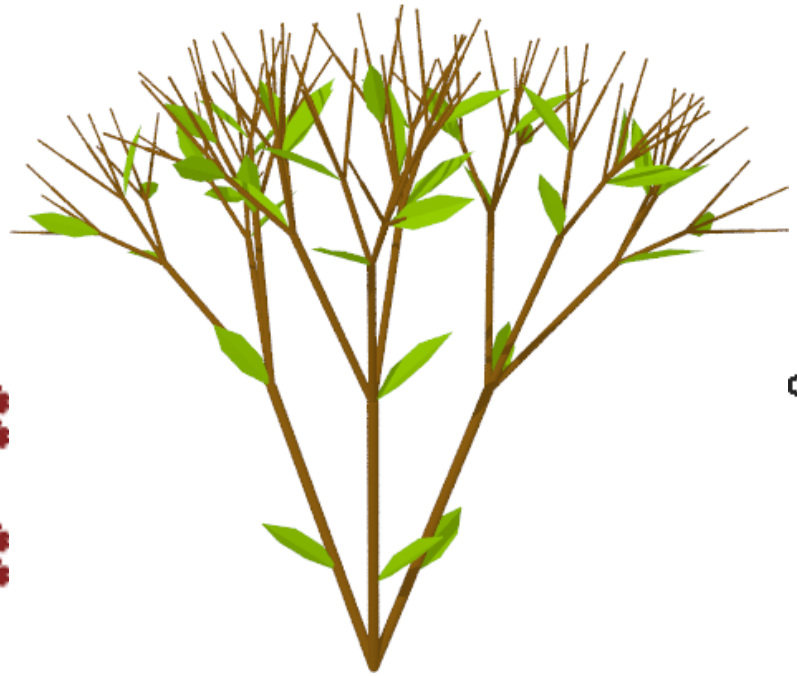
Lindenmayer Systems

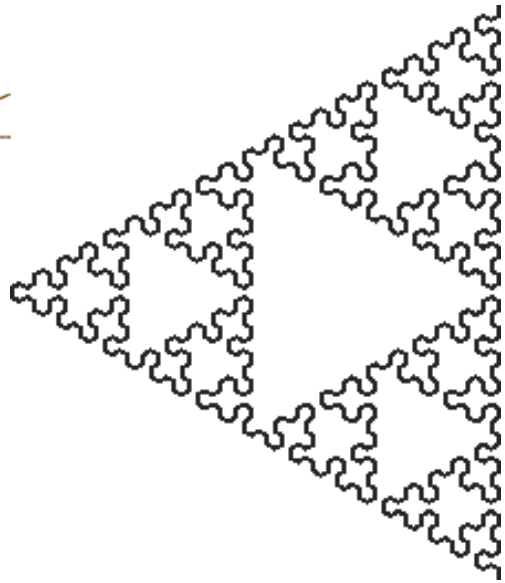
An efficient implementation

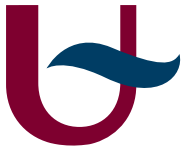
Wouter ibens

Universiteit Antwerpen

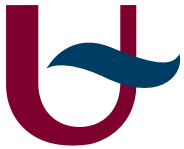






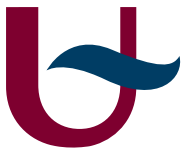


Inhoud



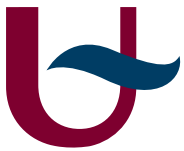
Turtle graphics

- Een alfabet, verzameling acties en een mapping hiertussen
- Een hoek σ
- Een pen met locatie en richting



Turtle graphics - Voorbeeld

- Alfabet $= \{$F, f, $+$, -$\}$
- $\sigma = 60^\circ$
- Acties $= \{$Teken lijn van lengte 1, beweeg vooruit met lengte 1, draai σ linksom, draai σ rechtsom$\}$



