Development of Games

Lecture 6

Fractals in Games

Why fractals ?

- Natural processes are uncertain but have any order
  - E.g. trees, forest, maintains, shape of island, waves of sea
- How to make simultaneously random and determined shapes ?
- Fractal mathematics
- Another task in contrast to polygons
  - Generation of any natural-like shape but not copy of already existing

History

- Mathematically recognized in 19th century
- Named and used practically by Benoit Mandelbrot
  - Studied stock-market, fluid flows, galaxy distribution in universe
  - “Father of Fractals”

Self similarity

- The ideas of self similarity are best illustrated by first looking at some “non-natural” examples.
- We can then look at how these ideas are easily extended to the 3d case.

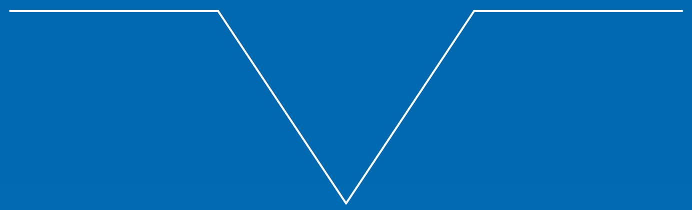
- In general, any fractal starts with a *base shape* and is then modified by repeated recursive application of a *generator rule*.

The Koch Curve (1)

- The Koch curve is a geometrical pattern that is generated by the recursive application of a simple rule:
- Start with a base shape - in this case a simple straight line

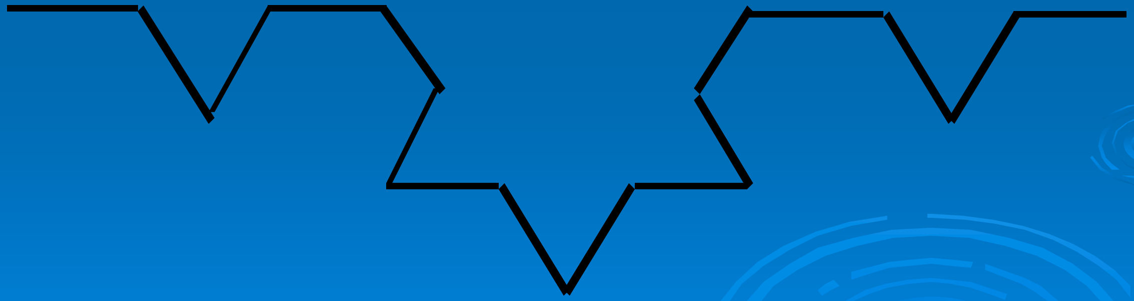
The Koch Curve (2)

- Apply the following rule:
  - Take the central third of the line and replace it with a triangle



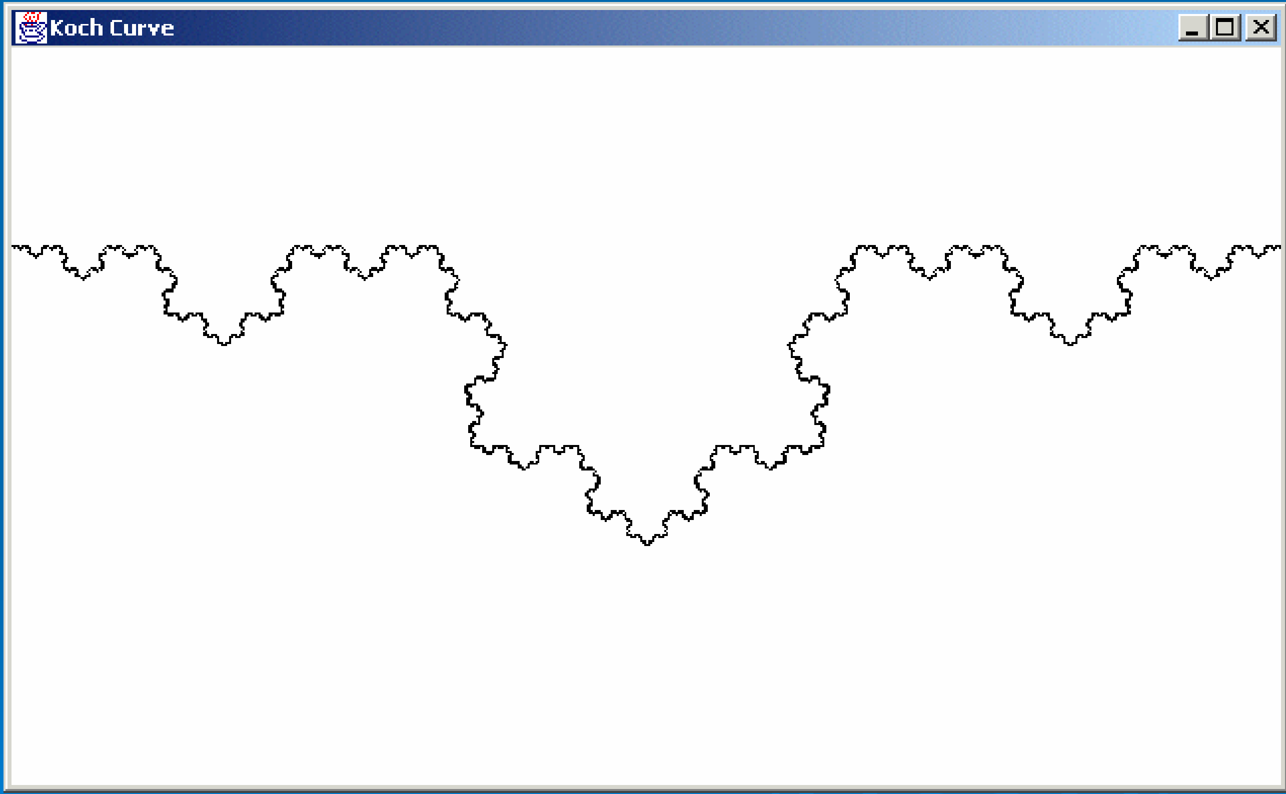
The Koch Curve (3)

