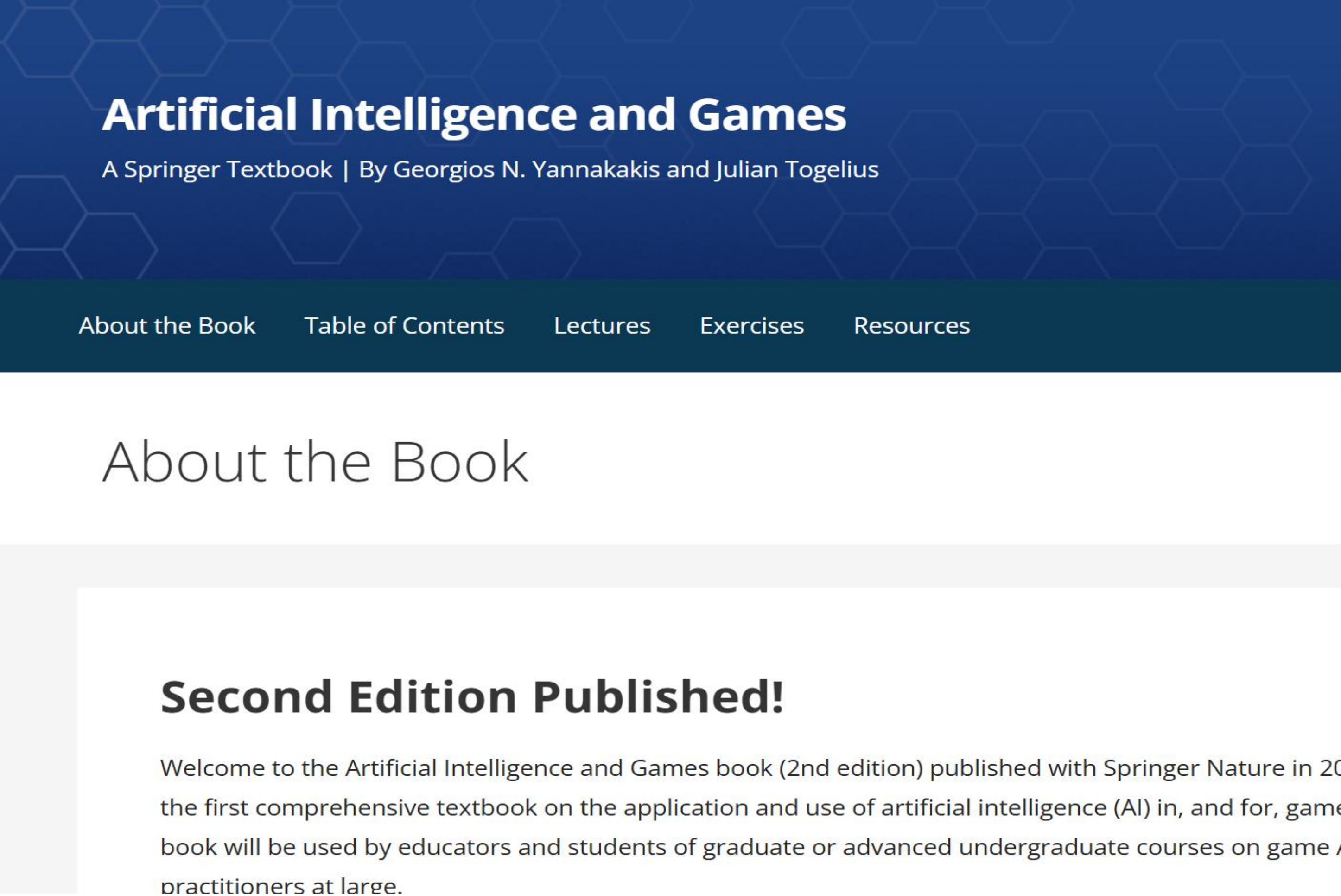
Artificial Intelligence and Games

Generate



Artificial Intelligence and Games

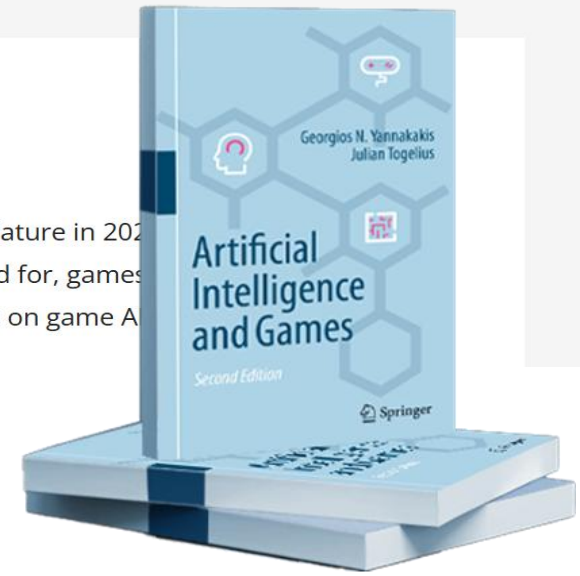
A Springer Textbook | By Georgios N. Yannakakis and Julian Togelius

About the Book · Table of Contents · Lectures · Exercises · Resources

About the Book

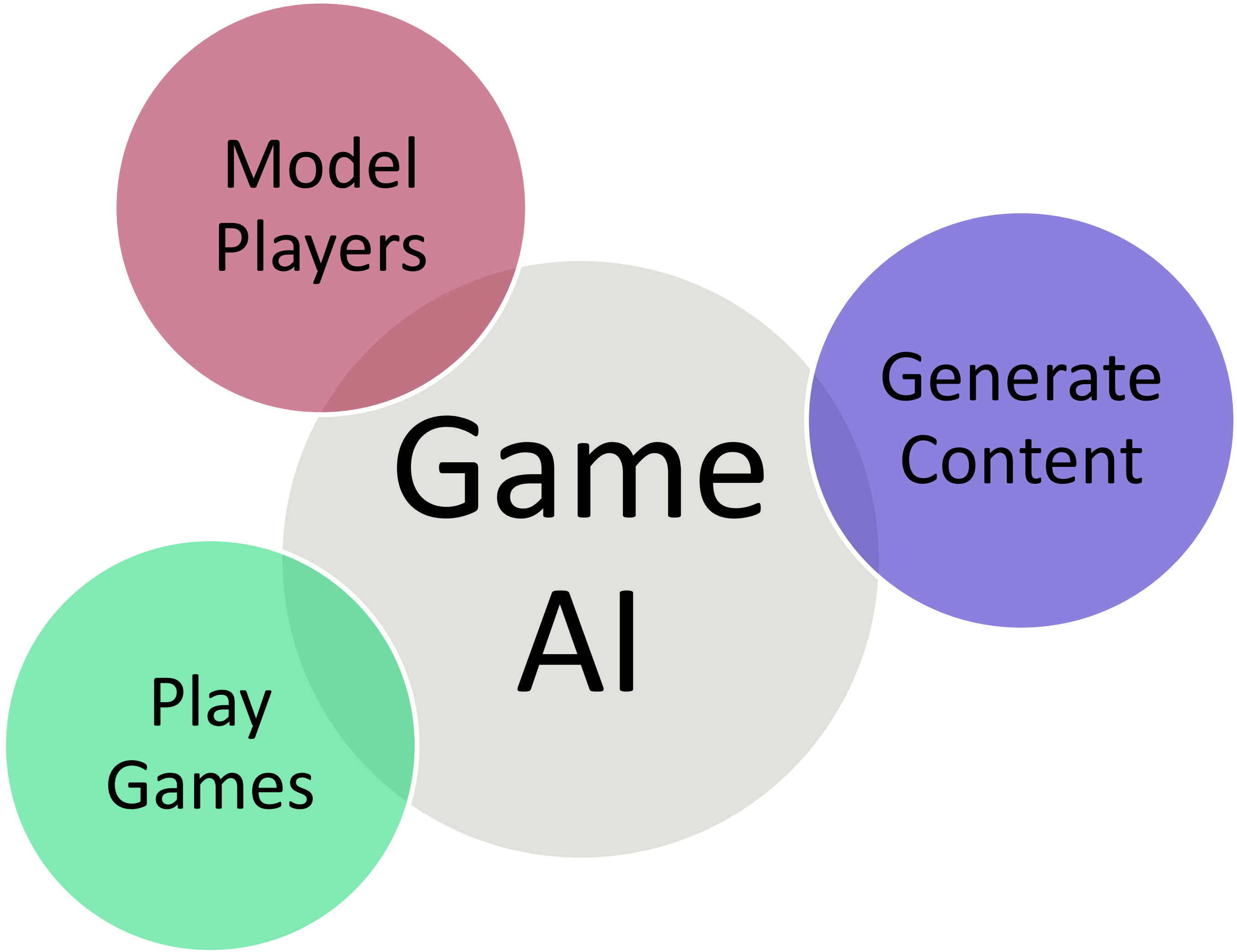
Second Edition Published!

Welcome to the Artificial Intelligence and Games book (2nd edition) published with Springer Nature in 202
the first comprehensive textbook on the application and use of artificial intelligence (AI) in, and for, games
book will be used by educators and students of graduate or advanced undergraduate courses on game A
practitioners at large.