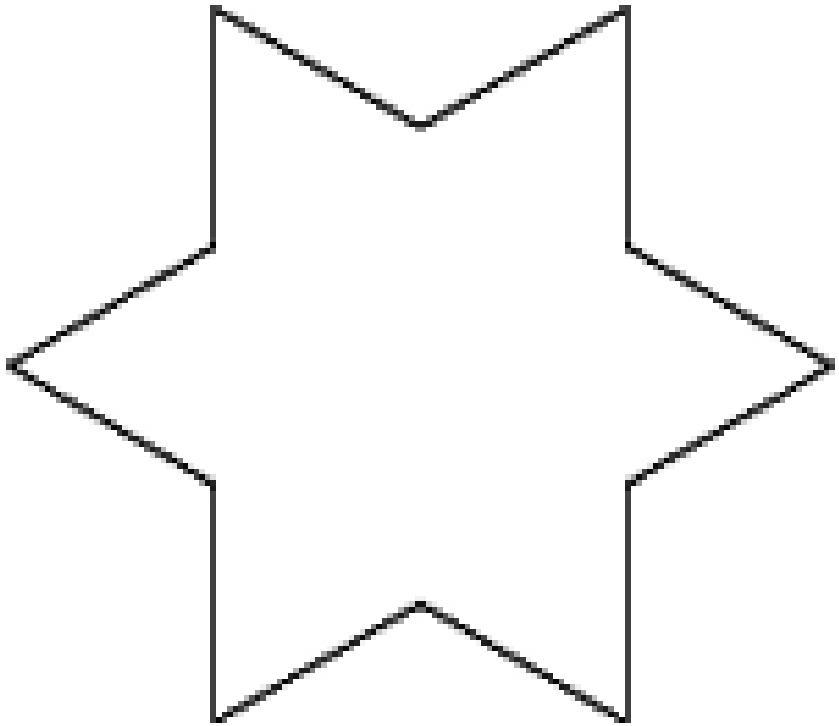
Turtle graphics - Voorbeeld

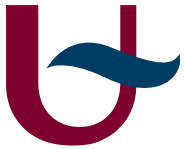
- Alfabet = {F, f, +, -}
- $\sigma = 60°$
- Acties = {Teken lijn van lengte 1, beweeg vooruit met lengte 1, draai σ linksom, draai σ rechtsom}
- Mapping:

  F $\rightarrow$ Teken lijn
  f $\rightarrow$ Beweeg vooruit (zonder te tekenen)
  + $\rightarrow$ Draai σ linksom
  - $\rightarrow$ Draai σ rechtssom

Turtle graphics - Voorbeeld

$$F + F - -F + F - -F + F - -F + F - -F + F - -F + F$$





L-Systemen

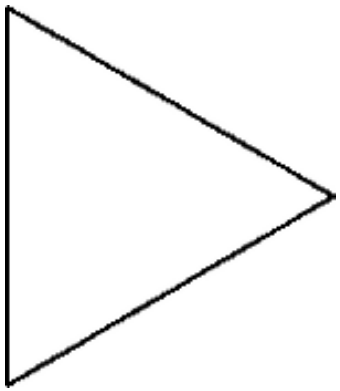
- De strings voor turtle graphics dynamisch opbouwen
- We voegen een beginstring (axiom) een een formele grammar toe.

L-Systemen

- De strings voor turtle graphics dynamisch opbouwen
- We voegen een beginstring (axiom) een een formele grammar toe.
- Voorbeeld:
  - Axiom $= F - -F - -F$
  - Grammar: $F \rightarrow F + F - -F + F$

L-Systemen - Voorbeeld

Iteratie 0
(Axiom)



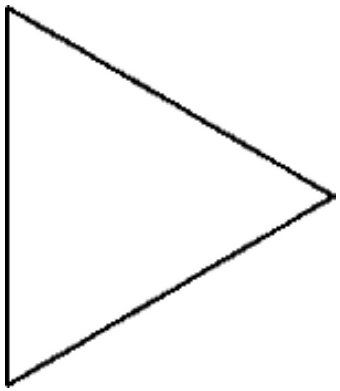
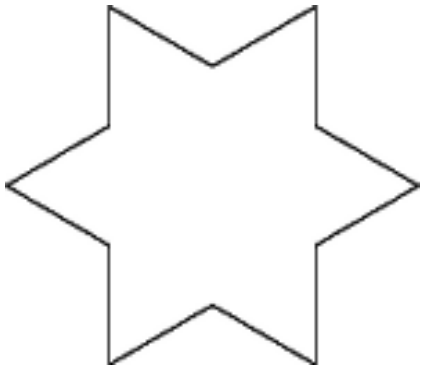
Iteratie 1