- Now apply this to all of the segments of the resulting line



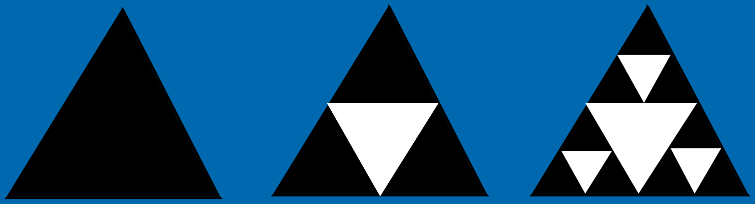
The Koch Curve (4)

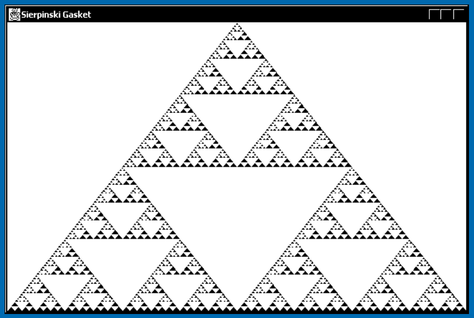
- After repeated applications of the rule
- length of the line tends to infinity.



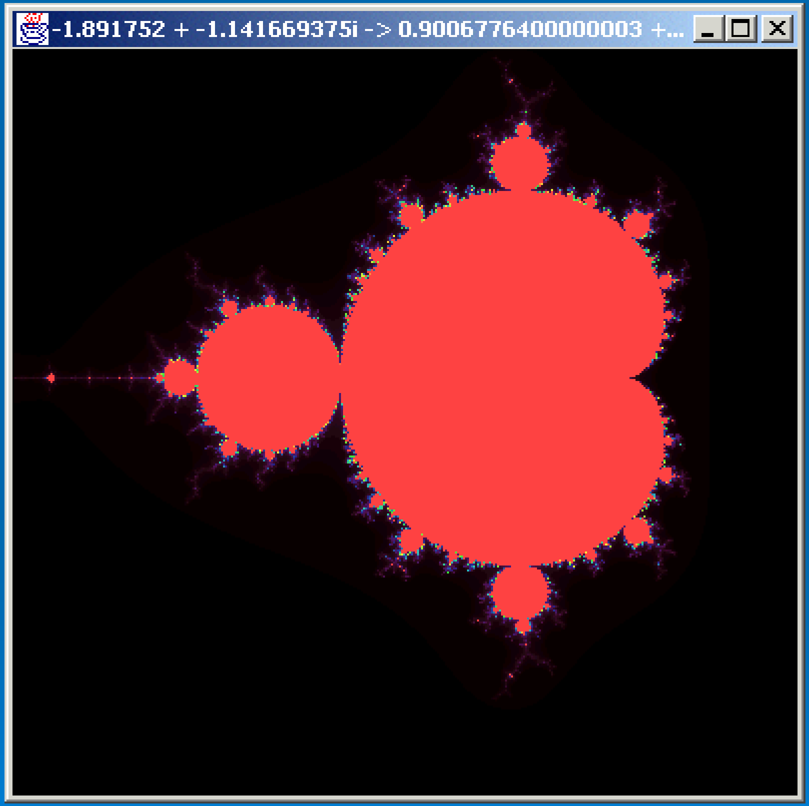
The Sierpinski Gasket

- Start with a shaded triangle as the base shape and repeatedly remove the middle of the remaining triangle





The Mandelbrot Set



```
-1.891752 + -1.141669375i -> 0.90067764000000003 +...
```

The Mandelbrot Set (2)

- Based upon a mathematical relationship:

$$\mathbf{z}_{\mathbf{i}+\mathbf{1}} \Leftarrow \mathbf{z}_{\mathbf{i}}^{2} + \mathbf{c}$$

  - where **z** and **c** are both complex numbers.
  - i.e. assign **z** the value **z**^{2} + **c**

