Procedural Content Generation

Lecture 1: Introduction

IT University of Copenhagen

Julian Togelius

What is PCG in games?

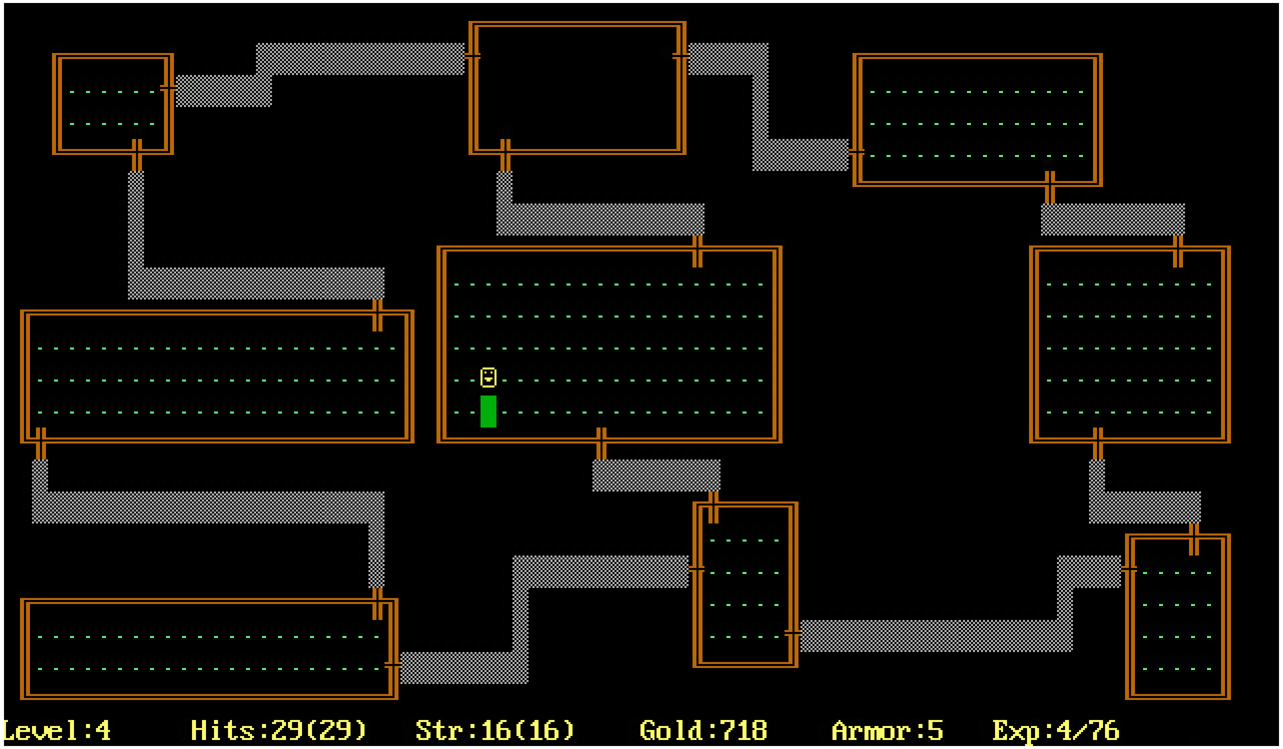
- Procedural Generation: with no or limited human intervention, algorithmically
- of Content: *not* NPC behaviour, *not* the game engine, things that affect gameplay
- in Games: computer games, board games... any kind of games

Game content, e.g.

- Levels, tracks, maps, terrains, dungeons, puzzles, buildings, trees, grass, fire, plots, descriptions, scenarios, dialogue, quests, characters, rules, boards, parameters, camera viewpoint, dynamics, weapons, clothing, vehicles, personalities...

History: Runtime random level generation

- Rogue-2D



1980

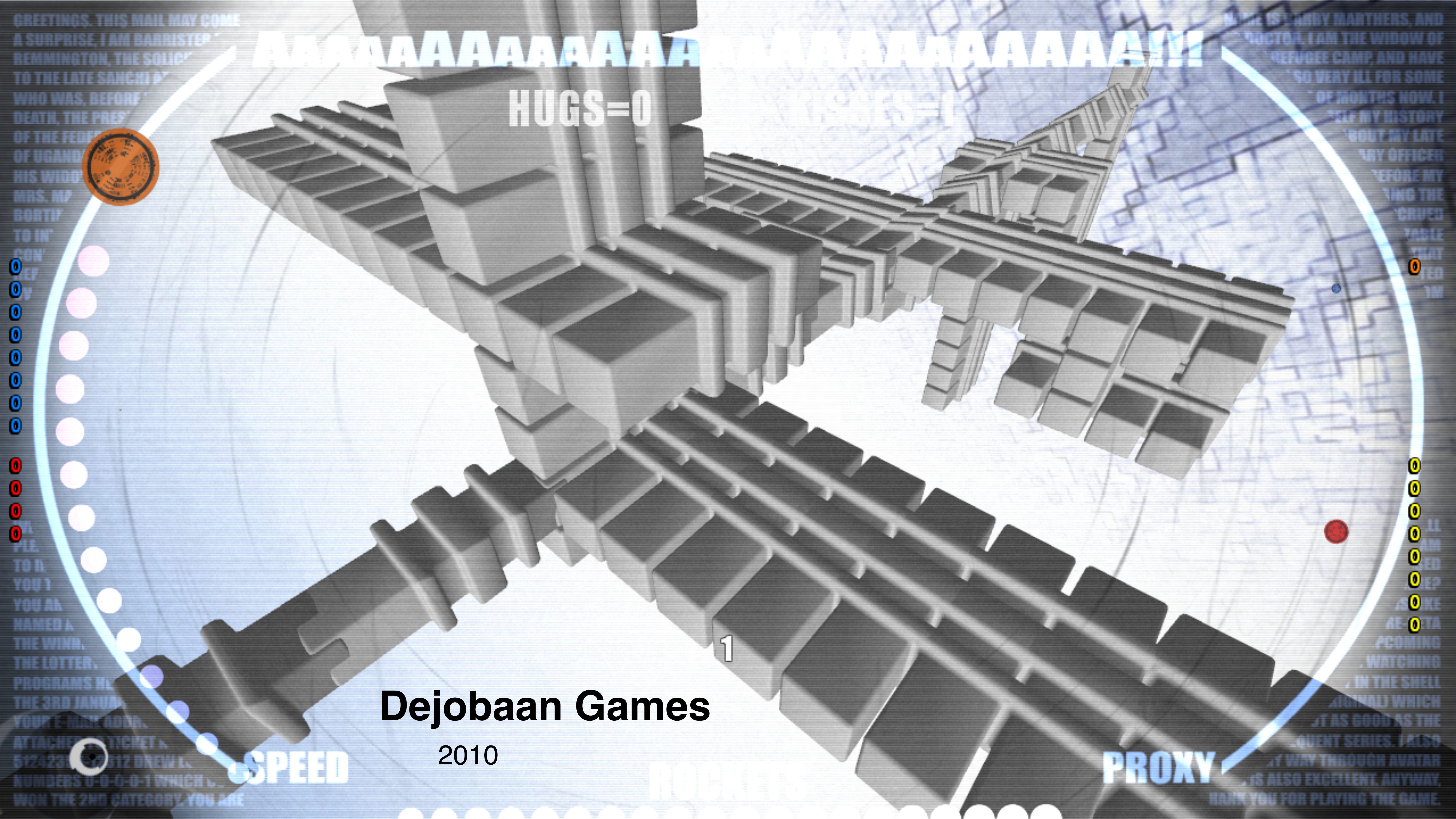
Civilization IV

2005

Diablo


LEVEL UP
CHAR
QUESTS
MAP
MENU
INV
SPELLS
GHOUL
TOTAL KILLS : 25
IMMUNE : MAGIC

2008



Dejobaan Games

2010

SpeedTree


KFS1
WEAPON
19 : 5
INFANTRY
F1 - mission
F2 - target
F3 - brigade
F4
F5
F6 - help
clear, cpdump, cpinfo, crs, defend, fbs, find, friends, gm, gmmode, grab, gridsnapshot, hc, hclist, hcmotd,
hcoff, hcon, help, home, hqoff, hqon, ignore, kick, kickall, kickban, lastcap, lastdeaths, lastmoves, listsquad,
liststos, m, makeco, makeleader, mlist, movebrigade , movedivision, msp, noechat, nowho, nukepersonas,
objectives, orders, overrun, own, pinfo, pwncp, rebuildcp, renamesquad, report, resetstrat, resync, roster,
setmotd, setrank, shutdown, snoop, spam, status, sys, sysmotd, time, trainers, um, ungrab, voteco, voting,
warn, warnall, whisper, who, wtf, yell

Sudoku

9			1					5
		5		9		2		1
8				4				
				8				
			7					
				2	6			9
2			3					6
			2			9		
		1	9		4	5	7	

The future...

- Can we drastically cut game development costs by creating content automatically from designers' intentions?
- Can we create games that adapt their game worlds to the preferences of the player?
- Can we create endless games?
- Can the computer circumvent or augment limited human creativity and create new types of games?

In general,

PCG > randomness

A taxonomy of PCG

- Online/Offline
- Necessary/Optional
- Random seeds/Parameter vectors
- Stochastic/Deterministic
- Constructive/Generate-and-test

Online/Offline

- Online: as the game is being played
- Offline: during development of the game

Necessary/Optional

- Necessary content: content the player needs to pass in order to progress
- Optional content: can be discarded, or bypassed, or exchanged for something else

Stochastic/ Deterministic

- Deterministic: given the same starting conditions, always creates the same content
- Stochastic: the above is not the case

Random seeds/ Parameter vectors

- a.k.a. dimensions of control
- Can we specify the shape of the content in some meaningful way?

Constructive/ Generate-and-test

- Constructive: generate the content once and be done with it
- Generate-and-test: generate, test for quality, and re-generate until the content is good enough

The Search-based Paradigm

- A special case of generate-and-test:
  - The test function returns a numeric fitness value (not just accept/reject)
  - The fitness value guides the generation of new candidate content items
- Usually implemented through evolutionary computation

Evolutionary computation?

