

# Procedural Content Generation

Lecture 1: Introduction

Autumn 2013

IT University of Copenhagen

Noor Shaker and Julian Togelius

# Who are we?

- Noor Shaker: course leader, main lecturer
- Julian Togelius: lectures
- Mark Nelson: lectures
- Yun-Gyung Cheong: lectures
- Anders Hartzen: labs

# This course

- September/October: Lectures on Thursdays 14.00, followed by labs at 16.00
- November/December: Supervised course project (Thursdays at 14)
- Core literature: the PCG book!
- Additional literature: selected research papers
- Labs: implementation of key PCG algorithms

# Examination

- Report on course project (quality of both project and report taken into account)
- To be handed in 17 December
- Oral exam 15/16 January, with questions on literature, labs and project



# The project

- Do something cool with PCG!
- Preferably within a game, or at least with a clear gameplay angle
- Preferably novel
- At least partly your own implementation
- Groups of 1 or 2 people
- Write a good report! (6 or 8 pages)

The course web page:  
[http://blog.itu.dk/](http://blog.itu.dk/MPGG-E2013/)  
MPGG-E2013/

# Course structure

Date	Lecture	Readings	Lab
Aug 29	Introduction (Julian)		None
Sep 5	The search-based approach (Julian)		
Sep 12	Agents and cellular automata (Noor)		
Sep 19	Fractals and noise (Noor)		
Sep 26	Grammars and L-systems (Julian)		
Oct 3	Rules and mechanics (Mark)		
Oct 10	Planning (Yun)		
Oct 17	Holiday		
Oct 24	ASP (Mark)		
Oct 31	The experience-driven perspective (Noor)		
Nov 7	Mixed-initiative and evaluation (Antonios)	-	-
Nov 14	Group project plan presentations (Noor & Julian )	-	-
Nov 21	Group project supervision.	-	-
Nov 28	Group project supervision.	-	-
Dec 5	Group project supervision.	-	-
Dec 12	Group project supervision.	-	-

# 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

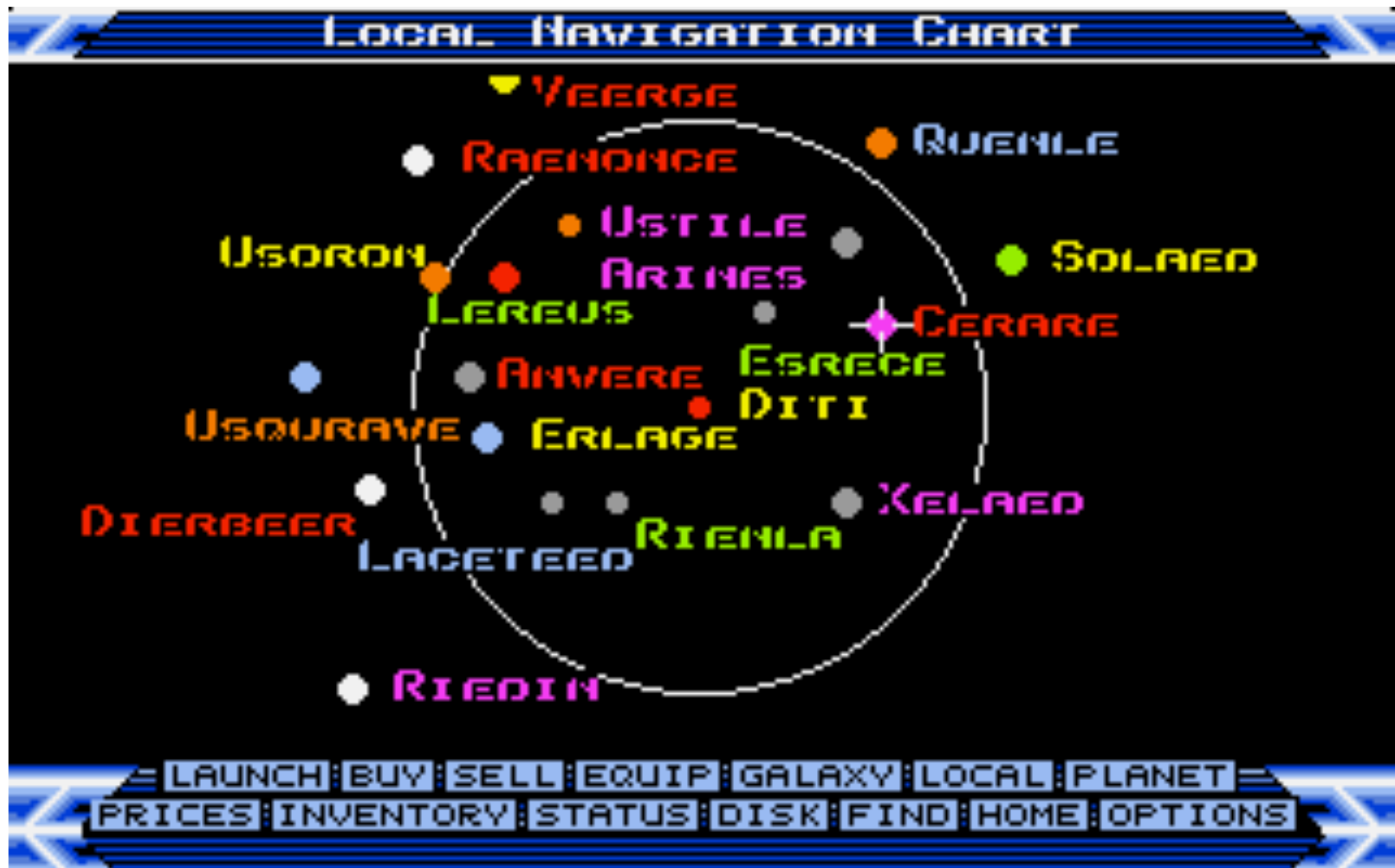
# Game content, e.g.

- 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...

