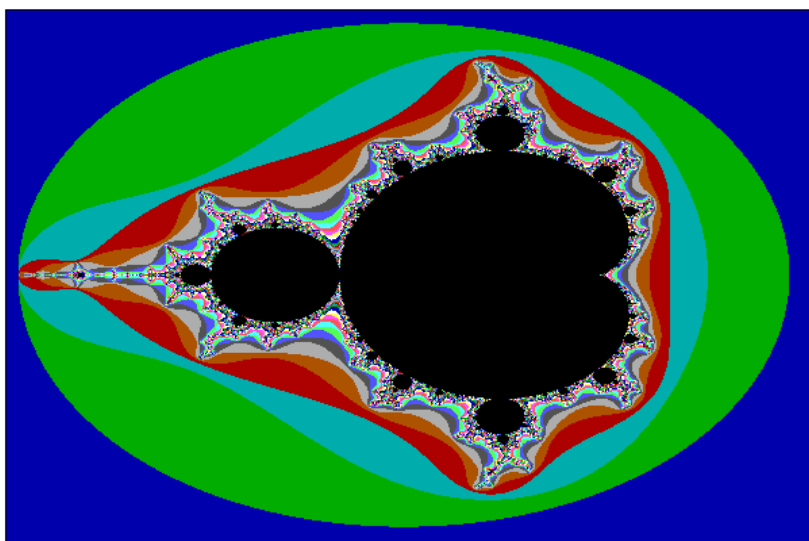


# Lecture 14

**Fractals** Ch 8

**Collision handling** Ch 12

**Most famous fractal: Mandelbrot set**

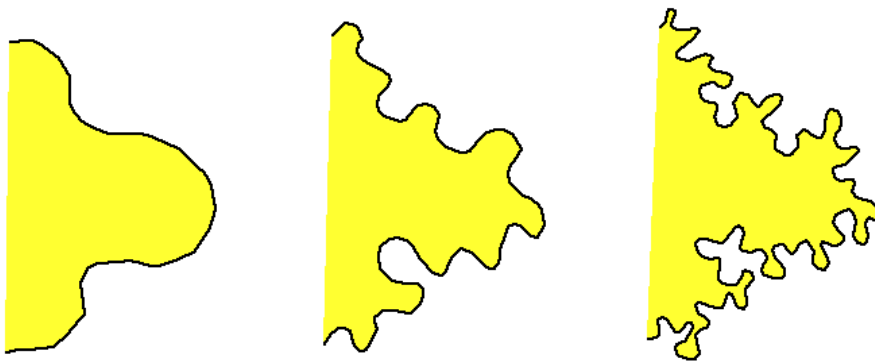


**What is it, more than a pretty image?**

**Natural objects have fractal features**

**Classic example: Coastline**

**Shape and length varies with resolution**



Ingemar  
Ragnemalm  
ingis@isy.liu.se

## **Fractals in computer graphics**

**Fractals are shapes with:**

- **self-similarity**
- **infinite resolution**

**Used for modelling such shapes**

Ingemar  
Ragnemalm  
ingis@isy.liu.se

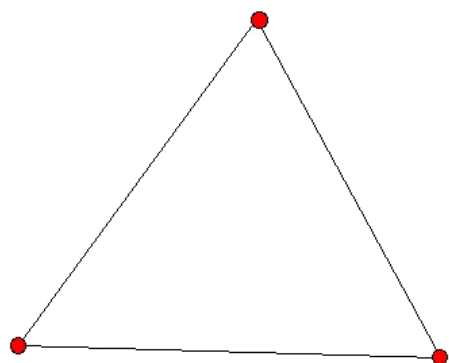
# Classification of fractals

- geometrical recursive construction
  - stochastic fractals
- mathematical formulas (in the complex plane)