- Keep a population of candidates
- Measure the fitness of each candidate
- Remove the worst candidates
- Replace with copies of the best (least bad) candidates
- Mutate/crossover the copies

Lecture 3: Plants and L-systems

Julian Togelius

(some material borrowed from Gabriela Ochoa)

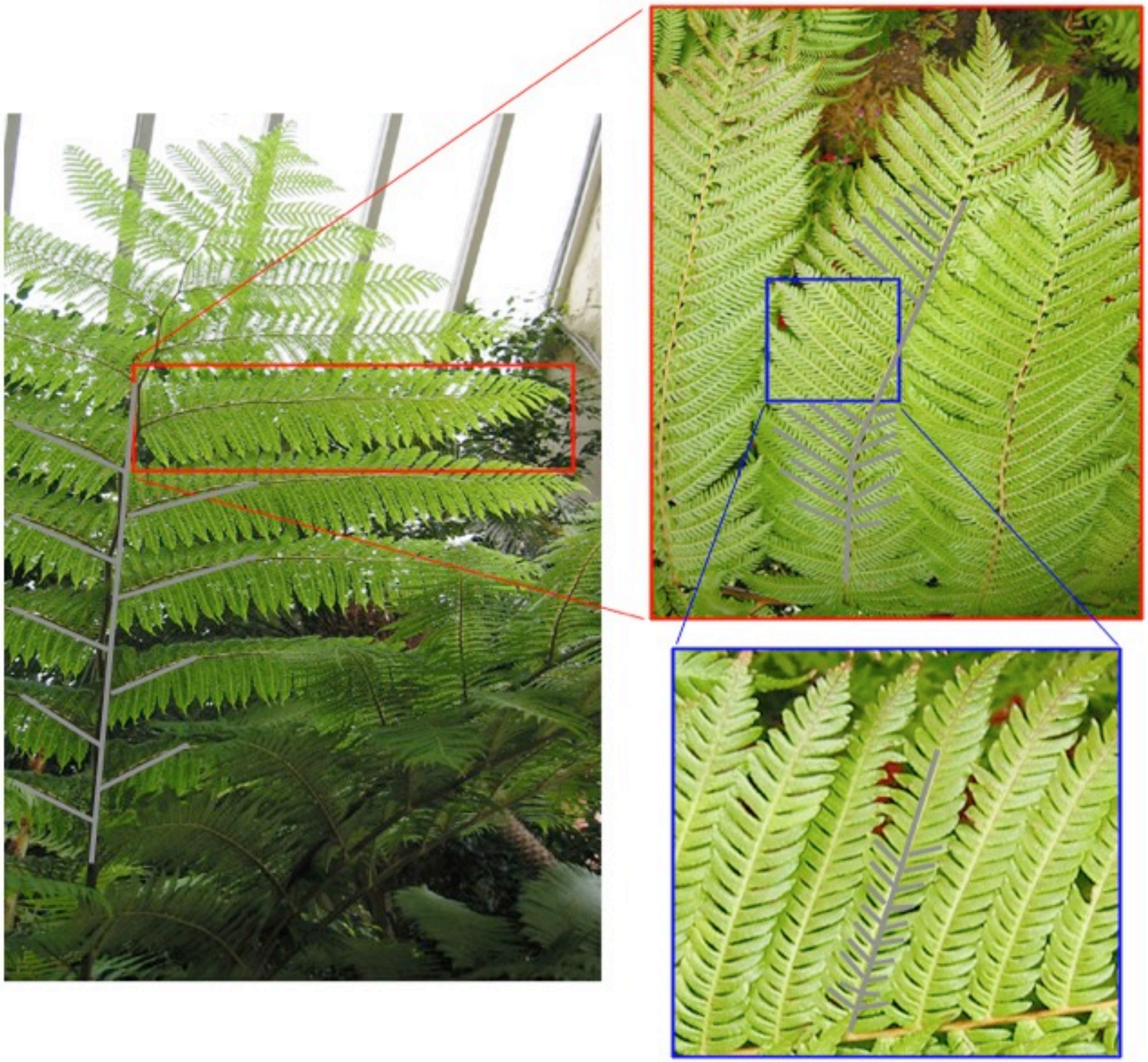
Plants?

- Core feature of the natural world... therefore of many games
- Need for believability
  - Infinitely detailed
  - Similar and recognizable, but not identical
- Need for compact representation
- Need for automatic large-scale generation

SpeedTree


KFS1
WEAPON
19 : 5
INFANTRY
F1 - mission
F2 - target
F3 - brigade
F4
F5
F6 - help
clear, cpdump, cpinfo, crs, defend, fbs, find, friends, gm, gmmode, grab, gridsnapshot, hc, hclist, hcmotd,
hcoff, hcon, help, home, hqoff, hqon, ignore, kick, kickall, kickban, lastcap, lastdeaths, lastmoves, listsquad,
liststos, m, makeco, makeleader, mlist, movebrigade , movedivision, msp, noechat, nowho, nukepersonas,
objectives, orders, overrun, own, pinfo, pwncp, rebuildcp, renamesquad, report, resetstrat, resync, roster,
setmotd, setrank, shutdown, snoop, spam, status, sys, sysmotd, time, trainers, um, ungrab, voteco, voting,
warn, warnall, whisper, who, wtf, yell

Self-similarity



Self-similarity

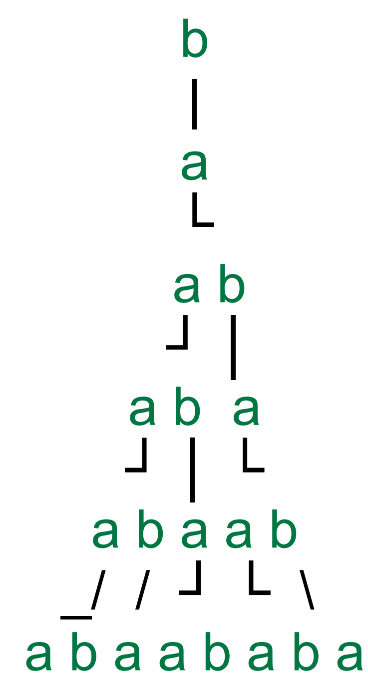
- Nature has obviously thought out some clever way of representing complex organisms using a compact description...
- ...permitting individual variation...
- ...why is this relevant for us?

L-systems

- Introduced by Aristid Lindenmeyer 1968, to model plant development
- Creates strings (text) from an *alphabet* based on a *grammar* and an *axiom*
- Closely related to Chomsky grammars (but productions carried out in parallel, not sequentially)

An example L-system

- Alphabet: {a, b}
- Production rules (grammar):
  a>ab
  b>a
- Axiom: b



Example of a derivation in a DOL-System

A graphical interpretation of L-systems

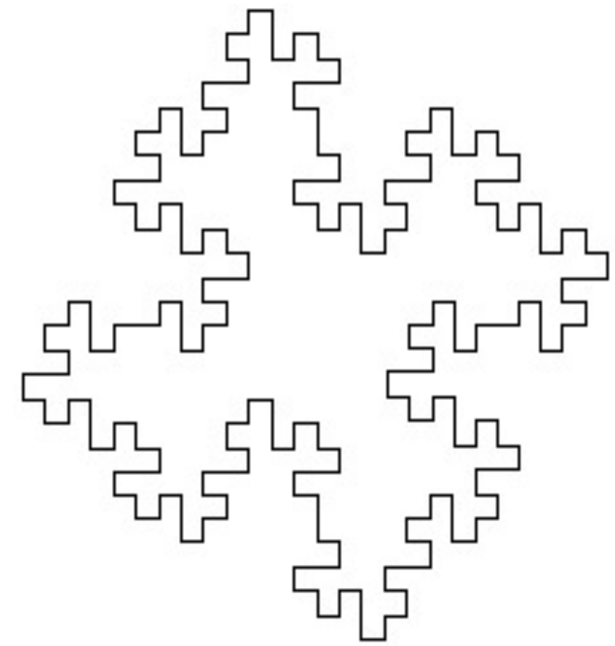
- Invented/popularized by Prusinkiewicz 1986
- Core idea: interpret generated strings as instructions for a turtle in turtle graphics
- Read the string from left to right, changing the state of the turtle (x, y, heading)

Example graphical L-system

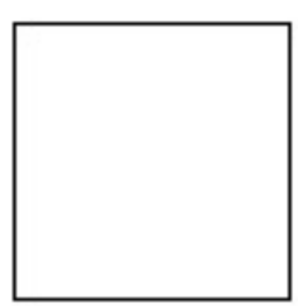
- Alphabet: {F, f, +, -}
- F: move the turtle forward (drawing a line)
- f: move the turtle forward (don't draw)
- +/-: turn right/left (by some angle)

Graphical L-system

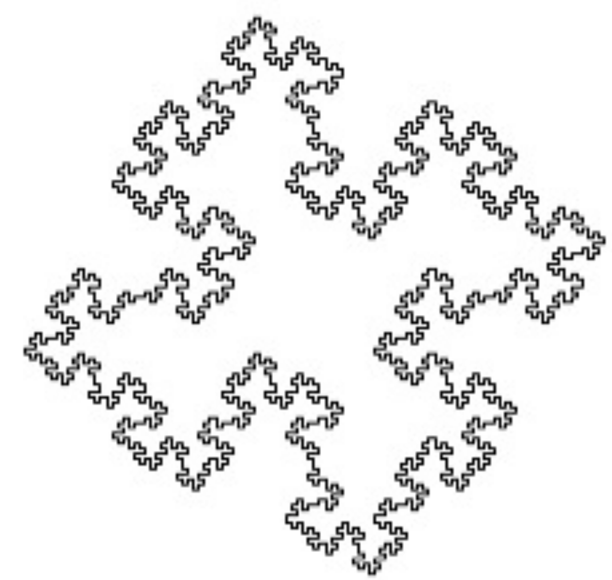
- axiom: F+F+F+F
- grammar: F>F+F-F-FF+F+F-F
- Turning angle: 90°



n=1



n=0



n=2

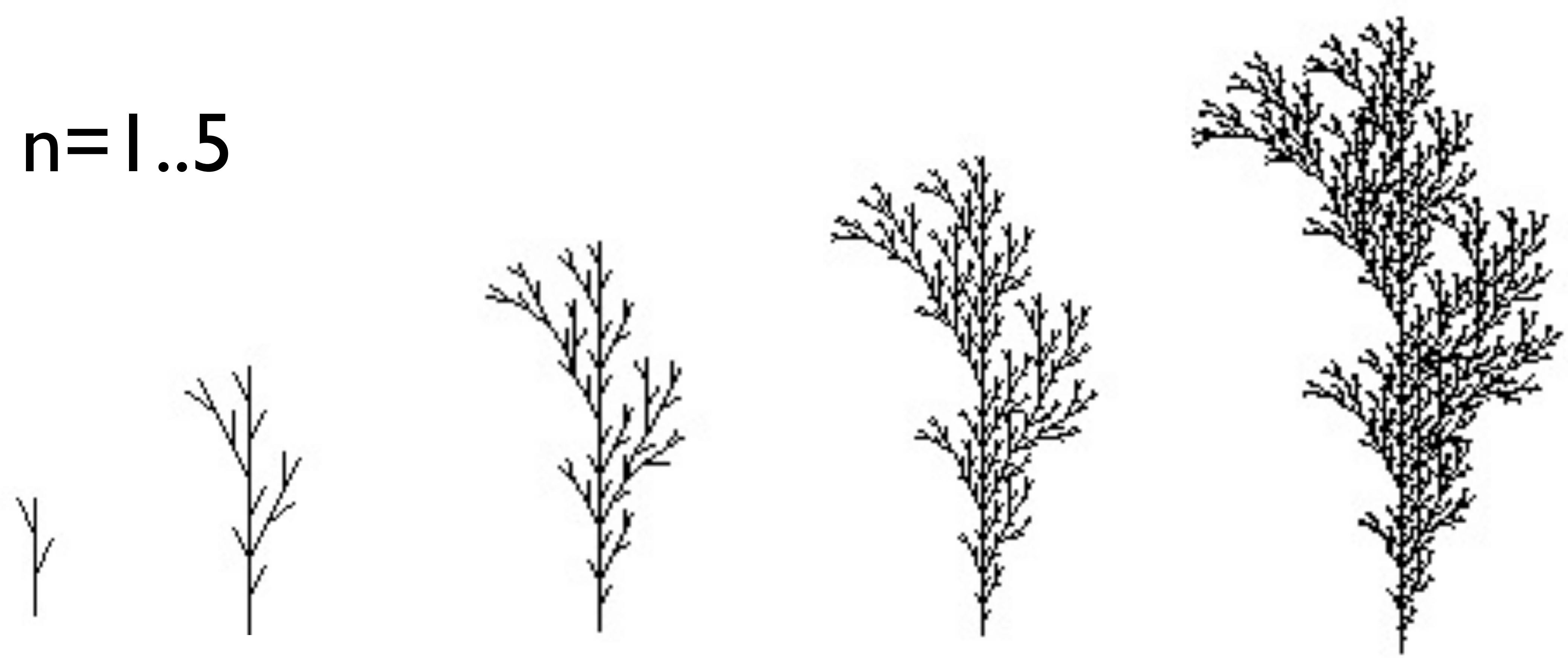
Bracketed L-systems

- Alphabet: {F, f, +, -, [,]}
- [: push the current state (x, y, heading of the turtle) onto a pushdown stack
-]: pop the current state of the turtle and *move* the turtle there *without drawing*
- Enables branching structures!