Three methods of generation of fractals

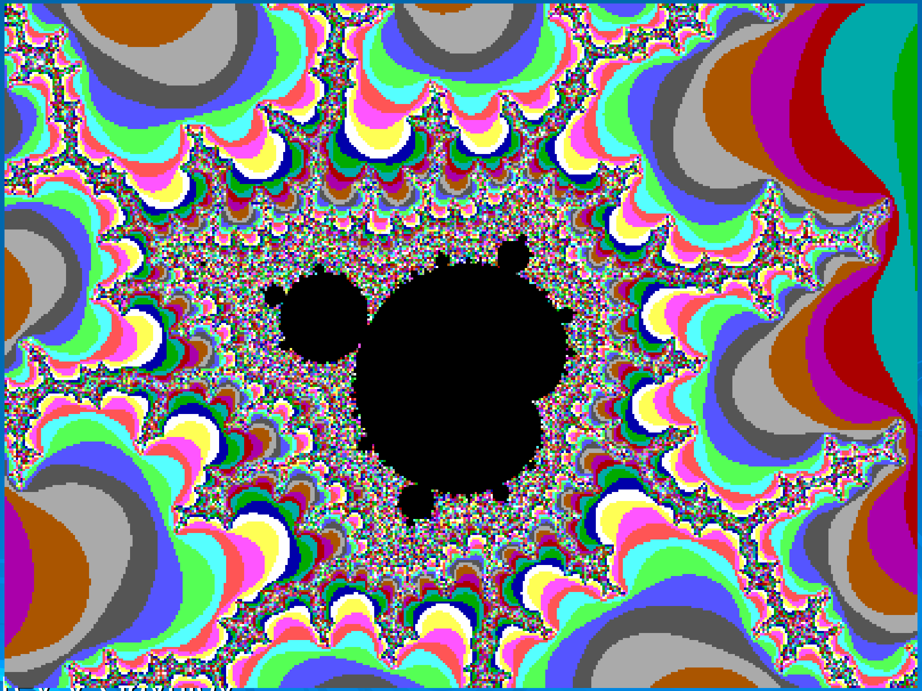
- Non-linear
- Iterated Function Systems
- L-Systems

Non-Linear Fractals (1)

- General procedure for non-linear fractal creation:
  - Input (x, y) location to a non-linear function
  - Repeatedly call the function with the output from the previous pass
  - Stop at some point and color pixel based on number of iterations and result

Non-Linear Fractals (2)

- Zooming in and out shows the fractal has continual self-symmetry
  - Plus these are quite cool
- Examples
  - Mandelbrot Set
  - Julia Set

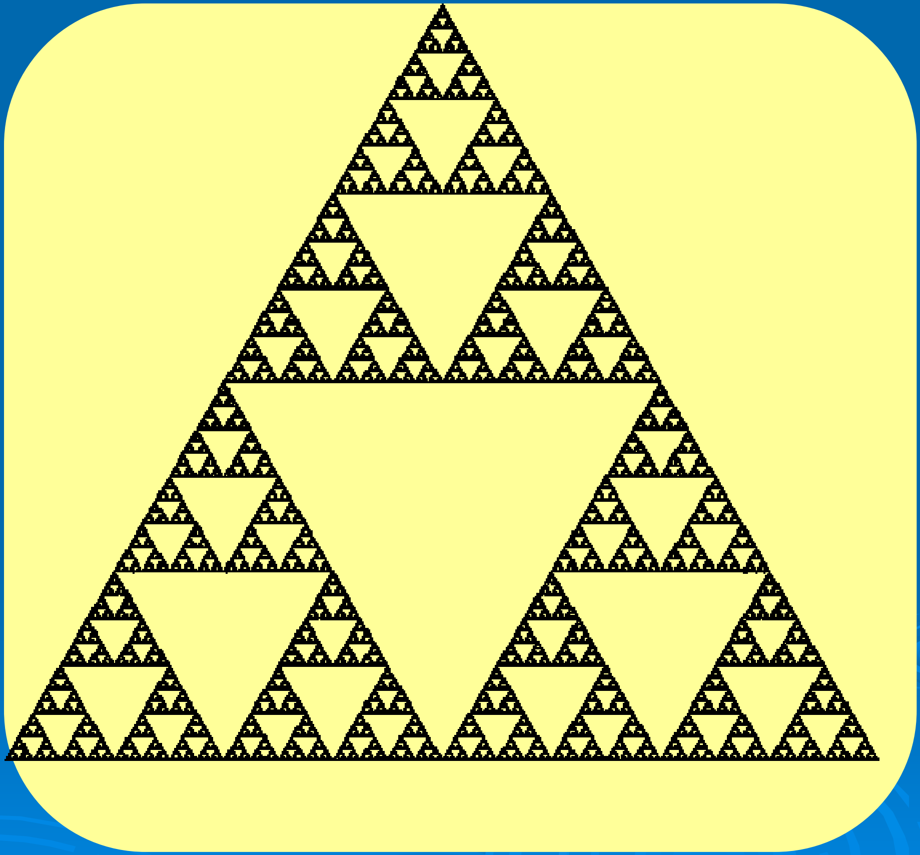


Iterated Function Systems (1)