$F - -F - -F$

Iteratie 2

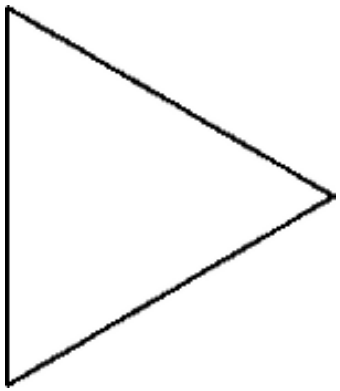
Iteratie 3

L-Systemen - Voorbeeld

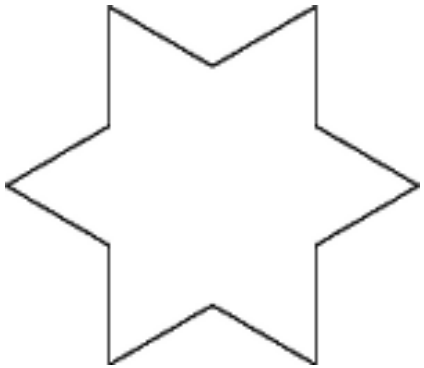
Iteratie 0 (Axiom)	Iteratie 1	Iteratie 2	Iteratie 3
			
	$F - -F - -F$		
	$\underline{F + F - -F + F} - -\underline{F + F - -F + F} - -\underline{F + F - -F + F}$		

L-Systemen - Voorbeeld

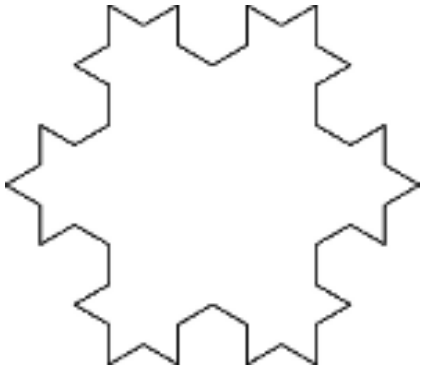
Iteratie 0 (Axiom)



Iteratie 1



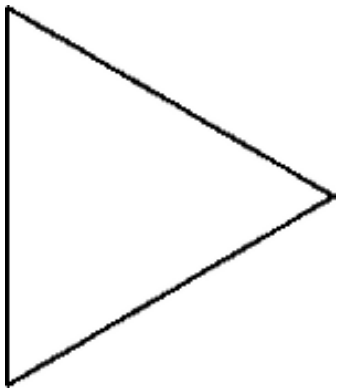
Iteratie 2



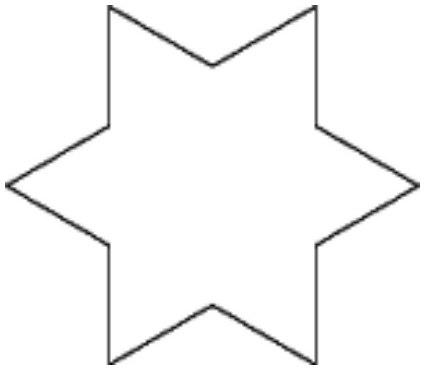
Iteratie 3

L-Systemen - Voorbeeld

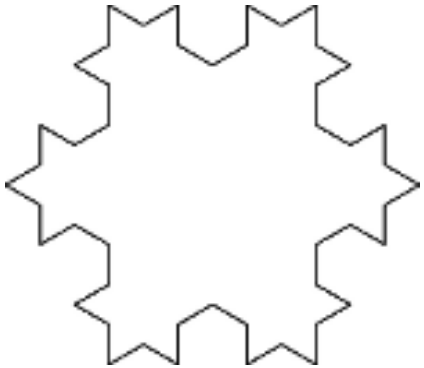
Iteratie 0 (Axiom)