Bracketed L-systems

- Axiom: F
- Grammar: F>F[-F]F[+F][F]
- Turning angle: 30°

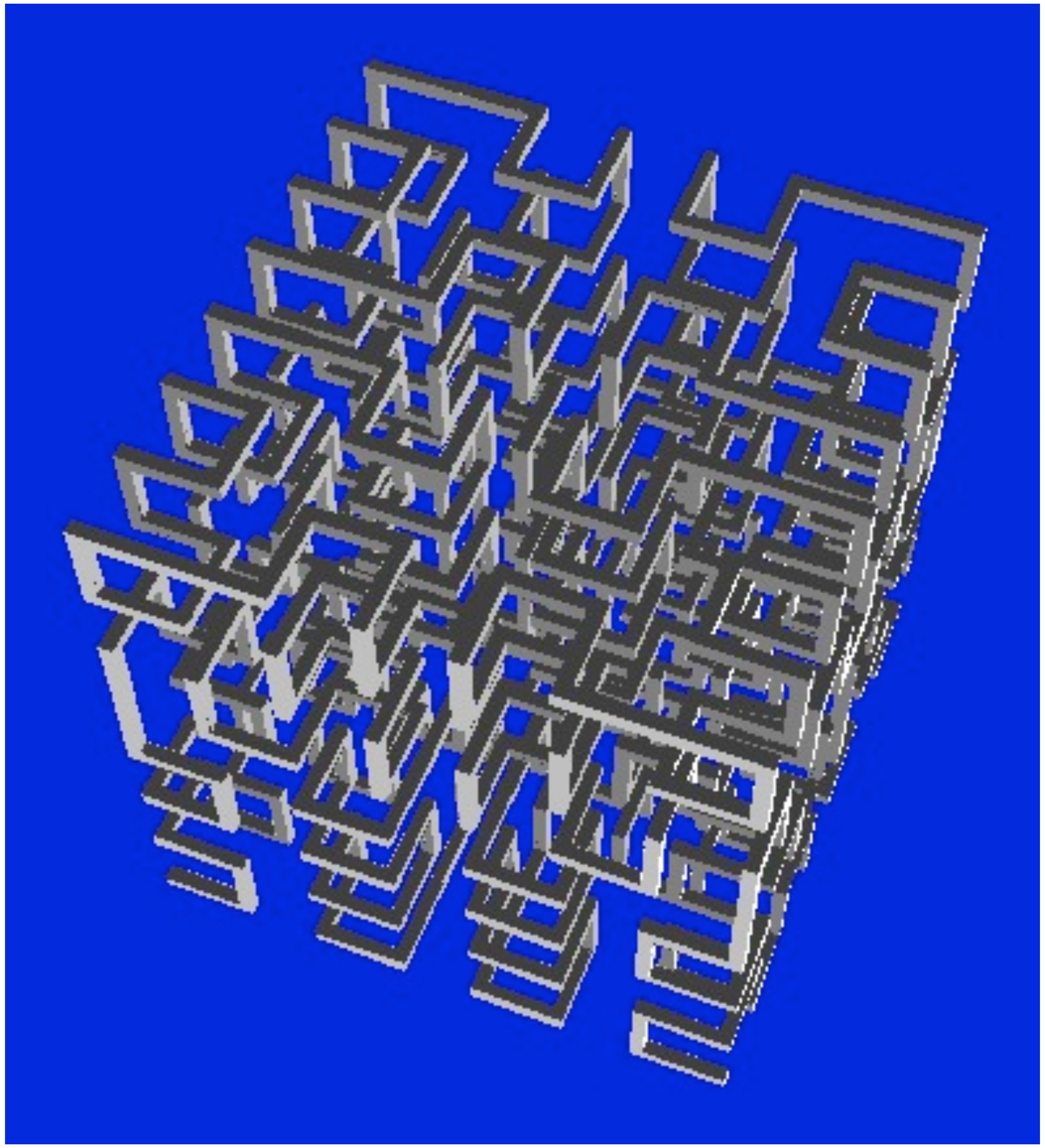
n=1..5

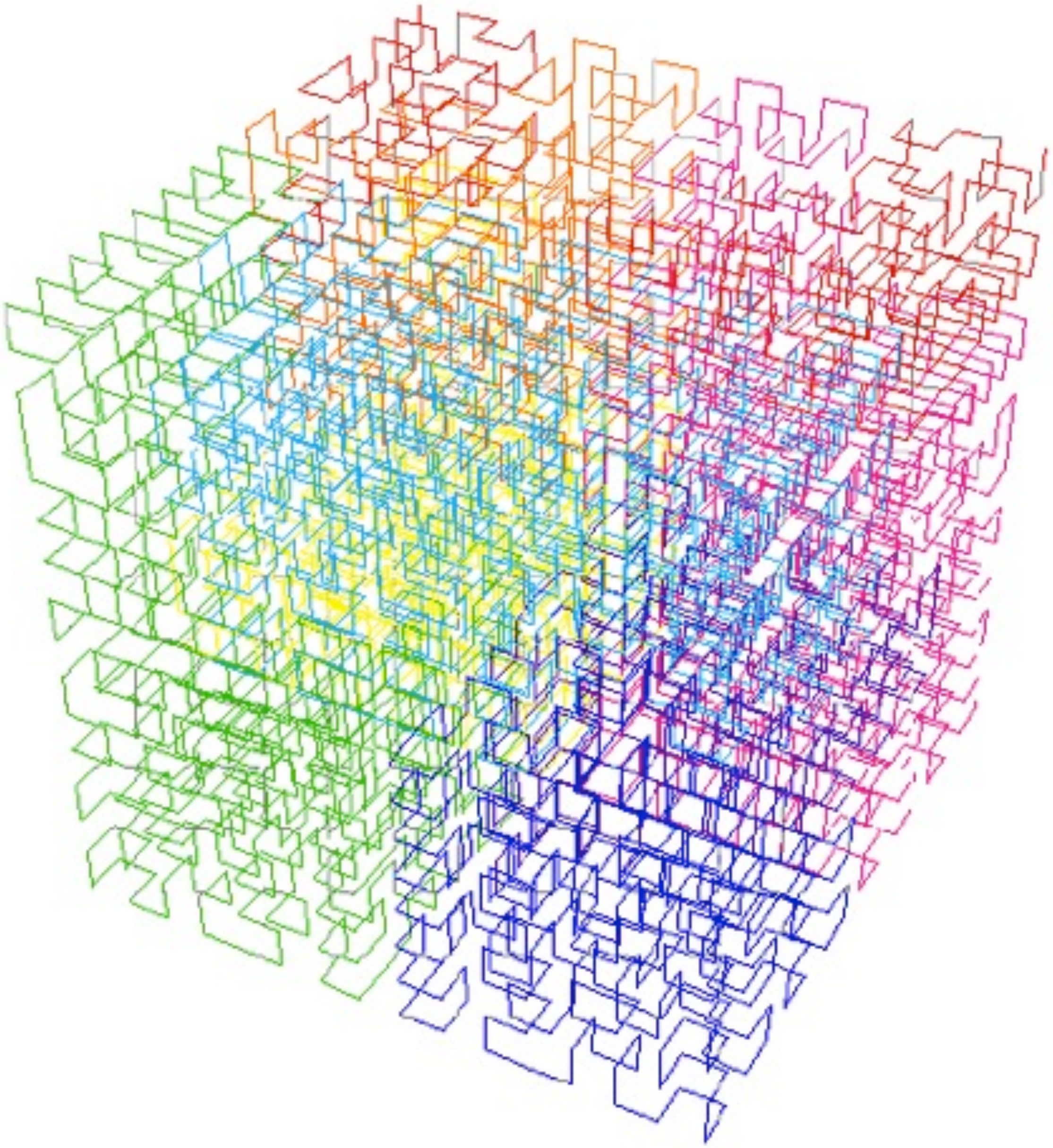


3D graphics

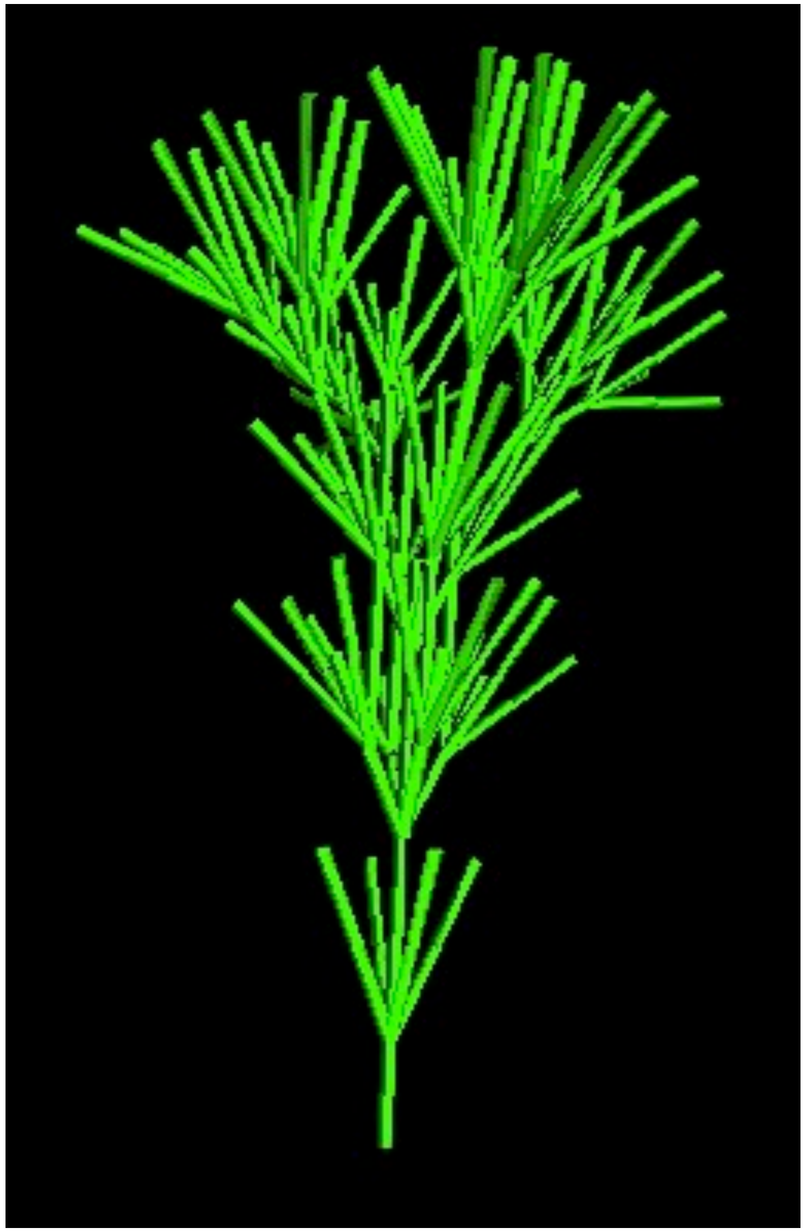
- Turtle graphics L-system interpretation can be extended to 3D space:
- Represent state as x, y, z and pitch, roll, yaw
- +, -: turn (yaw) left/right
- &, ^: pitch down/up
- \, /: roll left/right (counterclockwise/ clockwise)