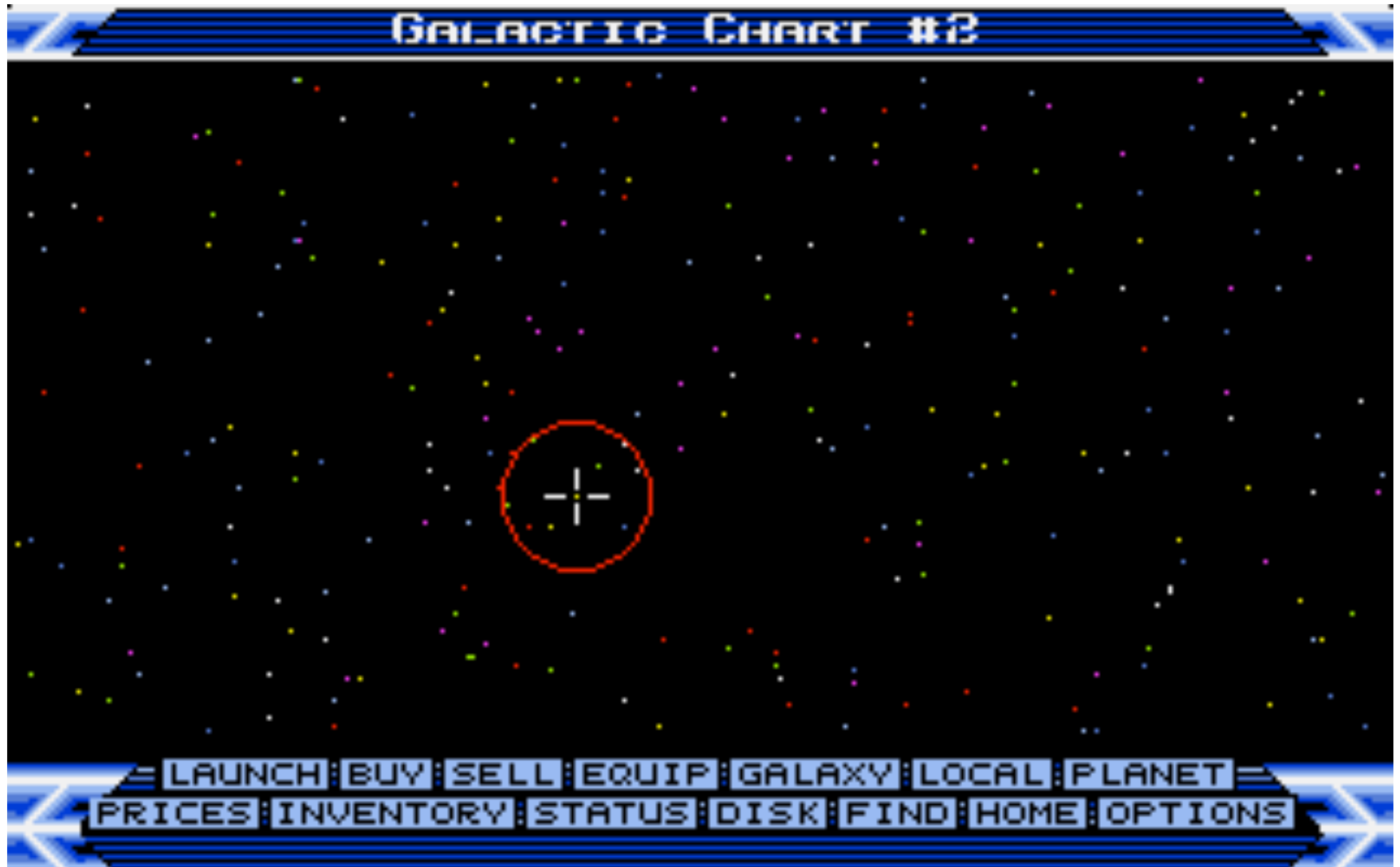
# Elite



# Elite



# Elite





# Elite



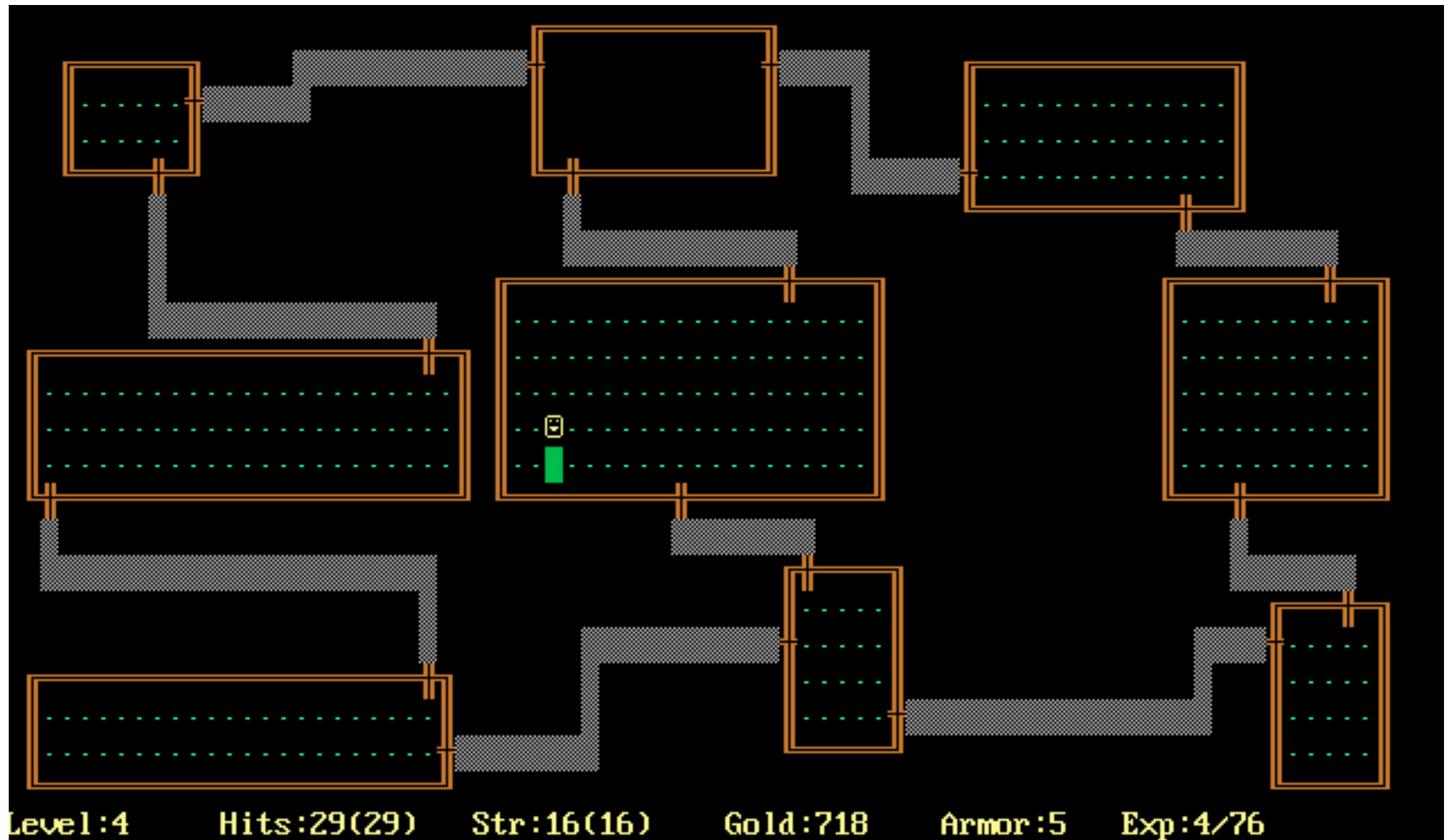
# Elite



# Elite

Fits in memory on a Commodore 64!