- Much simpler than non-linear
  - Conceptually
- Usually involves some randomness and some simple algebra
- Example:
  - Take fixed 3 points on a plane
  - Take random location on a plane
    - Call this “curPos”
  - Randomly choose any of 3 points
    - Move curPos to half way between curPos and chosen point
  - Repeat the last step 1000’s of times

Iterated Function Systems (2)

- Serpinsky Gadget
- Other Examples:
  - Perfect Fern
  - Other geometric pretties



L-Systems

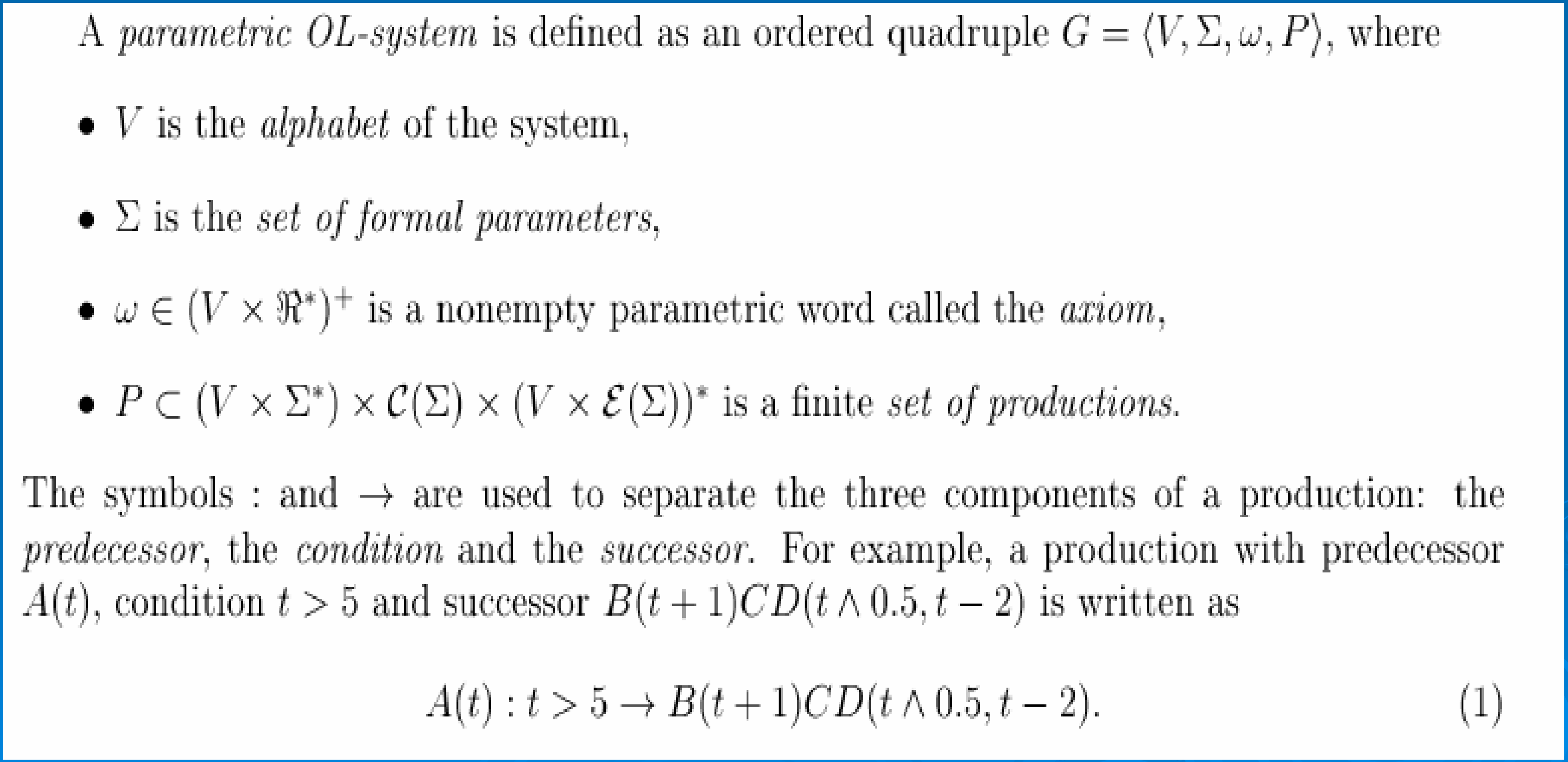
- In 1968, Aristid Lindenmayer
- Formalism for simulating of multicellular organisms
- Closely related with abstract automata and formal languages

L-Systems (1)

- These include a starting input, and a recursion rule
- Sometimes defined as a set of strings:
  - F --> F++F-F-
  - + --> F-F
  - F - move forward one unit
  - + rotate left by X degrees
  - - rotate right by X degrees
- Can also be defined via recursion

L-Systems (2)

Parametric L-systems operate on *parametric words*, which are strings of *modules* consisting of *letters* with associated *parameters*.