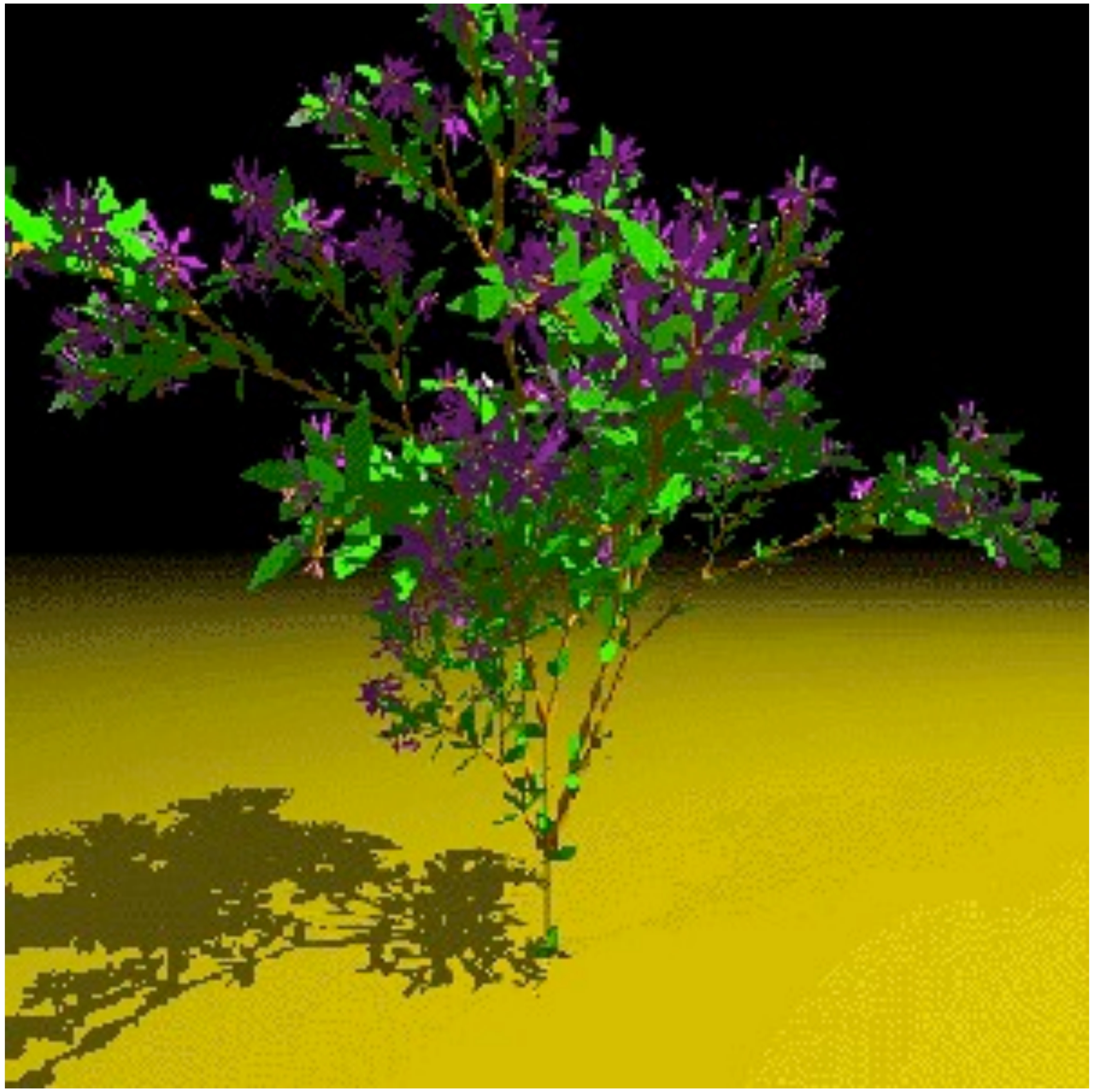
3D interpretation of L-systems



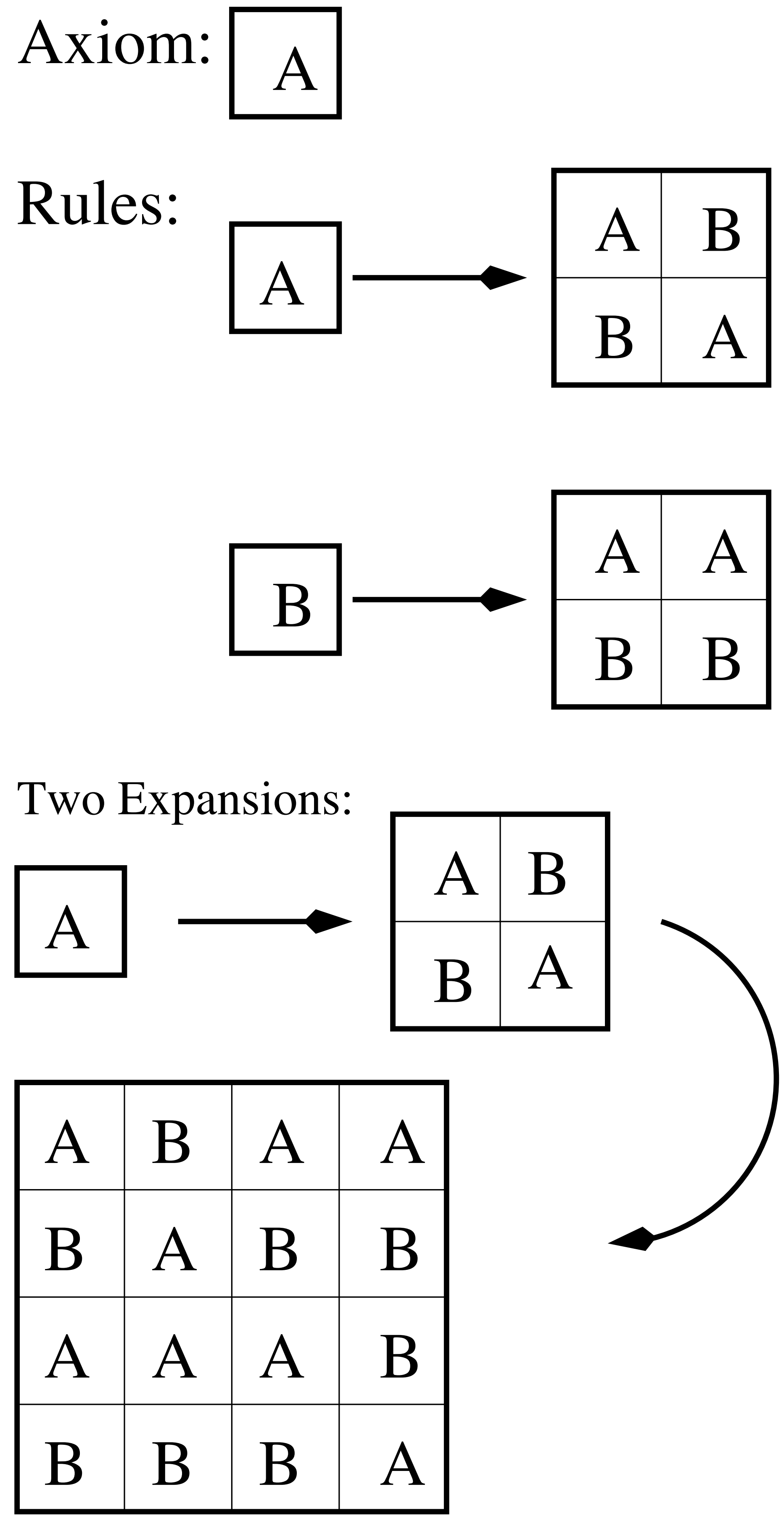


3D interpretation of bracketed L-systems



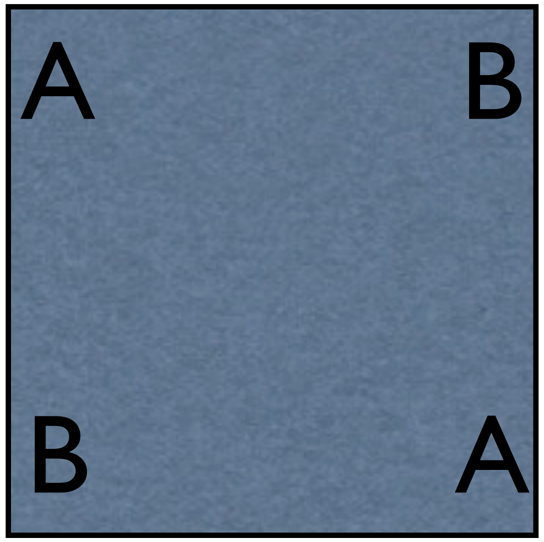


2D L-systems


Axiom:
A
Rules:
A
A B
B A
B
A A
B B
Two Expansions:
A
A B
B A
A B A A
B A B B
A A A B
B B B A