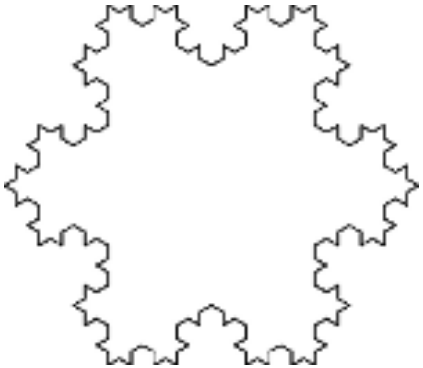
Iteratie 1



Iteratie 2

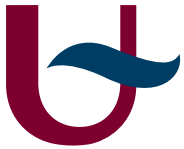


Iteratie 3



L-Systemen - Uitbreidingen

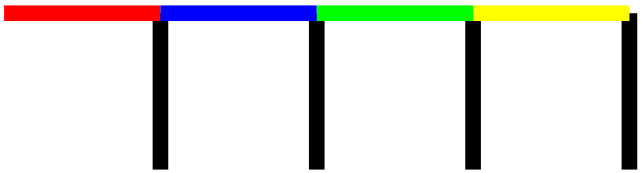
Brackets*	Een stackimplementatie om deelbomen te genereren en verder te gaan op de originele locatie.
3D	Een derde dimensie
Polygonen*	Lijnentekening vullen
Stijlen	Dikte, lengte en kleur aanpassen
Parameters	Acties verfijnen dankzij parameters
Stochastisch	Verschillende productieregels per symbool, met bepaalde kans