# Rogue





# Diablo III



# Dwarf Fortress





# Spelunky





# Far Cry 2





# SpeedTree





# Civilization IV



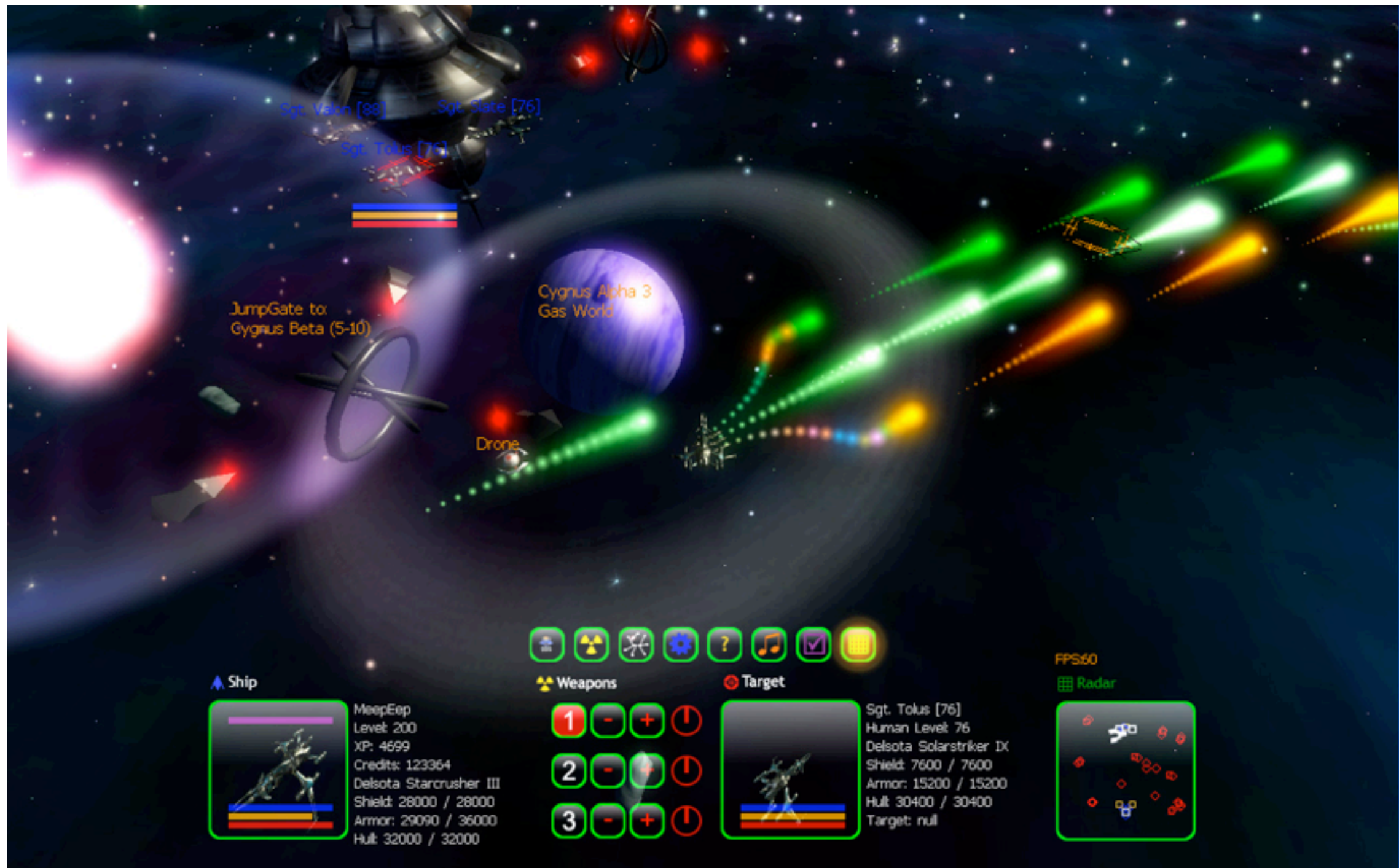


# Borderlands

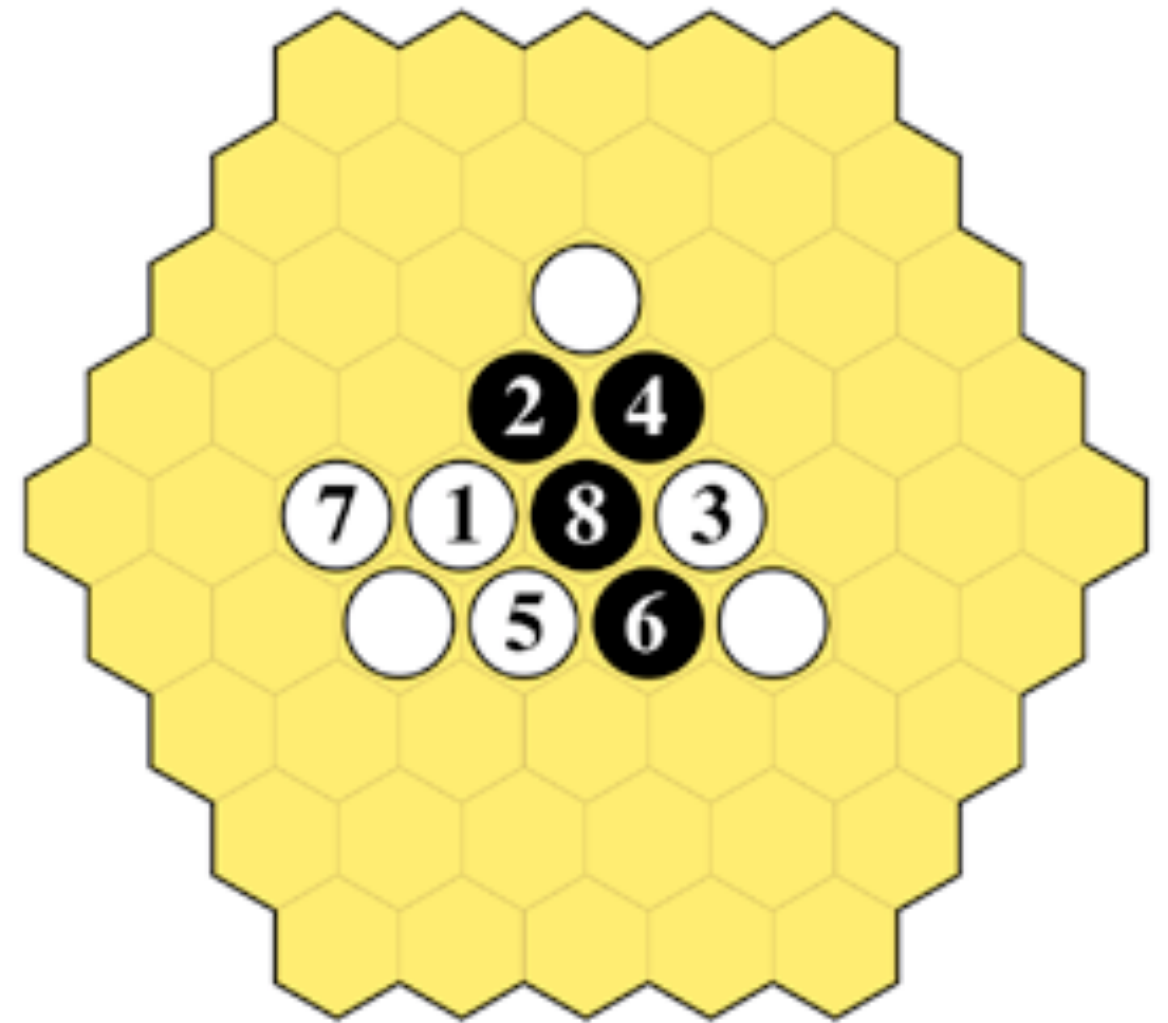
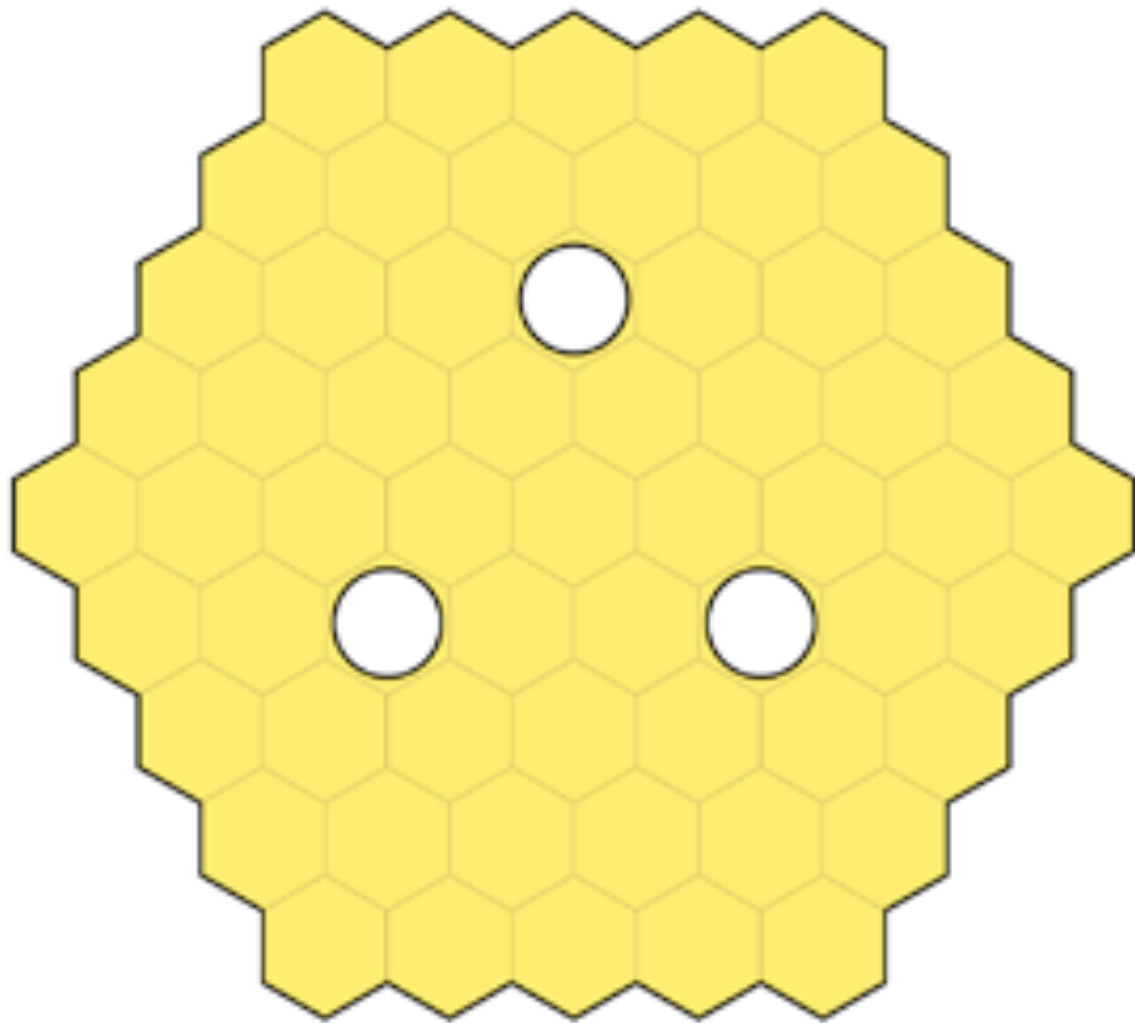