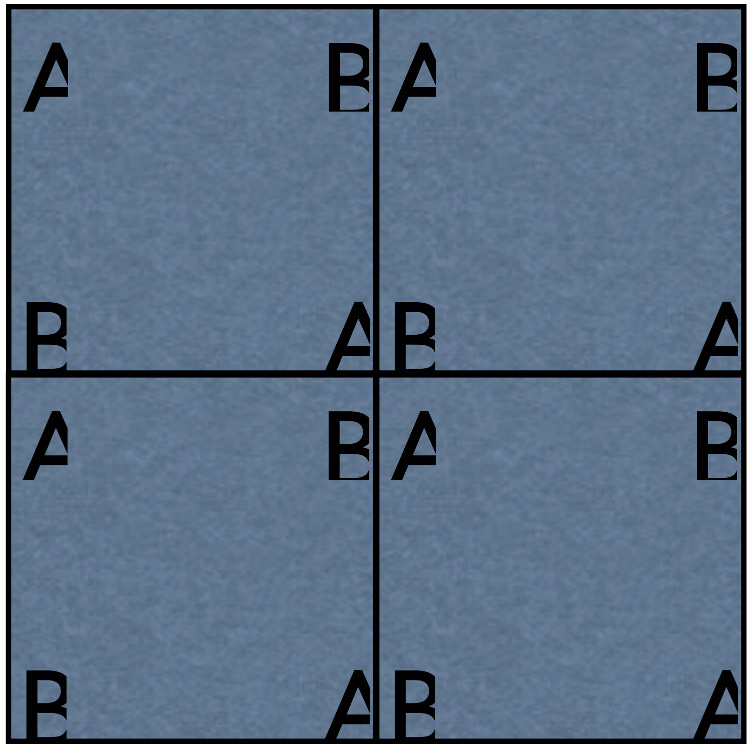
Terrain interpretation of 2D L-systems

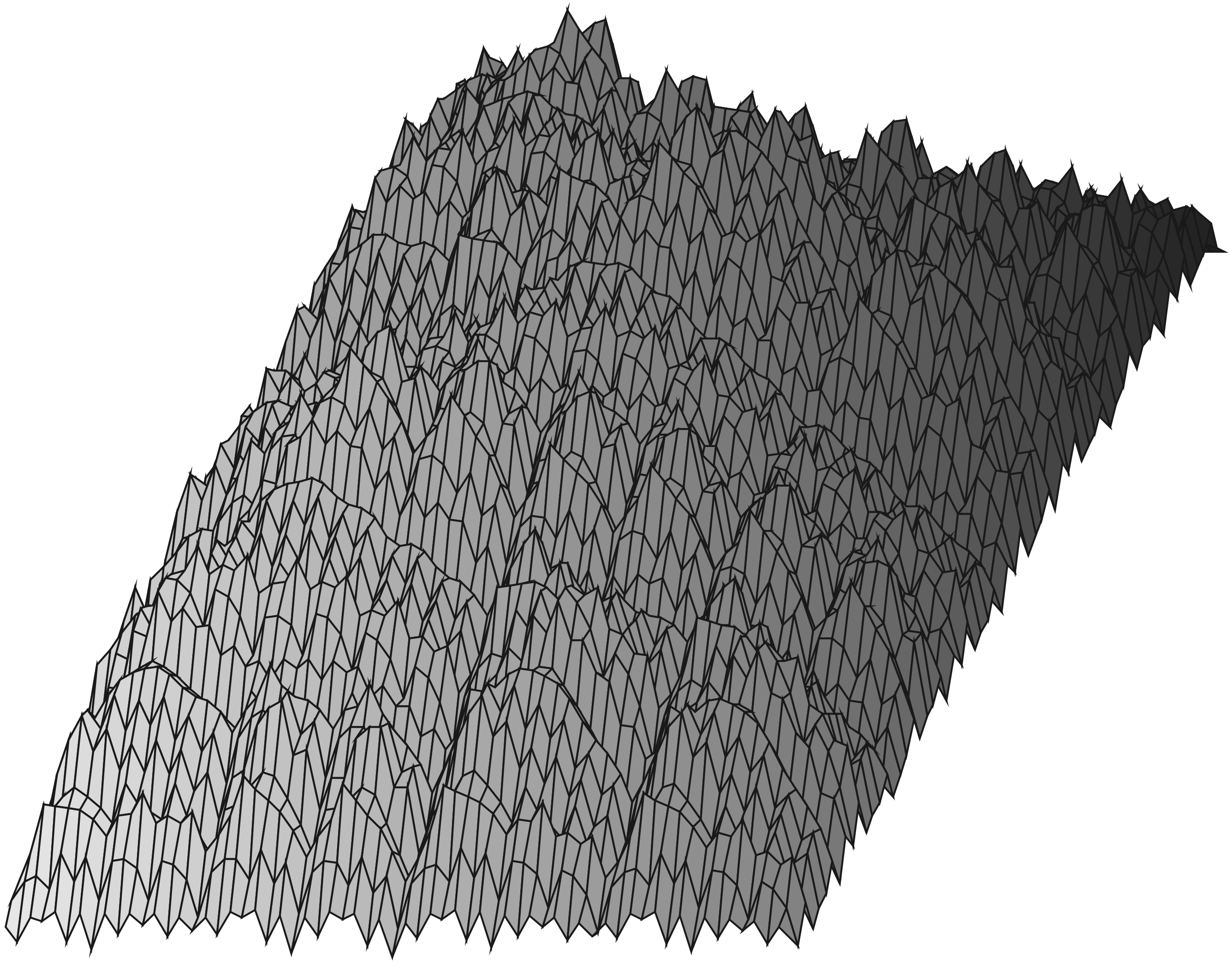
- Each group of four letters is interpreted as instructions for lowering or raising the corners of a square
- e.g. A=+0.5, B=-0.5



Terrain interpretation of 2D L-systems

- In next iteration, the 2D L-system is rewritten once, and each square is divided into two
- "Doubling the resolution"





Evolving L-systems

- How can we combine L-systems with evolutionary computation?

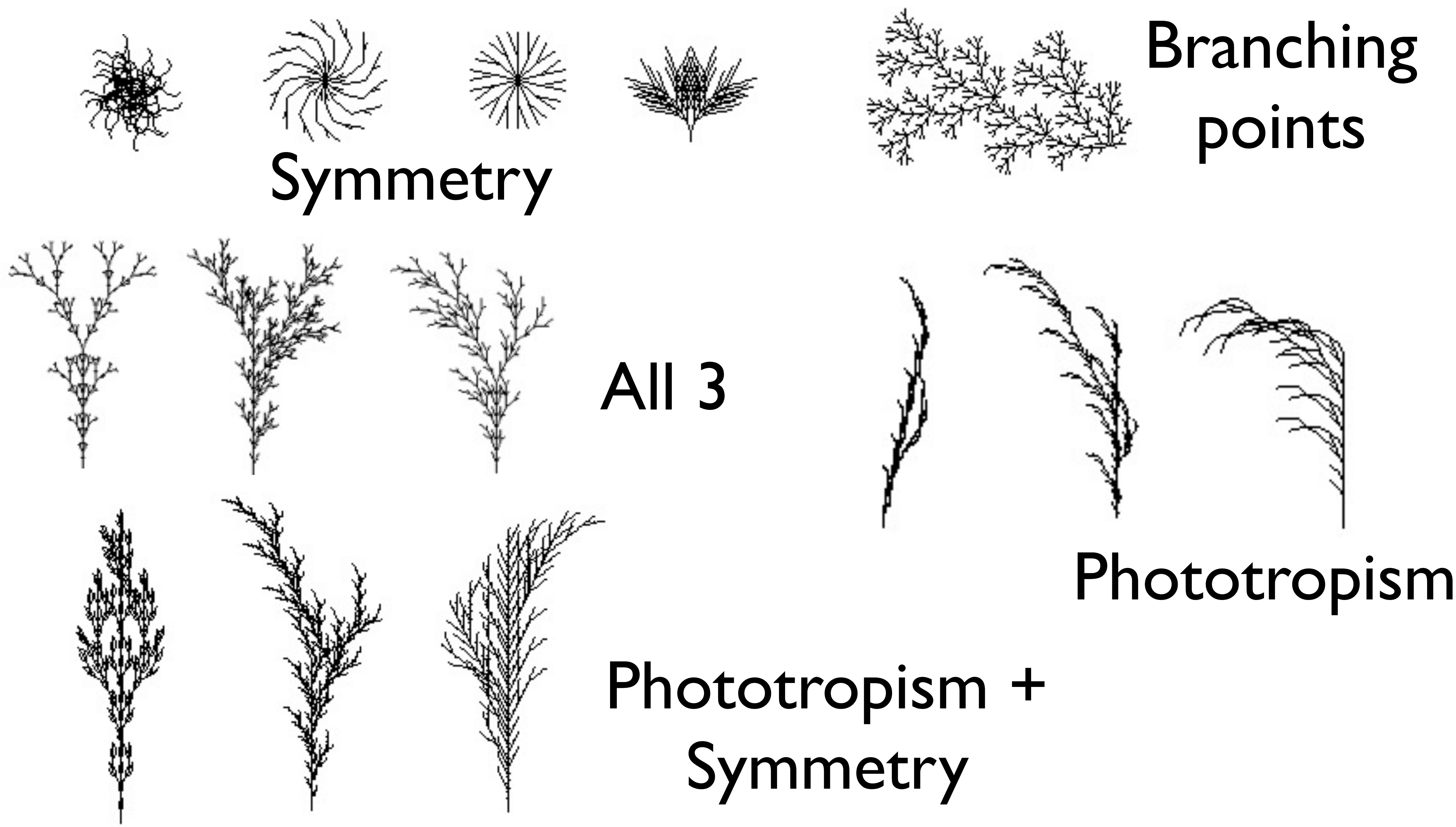
Evolving L-systems

- Evolving the axiom
- Evolving the grammar:
  - change the shape of one or more production rules, or
  - add/remove/replace productions
  - counter limits
- Evolving the interpretation:
  - Evolve production probabilities
  - Evolve other aspects (e.g. turning angles)

Fitness functions

- Phototropism
- Bilateral symmetry
- Proportion of branching points

Evolved L-systems



Multiobjective Exploration of the StarCraft Map Space

Julian Togelius, Mike Preuss,
Nicola Beume, Simon Wessing,
Johan Hagelbäck and Georgios N. Yannakakis

StarCraft

- Classic real-time strategy game
- Korea's unofficial national sport
- Two or three player competitive matches
- Three distinct races

Why generate maps?

- Give players an unlimited supply of new, unpredictable maps
  - Negates rote learning advantages
- Dynamically adapt the game to individual players' strengths...
  - ...or to groups of players!
- Help designers generate more novel and balanced maps
  - Help them with the "boring stuff"

Traditional (constructive) map generation

- Place features on maps according to some heuristic
  - e.g. fractals, growing islands, cellular automata
- Hard or impossible to optimize for gameplay properties
- Restrictions on possible content necessary in order to ensure valid maps

Our approach:

- Direct/indirect map representations
- An ensemble of fitness functions
- Multiobjective evolution

Our approach

- Define desirable traits of RTS maps
- Operationalize these traits as fitness functions
- Define a search space for maps
- Search for maps that satisfy the fitness functions as well as possible, using multiobjective evolution
  - (visualize trade-offs as Pareto fronts)

Desirable traits of an RTS map

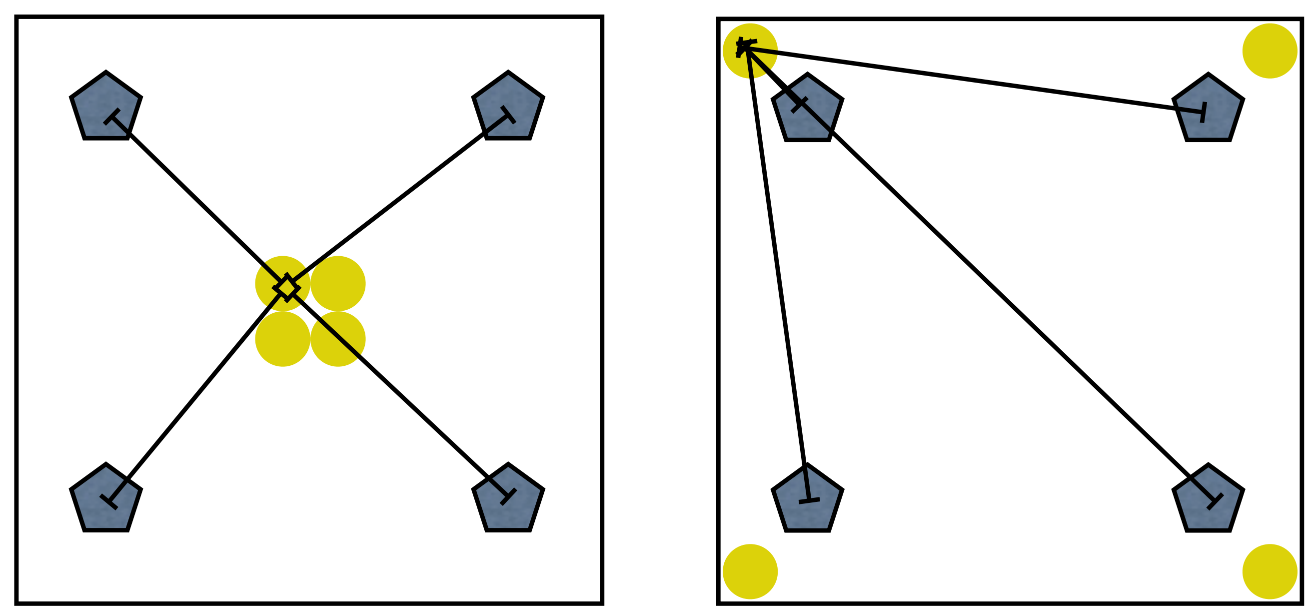
- Playability
- Fairness
- Skill differentiation
- Interestingness

Playability fitness functions

- Base space: minimum amount of space around bases
- Base distance: minimum distance between bases (via A*)

Fairness fitness functions

- Distance from base to closest resource
- Resource ownership
- Resource safety
- Resource fairness



(a) unsafe resources (b) safe resources

Skill differentiation fitness functions

(also contribute to interestingness)

