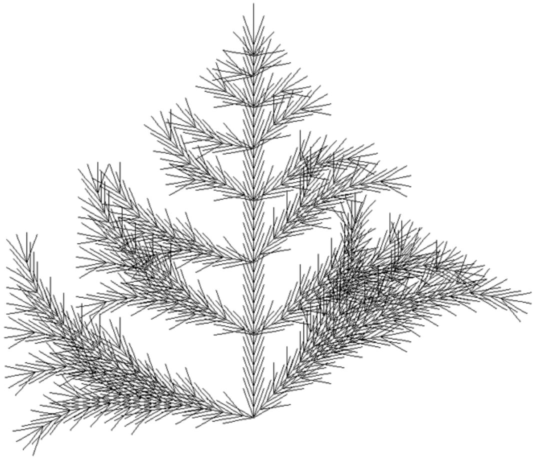
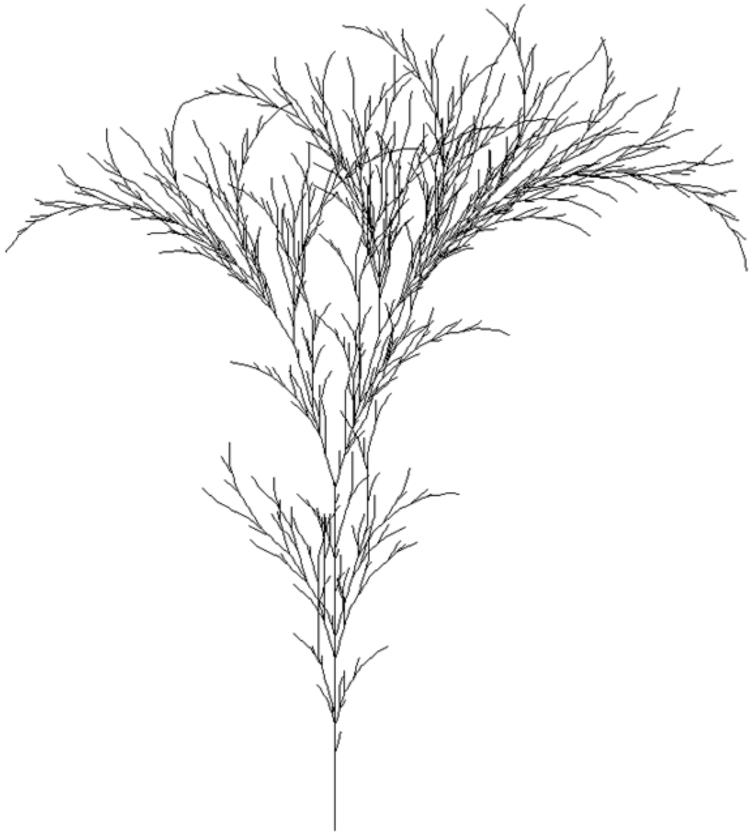
Morphogenesis, Lindenmayer Systems and Generative Encodings



Gabriela Ochoa
http://www.ldc.usb.ve/~gabro/

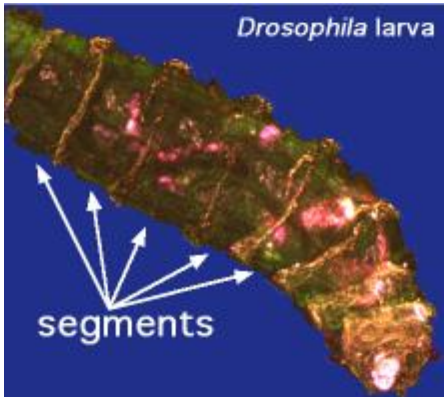
Content

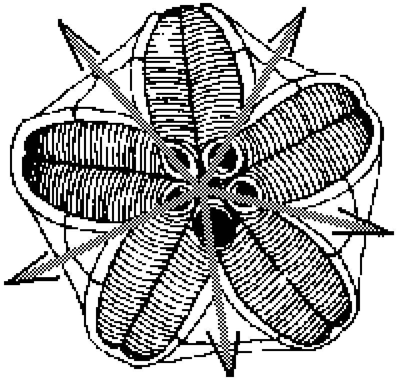
- Morphogenesis
  - Biology
  - Alife
- Lindenmayer Systems
  - Self-similarity, Rewriting
  - D0L-systems
  - Graphic Interpretation
- Generative or rule-based encodings for Evolutionary Algorithms



Morphogenesis in Biology

- One of the major outstanding problems in the biological sciences
- Fundamental question of how biological form and structure are generated
- Biological form at many levels, from individual cells, through the formation tissues, to the assembly of organs and whole organisms.



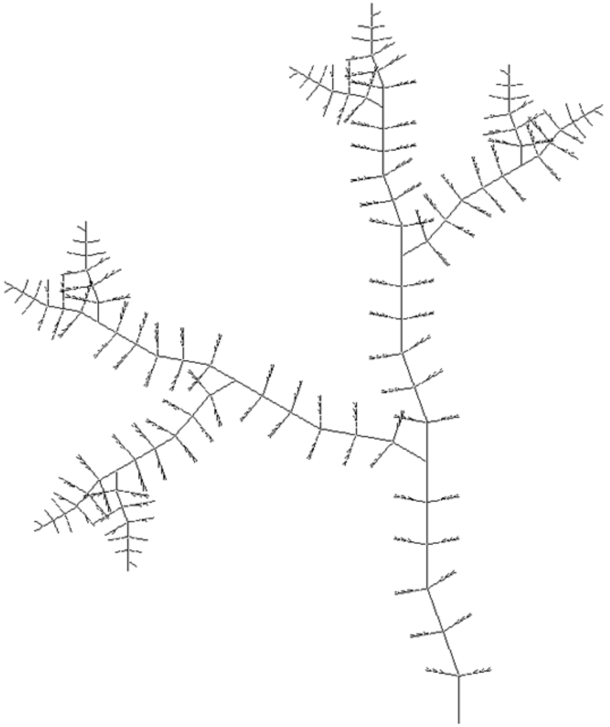


Morphogenesis in Alife

- Central Question in Morphogenesis: How the information coded in linear DNA molecules becomes translated into a three-dimensional form?
- Going from Genotype to Phenotype
- General assumption: the DNA does not specify 'as some kind of description' or ‘blueprint’ the final form of the body. More like 'a **recipe'** for baking a cake
- A typical Alife approach is to look at possible, very general, ways to generate complex forms from relatively simple rules -- often very abstract

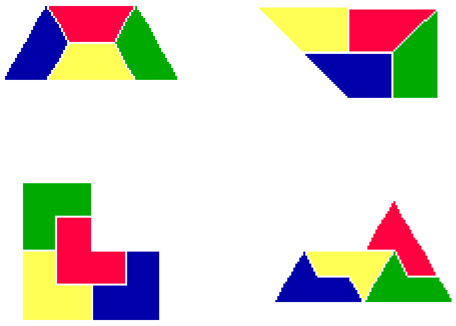
L-Systems

- A model of morphogenesis, based on formal grammars (set of rules and symbols)
- Introduced in 1968 by the Swedish biologist A. Lindenmayer
- Originally designed as a formal description of the development of simple multi-cellular organisms
- Later on, extended to describe higher plants and complex branching structures.