# Galactic Arms Race



# Ludi / Yavalath



# 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?

# Example projects from the previous three years

- Alistar, van Leeuwen: Infinite TD
- Dart, De Rossi: SpeedRock
- Kastbjerg, Schedl: Direct level adaptation in Super Mario Bros
- Hartzen, Justinussen: Compositional PCG
- Kerssemakers, Tuxen: PPLGG
- Borg Cardona: OpenTrumps
- All formed part of published papers!



# Infinite Tower Defense

# Infinite tower defense

- Creep adaptation: creeps are evolutionary optimised between each wave, so as to optimally fight the existing towers given limited resources
- Tower adaptation: tower selection part of an interactive evolutionary algorithm
- Path adaptation: difficulty adjusted through straightness of path

# Infinite tower defense



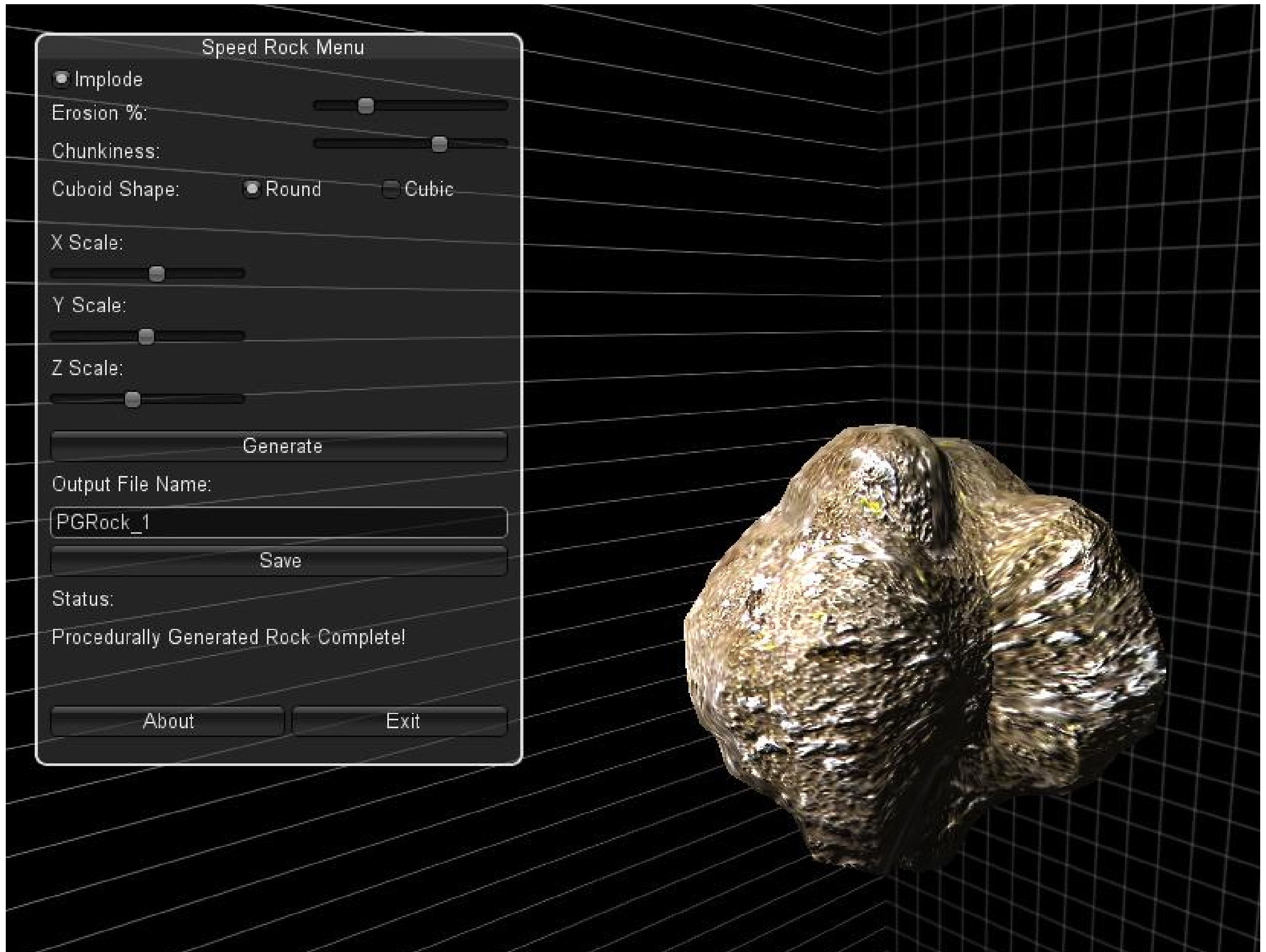
Avery, Togelius, Alistar, van Leeuwen: Computational Intelligence and Tower Defence Games. CEC 2011

# SpeedRock: Procedural rocks through grammars and evolution

Isaac M. Dart, Gabriele De Rossi and Julian Togelius  
IT University of Copenhagen, Denmark

# What?

- A tool for generating 3D models of rocks
- Offline
- Exports .OBJ files
- Somewhat controllable