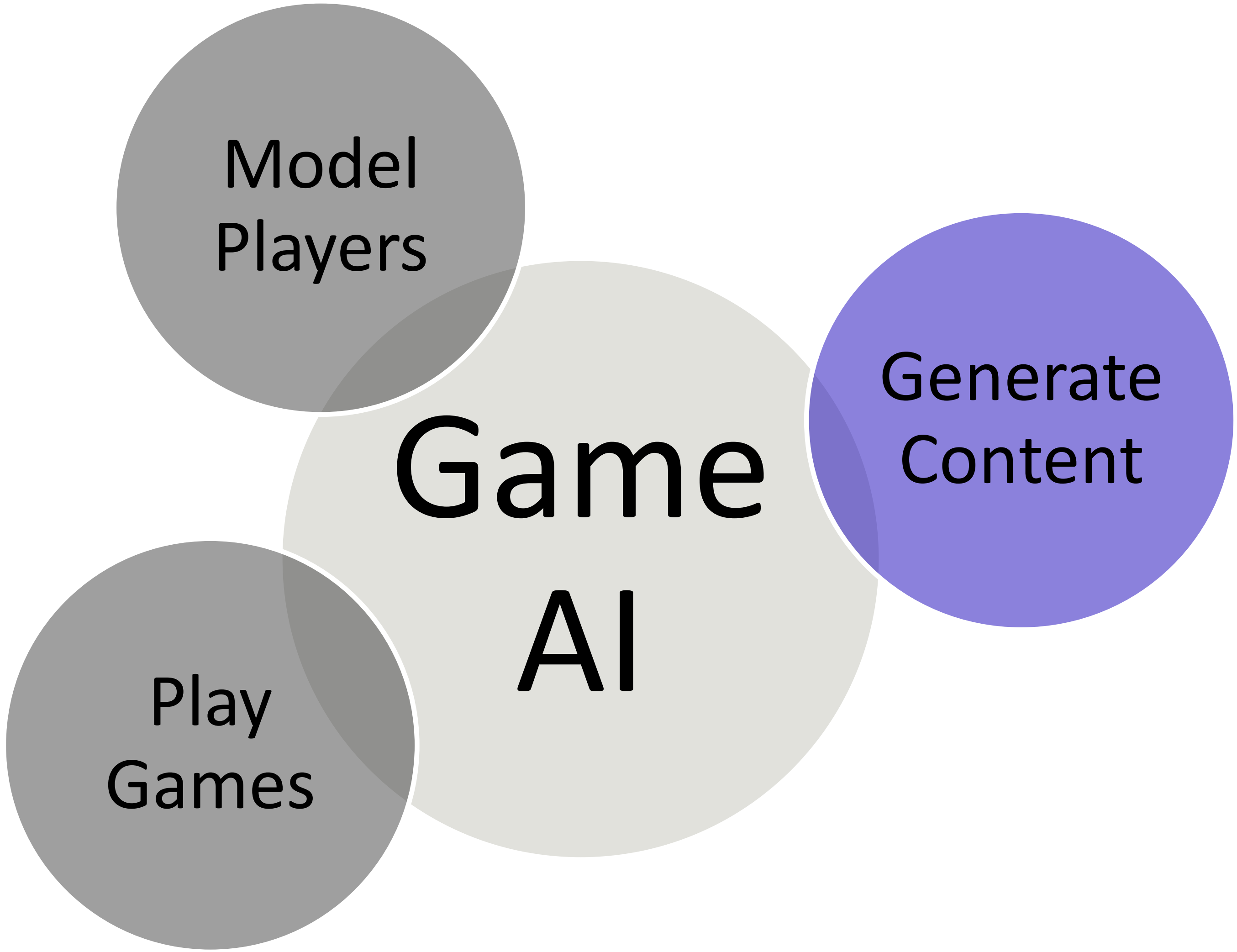


Readings: Part III

gameaibook.org



G. N. Yannakakis and J. Togelius, “**Artificial Intelligence and Games**,” Springer Nature, 2025.



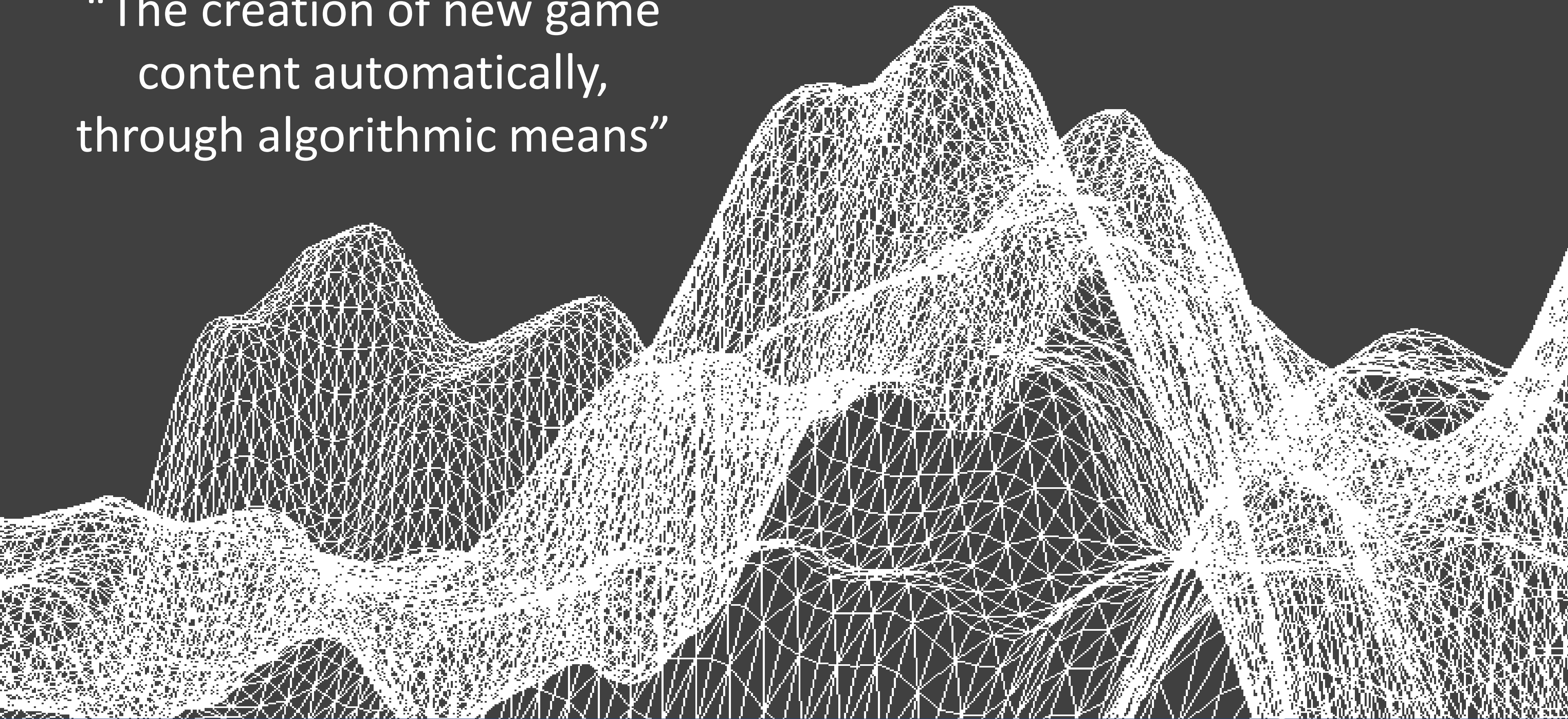
G. N. Yannakakis and J. Togelius, “**Artificial Intelligence and Games**,” Springer Nature, 2025.

Overview

- Procedural Content Generation (PCG)
  - What is it?
  - Why we need it?
- Constructive Approaches
- Search-Based PCG
- Machine Learning PCG
- PCG though QD
- PCG via RL
- PCG via LLM
- Mixed-Initiative PCG
- Experience-Driven PCG
  - Experience-Driven PCG via RL

Chapter 7: Procedural Content Generation

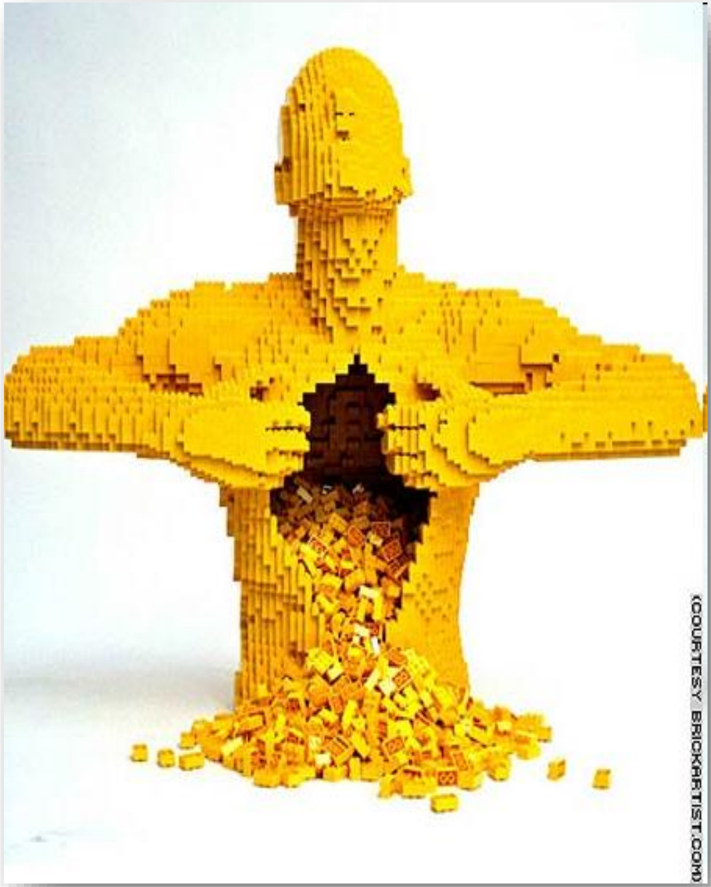
“The creation of new game content automatically, through algorithmic means”



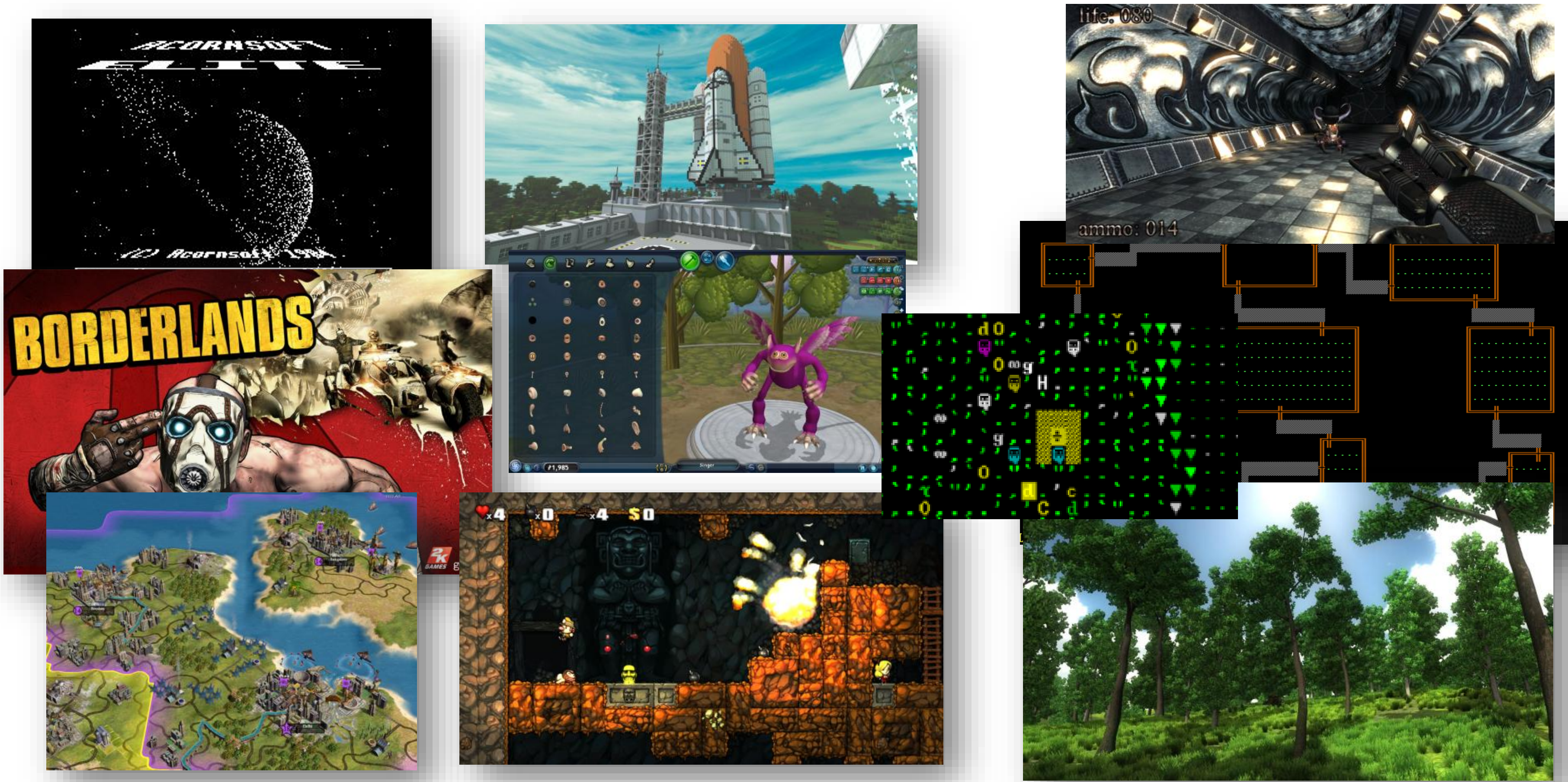
What is **Procedural Content Generation**?

What is Game Content?

- Content can be:
  - NPC behavior (aspects)
  - Quest/story/narrative
  - Camera profiles
  - Audiovisual settings
  - Levels/maps/tracks
  - Items
  - Game mechanics
  - Reward schedules
  - …
  - **Everything** together?
- Content **is** the game context
- Content has differing quality (ability to player experience elicitation)



PCG in Industry

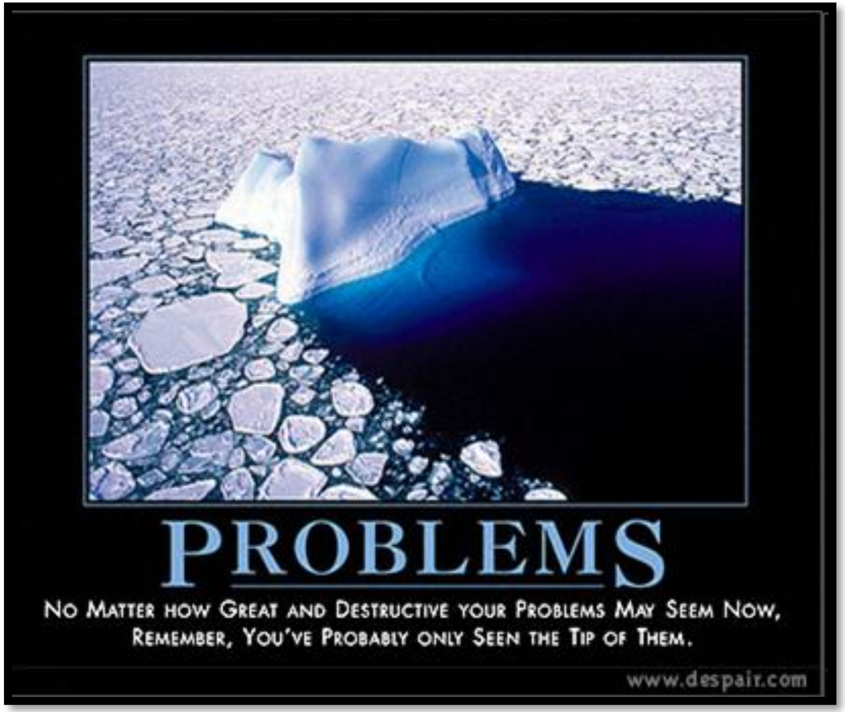


PCG in Academia

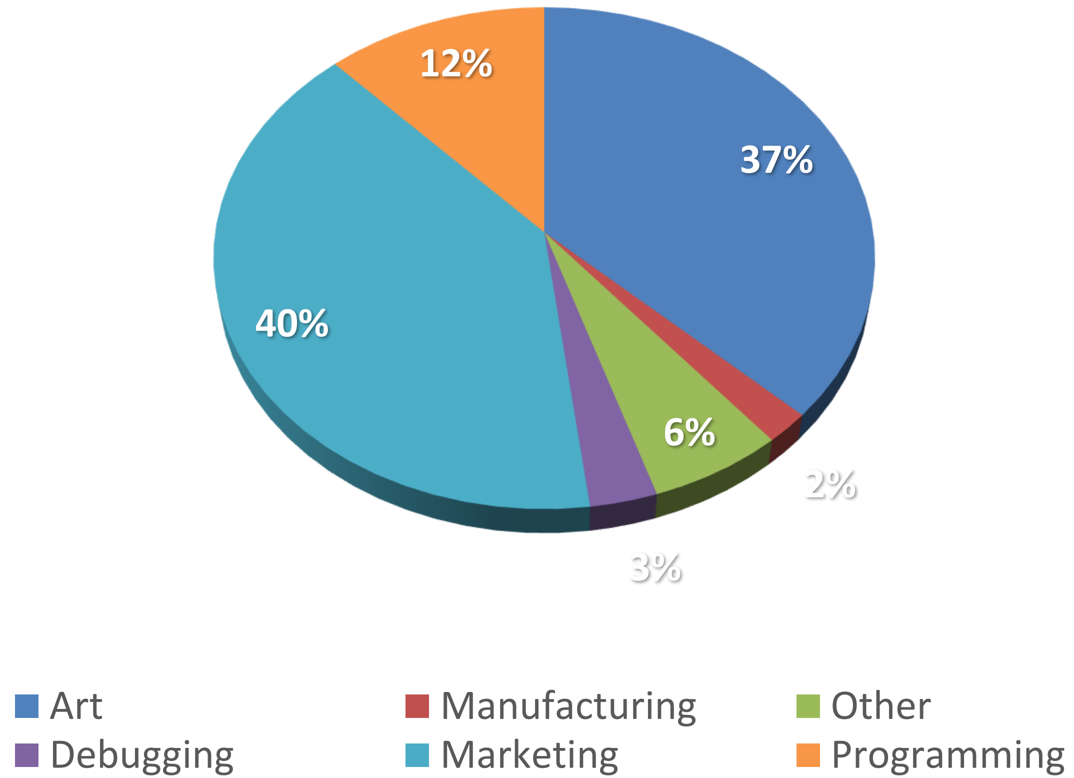
- Prior to 2005 or so: (crickets)
- After 2005, a steadily growing number of papers
- An IEEE Task Force
- Key venues: IEEE CoG, FDG, AIIDE, IEEE ToG, PCG Workshop
- A book: pcgbook.com
- Most papers are about level or map generation
  - Also papers about generation of texture, music, rules, quests, items, weapons, bosses…
- Somewhat constrained by a lack of good testbed games
- A paradigm shift over the years: ML-based PCG (generative AI in games), Mixed-Initiative PCG, Experience-driven PCG, PCGRL, PCGQD,…

Why Generate Content?