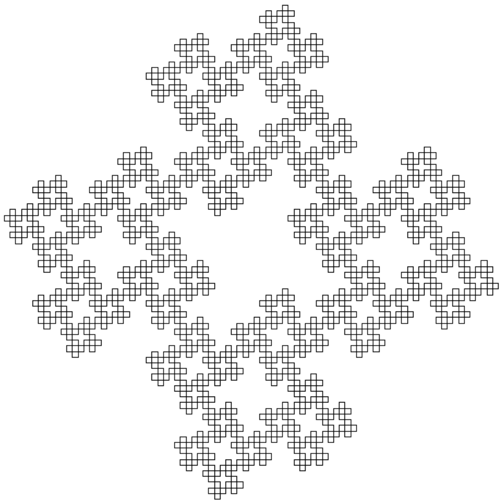


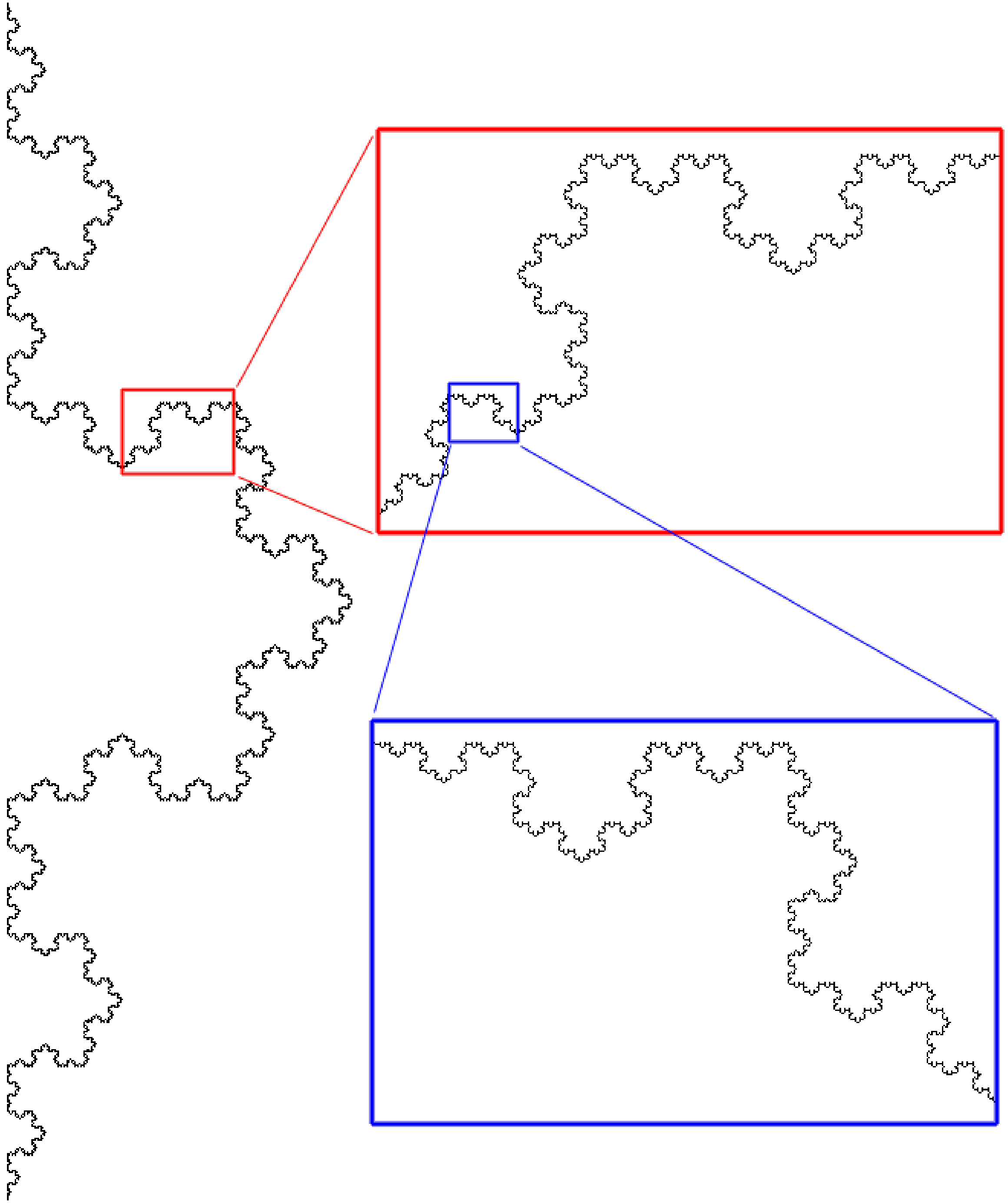
Self-Similarity

"When a piece of a shape is geometrically similar to the whole, both the shape and the cascade that generate it are called self-similar" (Mandelbrot, 1982)



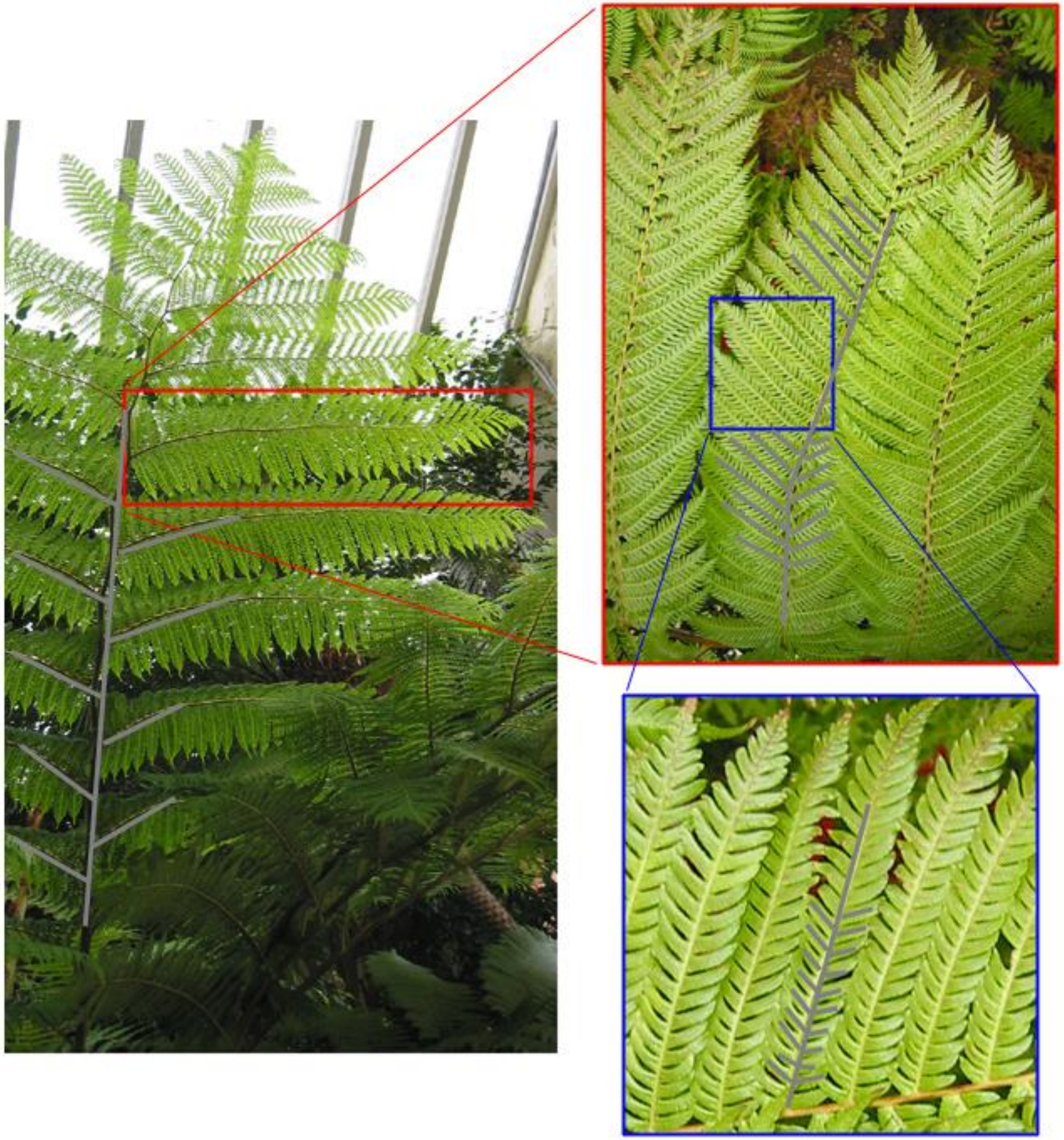
The recursive nature of the L-system rules leads to self-similarity and thereby fractal-like forms are easy to describe with an L-system.