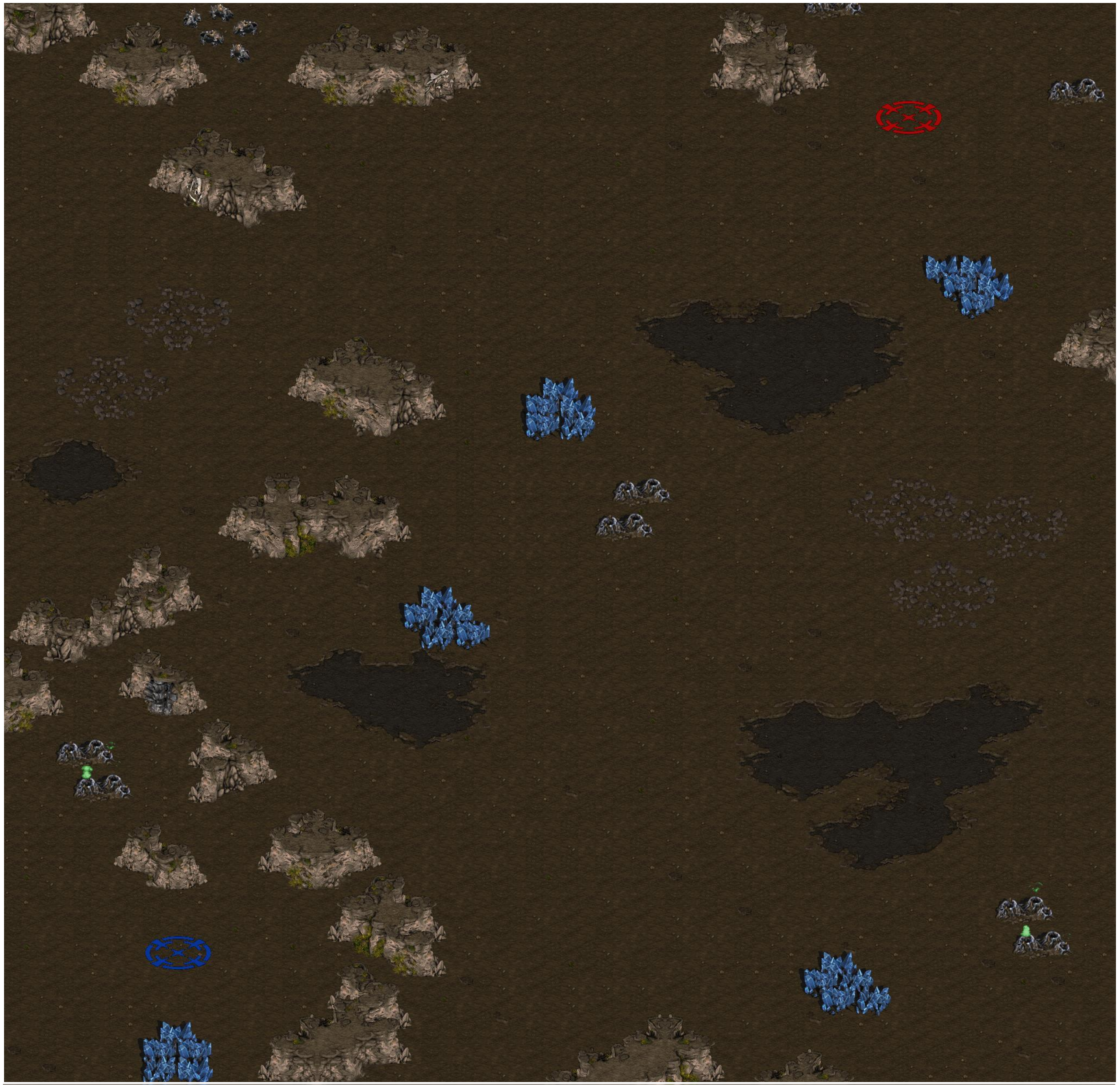
- Choke points
  (narrowest width of shortest path)
- Path overlapping

Dual map representation

- Indirect representation: a vector of real numbers in {0..1}
- Direct representation: a 64x64 grid corresponding to a StarCraft map, including impassable areas, bases, resource sites
- Genotype to phenotype mapping: before fitness calculation

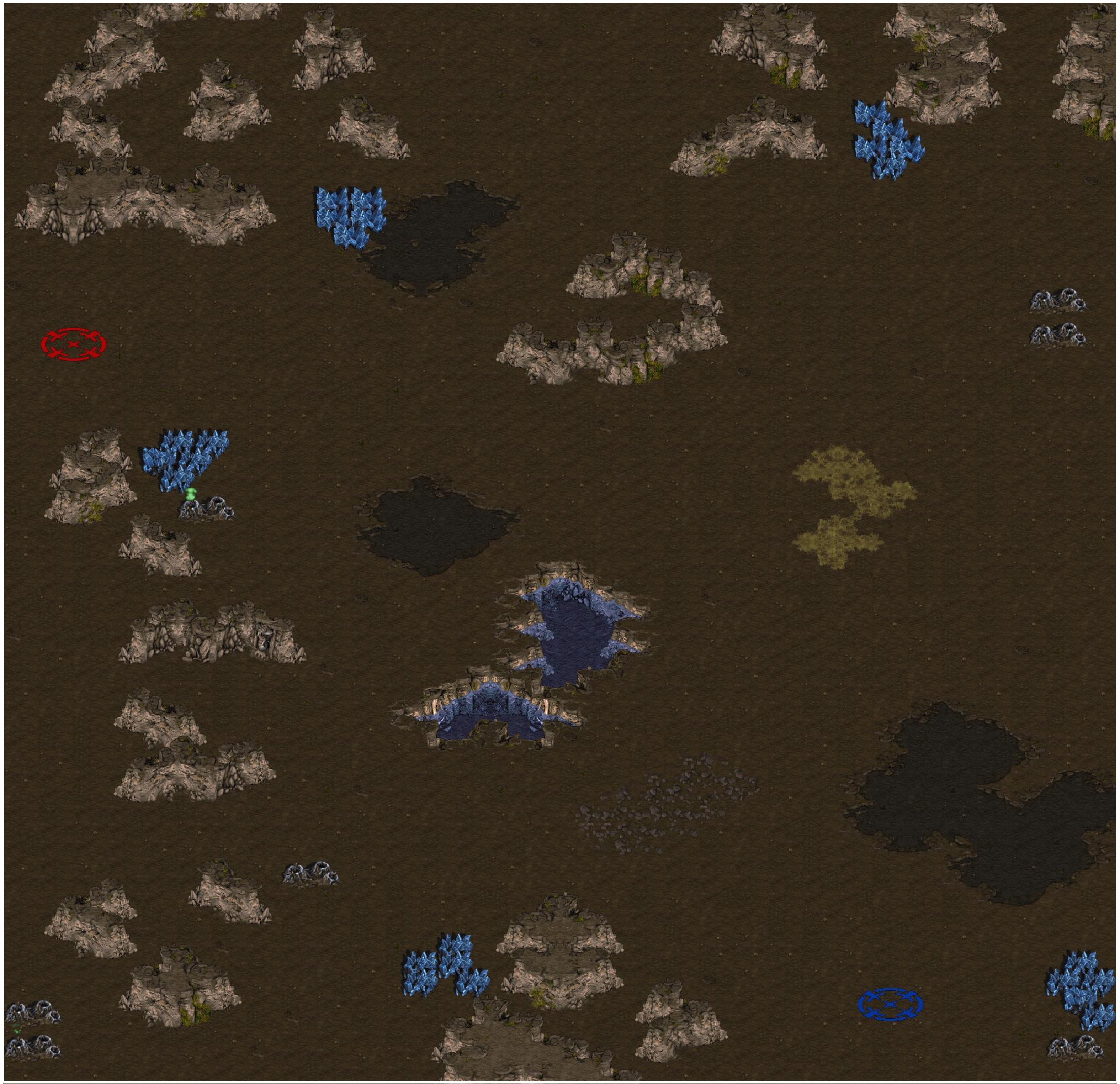
Genotype to phenotype

- Two or three bases, five mineral sources and five gas wells: *(phi, theta)* coordinates
- Rock formations represented indirectly using "turtle graphics". Each formation has:
  - *(x, y)* starting position
  - probability of turning left/right
  - probability of gaps ("lifting the pen")