- Achieve a specific aesthetic
- AI-assistive tools
- Replayability through variation
- Making infinite or player-adaptive games
- Surpassing human creativity
- Saving cost on content production
- Co-creativity
- Saving storage space

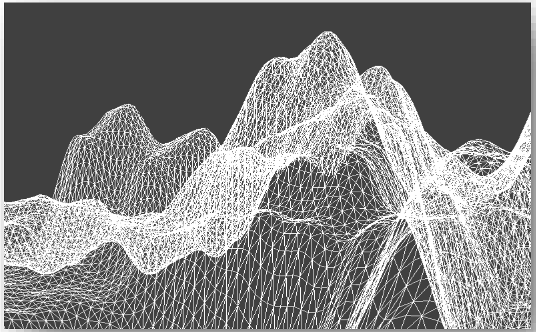


Pie Chart of Questionable Origin and Veracity



What Can PCG Do?

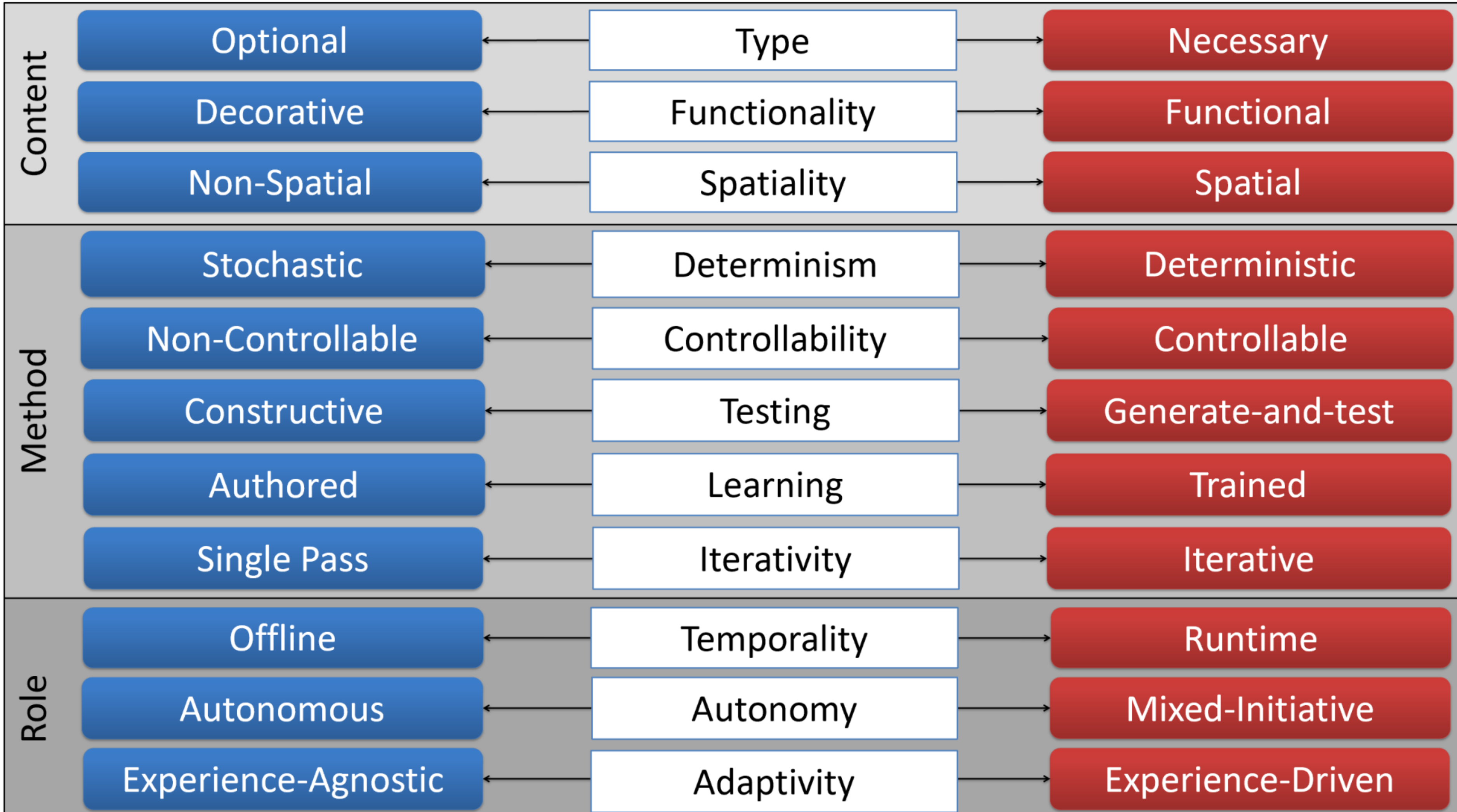
- Can we drastically cut game development costs by creating content automatically from designers' intentions?
- Can we create games that adapt their game worlds to the preferences of the player?
- Can we create endless games?
- Can the computer circumvent or augment limited human creativity and create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

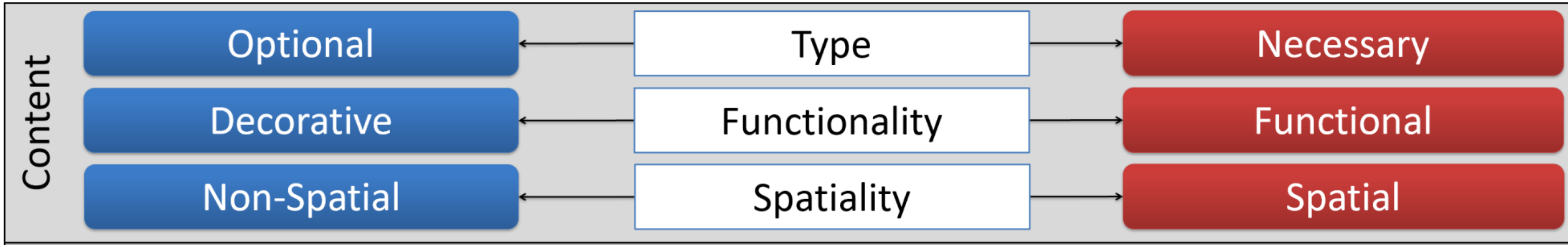


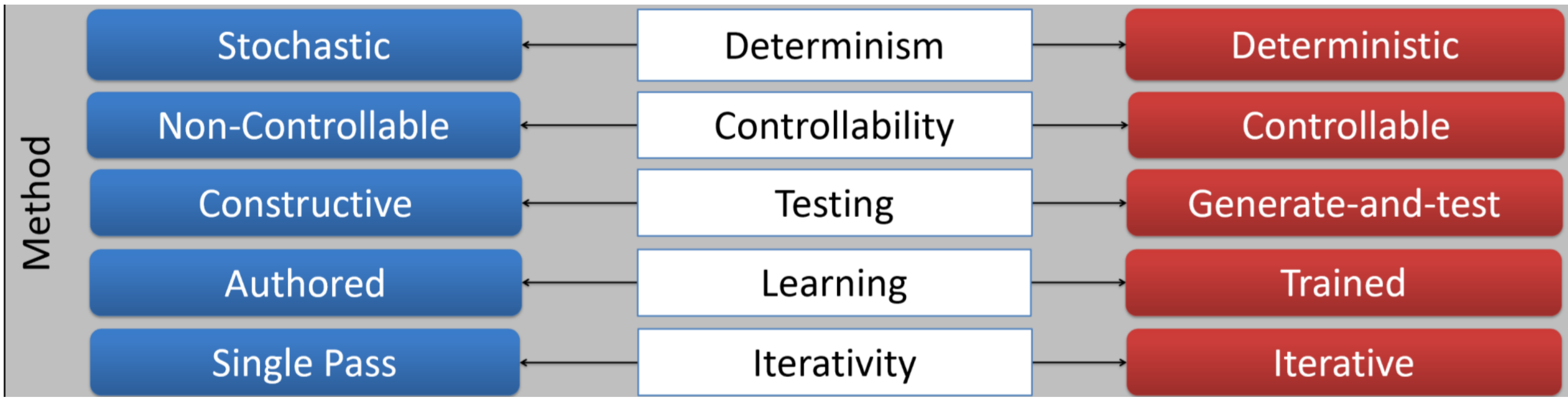
What Are the Trade-offs?

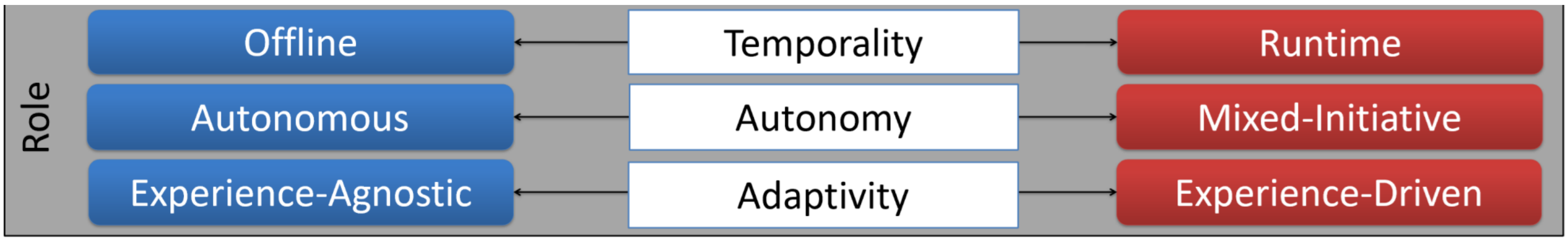
- *Speed*
  Real-time? Or design-time?
- *Reliability*
  Catastrophic failures break gameplay
- *Controllability*
  Allow specification of constraints and goals
- *Diversity*
  Content looks like variations on a theme
- *Creativity*
  Content looks “computer-generated”