Self-Similarity in Fractals

- Exact
- Example Koch snowflake curve
- Starts with a single line segment
- On each iteration replace each segment by 
- As one successively zooms in the resulting shape is exactly the same

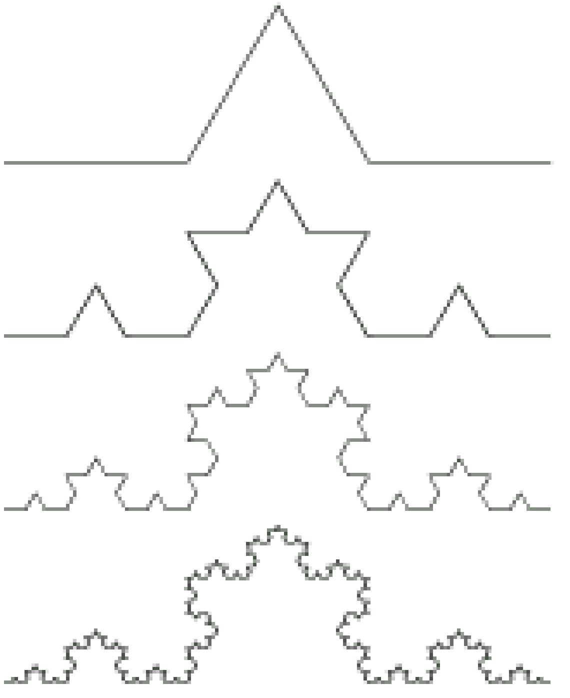


Self-similarity in Nature

- Approximate
- Only occurs over a few discrete scales (3 in this Fern)
- Self-similarity in plants is a result of developmental processes, since in their growth process some structures repeat regularly. (Mandelbrot, 1982)

Rewriting

- Define complex objects by successively replacing parts of a simple object using a set of rewriting rules or productions.
- Example: Graphical object defined in terms of rewriting rules - Snowflake curve
- Construction: recursively replacing open polygons



First four orders of the Koch Curve

Rewriting Systems on Character Strings

- The most extensively studied rewriting systems operate on character strings (Late 50s, Chomsky`s work on formal grammars)
- Later applications to Computer and formal Languges (BNF form)
- A. Lindenmayer (1968) new type of string-rewriting mechanism (L-systems).
- In L-systems productions are applied in parallel Reflects Biological motivation of L-systems

Types of L-systems

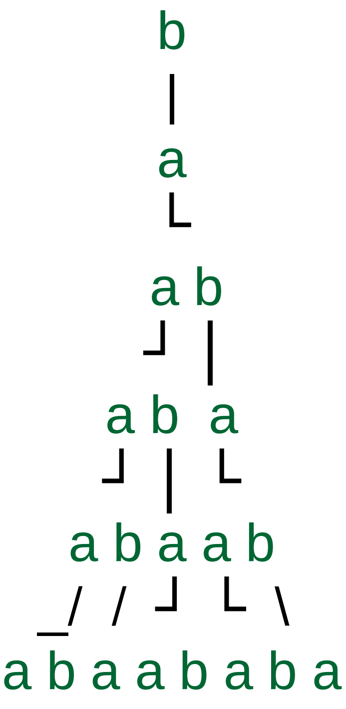
- *Context-free:* production rules refer only to an individual symbol
- *Context-sensitive:* the production rules apply to a particular symbol only if the symbol has certain neighbours
- *Deterministic*: If there is exactly one production for each symbol,
- *Stochastic*: If there are several, and each is chosen with a certain probability during each iteration

D0L-systems

- Simplest class of L-systems, deterministic and context free.
- Example:
  - Alphabet = {a,b}
  - Rules = {a → ab, b → a}
  - Axiom: b

Syntax of a production rule:

Initiator → Generator



Example of a derivation in a DOL-System

Graphic Interpretation

- L-systems were conceived as a formal theory of development. Geometric aspects were not considered
- Later, geometrical interpretations were proposed. Tool for fractal and plant modelling
- Graphic Interpretation of strings, based on turtle geometry (Prusinkiewicz et al, 89). State of the turtle: (x, y, α)
  - (x, y): Cartesian coordinates, turtle position
  - α: angle (heading) direction in which the turtle is facing
- Given the step size d and the angle increment δ, the turtle can respond to the commands represented by the following symbols:

Turtle Interpretation of Strings

F Move forward a step of length *d*. The state of the turtle changes to (x',y',α), where $x' = x + d\cos(\alpha)$ and $y' = y + d\sin(\alpha)$. A line segment between points (x,y) and (x',y') is drawn

f Move forward a step of length *d* without drawing a line. The state of the turtle changes as above

+ Turn left by angle δ. The next state of the turtle is $(x,y,\alpha+\delta)$

- Turn left by angle δ. The next state of the turtle is $(x, y, \alpha - b)$

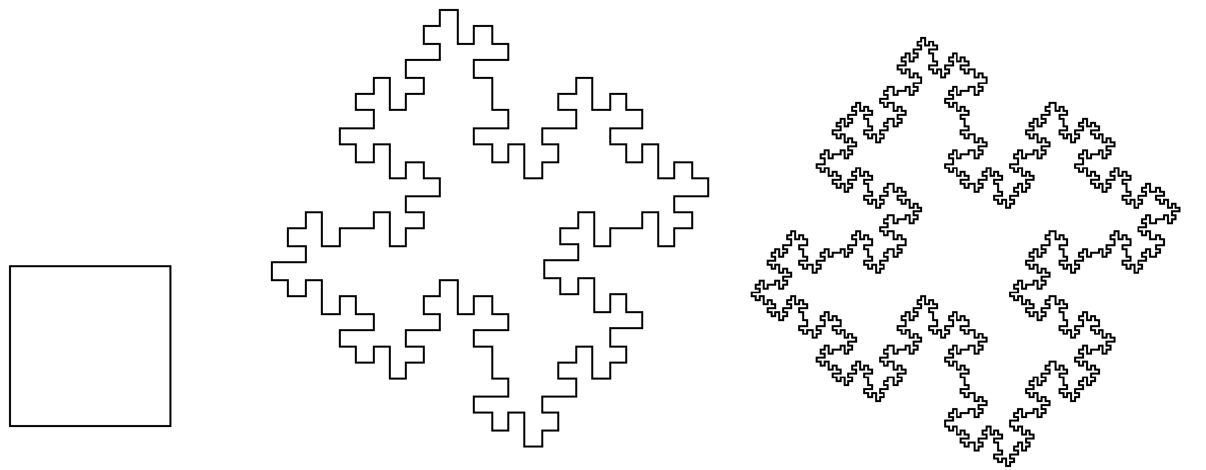
Turtle Interpretation of Strings

w: F+F+F+F

p: F →F+F-F-FF+F+F-F

Angle (δ) = 90º

Quadratic
Koch island



n= 0 n = 1 n = 2

Bracketed L-systems

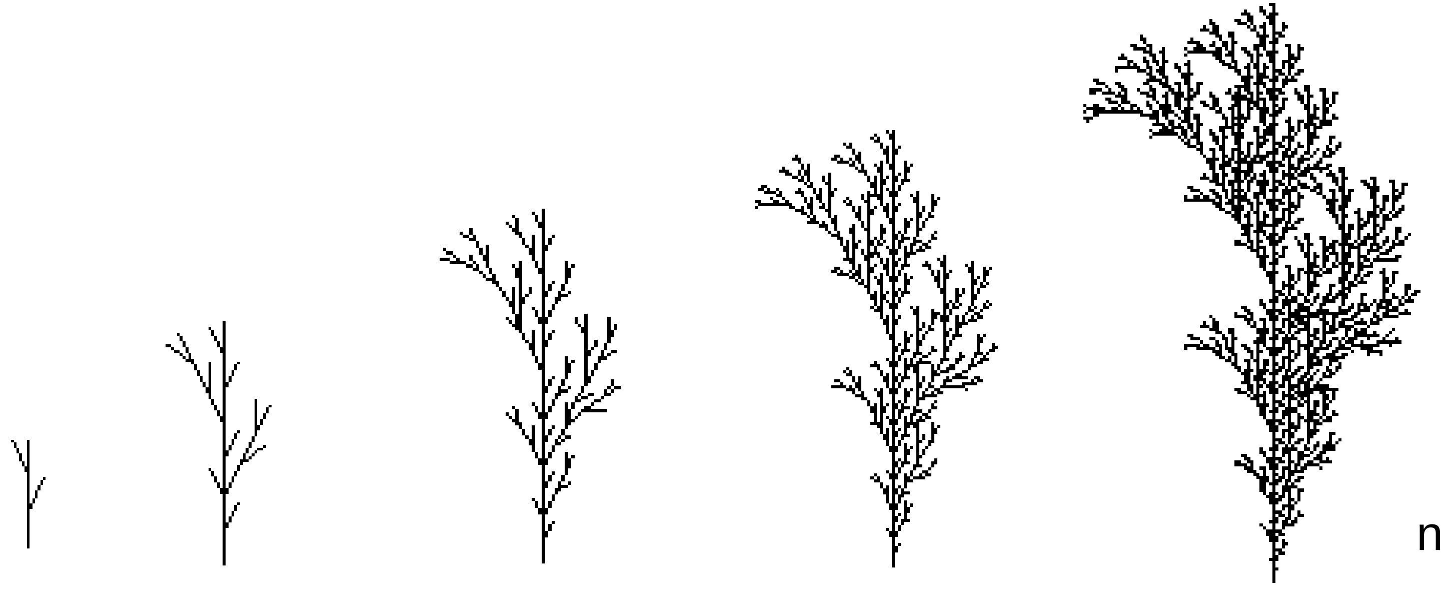
- To represent branching structures, L-systems alphabet is extended with two new symbols: [,], to delimit a branch. They are interpreted as follows:
  - [Push the current state of the turtle onto a pushdown stack.
  -] Pop a state from the stack and make it the current state of the turtle. No line is drawn, in general the position of the turtle changes