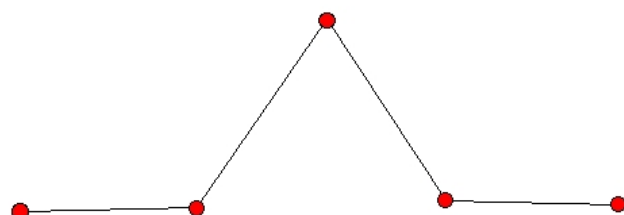
Ingemar  
Ragnemalm  
ingis@isy.liu.se

## Geometric construction of self-similar fractals

Example: Koch curve



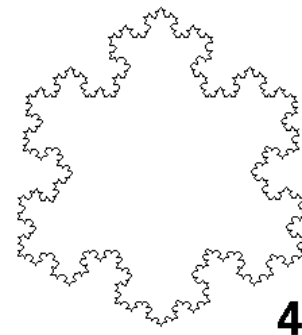
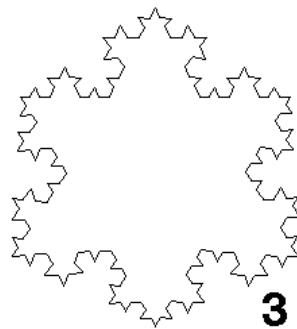
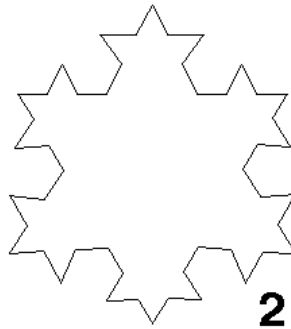
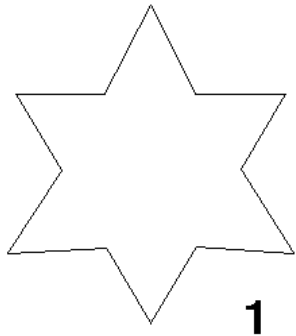
**Initiator**



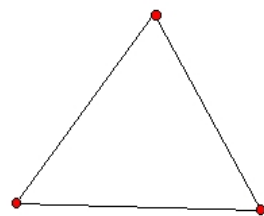
**Generator**

Ingemar  
Ragnemalm  
ingis@isy.liu.se

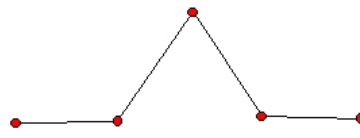
# Resulting Koch curves



Ingemar  
Ragnemalm  
ingis@isy.liu.se



Initiator



Generator

## Recursive function

Pass all parts to next level

Replace part with the generator, scaled to same length

Stop at desired recursion depth or when sections are small enough (e.g. 1 pixel long)

Ingemar  
Ragnemalm  
ingis@isy.liu.se

```
procedure DrawKoch(p1, p2, depth)
```

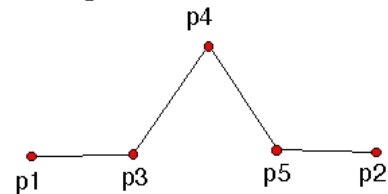
```
if depth >= maxDepth then
```

```
    MoveTo(p1)  
    LineTo(p2)  
    return
```

```
else
```

```
    calculate p3, p4, p5 as the three points inside the generator
```

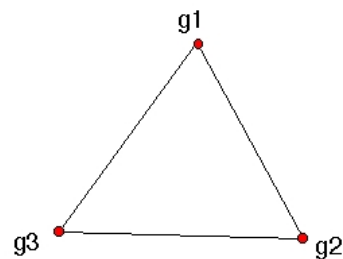
```
    DrawKoch(p1, p3, depth+1)  
    DrawKoch(p3, p4, depth+1)  
    DrawKoch(p4, p5, depth+1)  
    DrawKoch(p5, p2, depth+1)
```



```
main procedure:
```

```
Choose three generator points, g1, g2, g3
```

```
DrawKoch(g1, g2, 0)  
DrawKoch(g2, g3, 0)  
DrawKoch(g3, g1, 0)
```



Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Fractal dimension

A measure of how rough or fragmented the shape is

Definition:

$$ns^D = 1$$

n = number of subparts

s = scaling

D = fractal dimension

Solves to  $D = \ln(n) / \ln(1/s)$

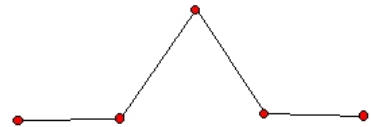
Ingemar  
Ragnemalm  
ingis@isy.liu.se

## Fractal dimension example:

### Koch curve

$$n = 4$$

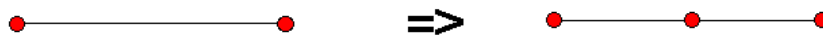
$$s = 1/3$$



$$D = \ln 4 / \ln 3 = 1.26$$

## Fractal dimension example:

### Splitting a line



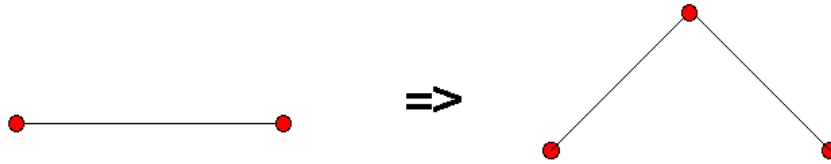
$$n = 2$$

$$s = 1/2$$

$$D = \ln 2 / \ln 2 = 1$$

## Fractal dimension example:

### Splitting a line and moving midpoint



$$n = 2$$

$$s = 1/\sqrt{2}$$

$$D = \ln 2 / \ln \sqrt{2} = 2$$

## Fractal dimension:

In 2D:

**1 to 2: Well-behaved fractal curve**

**>2: Self-intersecting, area-covering**

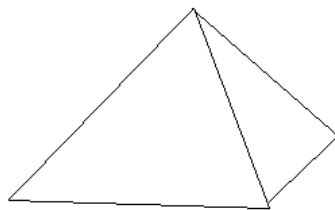
**Split line:  $D = 1$  minimum, no fractal**

**Koch:  $D = 1.26$ , moderate fractal**

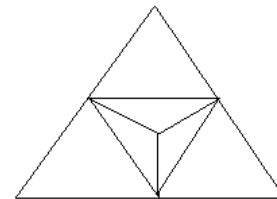
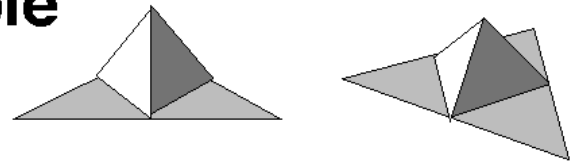
**Moved midpoint:  $D = 2$ , maximum**

# Geometric construction of self-similar fractals in 3D

## Example



Initiator



Generator

$$n = 6$$

$$s = 1/2$$

$$D = \ln 6 / \ln 2 = 2.58$$

## Interpretation of fractal dimension:

In 3D:

**2 to 3: Well-behaved fractal surface**

**>3: Self-intersecting, volume-covering**

## Example: Generation of plants

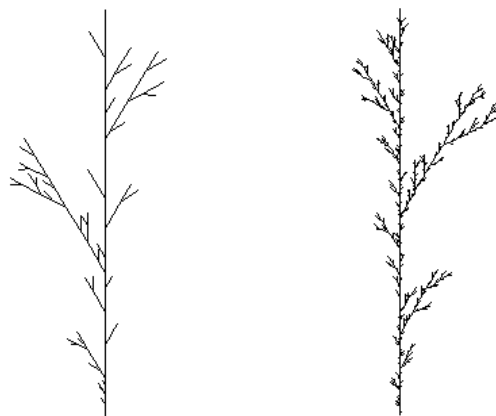


**Promising, but too self-similar!**

Ingemar  
Ragnemalm  
ingis@isy.liu.se

## Statistically self-similar fractals

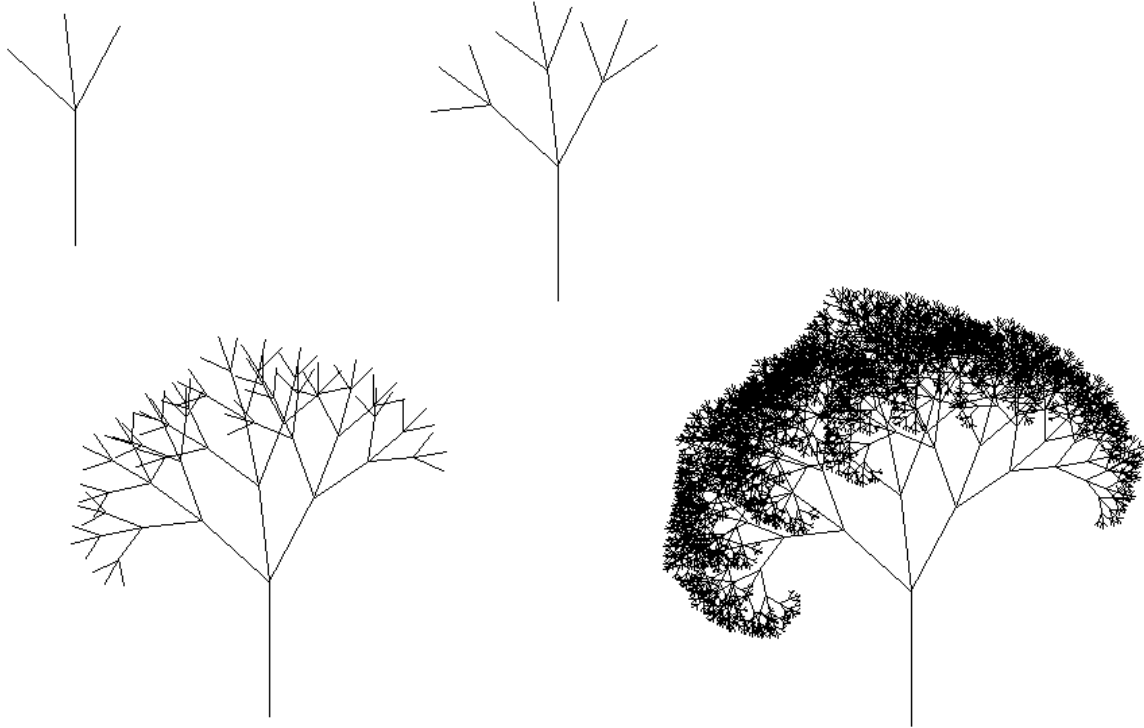
**Random variation of generator**



**Same branch generator as before,  
with some randomness!**

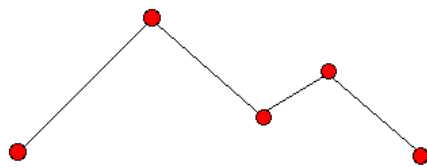
Ingemar  
Ragnemalm  
ingis@isy.liu.se

## Example: Generation of plants #2



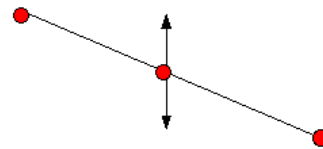
Ingemar  
Ragnemalm  
ingis@isy.liu.se

## Random midpoint-displacement Good for fractal terrain generation



Initiator

Desired rough  
overall whape



Generator

Find midpoint,  
displace along y only



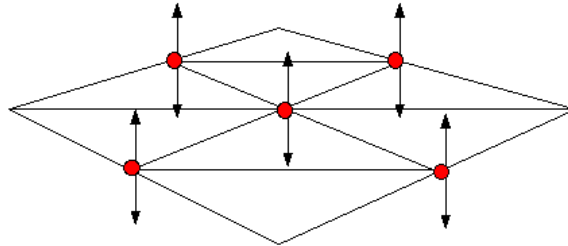
7 iterations

Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Fractal terrain generation in 3D

Split a square to four

Displace midpoints of each side



Middle point can be independent or calculated from the others

Edge points must match neighbor patches

Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Fractal terrain generation in 3D

Heightfield approach

Terrain level  $k$  is array of resolution  $2^k \times 2^k$

The next level has 4x the resolution

Generate new  $2 \times 2$  block from one, or filter over a small neighborhood

Add random offset to all values

Offset should be smaller for higher  $k$

=> magnitude of frequency components inverse proportional to frequency!

Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Terrain generation by FFT

Related method!

Fill frequency space (2D) with random numbers

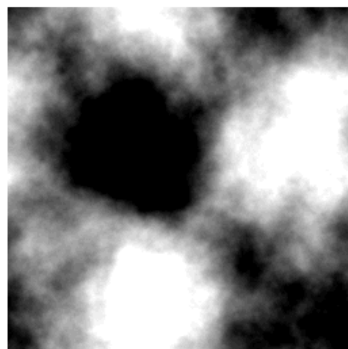
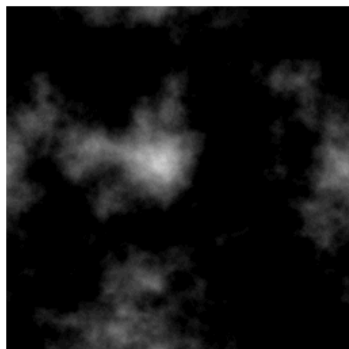
Filter by  $G(f) = F(f) * 1/|f|$