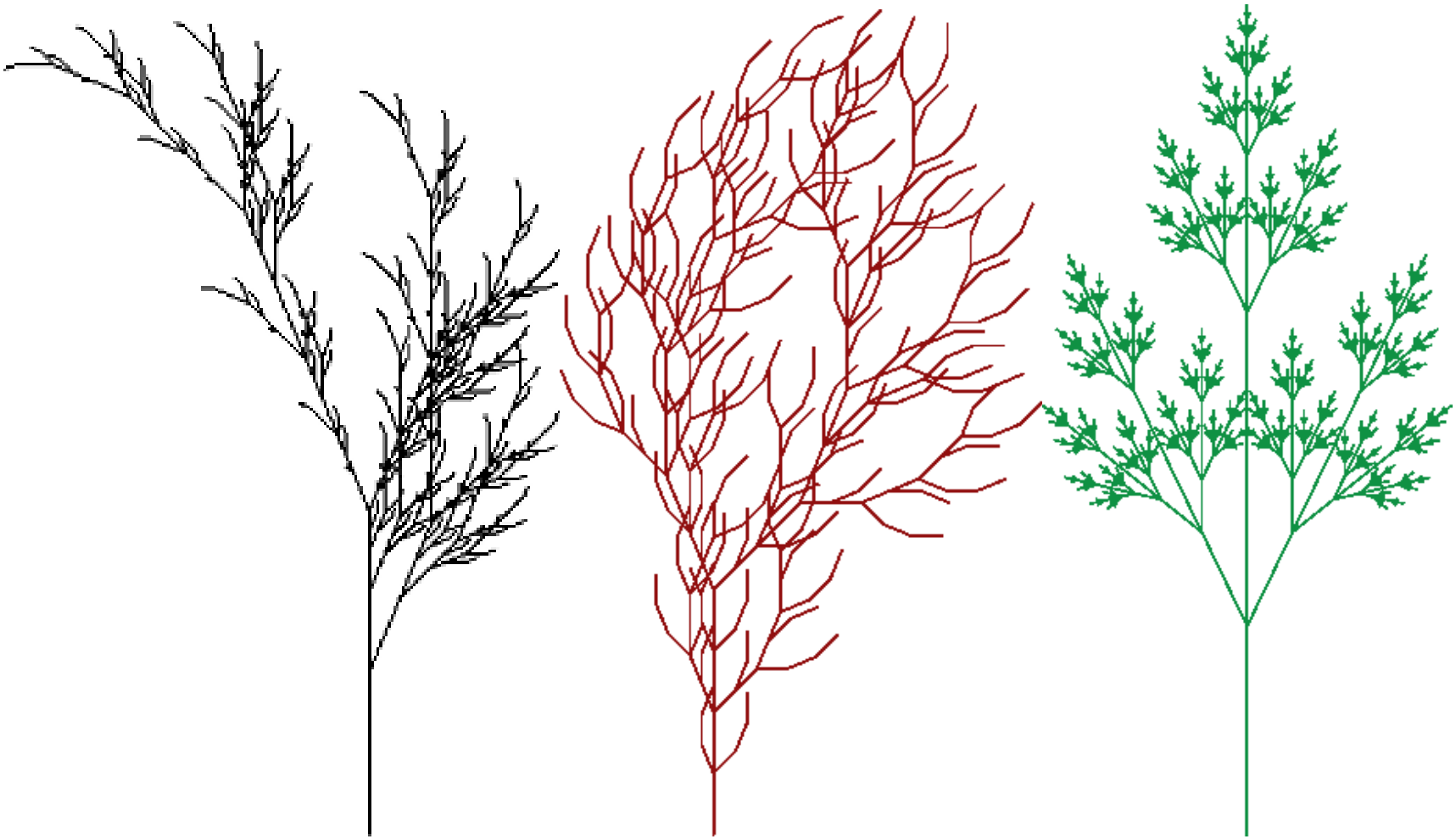


Brackets

- Brackets pushen/poppen de huidige staat op een stack
- Hierdoor kan een boom takken (met takken, met takken ...) hebben

Brackets

- Brackets pushen/poppen de huidige staat op een stack
- Hierdoor kan een boom takken (met takken, met takken ...) hebben
- F[+F]F[+F]F[+F]F[+F] ($\sigma = 90°$)
- 