Turtle Interpretation of Bracketed Strings

w: F

p: F → F[-F]F[+F][F]

Angle (δ) = 60º

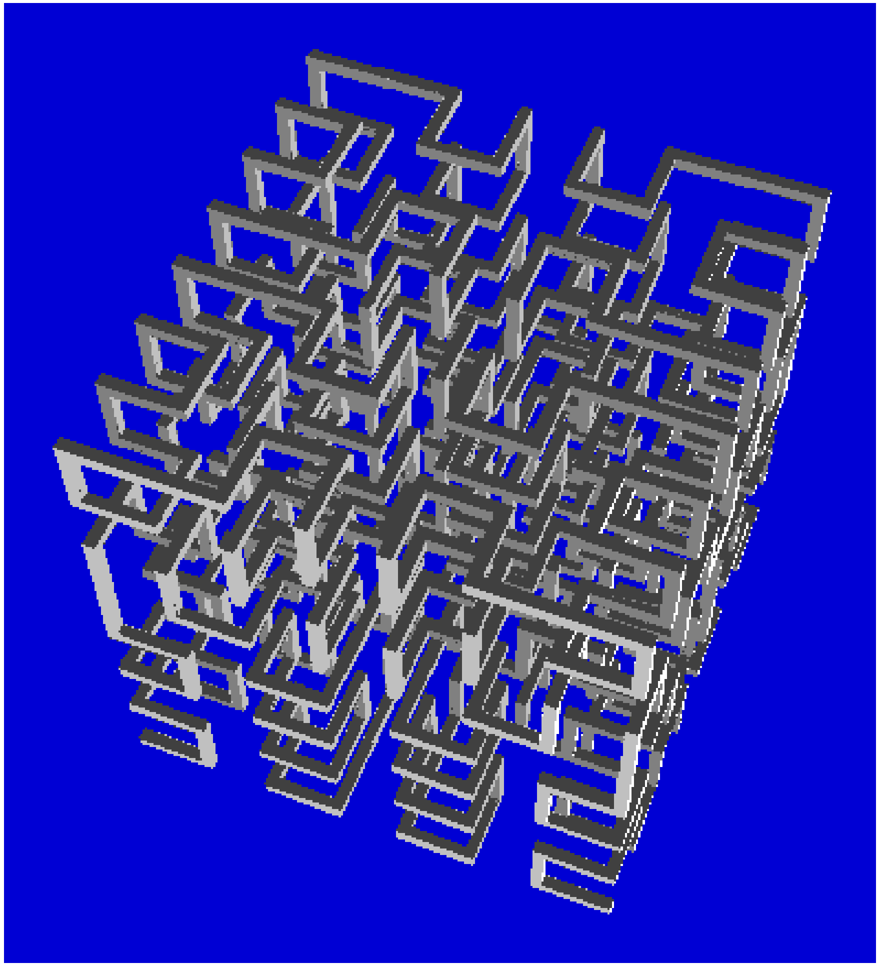


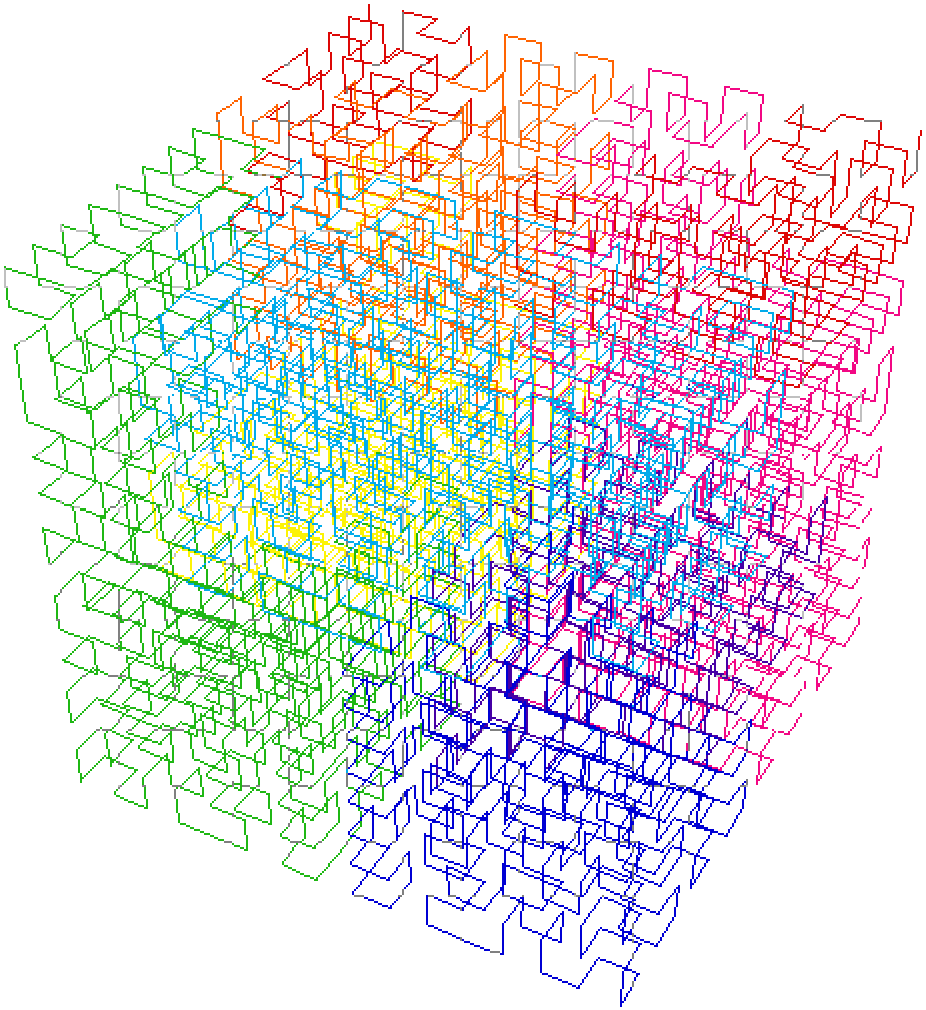
n = 1 - 5

Modeling in Three Dimensions

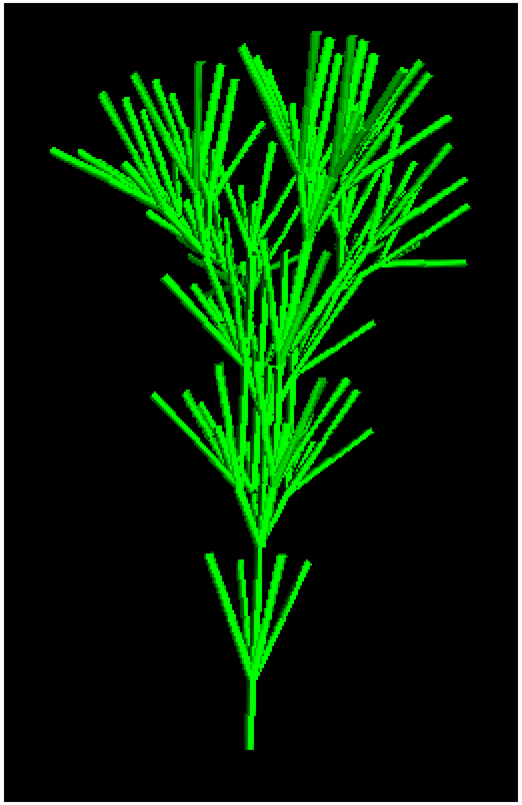
- Turtle interpretation of strings can be extended to 3D
- Represent the current orientation of the turtle in spave by 3 vectors: *H, L, U,* indicating turtle's *Heading*, the direction to the *Left*, and, the direction to the *Right*.
- 3 rotation matrices: R_U, R_L, and R_H and a fixed angle δ
- The following symbols control turtle orientation in space:
  - +, - : Turn left and right, using matrix $R_U(\delta)$
  - &, ^ : Pitch down and up, using matrix $R_L(\delta)$
  - \, / : Roll left and right, using matrix $R_H(\delta)$
  - | : Turning around, using matrix $R_U(180^{\circ})$

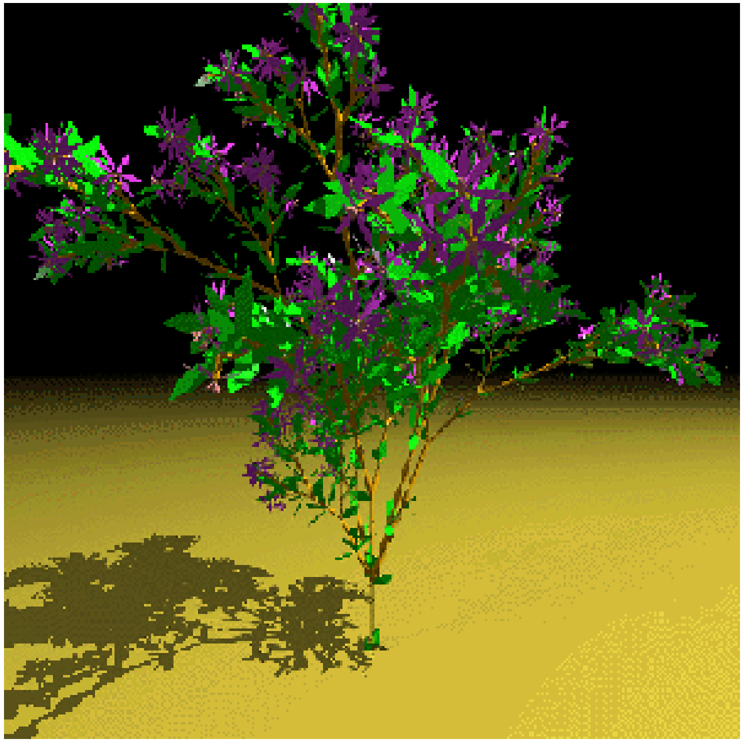
3D L-Systems





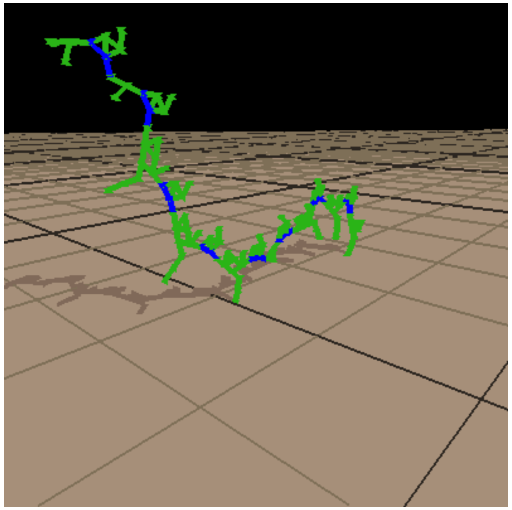
3D Bracketed L-Systems





Generative Encodings for Evolutionary Algorithms

- EAs has been applied to design problems. Past work has typically used a direct encoding of the solution
- Alternative: *Generative encoding*, i.e. an encoding that specifies how to construct the genotype
- Greater scalability through self-similar and hierarchical structure and reuse of parts