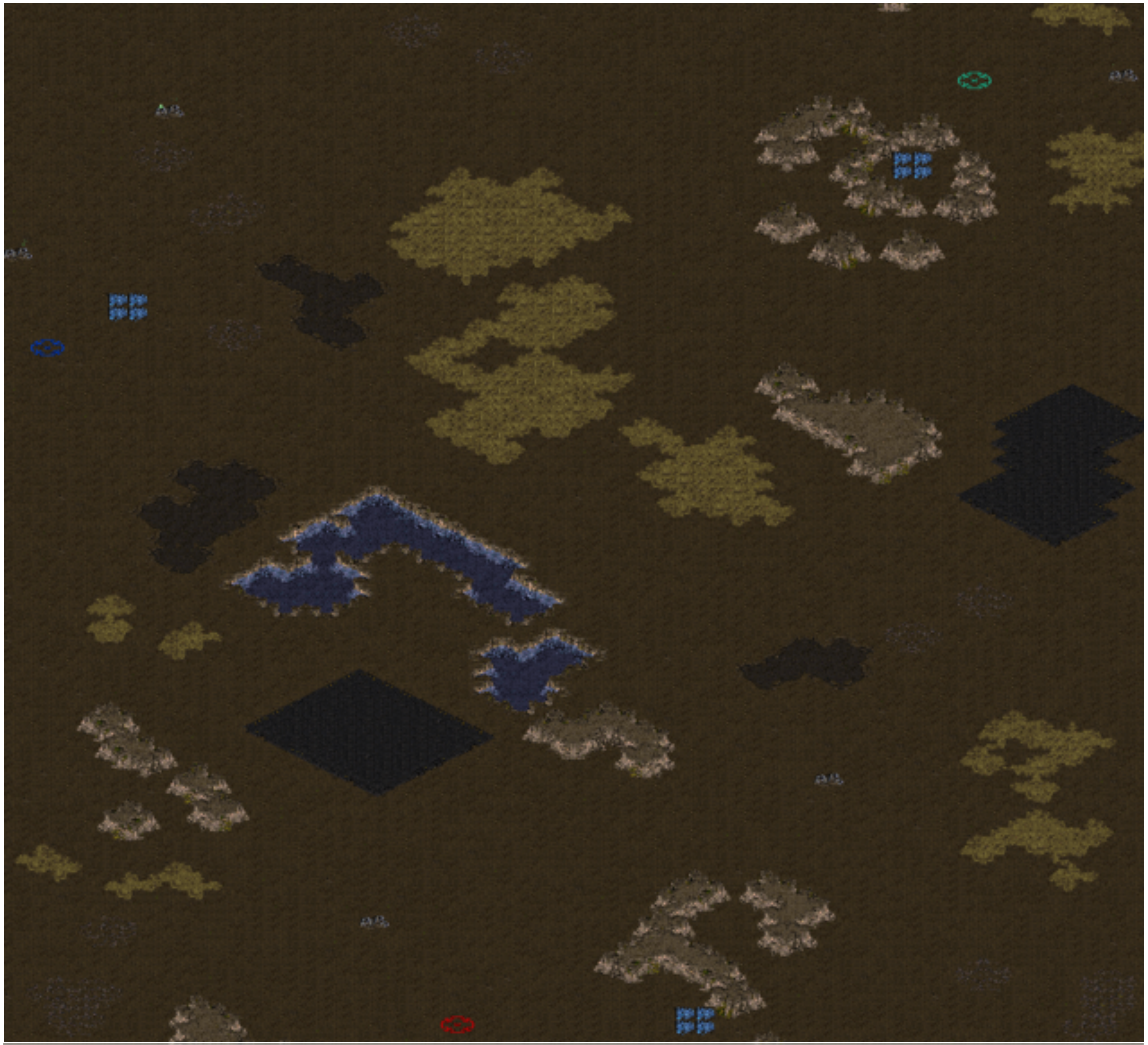
Evolved map

Resource fairness vs. choke points

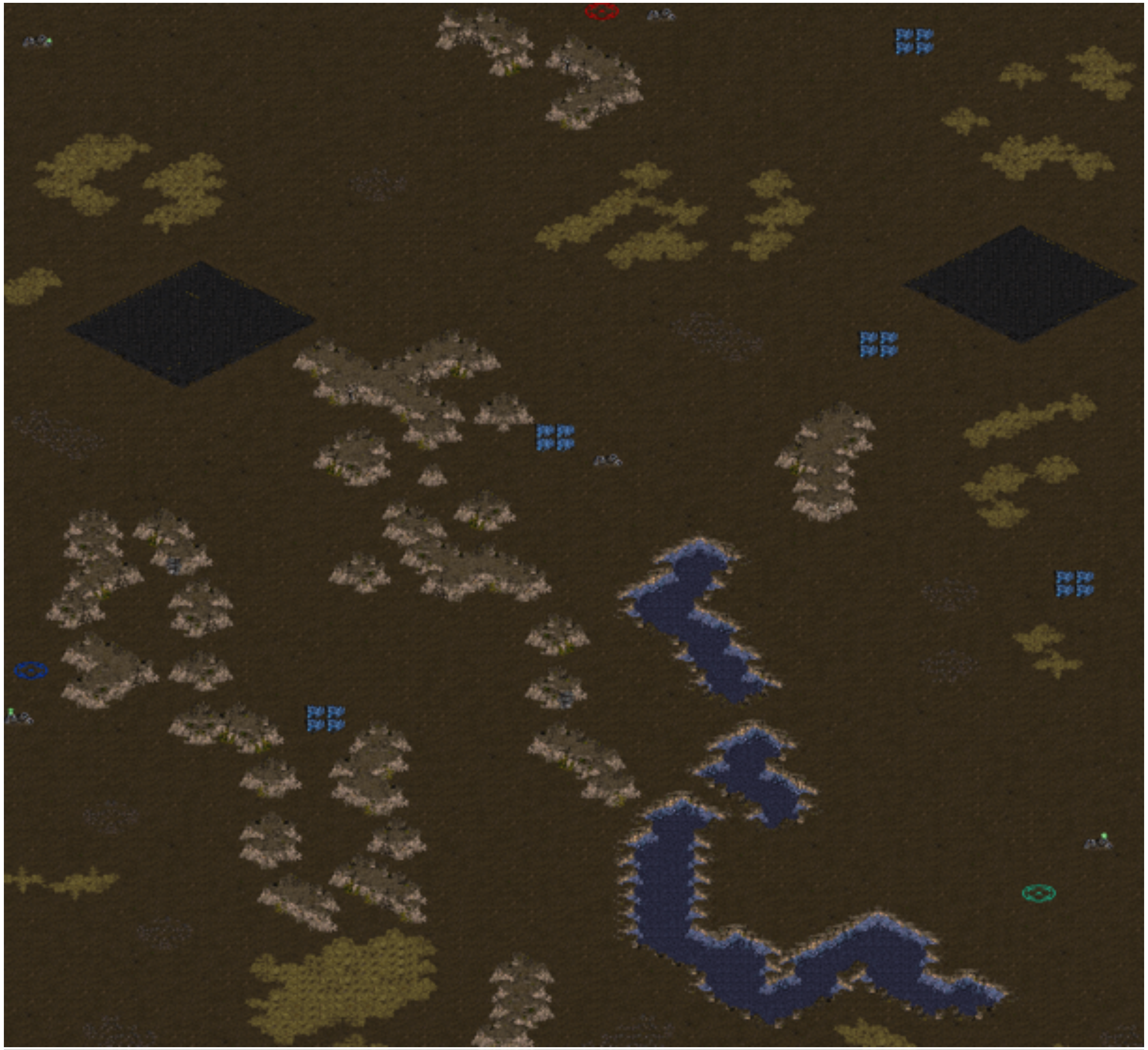


Another evolved map

Resource fairness vs. choke points



Three-player map



Another three-player map

Agent-based methods

- Use a number of "artificial agents" that construct the landscape by acting on it
- Agents of different types do different jobs
- Could be more controllable than diamond-square
- Could give rise to different types of landscapes

Lecture 6: Rules and mechanics

Julian Togelius

Salen and Zimmermann define games:

"A game is a system in which players engage in an artificial conflict, defined by rules, that results in a quantifiable outcome"

Can we create game rules automatically?

- If so, which types of rules?
- For which types of games?
- How would we represent them?
- How would we judge how good a set of rules is?
- And why would we do this?

Challenges

- How to represent game mechanics
  - Representation should be complete
  - Most games should make sense (?)
  - High locality (?)
  - Human-readable/editable (?)
- How to search the space
- How to evaluate the games

Automatic generation of recombination games

Cameron Browne

PhD Thesis, 2008
IEEE TCIAIG, 2010

“Combinatorial games”

- Finite: produce a well-defined outcome.
- Discrete: turn-based.
- Deterministic: chance plays no part.
- Perfect information: no hidden information.
- Two-player.

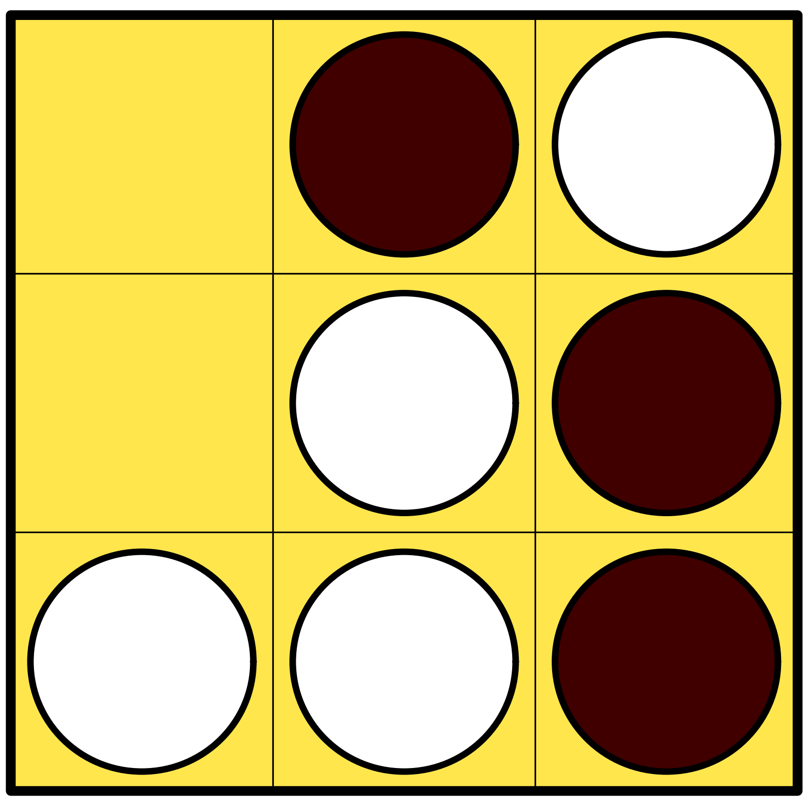
The Ludi Game Description Language

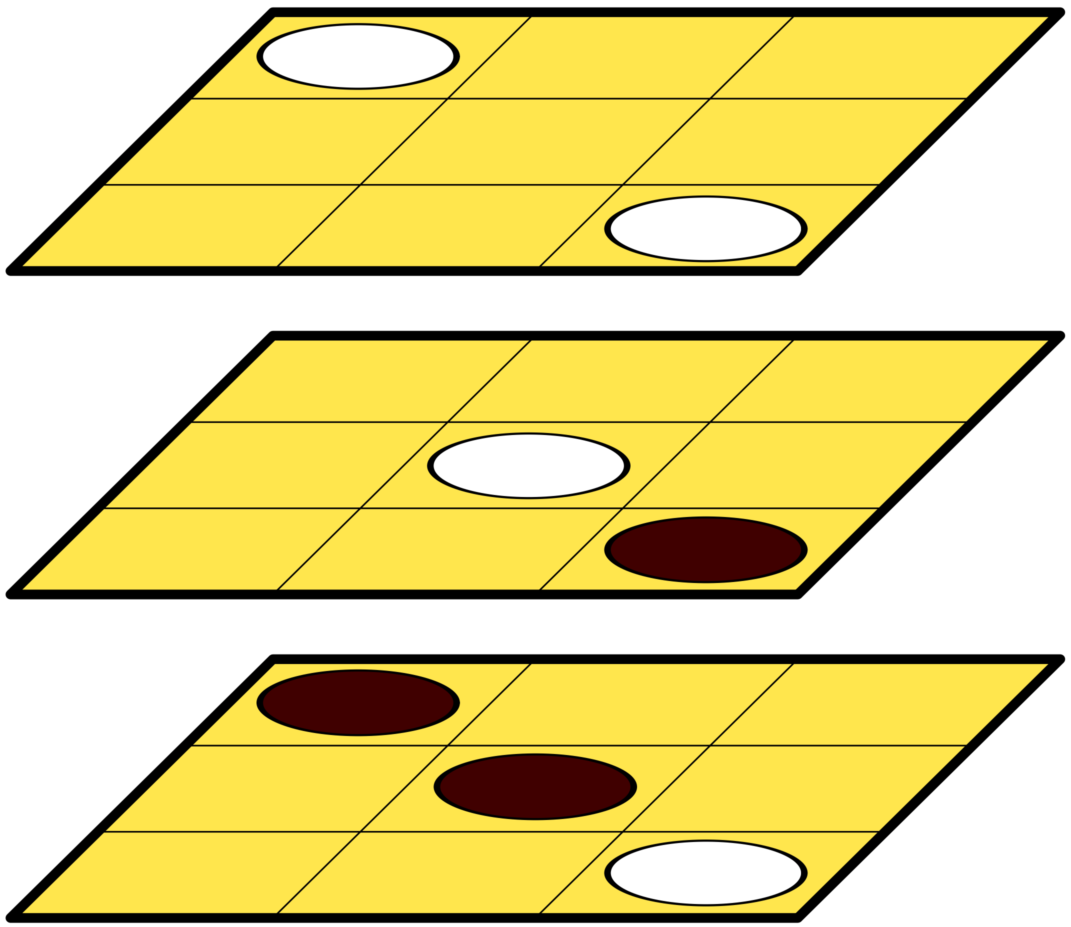
- In practice limited to board games
- *Ludeme:* Fundamental units of independently transferable game information ("game meme")
  - (tiling square)
  - (size 3 3)