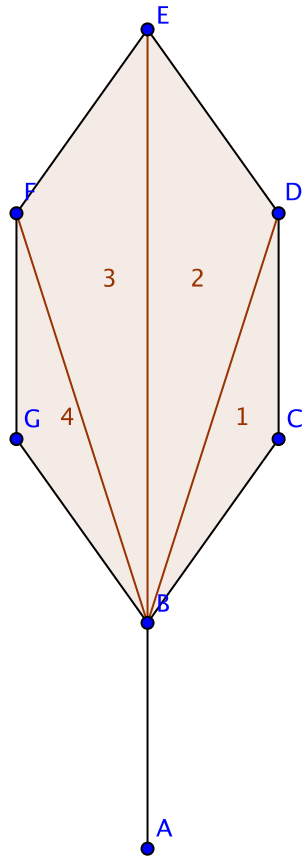


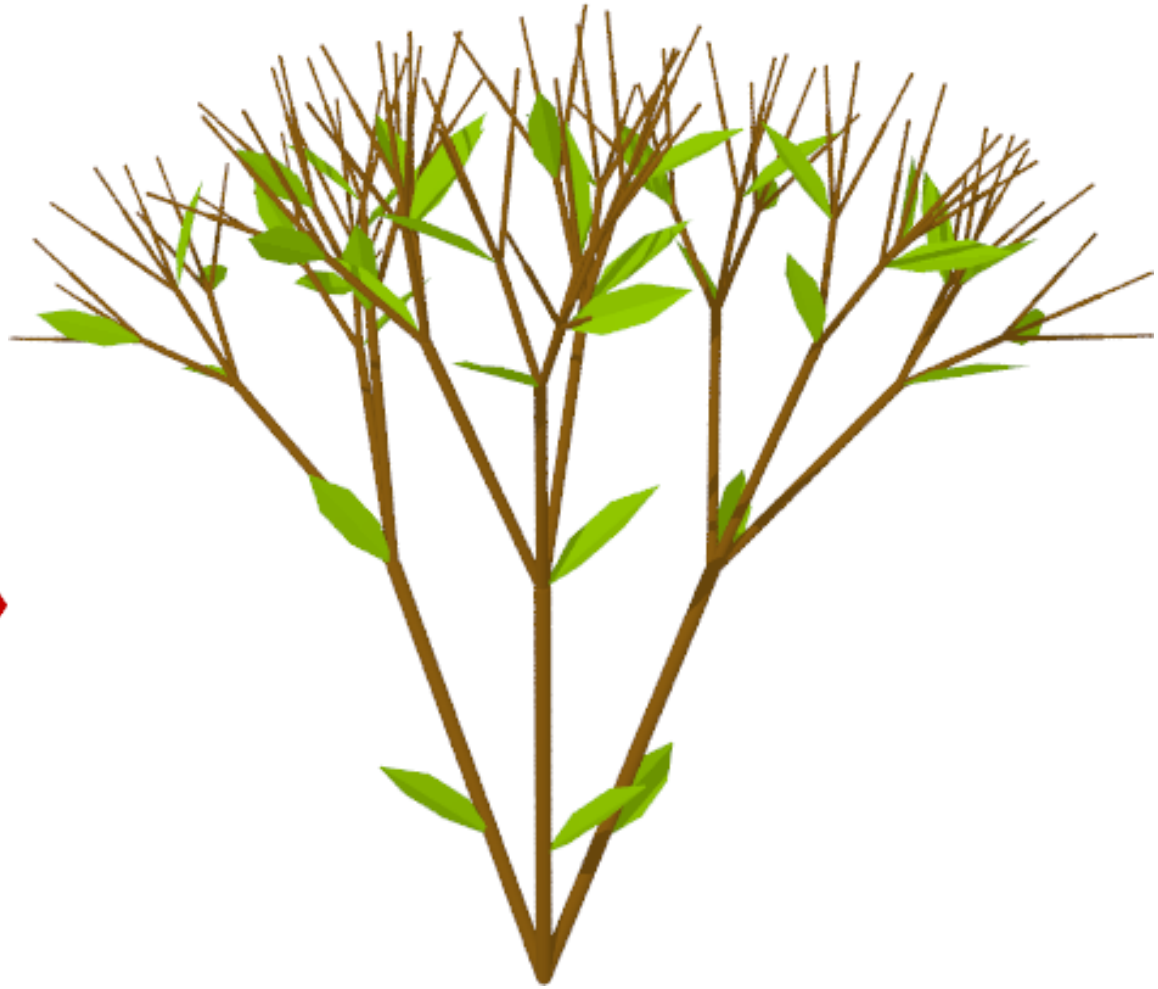
Polygonen

- Accolades vormen delen die aaneengesloten moeten zijn
- Elke beweging maakt een driehoek van het beginpunt en de 2 laatste punten.

Polygonen

- Accolades vormen delen die aaneengesloten moeten zijn
- Elke beweging maakt een driehoek van het beginpunt en de 2 laatste punten.
- $F\{-f+f+f-|-f+f+f\}$ $(\sigma = 30°)$

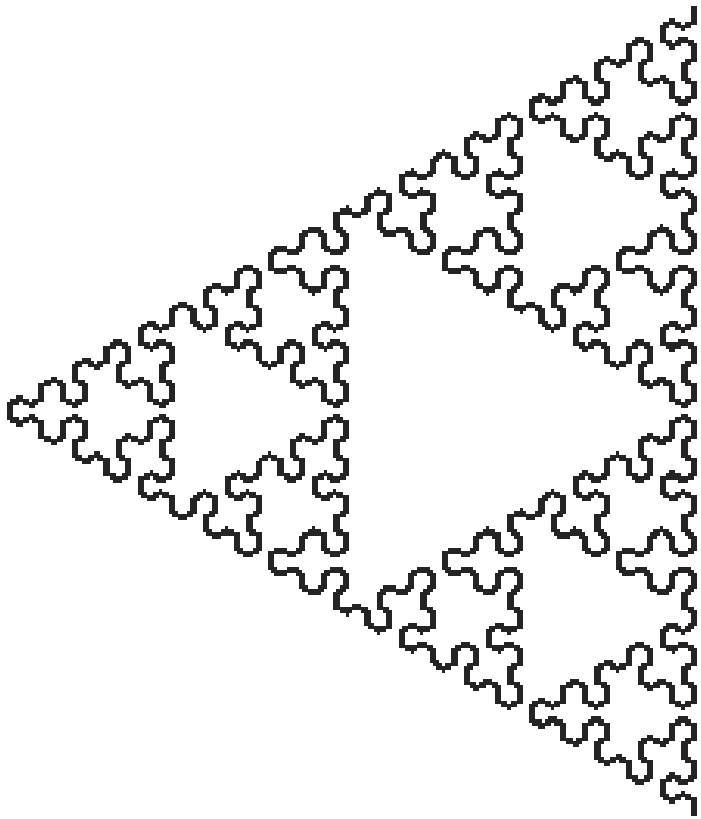


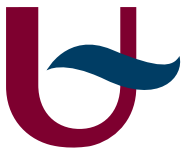


Sierpinski triangle

- Axiom $= A$
- $\sigma = 90°$
- Production rules:

$$\begin{aligned} \text{A} &\rightarrow B - A - B \\ \text{B} &\rightarrow A + B + A \end{aligned}$$