- Closer to Natural DNA encoding

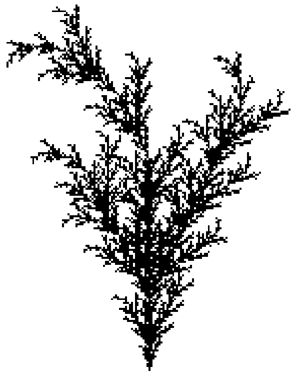


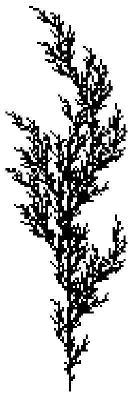
Examples of Generative Encoding for EAs

- Biomorphs, *The Blind Watchmaker* (R. Dawkins)
- Graph encoding for animated 3D creatures (K. Sims)
- L-Systems: plant-like structures, architectural floor design, tables, locomoting robots (C.Jacob, G. Ochoa, G. Hornby & J. Pollack, and others)
- Cellular automata rules to produce 2D shapes (H. de Garis)
- Context rules to produce 2D tiles (P. Bentley and S. Kumar)
- Cellular encoding for artificial neural networks (F. Gruau)
- Graph generating grammar for artificial neural networks (H. Kitano)

Evolving Plant-like Structures

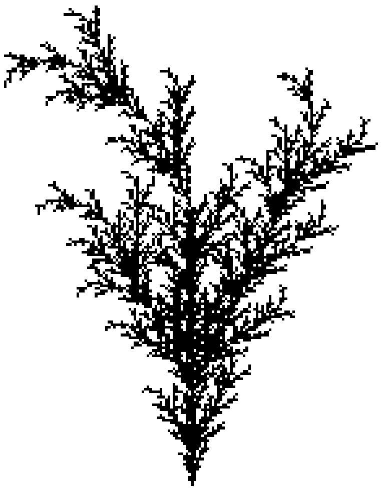
- Alife system for simulating the evolution of artificial plants
- Genotype: single ruled bracketed D0L-systems.
  - L-system: w: F,   p: F → F[-F]F[+F][F]
  - Chromosome: F[-F]F[+F][F]
- Phenotype: 2D branching structures, resulting from derivation and graphic interpretation of L-systems
- Genetic Operators: Recombination and mutation operators that preserve the syntactic structure of rules



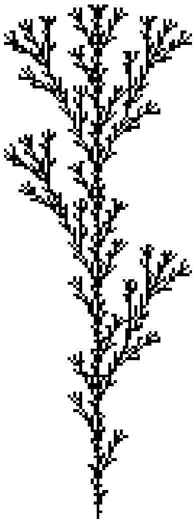


Recombination

Parents



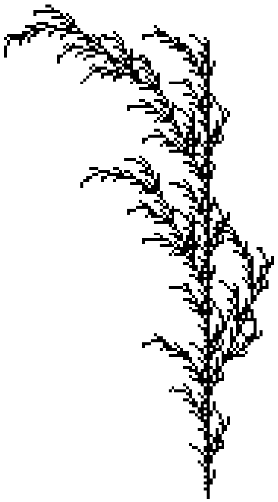
F[-FF]+[FFF]-FF**[-F-F]**



F[+F]+[-F-F]-FF**[+F]**[-F][F]