A *parametric OL-system* is defined as an ordered quadruple $G = \langle V, \Sigma, \omega, P \rangle$, where

- V is the *alphabet* of the system,
- Σ is the *set of formal parameters*,
- $\omega \in (V \times \Re^*)^+$ is a nonempty parametric word called the *axiom*,
- $P \subset (V \times \Sigma^*) \times \mathcal{C}(\Sigma) \times (V \times \mathcal{E}(\Sigma))^*$ is a finite *set of productions*.

The symbols : and $\rightarrow$ are used to separate the three components of a production: the *predecessor*, the *condition* and the *successor*. For example, a production with predecessor $A(t)$, condition $t > 5$ and successor $B(t+1)CD(t \wedge 0.5, t-2)$ is written as

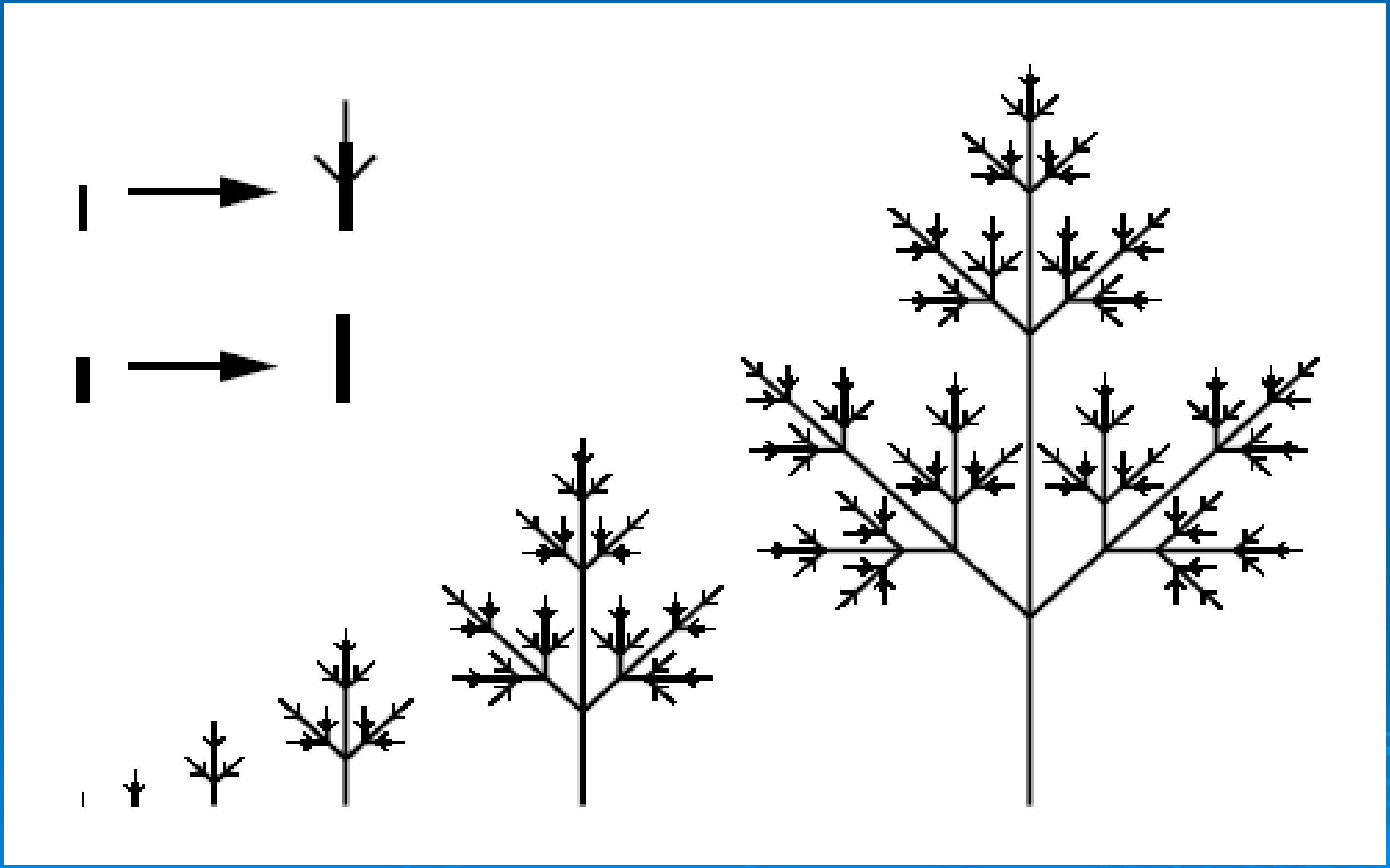
$$A(t) : t > 5 \rightarrow B(t+1)CD(t \wedge 0.5, t-2). \qquad (1)$$

L-Systems (3)

- A production matches a module in a parametric word if the following conditions are met:
  - the letter in the module and the letter in the production predecessor are the same,
  - the number of actual parameters in the module is equal to the number of formal parameters in the production predecessor, and
  - the condition evaluates to true if the actual parameter values are substituted for the formal parameters in the production.
- A matching production can be applied to the module, creating a string of modules specified by the production successor. The actual parameter values are substituted for the formal parameters according to their position.