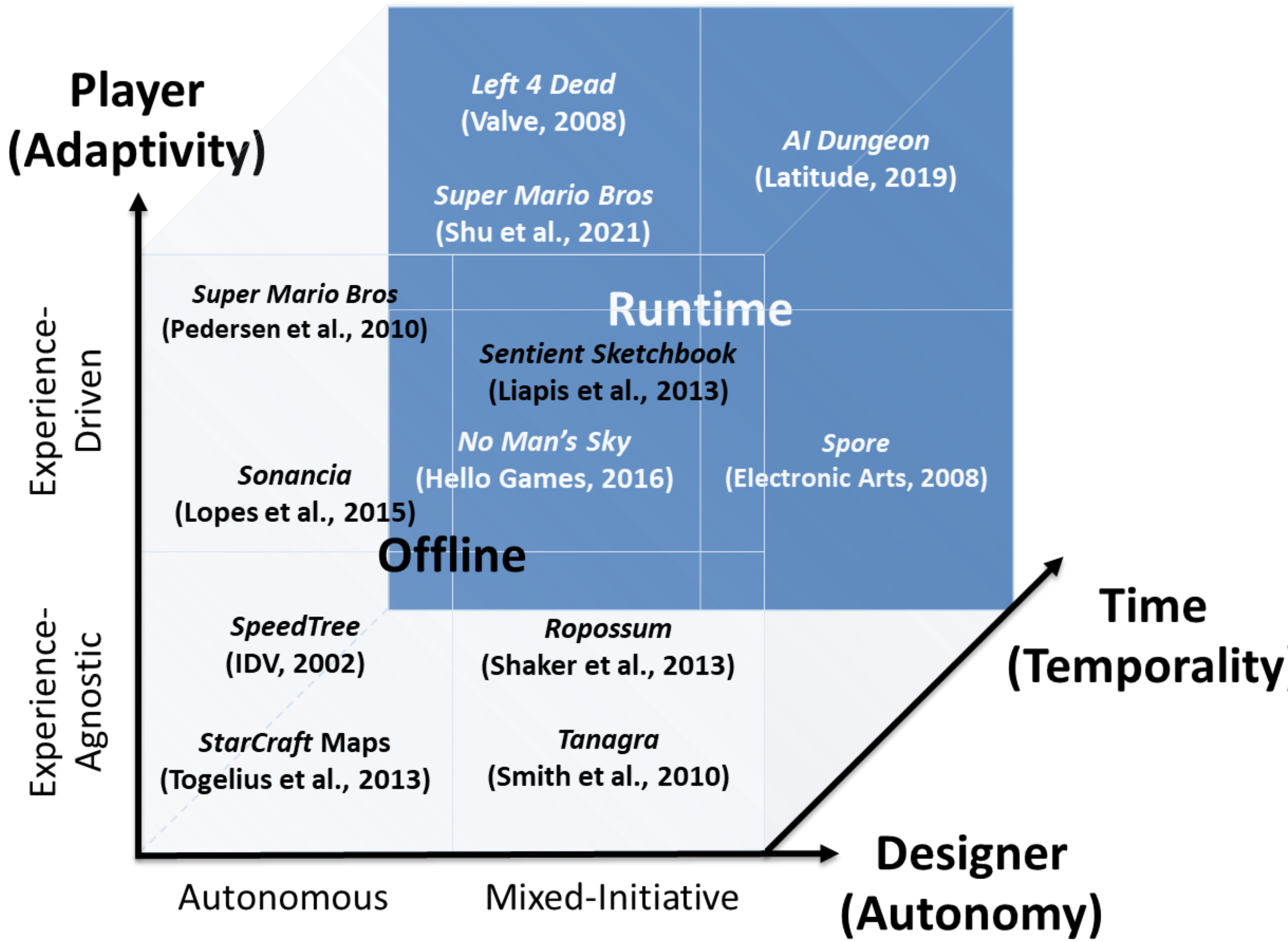
A PCG Taxonomy

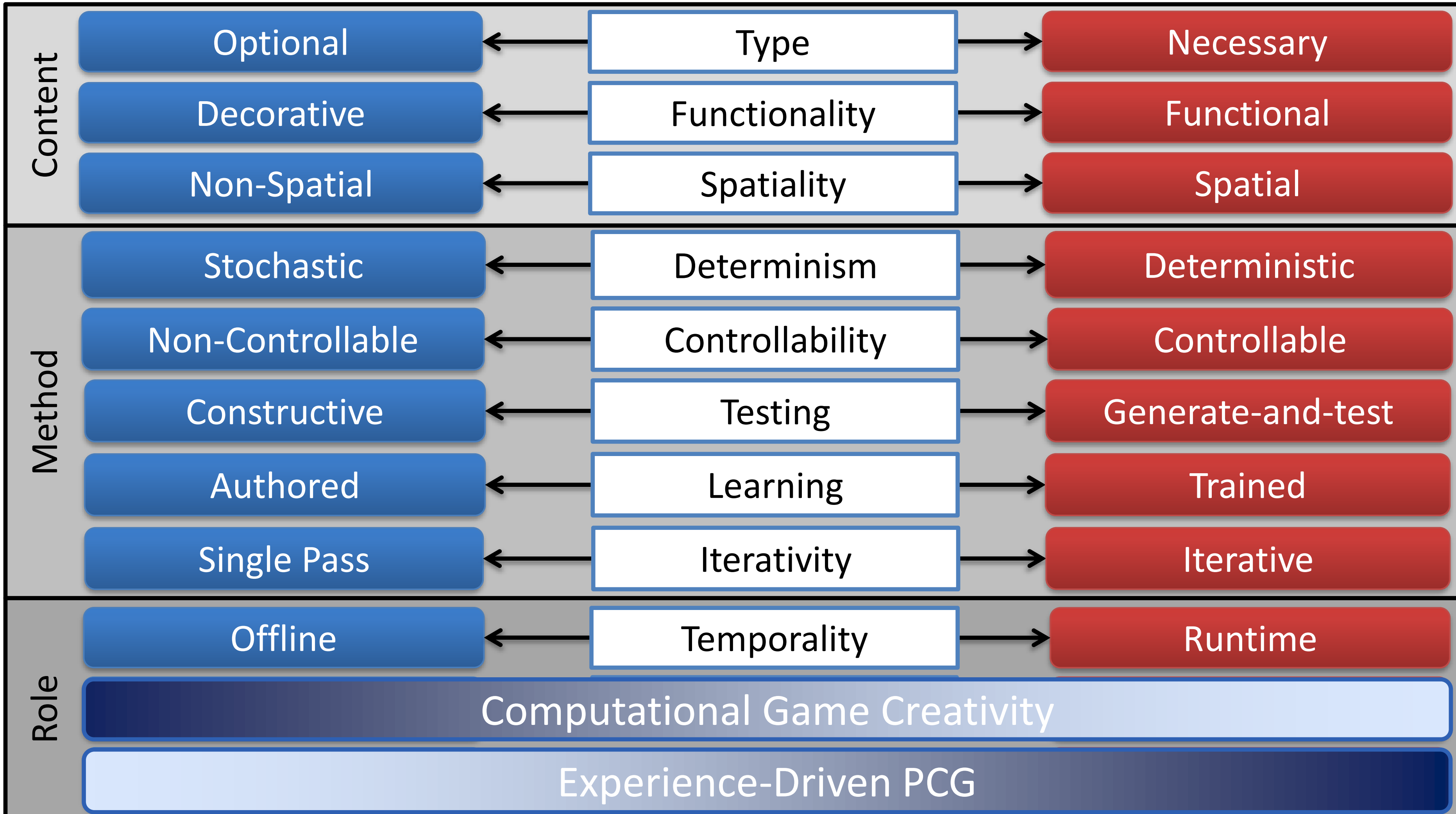

Content
Optional
Type
Necessary
Decorative
Functionality
Functional
Non-Spatial
Spatiality
Spatial
Method
Stochastic
Determinism
Deterministic
Non-Controllable
Controllability
Controllable
Constructive
Testing
Generate-and-test
Authored
Learning
Trained
Single Pass
Iterativity
Iterative
Role
Offline
Temporality
Runtime
Autonomous
Autonomy
Mixed-Initiative
Experience-Agnostic
Adaptivity
Experience-Driven


Content
Optional
Type
Necessary
Decorative
Functionality
Functional
Non-Spatial
Spatiality
Spatial


Method
Stochastic
Determinism
Deterministic
Non-Controllable
Controllability
Controllable
Constructive
Testing
Generate-and-test
Authored
Learning
Trained
Single Pass
Iterativity
Iterative


Role
Offline
Temporality
Runtime
Autonomous
Autonomy
Mixed-Initiative
Experience-Agnostic
Adaptivity
Experience-Driven


Player
(Adaptivity)
Left 4 Dead
(Valve, 2008)
Super Mario Bros
(Shu et al., 2021)
AI Dungeon
(Latitude, 2019)
Runtime
Super Mario Bros
(Pedersen et al., 2010)
Sentient Sketchbook
(Liapis et al., 2013)
No Man's Sky
(Hello Games, 2016)
Spore
(Electronic Arts, 2008)
Sonancia
(Lopes et al., 2015)
Offline
Experience-
Driven
Experience-
Agnostic
SpeedTree
(IDV, 2002)
Ropossum
(Shaker et al., 2013)
StarCraft Maps
(Togelius et al., 2013)
Tanagra
(Smith et al., 2010)
Time
(Temporality)
Designer
(Autonomy)
Autonomous
Mixed-Initiative


Content
Optional
Type
Necessary
Decorative
Functionality
Functional
Non-Spatial
Spatiality
Spatial
Method
Stochastic
Determinism
Deterministic
Non-Controllable
Controllability
Controllable
Constructive
Testing
Generate-and-test
Authored
Learning
Trained
Single Pass
Iterativity
Iterative
Role
Offline
Temporality
Runtime
Computational Game Creativity
Experience-Driven PCG

Computational Creativity **in** and **for** Games!

Liapis, Yannakakis, Togelius: **"Computational Game Creativity,"** in *Proceedings of the Fifth International Conference on Computational Creativity*, 2014.

Computational Creativity

"**Computational Creativity** is a recent area of creativity research that brings together academics and practitioners from diverse disciplines, genres and modalities, to explore **the potential of computers** to be **autonomously creative** or to **collaborate as co-creators** with humans."

-- **PROSECCO** Network of Excellence

PROSECCO
PROMOTING THE SCIENTIFIC EXPLORATION OF COMPUTATIONAL CREATIVITY

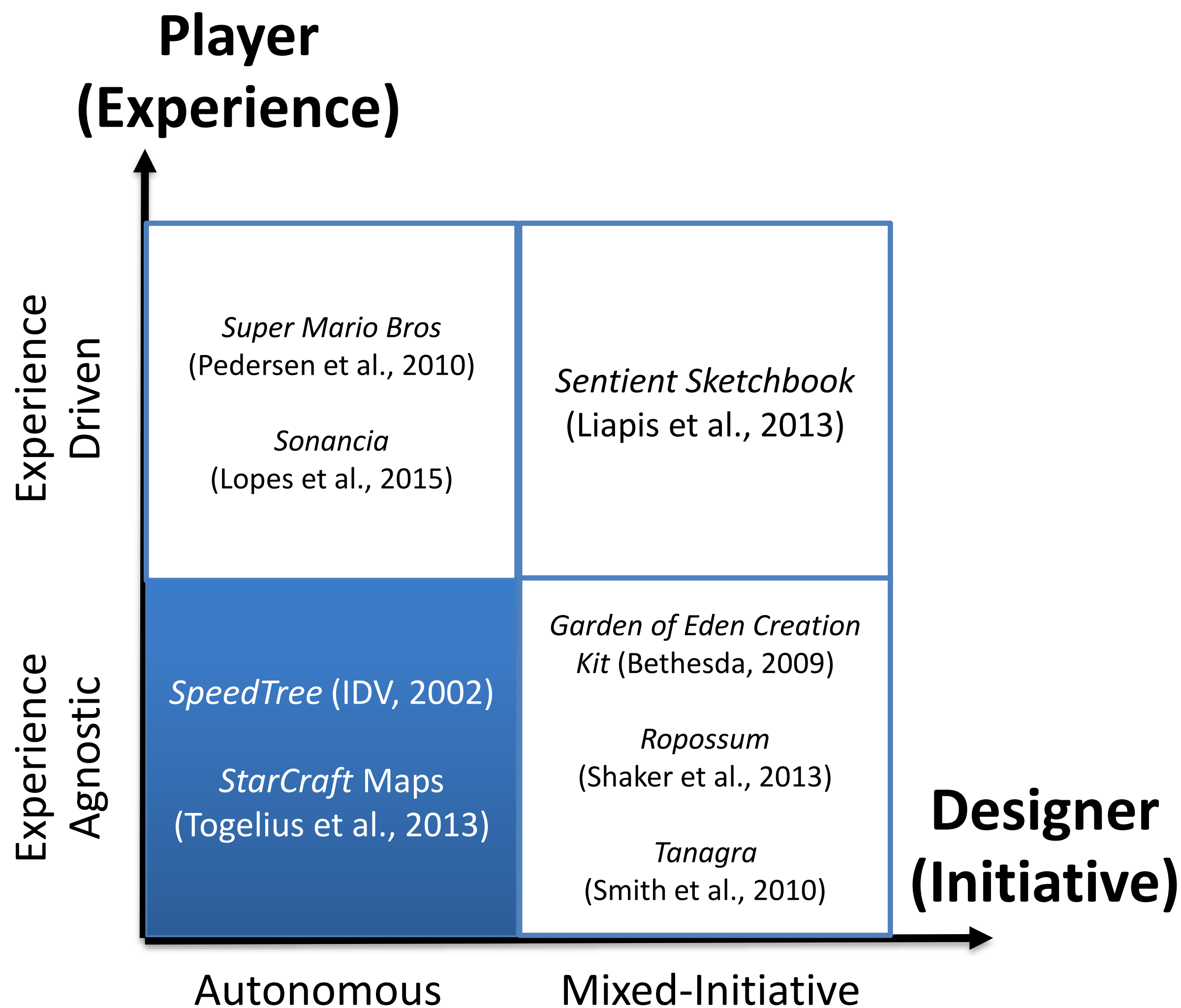
Computational **Game** Creativity

“The study of Computational Creativity **within** and **for** digital games”