# How?

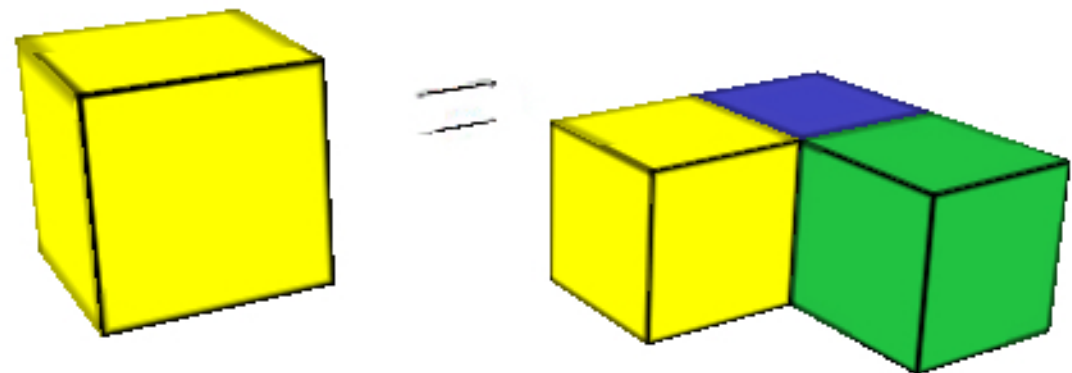
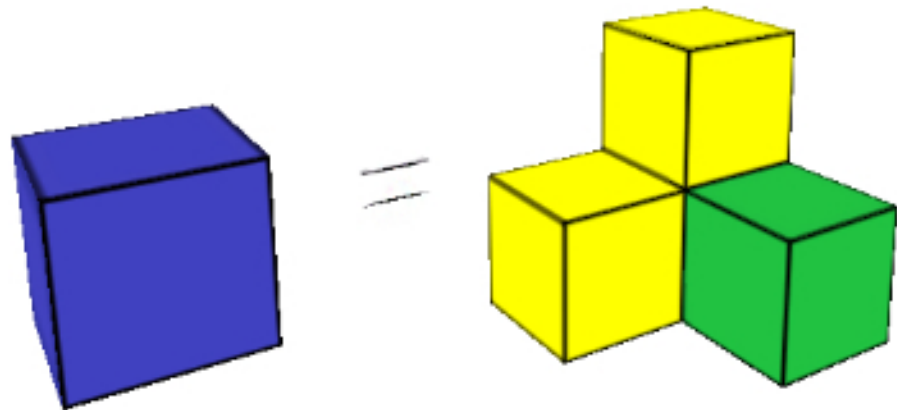
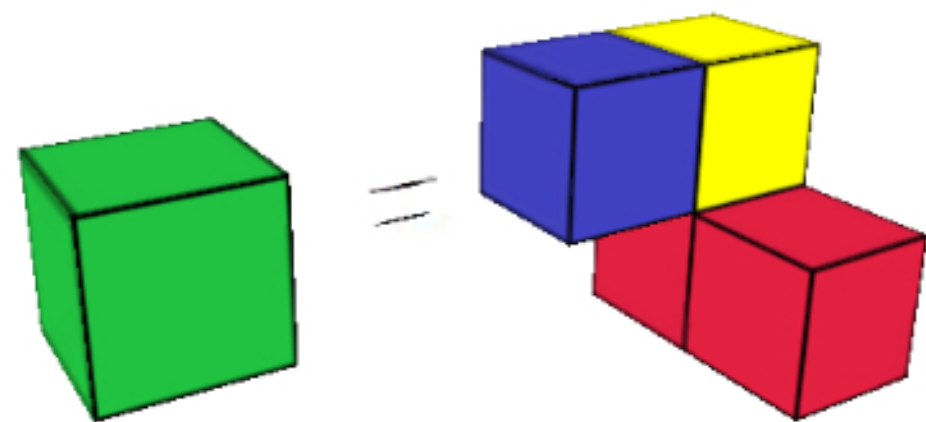
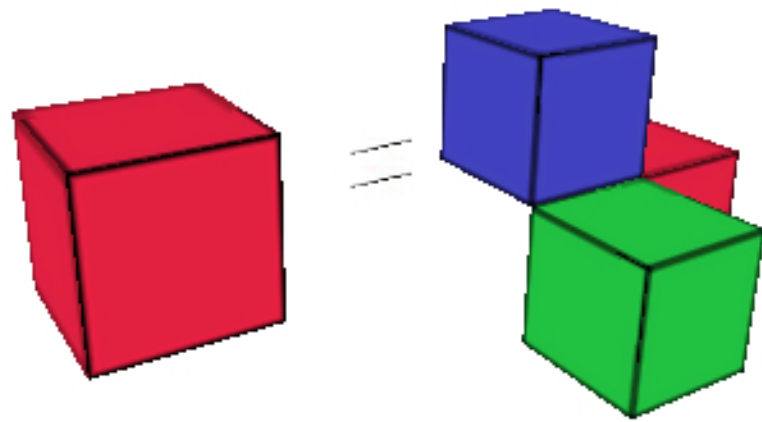
- 3D L-system subdivision
- Implosion
- Erosion
- Vacuum sealing
- Optimization by evolutionary computation

# 3D L-systems

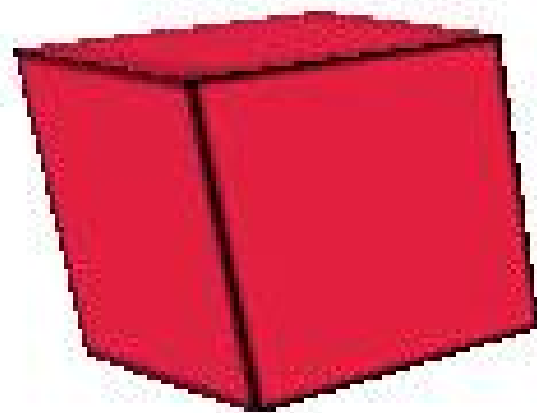
- Cubes: each has one of four colours
- Axiom: an initial set of cubes that occupy the entire space of the future rock
- Rules: how to subdivide a cube into eight smaller cubes