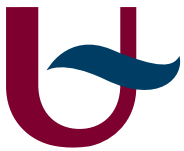




Straight-forward Implementatie

```
//Given: axiom (string), rules (array of string)
currentstring = axiom
iteration = 0

function iterate()
    newstring = ""
    iteration = iteration + 1
    foreach c in currentstring
        if (c in rules)
            newstring = newstring + rules[c]
        else
            newstring = newstring + c
    currentstring = newstring
```



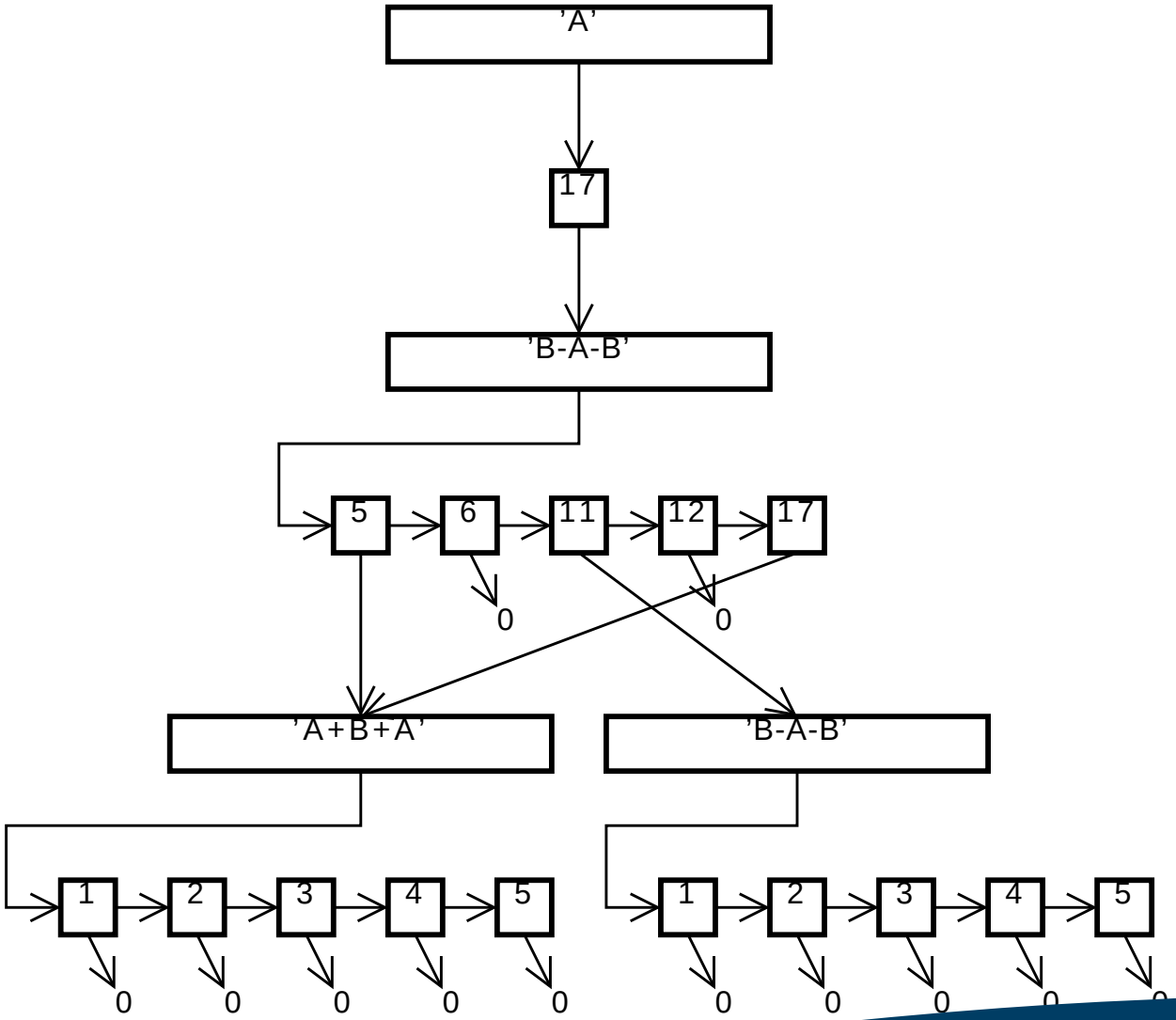
Complexiteiten Sierpinski-driehoek

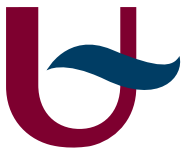
- Plaatscomplexiteit: $O(3^l)$
- Tijdscomplexiteit (opbouw structuur): $O(3^l)$
- Zowel tijd- als plaatscomplexiteit zijn exponentieel

Tree-based structuur

- Het axiom is een root-node
- Elke productie-body + iteratieniveau is een leaf-node
- Hierin staat pointers naar de verwijzende leaf-nodes op diepere niveau

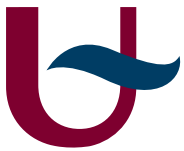
Tree-based structuur





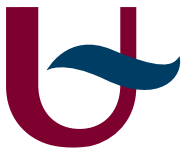
Complexiteiten Sierpinski-driehoek

- Plaatscomplexiteit: $O(I)$
- Tijdscomplexiteit* (opbouw structuur): $O(I)$



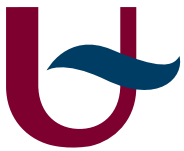
Complexiteiten Sierpinski-driehoek

- Plaatscomplexiteit: $O(I)$
- Tijdscomplexiteit* (opbouw structuur): $O(I)$