Within: Games as an ideal canvas for studying CC

For: Games benefit as products from artifacts of CC

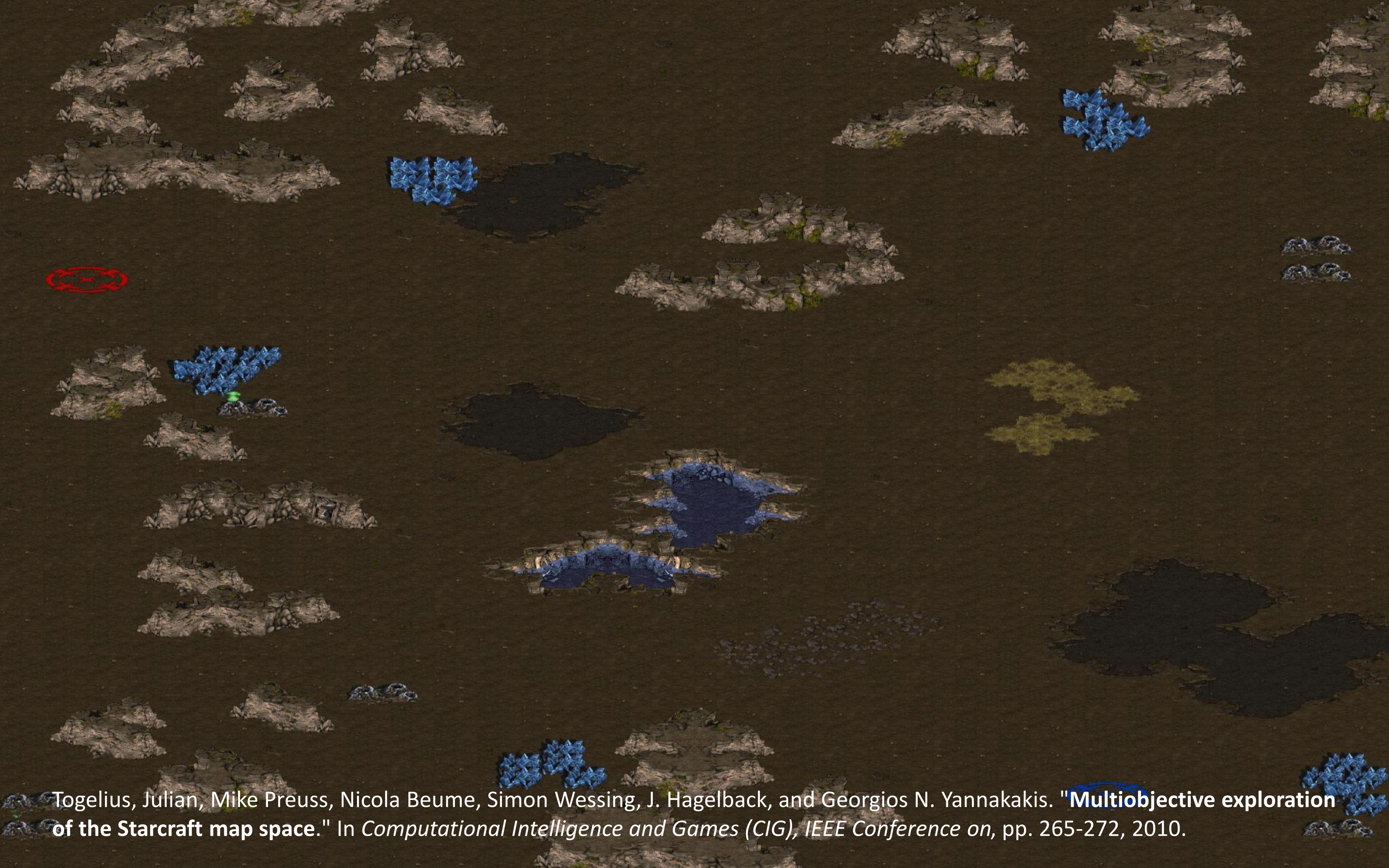
Liapis, Yannakakis, Togelius: **"Computational Game Creativity,"** in *Proceedings of the Fifth International Conference on Computational Creativity*, 2014.


Player
(Experience)
Experience
Driven
Experience
Agnostic
Super Mario Bros
(Pedersen et al., 2010)
Sonancia
(Lopes et al., 2015)
Sentient Sketchbook
(Liapis et al., 2013)
SpeedTree (IDV, 2002)
StarCraft Maps
(Togelius et al., 2013)
Garden of Eden Creation
Kit (Bethesda, 2009)
Ropossum
(Shaker et al., 2013)
Tanagra
(Smith et al., 2010)
Designer
(Initiative)
Autonomous
Mixed-Initiative

Generation
Forces
Geometry Types
Random Seeds
Hand Drawn
Segments
Spine
Branch
Texture Coordinates
Materials
Displacement
Ambient Occlusion
Level of Detail
Wind
Physics
Spine Generator
Ready

Procedural FPS Level Generation

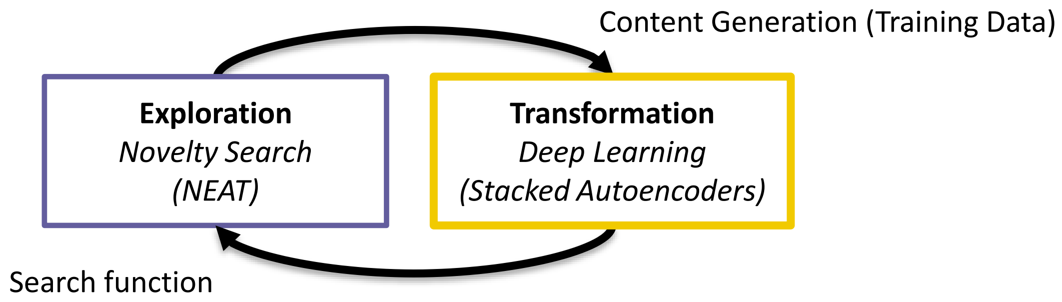
W. Cachia, A. Liapis, and G. N. Yannakakis, "**Multi-Level Evolution of Shooter Levels**", in *Proceedings of AIIDE*, 2015.



Togelius, Julian, Mike Preuss, Nicola Beume, Simon Wessing, J. Hagelback, and Georgios N. Yannakakis. "**Multiobjective exploration of the Starcraft map space**." In *Computational Intelligence and Games (CIG), IEEE Conference on*, pp. 265-272, 2010.

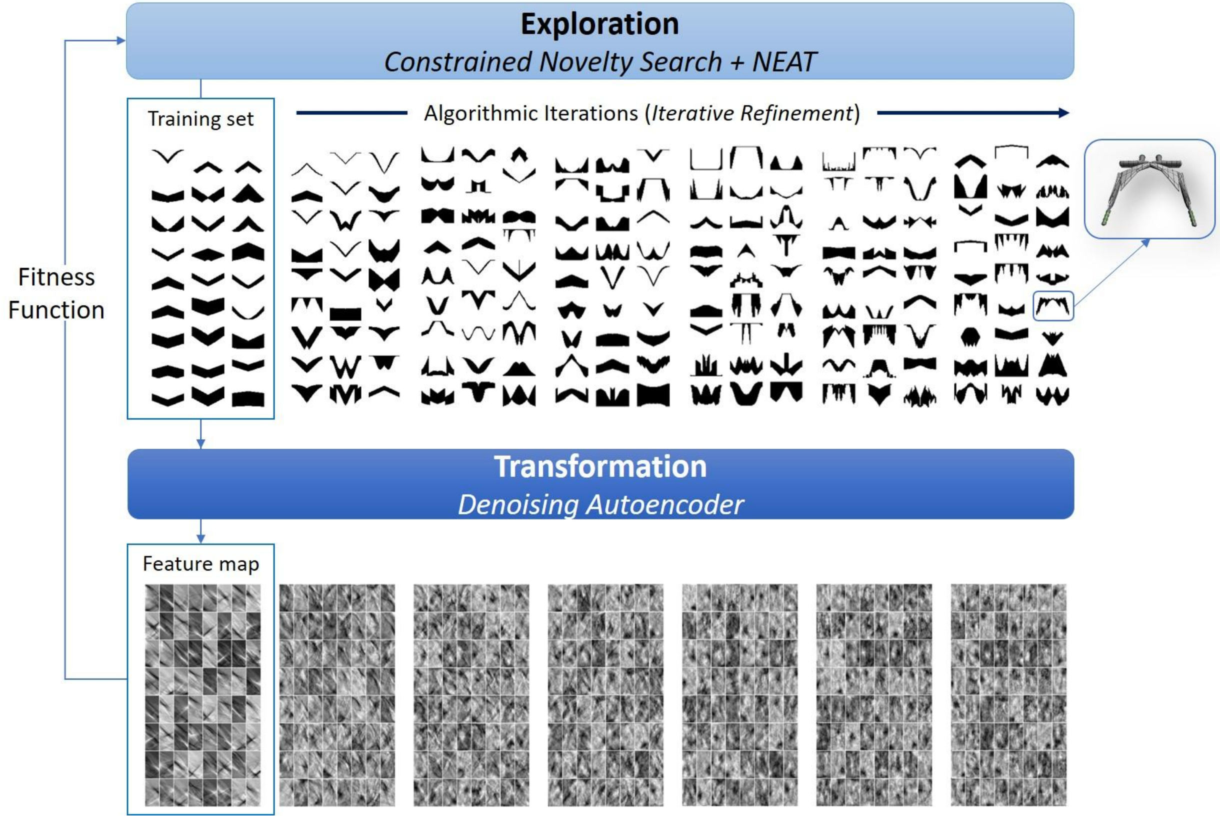
Deep Learning Meets Novelty Search

Liapis, Martínez, Togelius, and Yannakakis: **"Transforming Exploratory Creativity with DeLeNoX,"** in *Proceedings of the Fourth International Conference on Computational Creativity*, 2013.