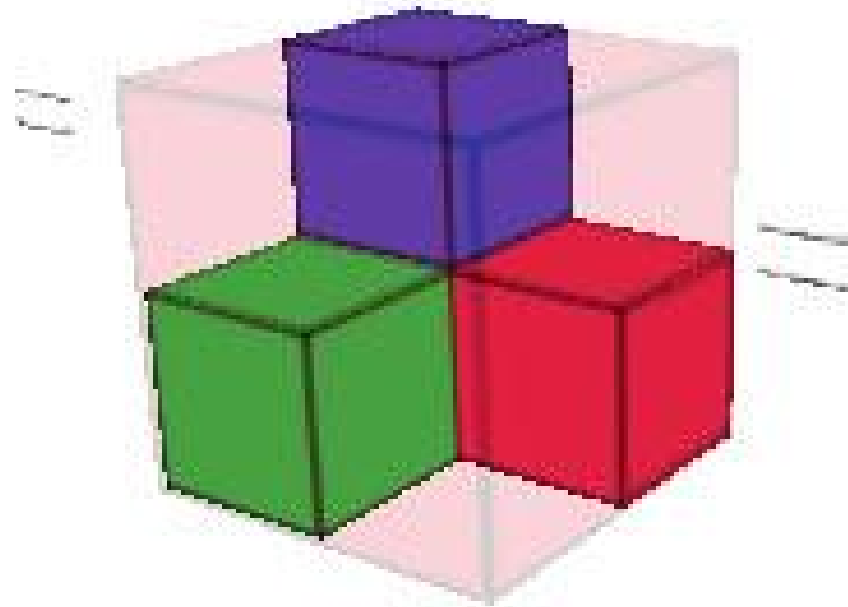
# 3D L-systems



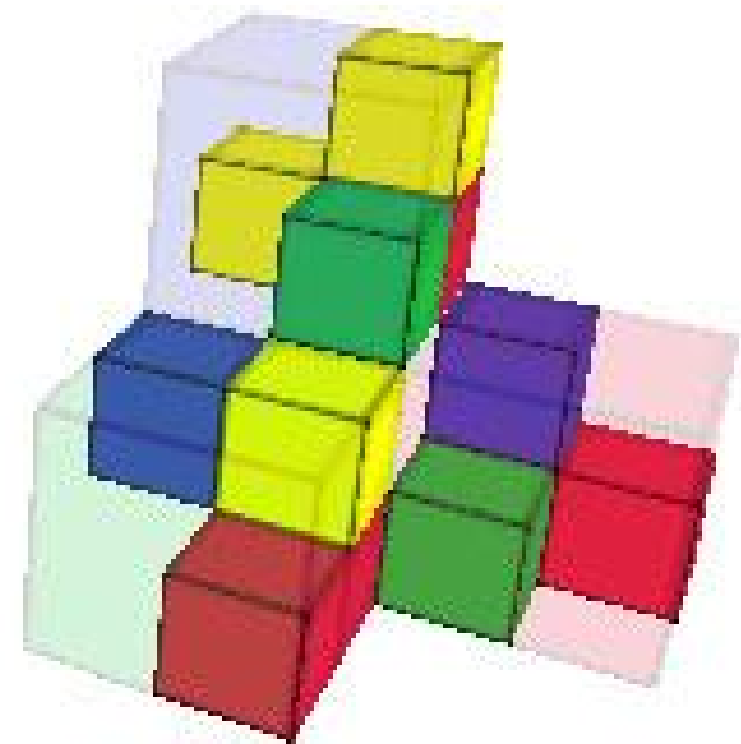
# 3D L-systems



Expansion 0

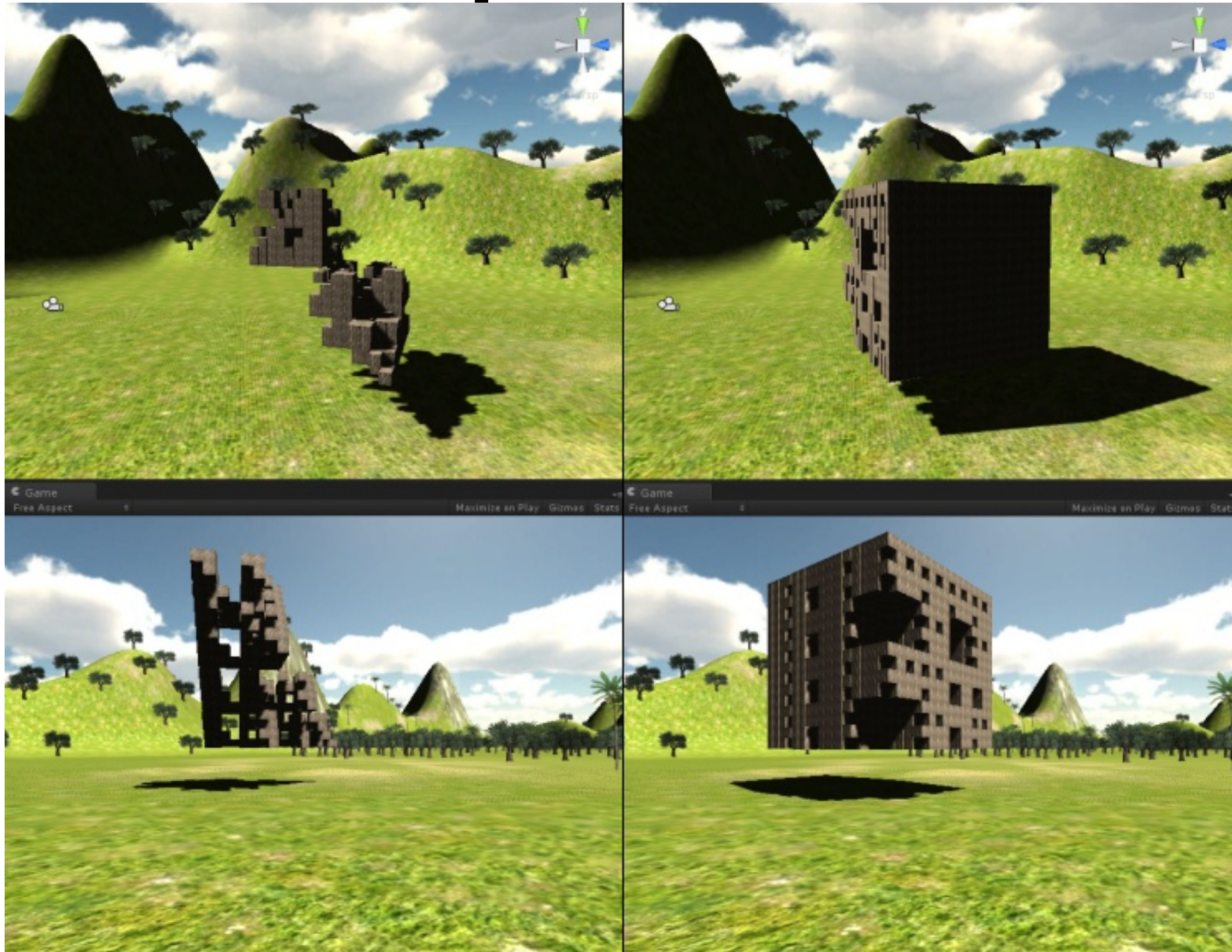


Expansion 1



Expansion 2

# Random ruleset expansion

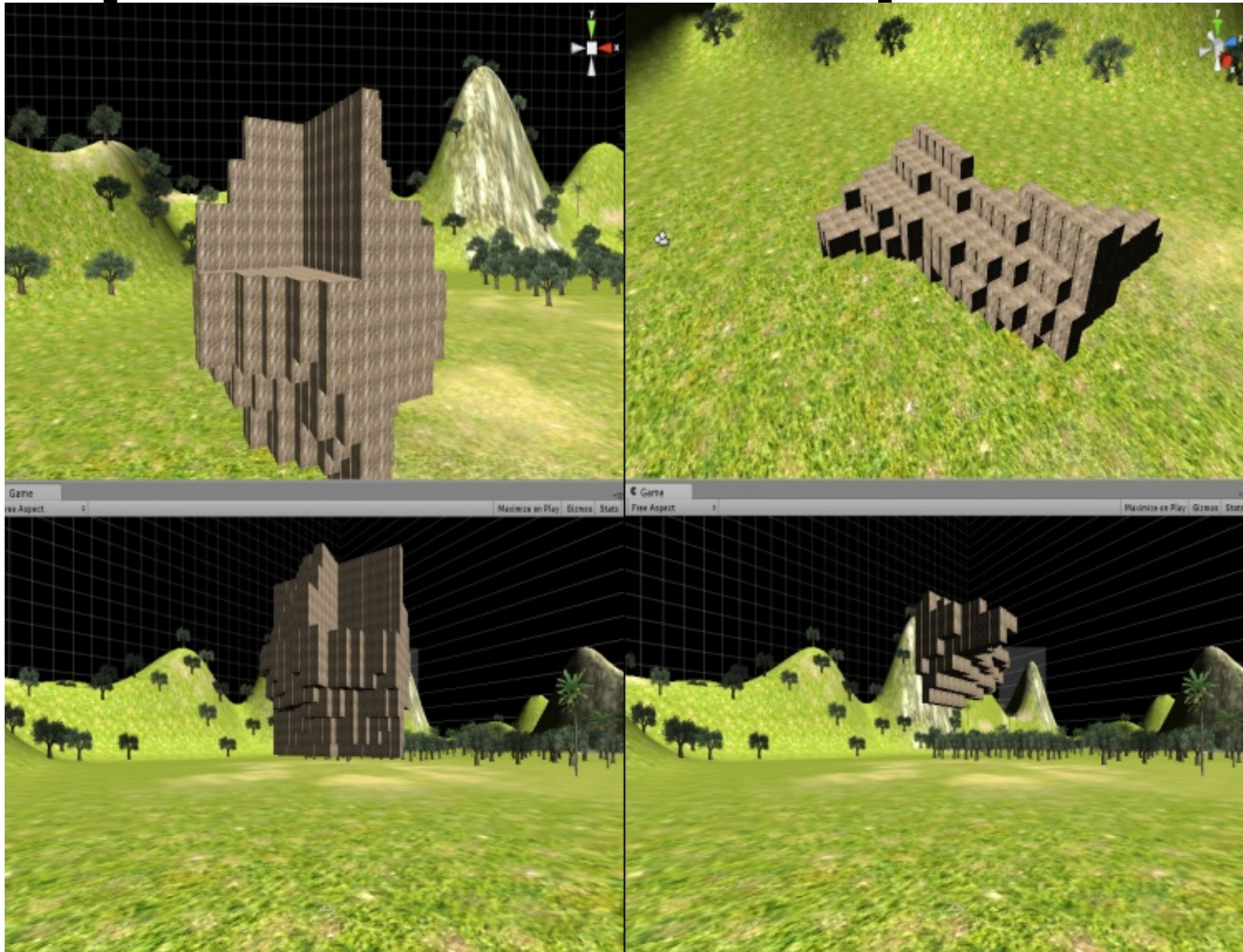


# Rock implosion

- After rewriting/expansion: implosion
- All cubes are moved toward center  $(x, y, z)$  if free space exists
- Repeat as long as possible



# Random ruleset expansion + implosion



# Erosion

- One-step “cellular automaton”
- Calculate number of neighbours of cube (max 26)
- Probability of deleting cube: 45% if 4 neighbours, 65% if 3 etc.