Fractals in Games and Animation

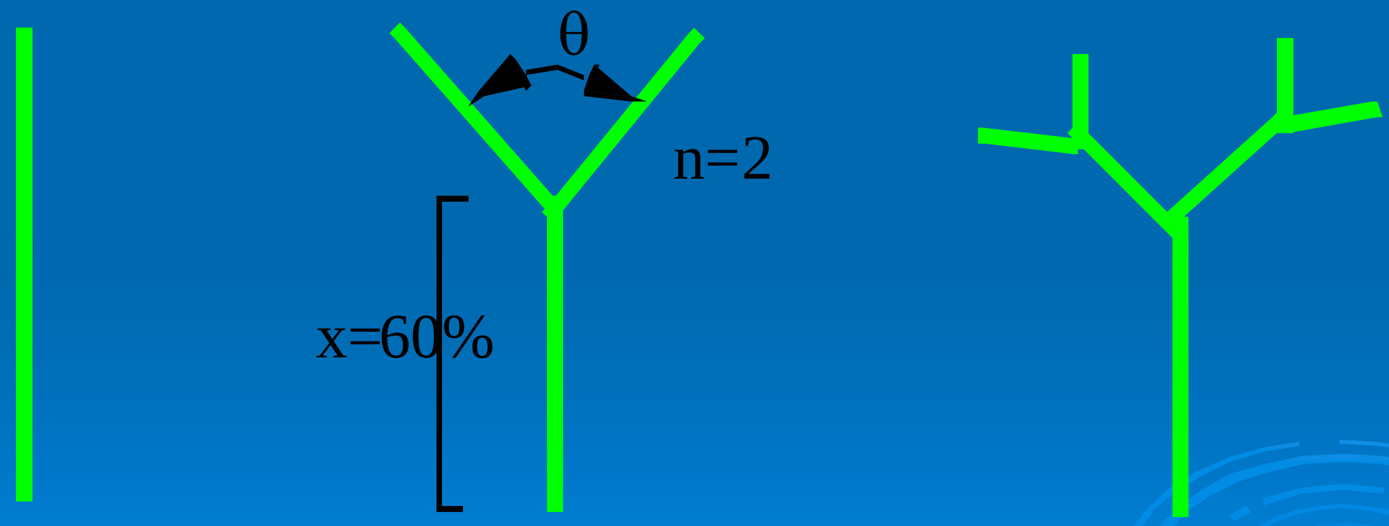
Simulation of trees



Trees

- Trees share a common fractal structure - a trunk which subdivides in branches which sub-divide into more branches which.....
- Base shape:
  - straight line.
- Generator rule:
  - Take the top x% of the line and split it into n branches, with q degrees between each branch. -

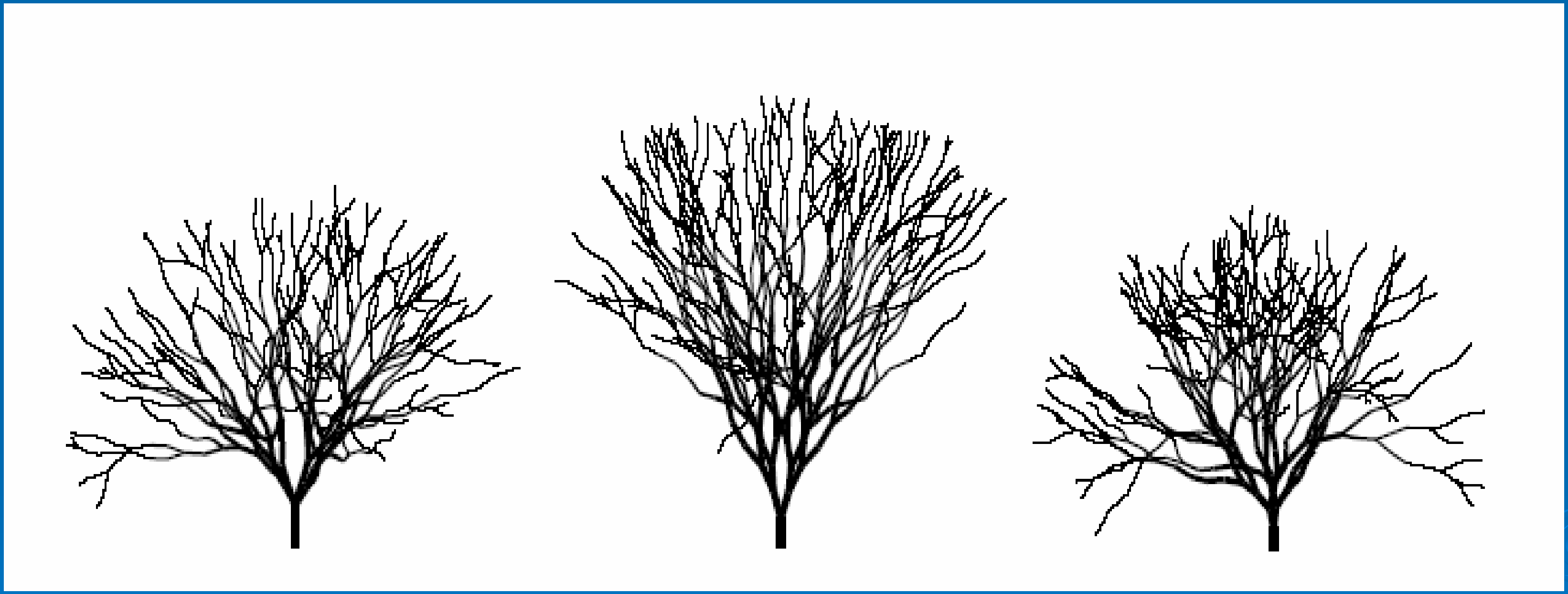
Trees (2)