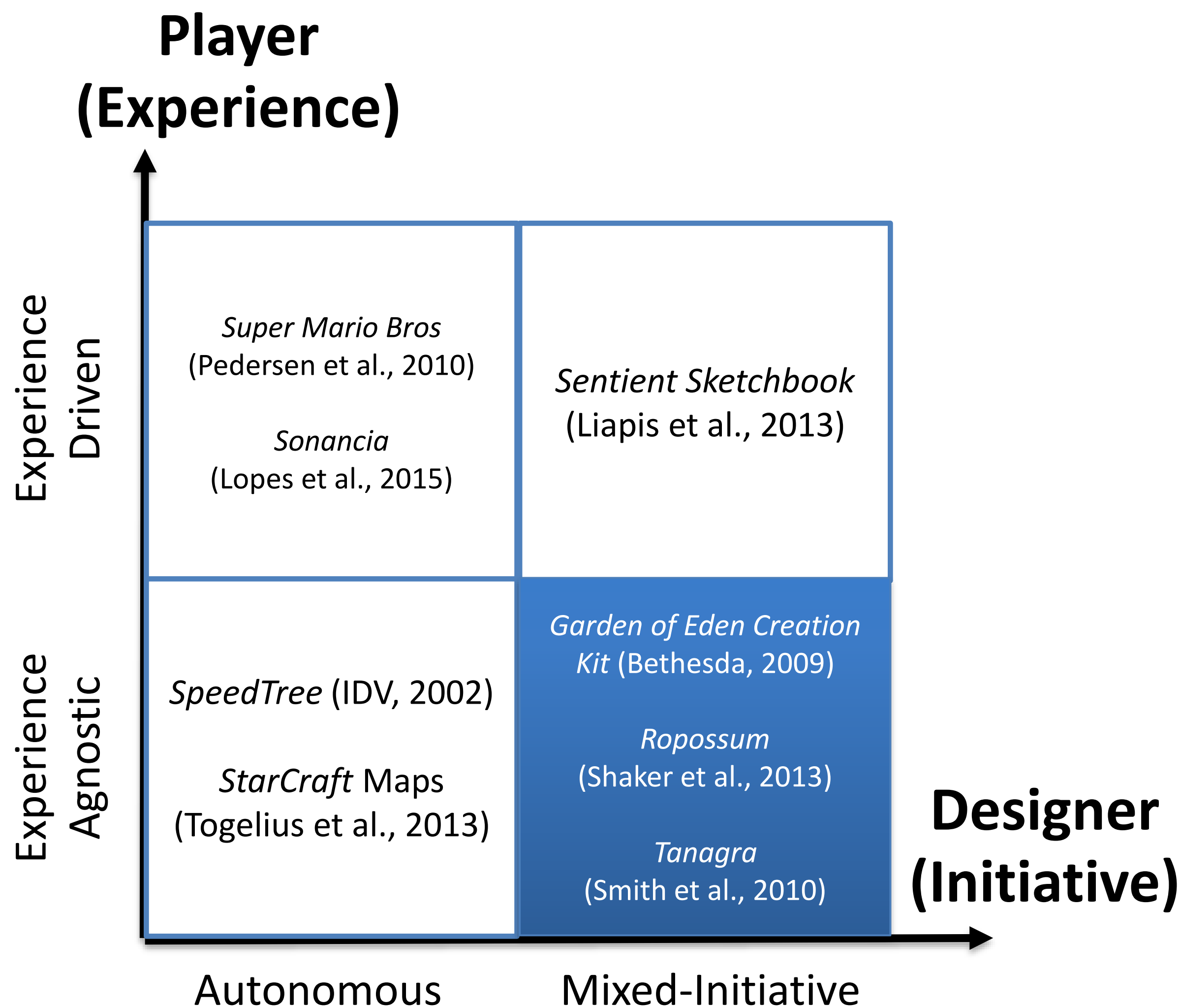


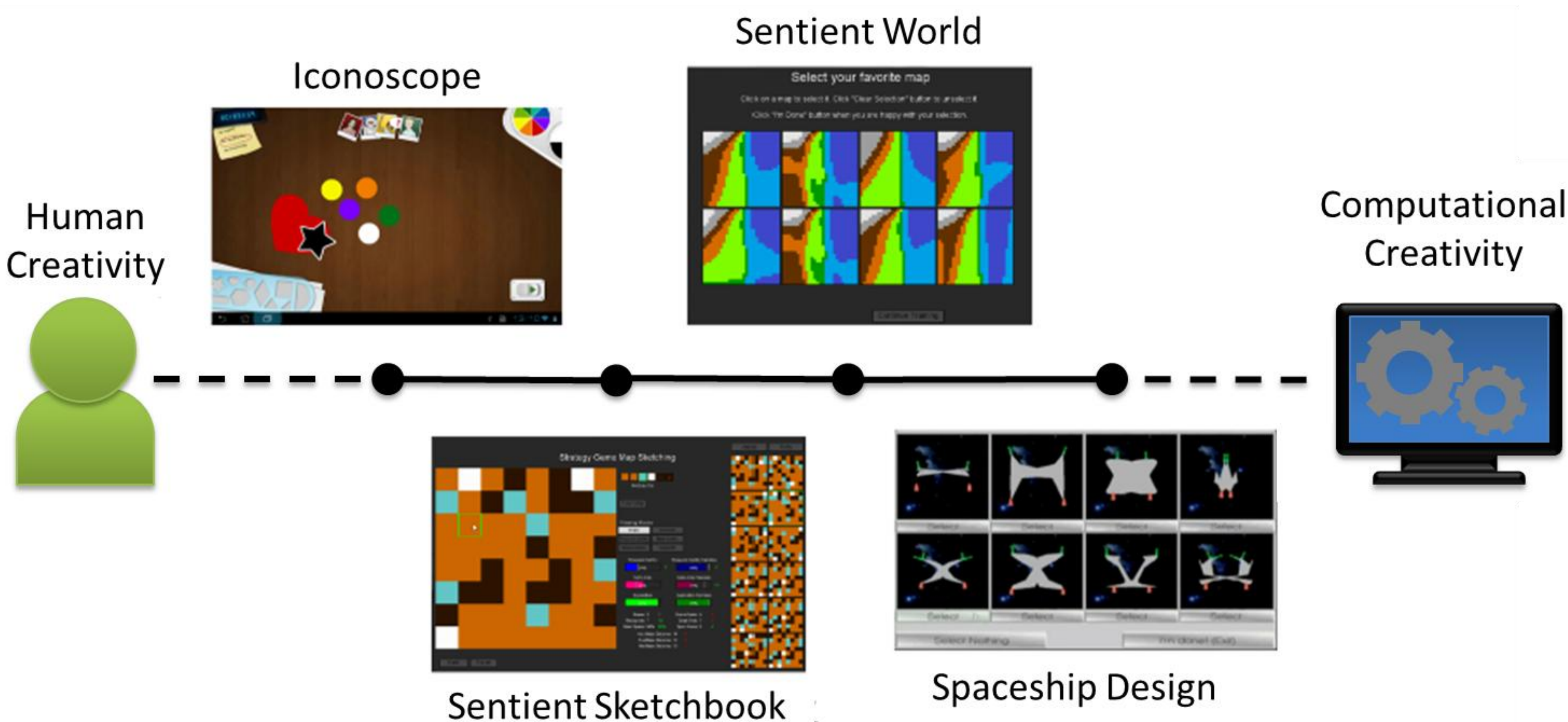
Deep Learning Meets Novelty Search

Liapis, Martínez, Togelius, and Yannakakis: **"Transforming Exploratory Creativity with DeLeNoX,"** in *Proceedings of the Fourth International Conference on Computational Creativity*, 2013.

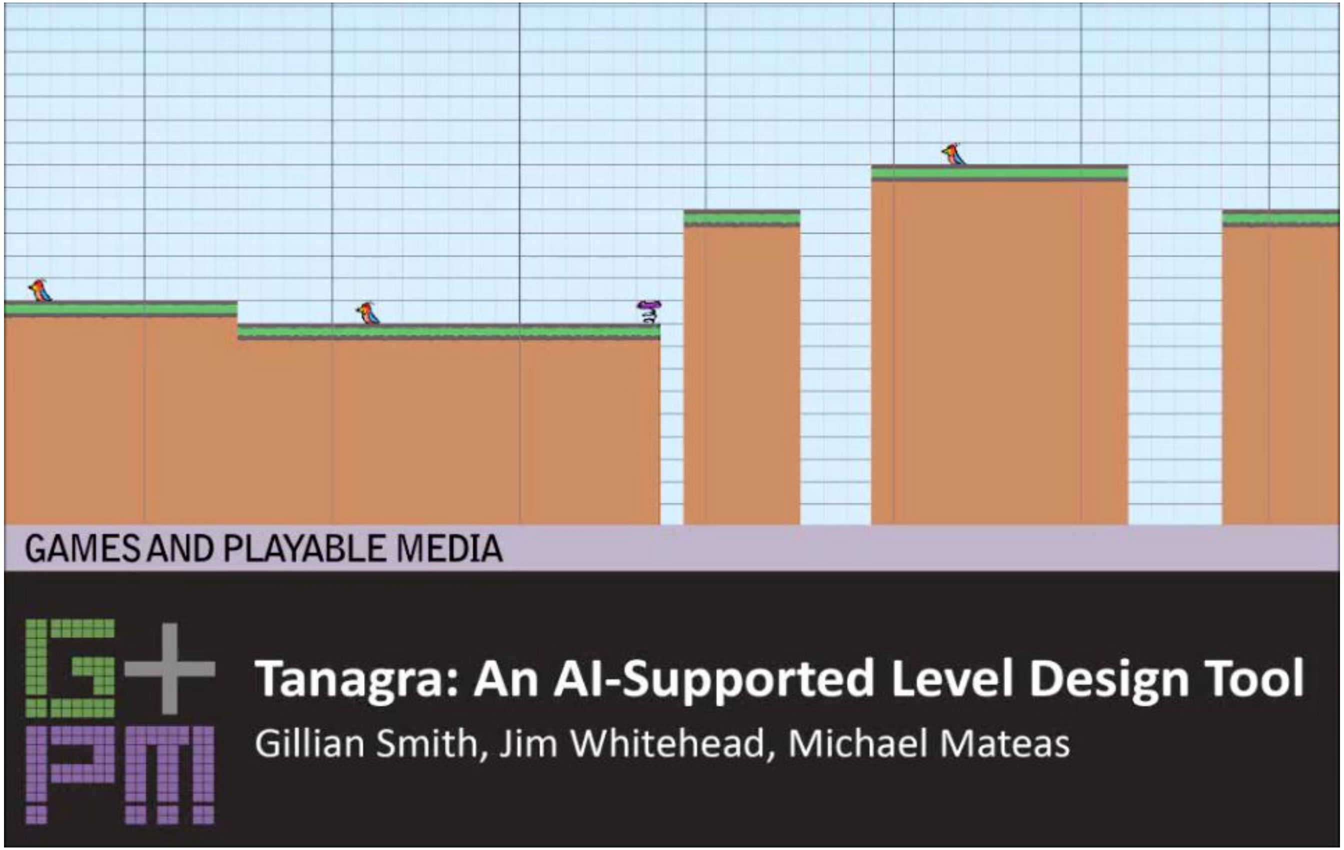


Barthet, Liapis, and Yannakakis. **Open-Ended Evolution for Minecraft Building Generation** *IEEE Transactions on Games* (2022)


Player
(Experience)
Experience
Driven
Experience
Agnostic
Super Mario Bros
(Pedersen et al., 2010)
Sonancia
(Lopes et al., 2015)
Sentient Sketchbook
(Liapis et al., 2013)
SpeedTree (IDV, 2002)
StarCraft Maps
(Togelius et al., 2013)
Garden of Eden Creation
Kit (Bethesda, 2009)
Ropossum
(Shaker et al., 2013)
Tanagra
(Smith et al., 2010)
Designer
(Initiative)
Autonomous
Mixed-Initiative


Sentient World
Select your favorite map
Iconoscope
Human
Creativity
Computational
Creativity
Sentient Sketchbook
Spaceship Design

Tanagra: Constraint Solver for MI-PCG



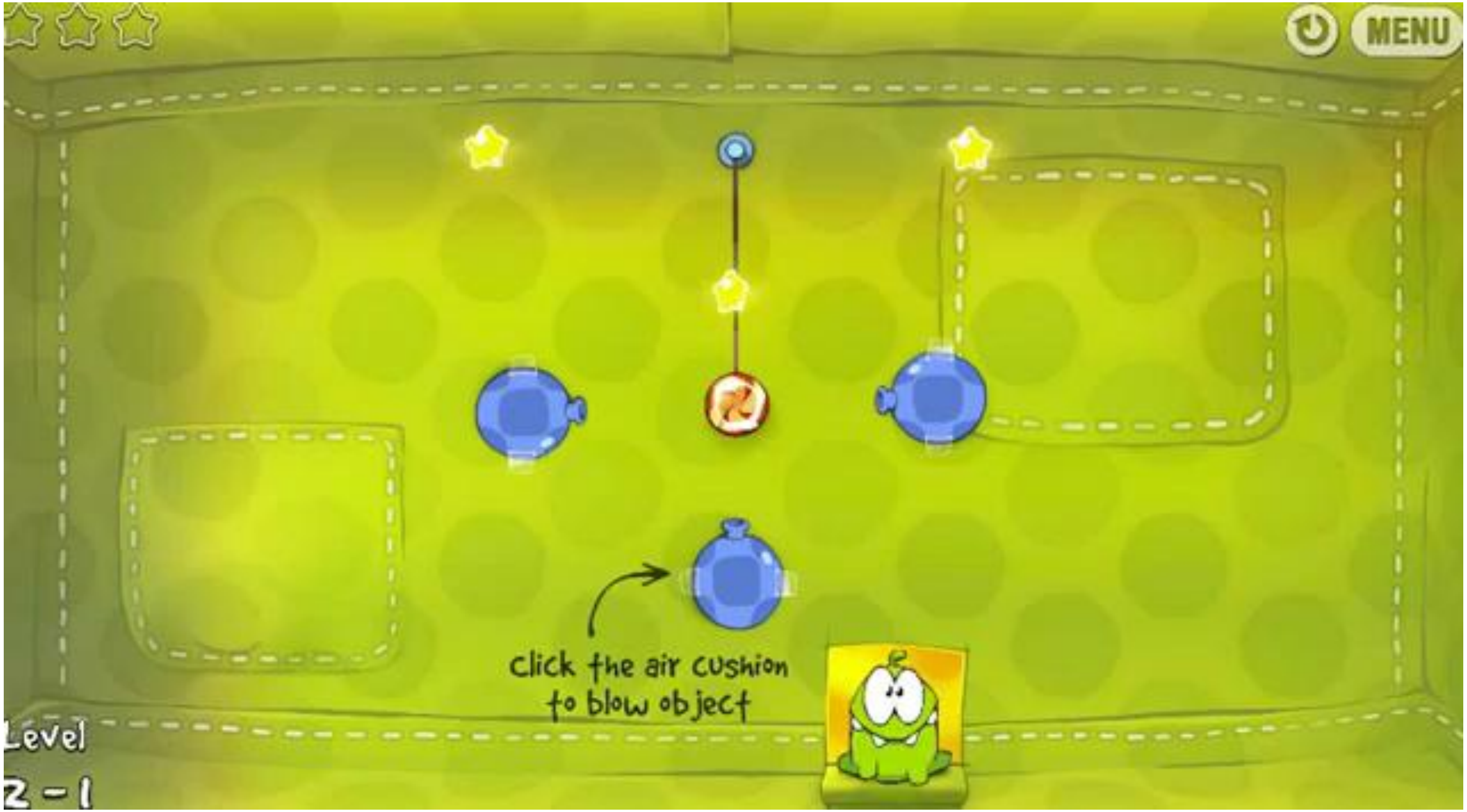
Ropossum

Physics-based puzzle “cut the rope”

- evolutionary grammars for creating new puzzles
- playability module for testing how (if?) to solve a puzzle
- using the designer’s input in complete or partial designs

M. Shaker, M. H. Sarhan, O. A. Naameh, N. Shaker, and J. Togelius. Automatic generation and analysis of physics-based puzzle games. In Computational Intelligence in Games (CIG), 2013 IEEE Conference on.

Ropossum



Ropossum

coralize

A reef / coral PCG tool for Unity

R. Abela, A. Liapis, G. N. Yannakakis: **"A Constructive Approach for the Generation of Underwater Environments,"** in *Proceedings of FDG*, 2015.