Tic-Tac-Toe

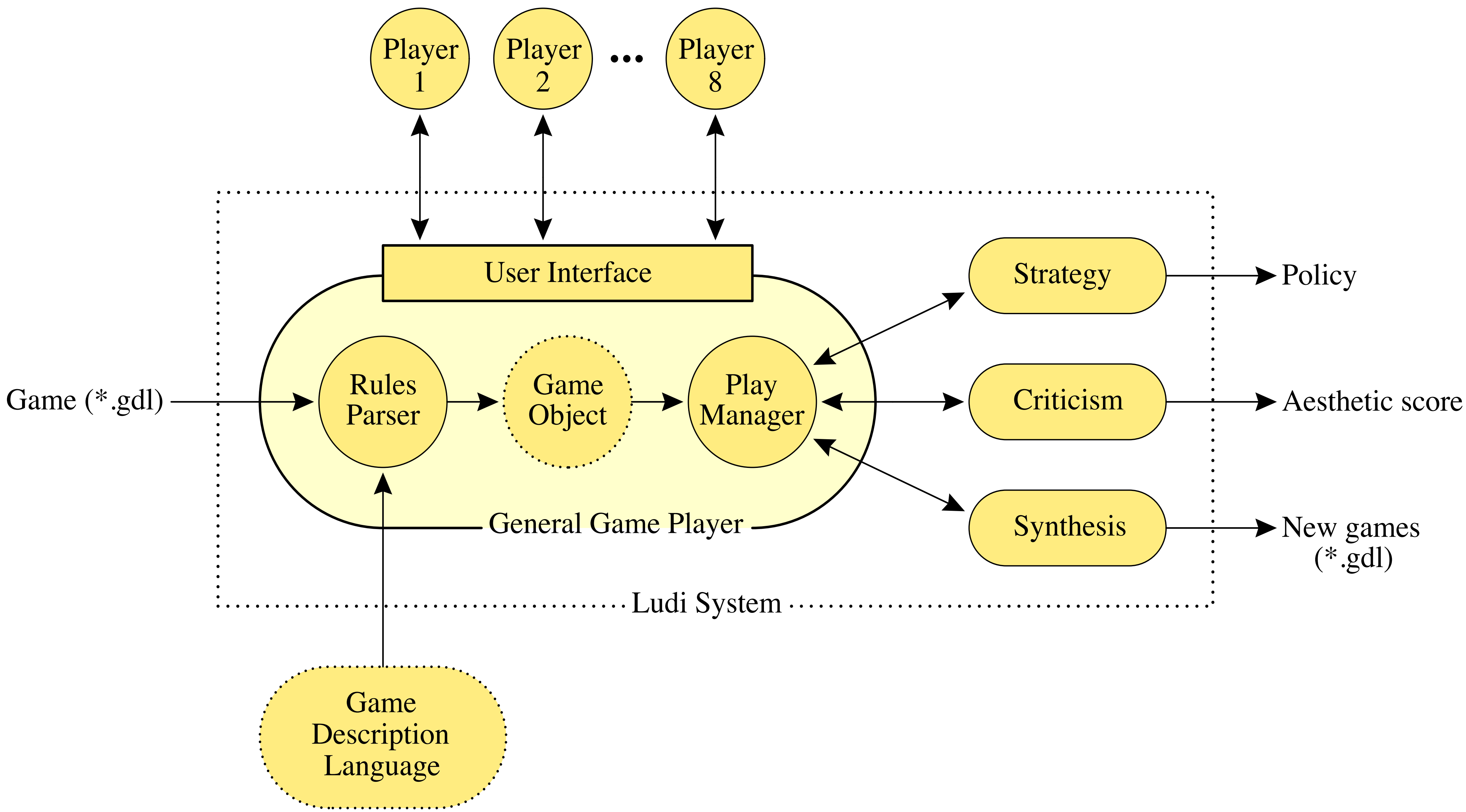
```
(game Tic-Tac-Toe
  (players White Black)
  (board
    (tiling square i-nbors)
    (size 3 3)
  )
  (end (All win (in-a-row 3)))
)
```

(size 3 3) vs (size 3 3 3)





The *Ludi* system

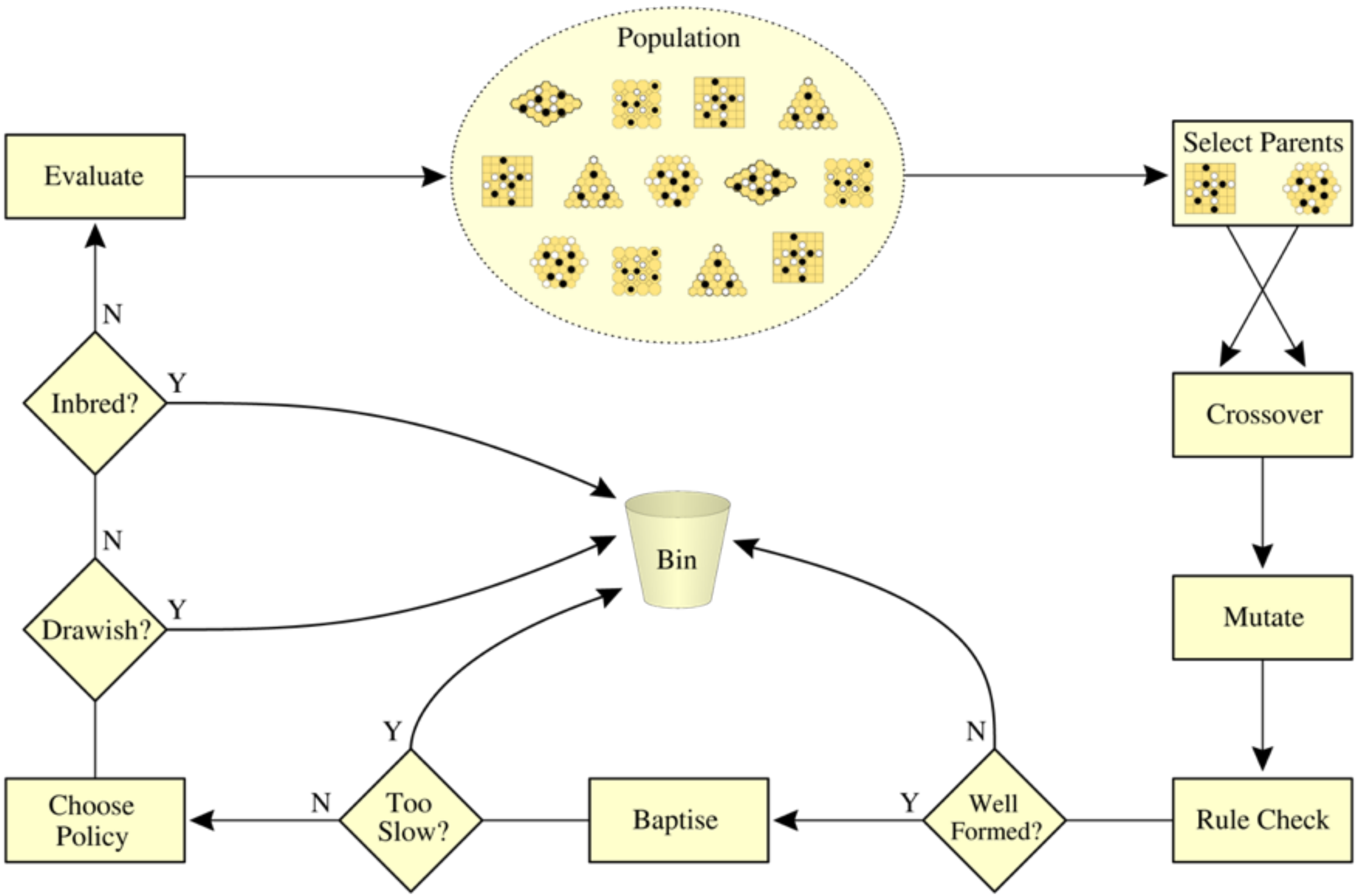


Evaluating a game

- Play the game (both player use same algorithm, with optimized board evaluation)
- Measure various *aesthetic criteria*: aspects of how the game is played, of the ruleset, and of the outcomes
- Combine the scores into a fitness value somehow

Aesthetic criteria

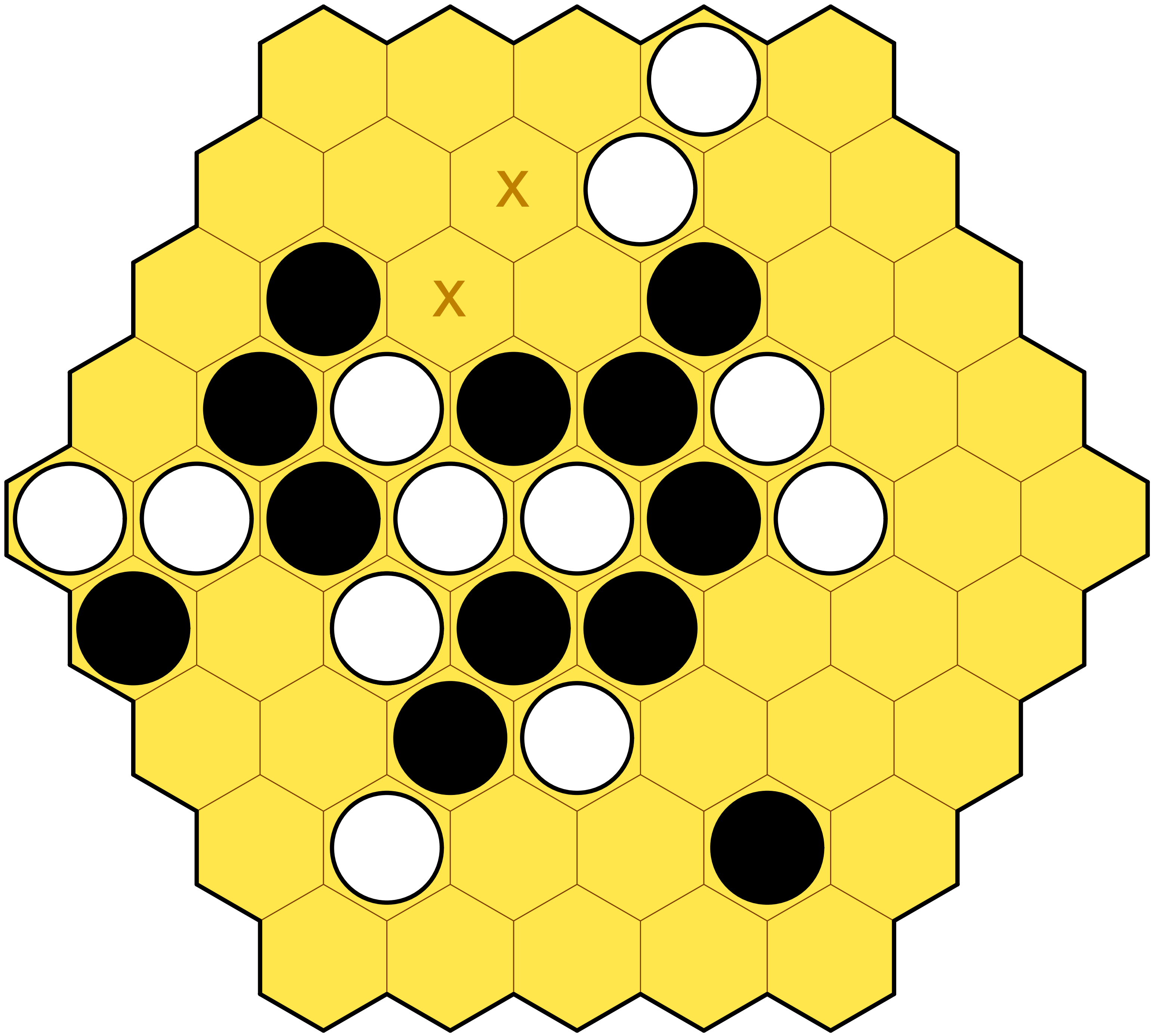
- 16 Intrinsic: based on rules and equipment
- 11 Viability: based on game outcomes
  - e.g. completion, duration
- 30 Quality: based on trends in play
  - e.g. drama, uncertainty


Population
Evaluate
Select Parents
Crossover
Mutate
Rule Check
Well Formed?
Baptise
Too Slow?
Choose Policy
Drawish?
Inbred?
Bin
Y
N

Yavalath

```
(game Yavalath
  (players White Black)
  (board (tiling hex) (shape hex) (size 5))
  (end
    (All win (in-a-row 4))
    (All lose (and (in-a-row 3) (not (in-a-row 4))))
  )
)
```

Yavalath



Combining human and computer creativity

Procedural Content Generation, Autumn 2010

Julian Togelius

Who creates a game's content?

- The designer(s)/developer(s)?
- A computer-implemented algorithm?
- The players?

PCG and authorship

- How can we combine a human designer's authorial control and expressive ability with PCG capabilities?
- Dimensions of control
- Ease of use
- Multi-level editing / two-way flow of control

the death of level designer

seriously ?