Convert to spatial image with FFT

Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Terrain generation by FFT

## Examples

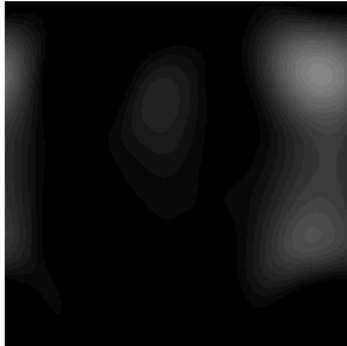


Frequency space:

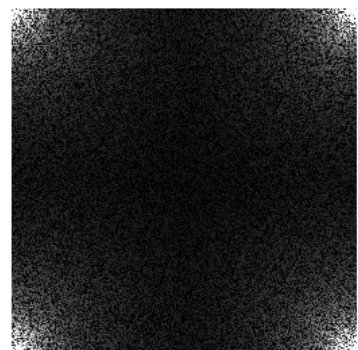
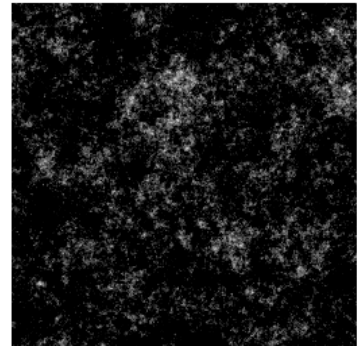
Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Other falloffs than $1/|f|$

$1/|f|^2$



$1/\sqrt{|f|}$



Signal space:

Frequency space:

Ingemar  
Ragnemalm  
ingis@isy.liu.se

## Geometric construction of fractals

- Deterministic self-similar fractals
- Statistically self-similar fractals
- Random midpoint displacement

Easy to control

Intuitive parameters

Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Self-squaring fractals

Based on simple functions in complex space

Insert complex numbers (points) into a function

Apply function recursively, and analyze the behaviour.

- Diverge?
- Converge?
- Chaotic?

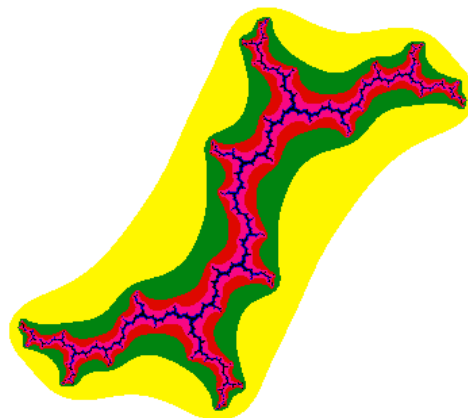
Converge or chaotic: Does it keep within some limit in a number of iterations?

Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Self-squaring fractals

## The Julia set

$$z_{k+1} = z_k^2 + \lambda$$



Julia set for  $\lambda = (0, 1) = 0 + j$

Ingemar  
Ragnemalm  
ingis@isy.liu.se

# The Julia set

## Implementation

for  $y = \text{miny}$  to  $\text{maxy}$   
for  $x = \text{minx}$  to  $\text{maxx}$   
 $(z_r, z_i) = \text{scaling of } (x, y)$

for  $i = 0$  to  $\text{maxiterations}$

$$z = z^2 + \lambda$$

if  $|z| > R$  then Leave

Draw pixel  $(x, y)$  (different colors for different  $i$ )