Baba is Y'all: Collaborative Mixed-Initiative Level Design

Megan Charity
Game Innovation Lab
New York Univeristy
Brooklyn, USA
mlc761@nyu.edu

Ahmed Khalifa
Game Innovation Lab
New York Univeristy
Brooklyn, USA
ahmed@akhalifa.com

Julian Togelius
Game Innovation Lab
New York Univeristy
Brooklyn, USA
julian@togelius.com

***Abstract*—We present a collaborative mixed-initiative system for building levels for the puzzle game "Baba is You". Unlike previous mixed-initiative systems, Baba is Y'all is designed for collaborative asynchronous creation by multiple users over the internet. The system includes several AI-assisted features to help designers, including a level evolver and an automated player for playtesting. The level archives catalogues levels according to which mechanics are implemented and not implemented, allowing the system to ask users to design levels with specific combinations of mechanics. We describe the operation of the system and the results of small-scale informal user test, and discuss future development paths for this system as well as for collaborative mixed-initiative systems in general.**

***Index Terms*—PCG, Level Generation, Mixed-Inititive, Evolutionary Computation, Quality Diversity**

I. Introduction

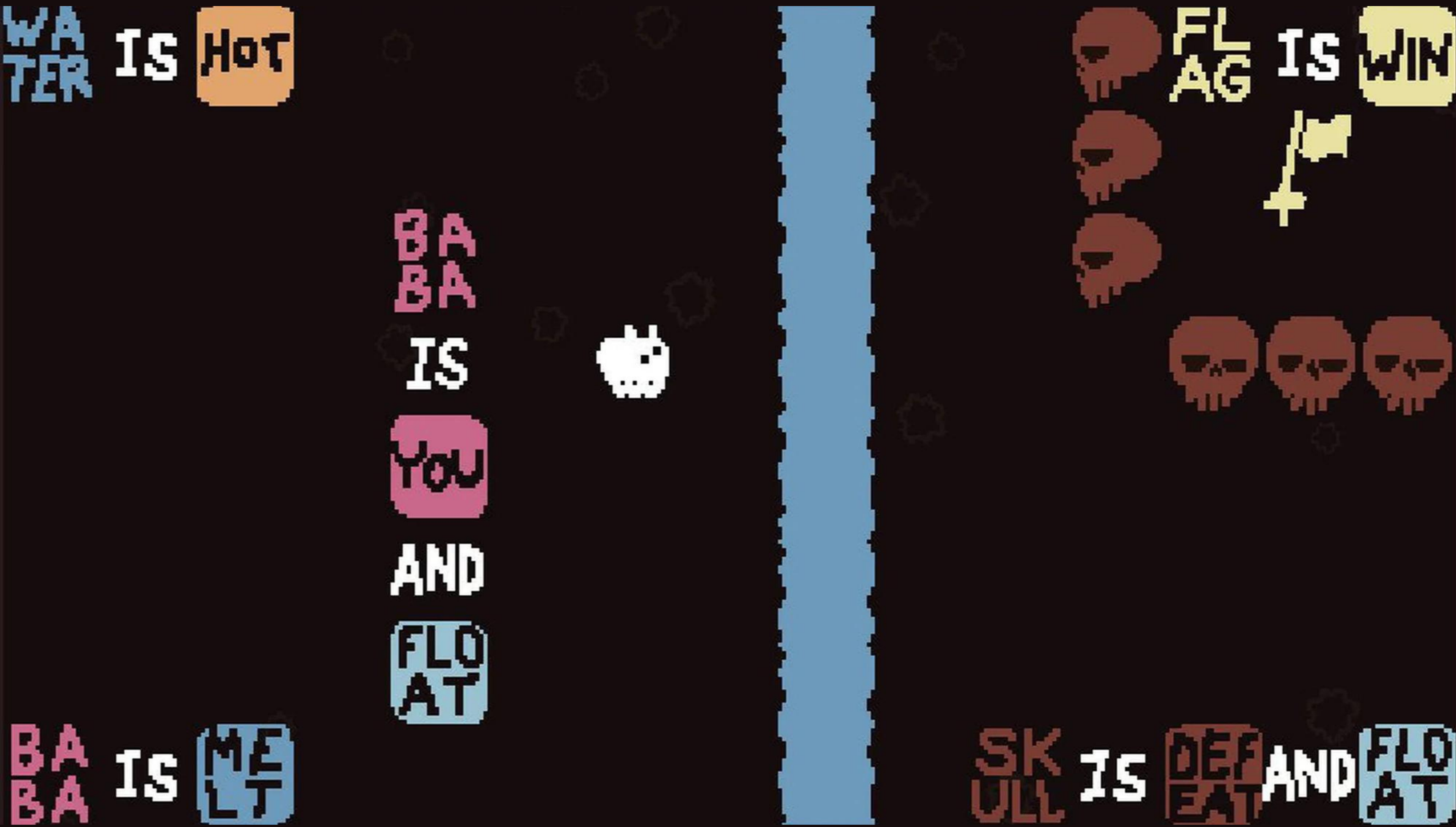
How to best design game content together with content generation algorithms is a hard and important question. A number of prototype systems for *mixed-initiative design* have been created to showcase ways in which humans and algorithms can design game content together [1, 2, 3]. Many different modes of interaction have been devised, including those where the

levels for the puzzle game Baba is You (Arvi Teikari, 2019). The system includes features for editing levels, automatically playtesting levels, helping design levels through an evolutionary algorithm, rating levels, and suggesting novel levels to design. All features are built around a central level archive which is structured like the map of elites from the MAP-Elites algorithm. We also report preliminary results from an informal user study, shining some light on how the system can be used.

II. Background

A. Procedural Content Generation

Procedural Content Generation is the process of using a computer program to create content [4]. These techniques have been used since the early days of computer games and still remain a popular technique today. PCG is typically divided based on the technology behind the generation process into three main categories: Constructive techniques [4], Search-Based techniques [5], and Machine Learning techniques [6]. Search based approaches are more common in academia for their generality and ease of use. These techniques use a


WATER IS HOT
FLAG IS WIN
BABA IS YOU AND FLOAT
BABA IS MELT
SKULL IS DEFEAT AND FLOAT

BABA IS Y'ALL

New | Top | Unmade | Rate | Search | My Levels

Gallotta, Liapis, Yannakakis, **LLMaker: A Game Level Design Interface Using (Only) Natural Language**, in *Proc. of IEEE CoG,* 2024

LLMaker

Gallotta, Liapis, Yannakakis, **LLMaker: A Game Level Design Interface Using (Only) Natural Language**, in Proc. of IEEE CoG, 2024

You: Create a room set in USSR with two enemies
AI: The "USSR Room" has been created, and it's filled with two enemies: Ivan, a burly soldier with 20 health points, and Nikolai, a sharpshooter with 15 health points. Both enemies have unique descriptions and characteristics.
You: Add a mine trap
AI: It seems that I made an error. Traps can only be added to corridors, not to rooms. If you'd like, I can create a corridor and add the mine trap there.

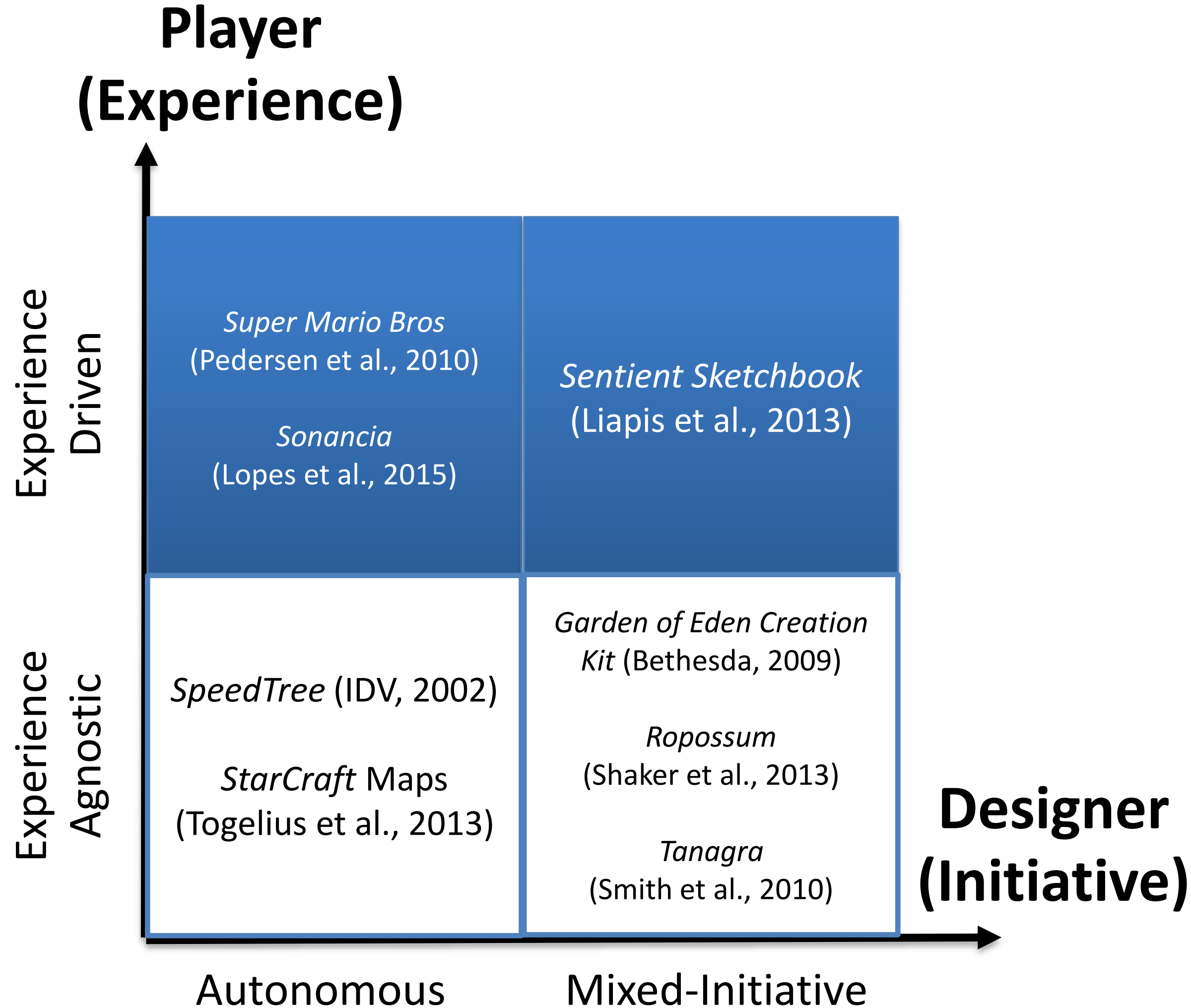
LLMaker

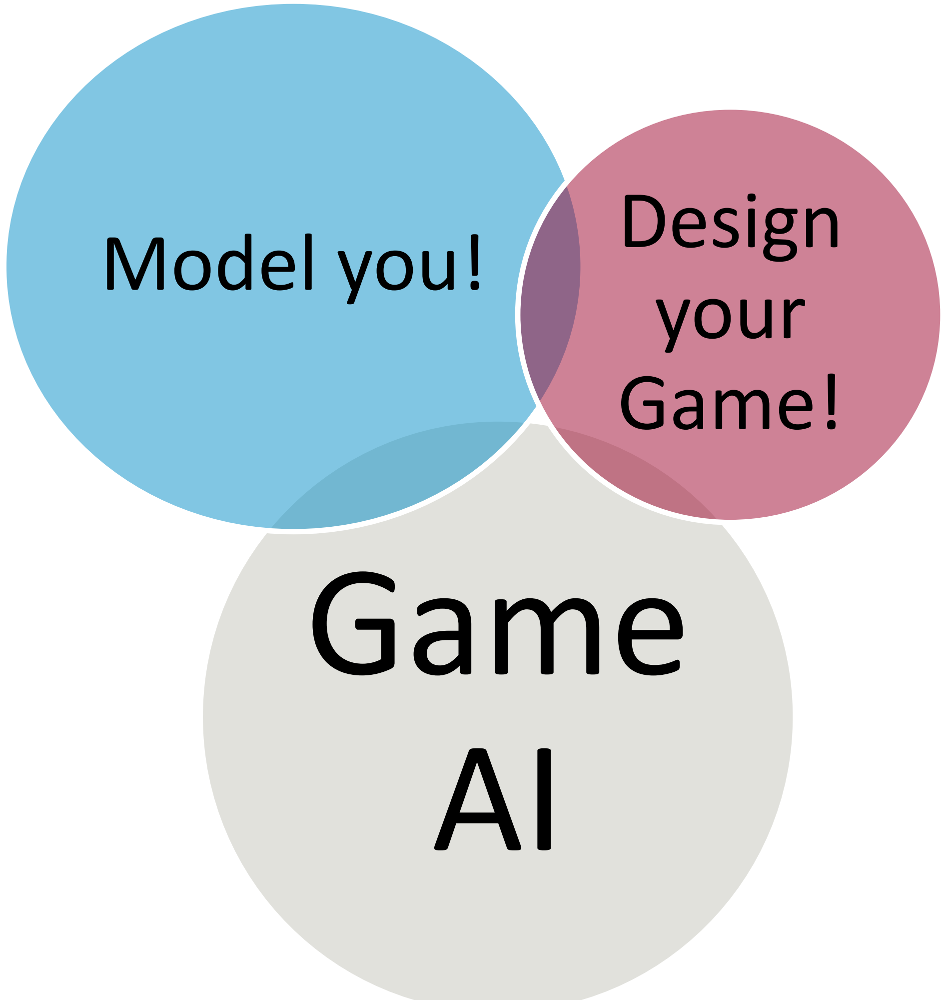
You: Make a Super Mario Bros room
AI: The "Super Mario Bros" room inspired by the iconic game has been added to the level.
You: Add a couple of enemies
AI: Two enemies, a Goomba and a Koopa Troopa, have been added to the "Super Mario Bros" room.

Room: ***Super Mario Bros***

A room inspired by the iconic Super Mario Bros game, filled with colorful blocks and pipes, with a flagpole at the end.