Offspring

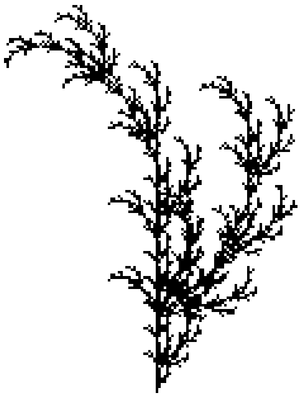
F[-FF]+[FFF]-FF**[+F]**



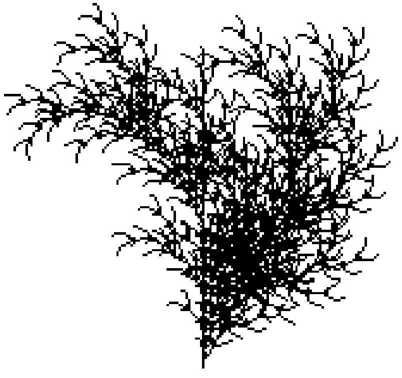
F[+F]+[-F-F]-FF**[-F-F]**[-F][F]

Mutation

Symbol Mutation

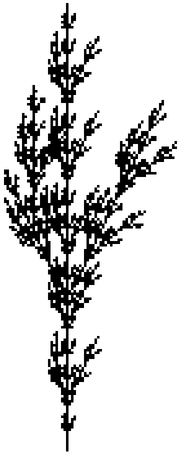


F[+F]+[+F-F-F]-F**F**[-F-F]



F[+F]+[+F-F-F]-F**[-F]**[-F-F]

Block Mutation



FF[+FF][-F+F]**[FFF]**F

FF[+FF][-F+F]**[-F]**F

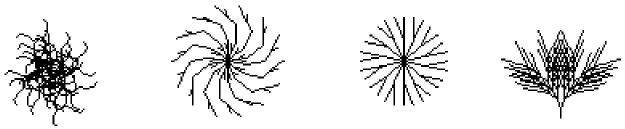
Evolving Plant-like Structures

- Selection
  - Automated: fitness Function inspired by evolutionary hypothesis concerning the factors that have had the greatest effect on plant evolution.
  - Interactive: allowing the user to direct evolution towards preferred phenotypes
- It is difficult of automatically measuring the aesthetic visual success of simulated objects or images. In most previous work the fitness is provided through visual inspection by a human

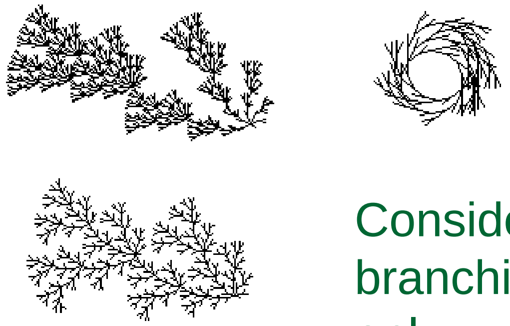
Automated Selection

- Hypotheses about plant evolution (K.Niklas, 1985):
  - Plants with branching patterns that gather the most light can be predicted to be the most successful (photosynthesis).
  - Evolution of plants was driven by the need to reconcile the ability to support vertical branching structures
- Analytic procedure, components:
  - (a) phototropism (growth movement of plants in response to stimulus of light),
  - (b) bilateral symmetry,
  - (c) proportion of branching points.

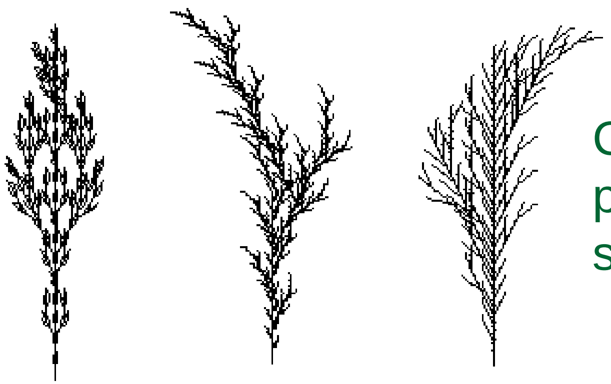
Results



Considering symmetry only

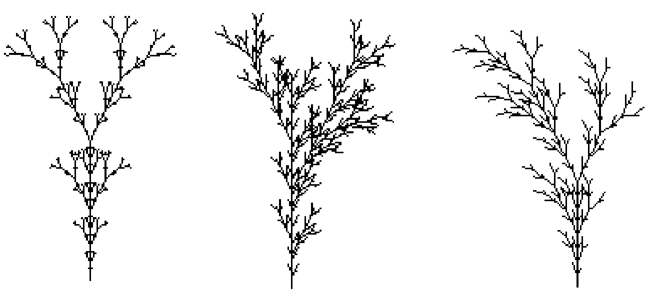


Considering branching points only



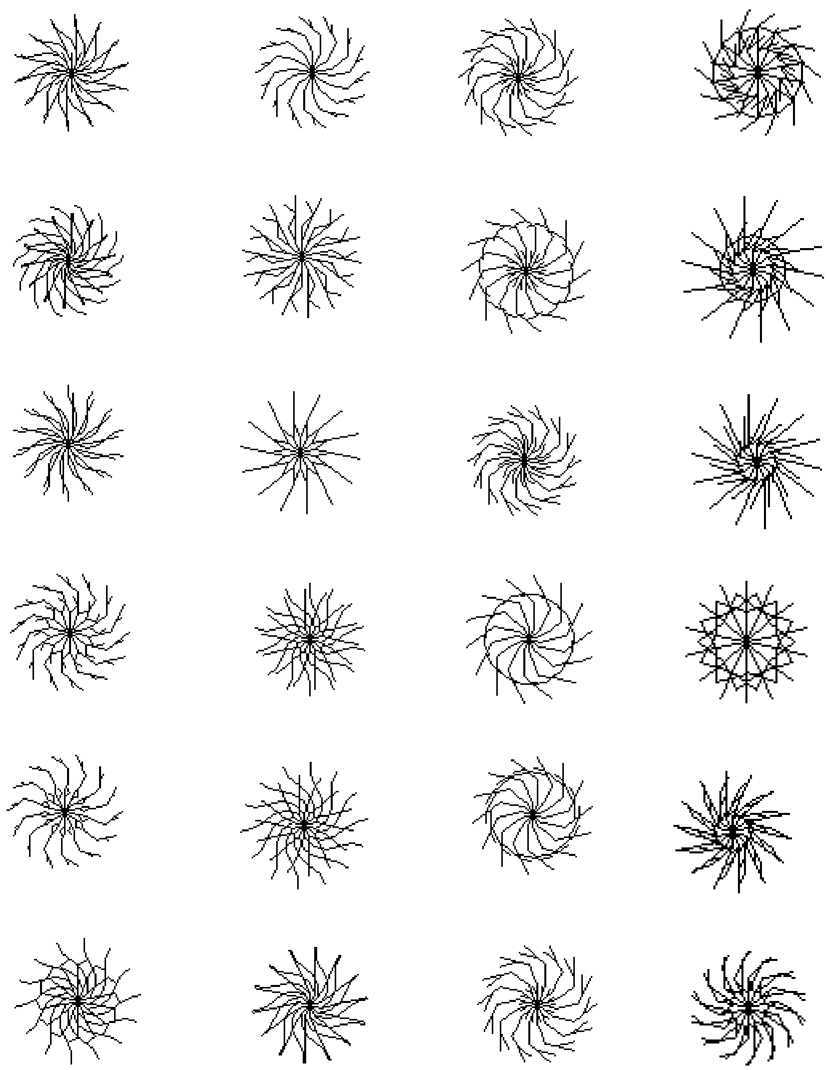
Considering phototropism, and symmetry

Considering phototropism only



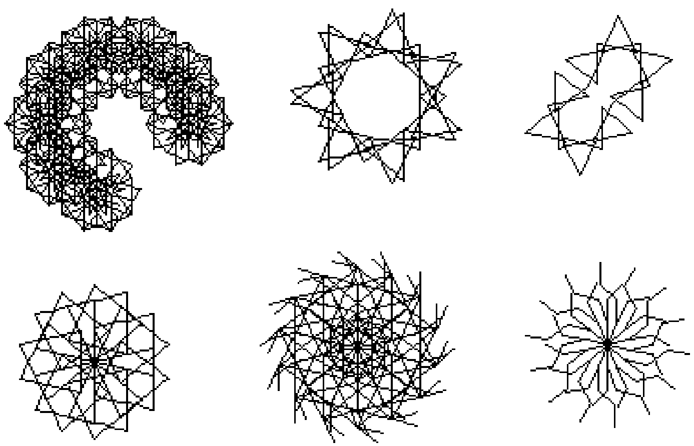
Considering phototropism, symmetry and branching points

Sea Stars and Urchins

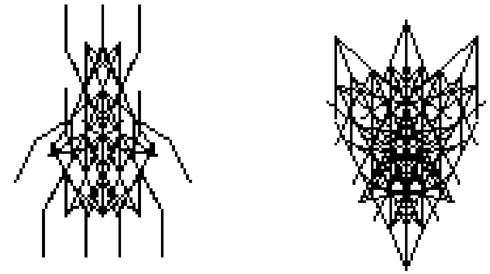


Obtained by a fitness function considering symmetry only. And interactively mutating and recombining organisms

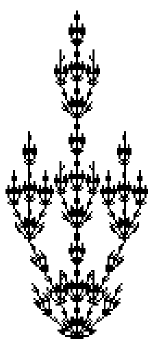
Some others unexpected figures!