# Skinning by vacuum sealing

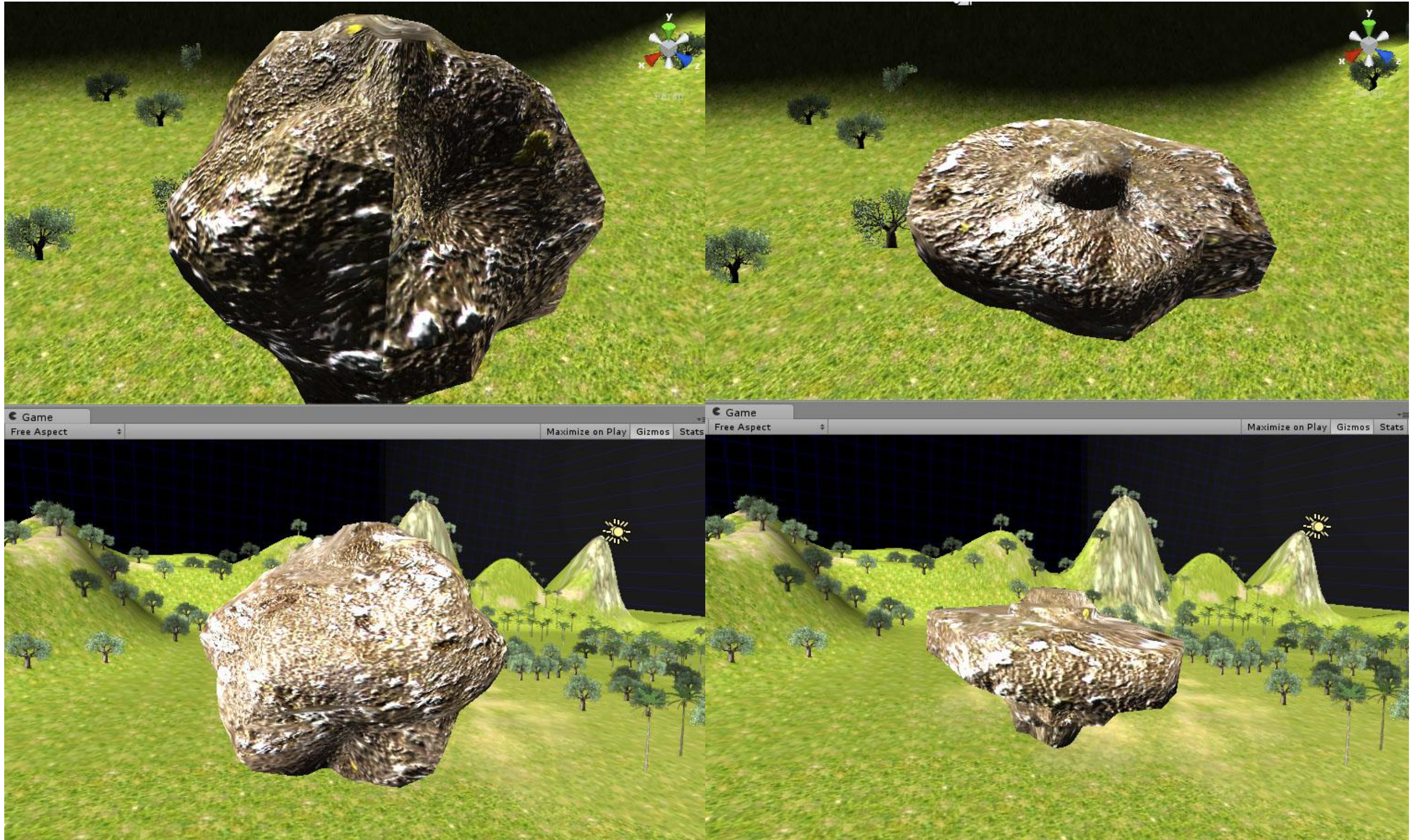
- A sphere mesh is created around the rock
- For each mesh vertex, cast a ray to the center
- Move the sphere to where it intersects with a cube

# Finding rules

- Evolutionary algorithm
- Fitness function: correspondence between shape and specified dimensions



# Examples





# Direct level generation in Infinite Mario Bros



Emil Kastbjerg and David Schedl

# Offline version

- The player plays a simple hand-crafted level
- All of the player's actions are recorded
- The recorded actions are used to create the next level
- All level edits are local: at the place in the level where the event took place

# Generation rules

- Jump button pressed: block is created above
- Jump button released: ground raised/  
platform created
- Speed/fire pressed: enemy created



# Online version

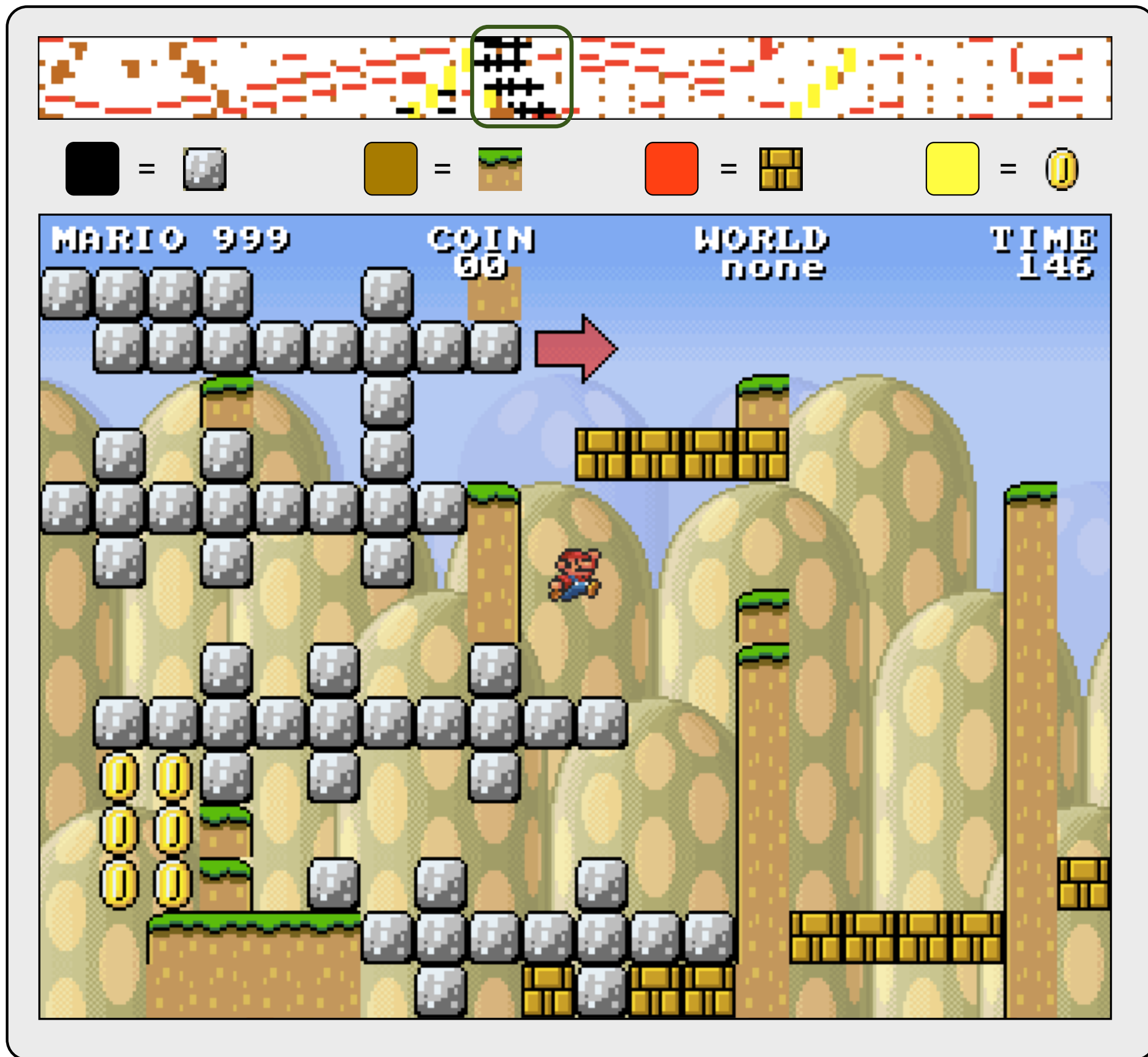
- All edit actions are *instantaneous* (and still local)
- Extra rules to balance the level process:  
enemies created when coins collected,  
coins created when enemies killed