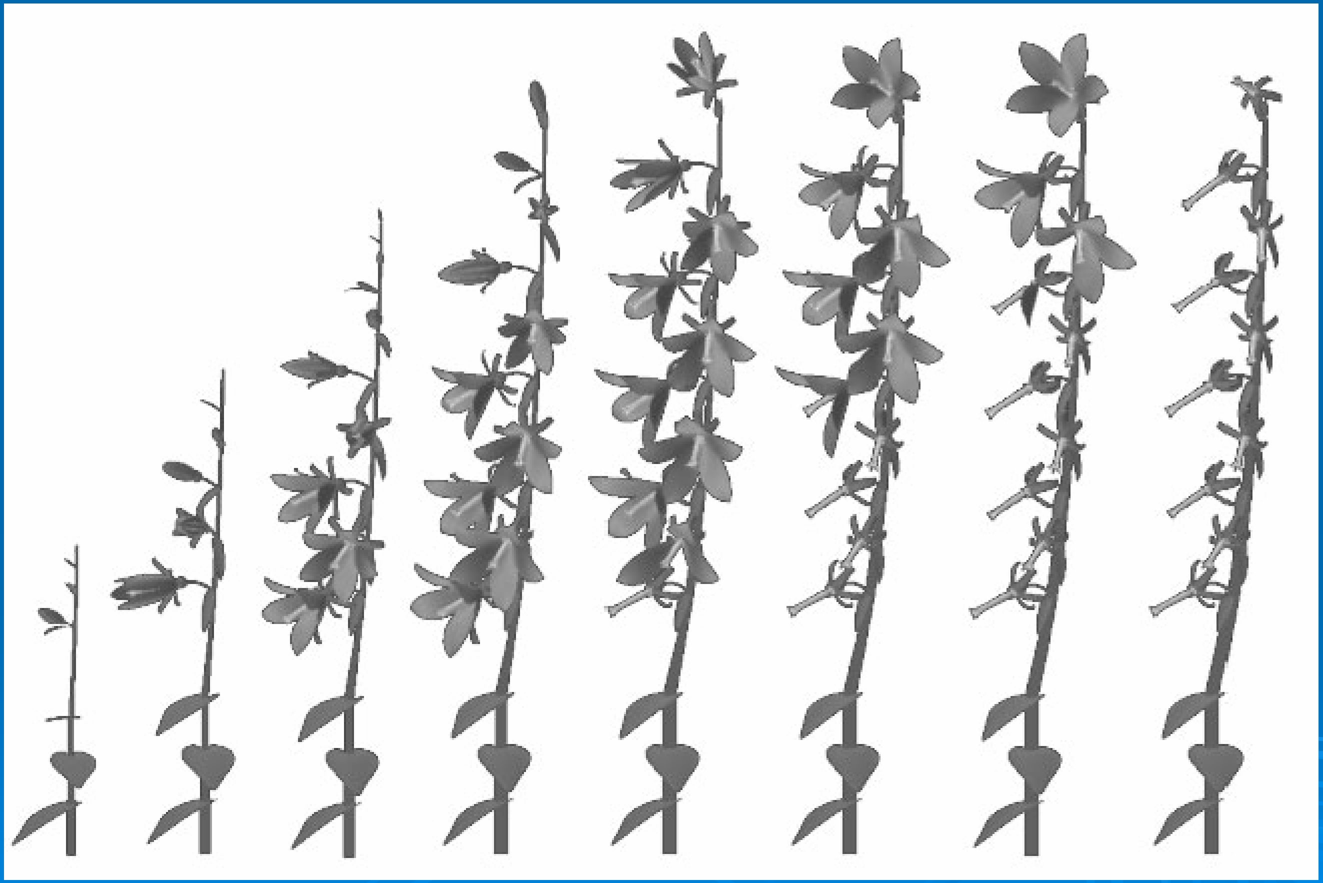
Trees (3)