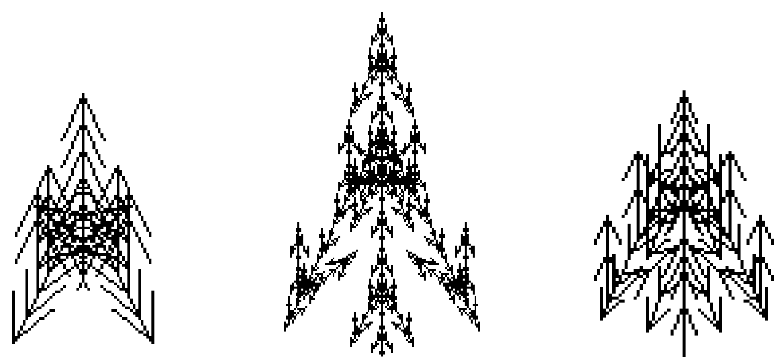
Stars



Animals



Candlestick



Rockets

Developmental rules for Neural Networks - 1

Firstly, biological neural networks:
there is simply not enough information in all our DNA to specify all the architecture, the connections within our nervous systems.

So DNA (... with other factors ...) must provide a developmental **'recipe'** which in some sense (partially) determines nervous system structure -- and hence contributes to our behaviour.

Developmental rules for Neural Networks - 2

Secondly, artificial neural networks (ANNs):
we build robots or software agents with ANNs which act as their nervous system or control system

Alternatives: (1) Design, (2) Evolve ANN architecture.
Evolving: (2.1) Direct encoding, (2.2) Generative encoding
Early References: Frederick Gruau, and Hiroaki Kitano.

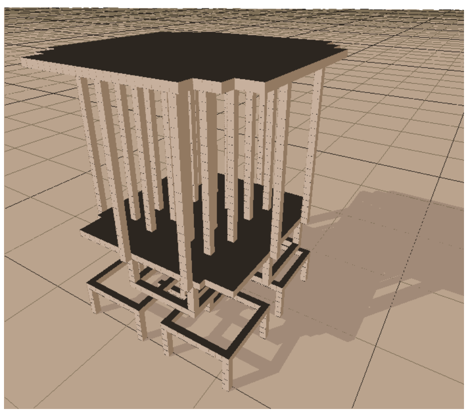
Gruau invented 'Cellular Encoding', with similarities to L-Systems, and used this for evolutionary robotics.

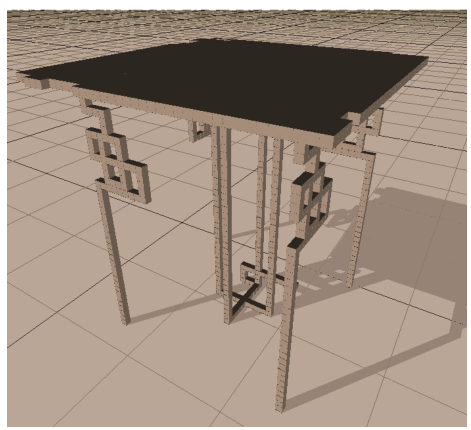
Kitano invented a 'Graph Generating Grammar‘.: A Graph L-System that generates not a 'tree', but a connectivity matrix for a network

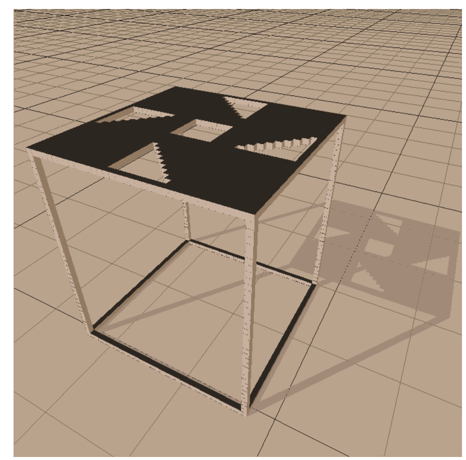
Generative Representations for Design Automation

- Dynamical & Evolutionary Machine Organization (DEMO). Brandeis University, Boston, USA

Evolved Tables: Fitness function rewarded structures for maximizing: height; surface area; stability/volume; and minimizing the number of cubes.

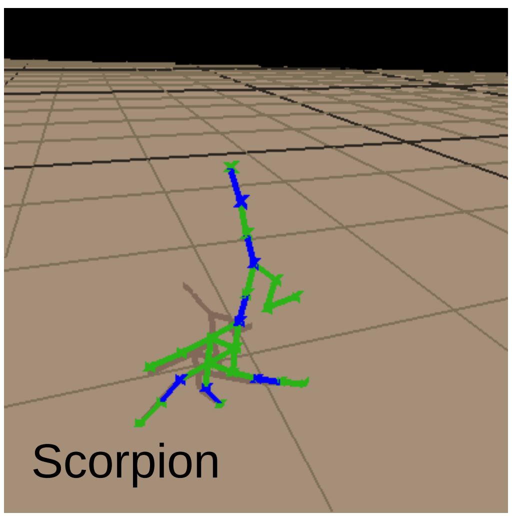




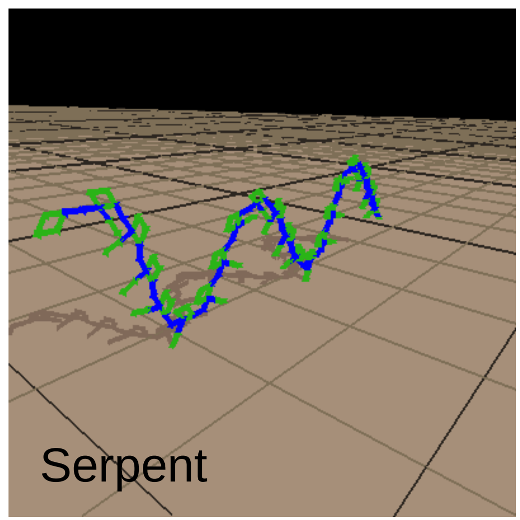


Hierarchically Regular Locomoting Robots

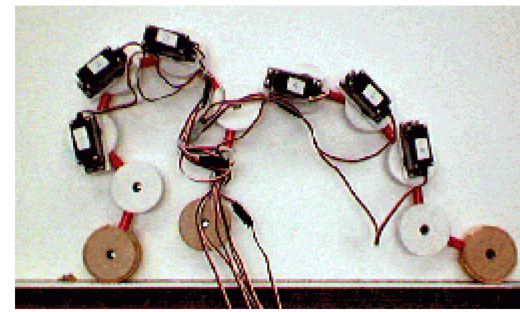
Evolve both the morphology and the controllers for different robots. Generative encoding based on L-systems