$\text{maxiterations} \approx 15$

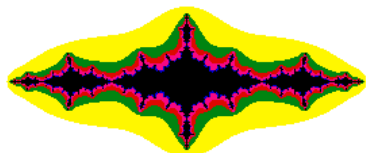
$R^2 \approx 10$

Ingemar  
Ragnemalm  
ingis@isy.liu.se

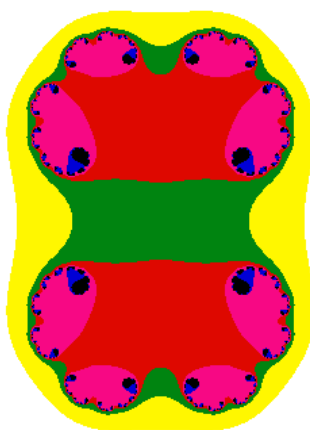
## Other Julia sets

$$z_{k+1} = z_k^2 + \lambda$$

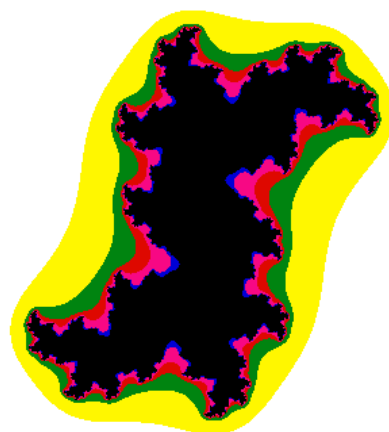
Other  $\lambda$  values



$\lambda = (-1.3, 0)$



$\lambda = (0.4, 0)$



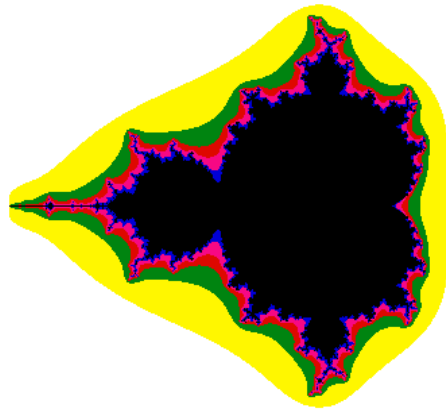
$\lambda = (0.3, 0.5)$

Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Self-squaring fractals

## The Mandelbrot set

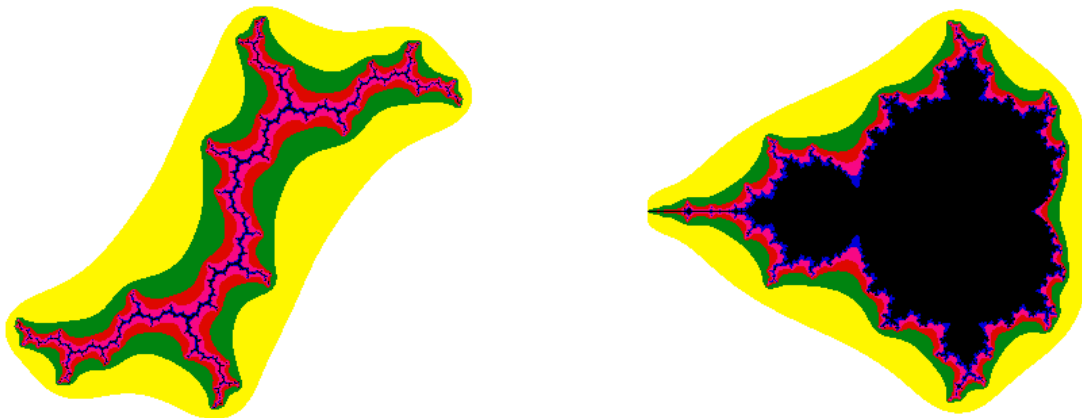
$$z_{k+1} = z_k^2 + z_0$$



Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Self-squaring fractals

- Beautiful
- Non-predictable
- Limited usability



## Mathematical curiosity

Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Fractals, summary

## 1) Geometrically constructed fractals

Very useful for generating many kinds of natural objects

Allows design of complex models with arbitrary resolution

## 2) Self-squaring fractals (and other adventures in the complex plane)

Questionable practical usability

Hard to do planned designing

Ingemar  
Ragnemalm  
ingis@isy.liu.se

Related methods:

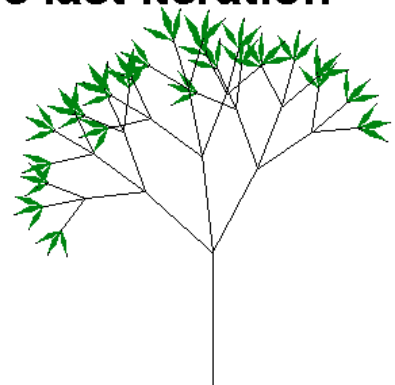
## Shape grammars and procedural methods

No unlimited resolution

Different rules at different levels

Example: Tree with leaves: replace last iteration with leaf generator

“graftals”



Ingemar  
Ragnemalm  
ingis@isy.liu.se