- The algorithm can be made more complex in several ways:
  - By varying those parameters, different styles of tree can be obtained.
  - By allowing those parameters to vary randomly within limits over the generations some realistic branching patterns can be obtained.
  - By varying the limits over generations some really quite convincing trees can be generated

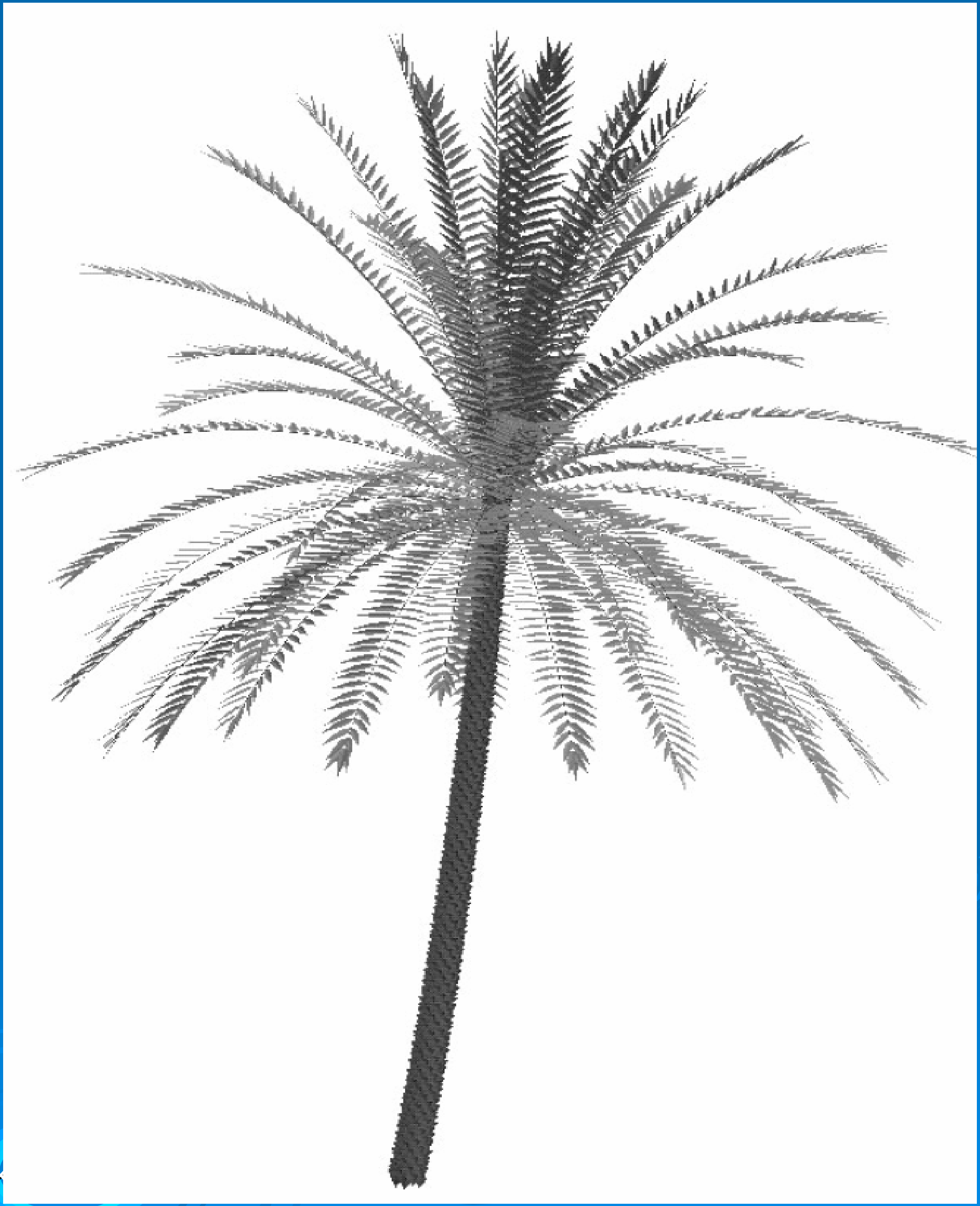
Trees (4) Stochastic



Simulated development of bluebell flower



Simulation of date palm



L-System used for palm

$$
\begin{array}{lllll}
\omega : & A & & & \\
p_1 : & A & & \rightarrow & F(1)[-X(3)B][+X(3)B]A \\
p_2 : & B & & \rightarrow & F(1)B \\
p_3 : & X(d) & : d > 0 & \rightarrow & X(d-1) \\
p_4 : & X(d) & : d == 0 & \rightarrow & U\% \\
p_5 : & U & & \rightarrow & F(0.3)
\end{array}
$$

Fractals and 3D

- All of the examples of fractals have been given in 2d versions so as not to obscure the functioning of the underlying generating mechanism with the “machinery” of 3D graphics. The basic approach in using fractal techniques in 3D is to generate the objects in the scene using the appropriate fractal techniques and then render the scene using any of the previously covered techniques.

Summary

- Base shape
- Generator rule