Scorpion

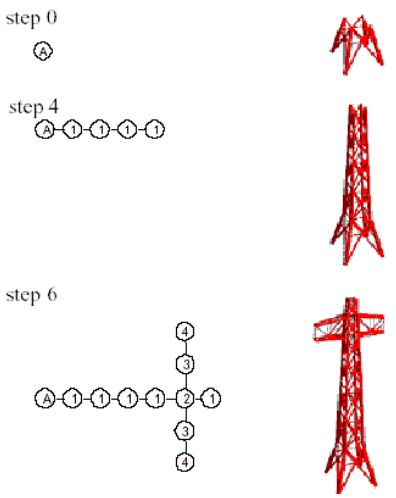


Serpent

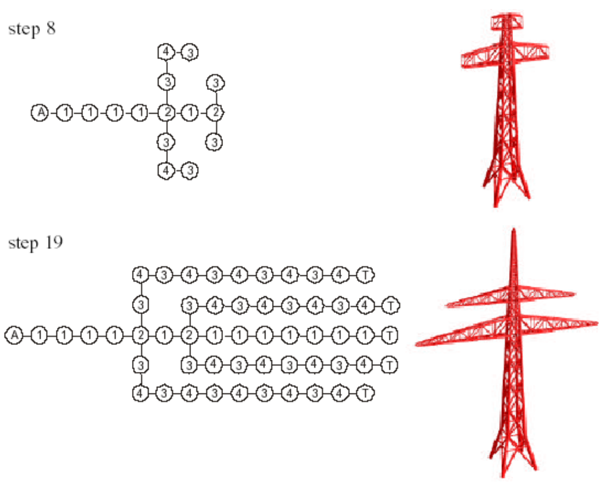


A constructed genobot

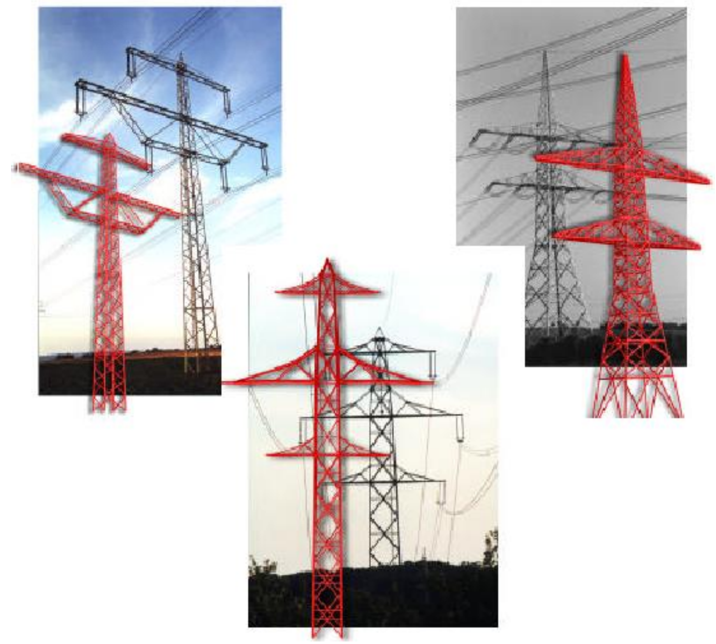
Grammar Based Representation of Transmission Towers



Evolutionary approach was applied to the Inverse Problem



i.e. The identification of a grammar that generates a predetermined tower



Real world towers translated into the grammar language

Conclusions (based on Hornby et. al)

- Main criticism for the use of EAs for design: it is doubtful whether it will reach the high complexities necessary for real applications
- Since the search space grows exponentially with the size of the problem, EAs with direct encoding will not scale to large designs
- Generative encoding (i.e. a grammatical encoding that specifies how to construct a design) can achieve greater scalability through self-similar and hierarchical structure
- Trough reuse of parts generative encoding is a more compact encoding of a solution

References

- Aristid Lindenmayer. Mathematical models for cellular interaction in development. parts I and II. *Journal of Theoretical Biology*, 18:280–299 and 300–315, 1968.
- P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, 1990.
- Richard Dawkins. The Evolution of Evolvability. In *Artificial Life*, C. Langton (ed) Addison Wesley 1989
- Richard Dawkins. *The Blind Watchmaker*. Harlow Longman (1986)
- Karl Niklas. *Computer Simulated Plant Evolution*. Scientific American (May 1985), (1985)
- Karl Niklas. *Biophysical limitations on plant form and evolution*. Plant Evolutionary Biology*,*Ed. L. D. Gottlieb and S. K. Jain. Chapman and Hall Ltd, (1988)
- H. Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*,4:461–476, 1990.
- Hugo de Garis. Artificial embryology : The genetic programming of an artificial embryo. In Branko Soucek and the IRIS Group, editors, *Dynamic, Genetic and Chaotic Programming*.Wiley, 1992.
- Karl Sims. Evolving Virtual Creatures. In *SIGGRAPH 94 Conference Proceedings*, Annual Conference Series,pages 15–22, 1994.
- Karl Sims. Evolving 3d morphology and behavior by competition. In R. Brooks and P. Maes, editors, *Proceedings of the Fourth Workshop on Artificial Life*,pages 28–39, Boston, MA, 1994. MIT Press.
- Gabriela Ochoa. On genetic algorithms and lindenmayer systems.In A. Eiben, T. Baeck, M. Schoenauer, and H. P.Schwefel, editors, *Parallel Problem Eolving from NatureV*, pages 335–344. Springer-Verlag, 1998

References

- C. Jacob. Genetic L-system Programming. In Y. Davidor and P. Schwefel, editors, *Parallel Problem Solvingfrom Nature III, Lecture Notes in Computer Science*,volume 866, pages 334–343, 1994.
- P. Coates, T. Broughton, and H. Jackson. Exploringthree-dimensional design worlds using lindenmayersystems and genetic programming. In P. J. Bentley, editor,*Evolutionary Design by Computers*, 1999
- C. Traxler and M. Gervautz. Using genetic algorithms to improve the visual quality of fractal plants generated with csg-pl-systems. In *Proc. Fourth International Conference in Central Europe on Computer Graphics andVisualization 96*, 1996.
- P. Bentley and S. Kumar. *Three ways to grow designs: Acomparison of embryogenies of an evolutionary design* problem. In Banzhaf, Daida, Eiben, Garzon, Honavar,Jakiel, and Smith, editors, *Genetic and EvolutionaryComputation Conference*, pages 35–43, 1999.
- Frederic Gruau. *Neural Network Synthesis Using Cellular Encoding and the Genetic Algorithm*. PhD thesis, Ecole Normale Sup´erieure de Lyon, 1994.
- Frederic Gruau and Kameel Quatramaran. Cellular encoding for interactive evolutionary robotics. Technical Report 425, University of Sussex, 1996.
- Rudolph, S., Alber, R.: An Evolutionary Approach To The Inverse Problem In Rule-based Design Representations. *Proceedings 7th International Conference onArtificial Intelligence in Design (AID'02)*, Kluwer Academic Publishers, 2002.

References

- Hornby, Gregory S., Lipson, Hod, and Pollack, Jordan B. **Generative Representations for the Automated Design of Modular Physical Robots**. *IEEE Transactions on Robotics and Automation*. (conditionally accepted).
- Pollack, Jordan B., Hornby, Gregory S., Lipson, Hod, and Funes, Pablo. **Computer Creativity in the Automatic Design of Robots**. *Leonardo, Journal for the International Society for Arts Sciences and Technology*. 36:2, 2003.
- Hornby, Gregory S. and Pollack, Jordan B. **Creating High-level Components with a Generative Representation for Body-Brain Evolution**. *Artificial Life*, 2002, 8:3.
- Hornby, Gregory S. and Pollack, Jordan B. **Evolving L-Systems to Generate Virtual Creatures**.
  Computers and Graphics, 2001, 25:6, pp 1041-1048.
- Pollack, Jordan B., Lipson, Hod, Hornby, Gregory S., and Funes, Pablo. **Three Generations of Automatically Designed Robots**. *Artificial Life*, 7:3, pg 215-223. 2001.
- Hornby, Gregory S. and Pollack, Jordan B. **Body-Brain Coevolution Using L-systems as a Generative Encoding**.
  Genetic and Evolutionary Computation Conference (GECCO) 2001.
- Hornby, Gregory S., Lipson, Hod, and Pollack, Jordan B. (2001). **Evolution of Generative Design Systems for Modular Physical Robots**.
  IEEE International Conference on Robotics and Automation (ICRA).
- Hornby, Gregory S. and Pollack, Jordan B. **The Advantages of Generative Grammatical Encodings for Physical Design**.
  Congress on Evolutionary Computation (CEC) 2001.