  *Enkel de tijd voor het opbouwen van de structuur is berekent, een traversal van de tree zal elke karakter moeten opvragen (mbv Binary Search).



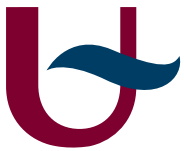
Optimalisaties voor het tekenen

- Turtle graphics gebruiken 1 karakter/actie
- Parameters lezen is vrij intensief



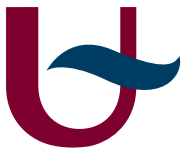
Run-Length Encoding

- $++++ \Rightarrow +'4$



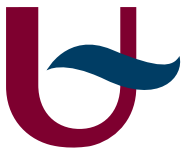
Run-Length Encoding

- $++++ \Rightarrow +'4$
- Korter
- Snellere uitvoering



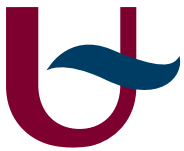
Run-Length Encoding

- $++++ \Rightarrow +'4$
- Korter
- Snellere uitvoering
- $++--++-+ \Rightarrow +'2$



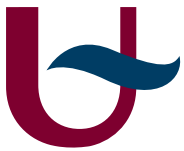
Run-Length Encoding

- $++++ \Rightarrow +'4$
- Korter
- Snellere uitvoering
- $++--++-+ \Rightarrow +'2$
- Speedup: $18 - 24\%$



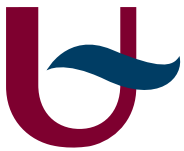
Parameters inlinen

- Parameters=$\{\}$ $F(0.3333)$
- $\Rightarrow$ Parameters=$\{0.3333\}$ $F * 0$



Parameters inlinen

- Parameters=$\{\}$ $F(0.3333)$
- $\Rightarrow$ Parameters=$\{0.3333\}$ $F * 0$
- Korter
- Sneller (niet telkens parsen van een float maar lookup in tabel)



Parameters inlinen

- Parameters=$\{\}$ $F(0.3333)$
- $\Rightarrow$ Parameters=$\{0.3333\}$ $F * 0$
- Korter
- Sneller (niet telkens parsen van een float maar lookup in tabel)
- Speedup: $14 - 21\%$