# Compositional PCG

- Anders Hartzen and Tróndur Justinussen
- Combining evolution with answer set programming
- Evolving parameters for ASP programs that generate mazes

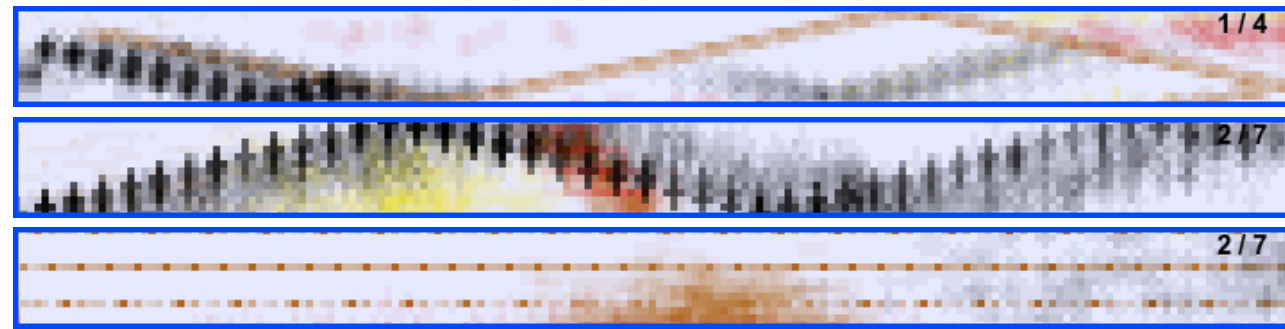
# A procedural procedural level generator generator

- Manuel Kerssemakers and Jeppe Tuxen
- Interactive evolution of agent-based Mario level generators

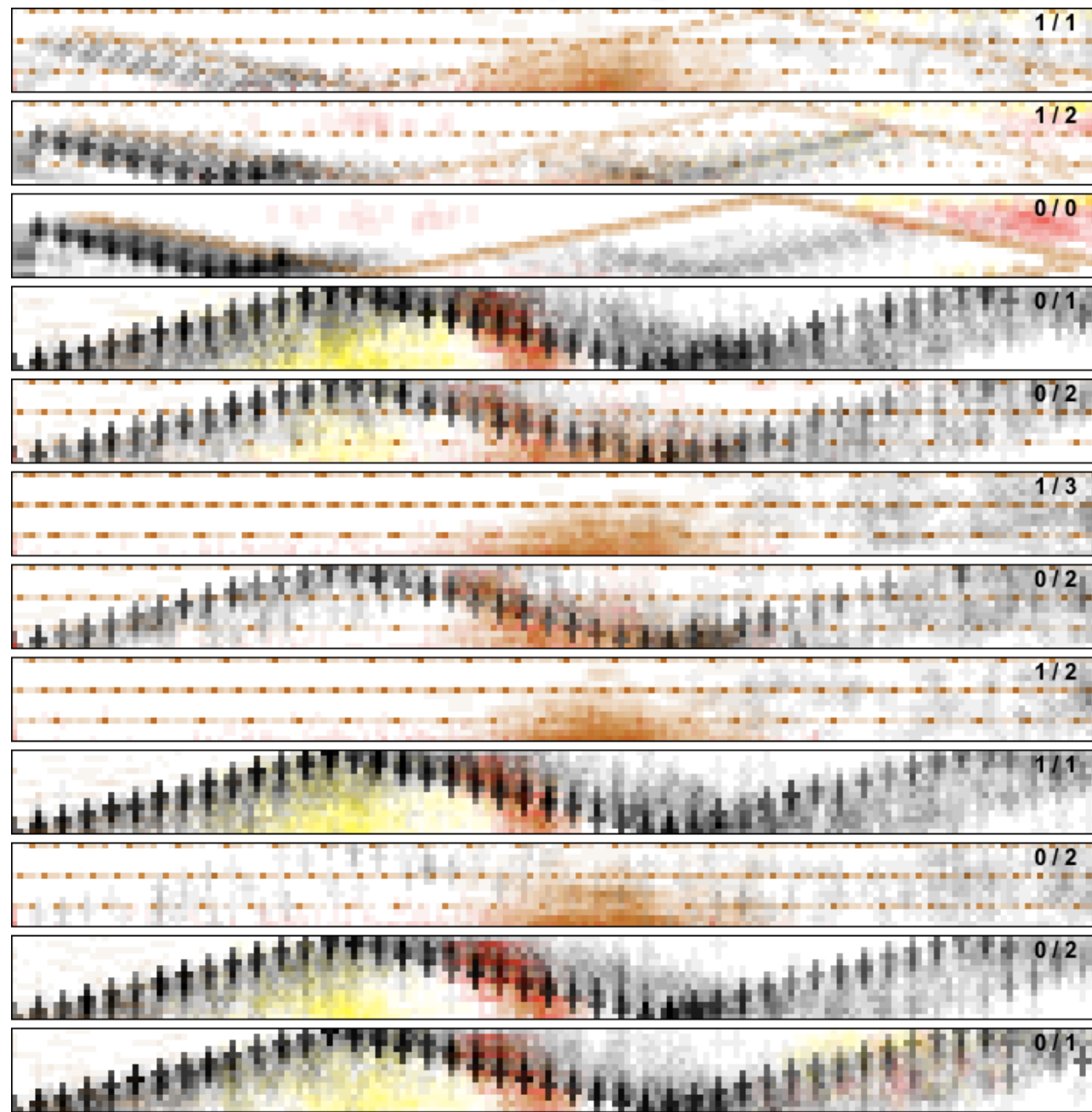




## Selected Parents



## Offspring



# A taxonomy of PCG

- Online/offline
- Necessary/optional
- Random seed/parameter vector
- Generic/adaptive
- Stochastic/deterministic
- constructive/generate-and-test
- Algorithmic/mixed-authorship

# 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

# Generic/adaptive

- Generic: for all types of players
- Adaptive: the content changes according to player behaviour

# Algorithmic/mixed authorship

- Algorithmic: designer tweaks the algorithm parameters
- mixed-authorship: richer designer input