Gallotta, Liapis, Yannakakis, **LLMaker: A Game Level Design Interface Using (Only) Natural Language**, in Proc. of IEEE CoG, 2024

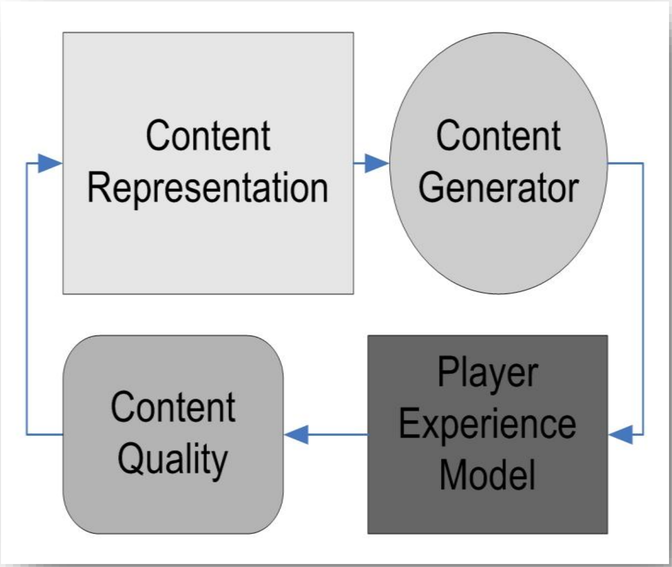

Player
(Experience)
Experience Driven
Experience Agnostic
Super Mario Bros
(Pedersen et al., 2010)
Sonancia
(Lopes et al., 2015)
Sentient Sketchbook
(Liapis et al., 2013)
SpeedTree (IDV, 2002)
StarCraft Maps
(Togelius et al., 2013)
Garden of Eden Creation Kit (Bethesda, 2009)
Ropossum
(Shaker et al., 2013)
Tanagra
(Smith et al., 2010)
Designer
(Initiative)
Autonomous
Mixed-Initiative



Yannakakis and Togelius, **Experience-driven Procedural Content Generation**, *IEEE Transactions on Affective Computing*, 2011.

A General PCG Framework

- Content is the **building block** of player experience
- **Search-based PCG**: use optimization algorithms (such as evolutionary algorithms) to search the space of content
- **Experience-driven PCG**: base evaluation function on player experience models



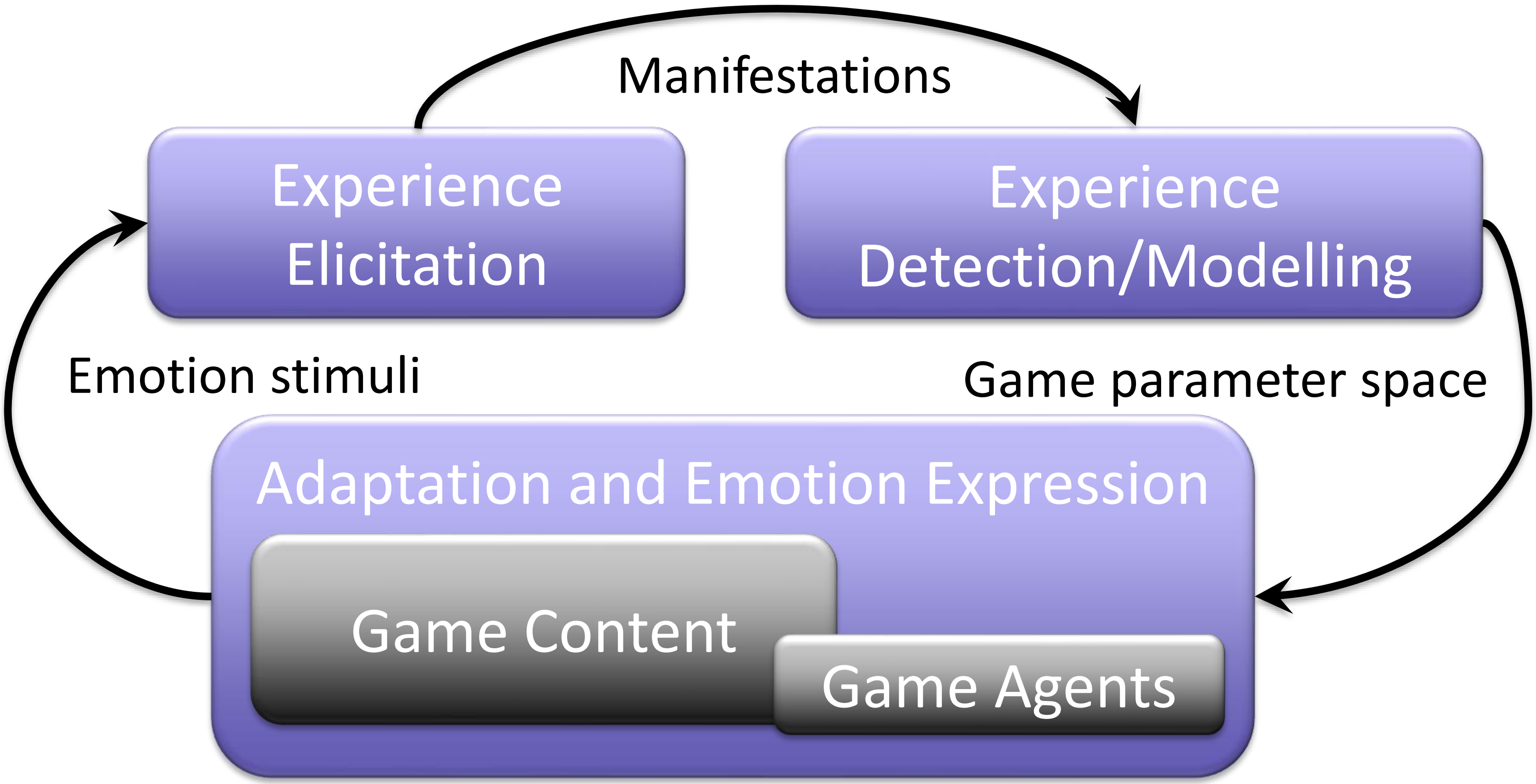
EDPCG: What is it?

“**A framework** for **personalised generation of content** in human computer interaction (in particular in games). It views **(game) content as the *building block* of user (player) experience**”

Yannakakis, G. N., & Togelius, J. (2011). **Experience-driven procedural content generation**. *IEEE Transactions on Affective Computing, 2*(3), 147-161.

EDPCG Best Realizes the Affective Loop

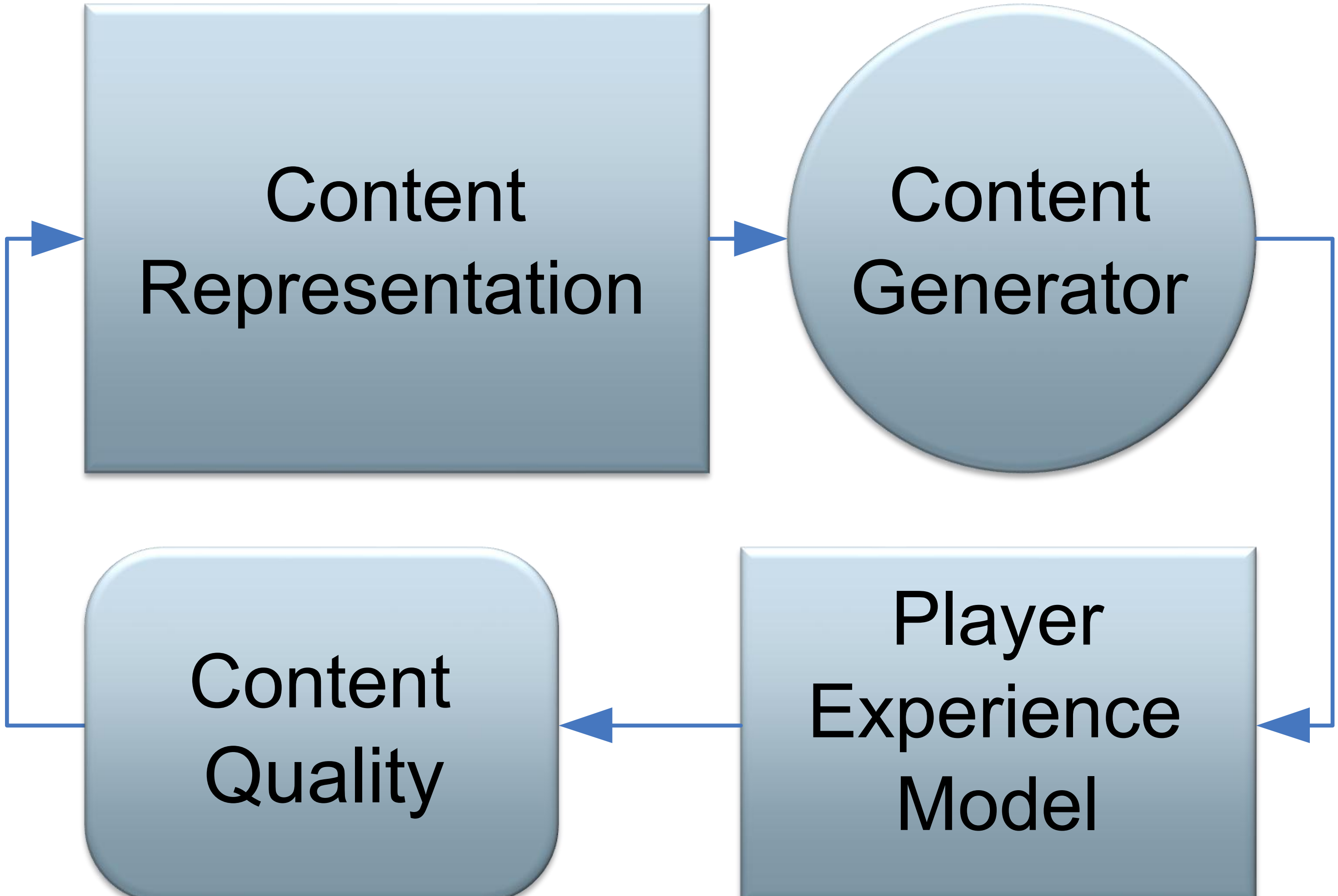
Yannakakis and Paiva, **Emotion in Games**, in *Handbook of Affective Computing*, 2013

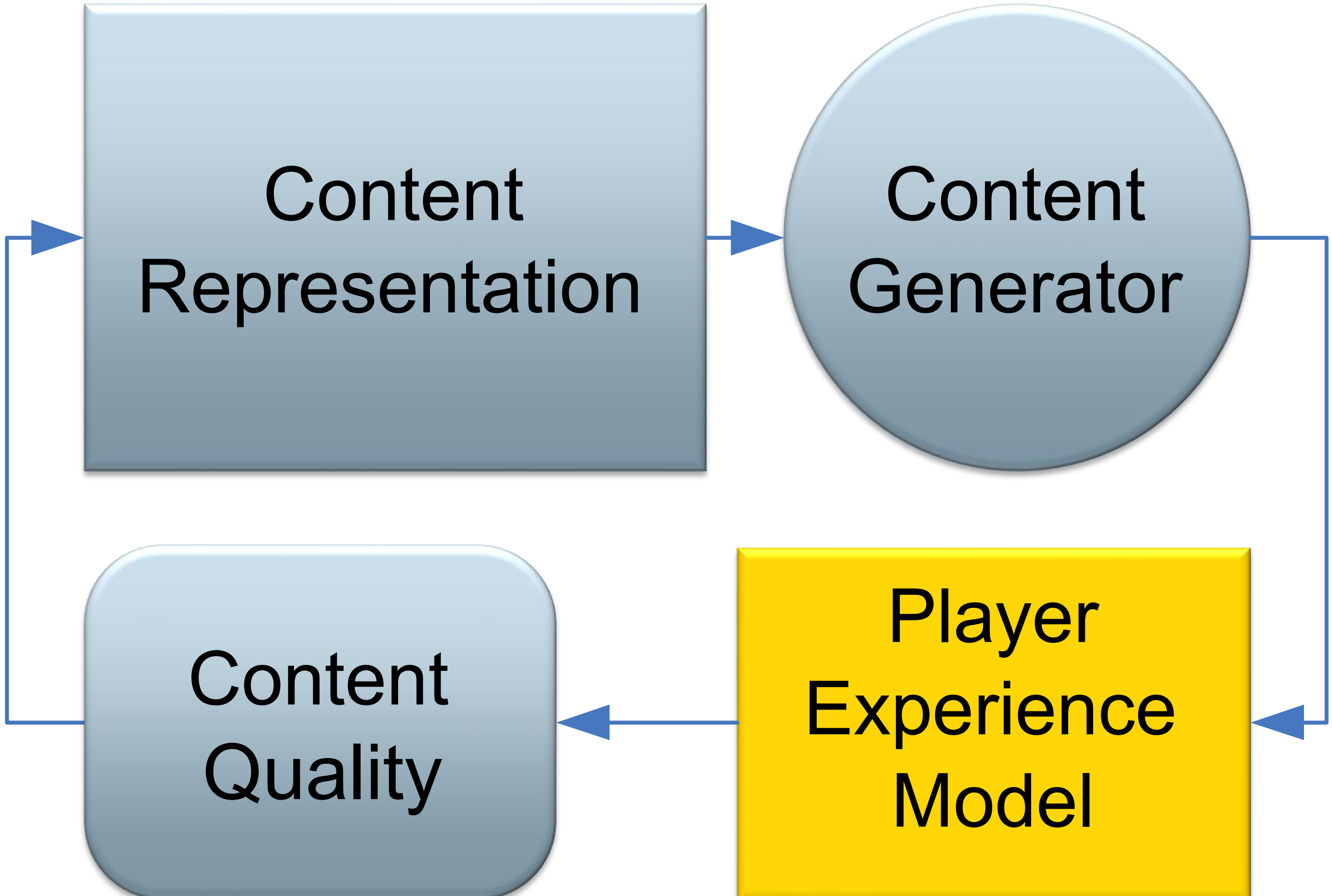


Experience-Driven Procedural Content Generation

“… collection of **affective patterns** elicited, **cognitive processes** emerged and **behavioral traits** observed during interaction (gameplay)”

Key Components

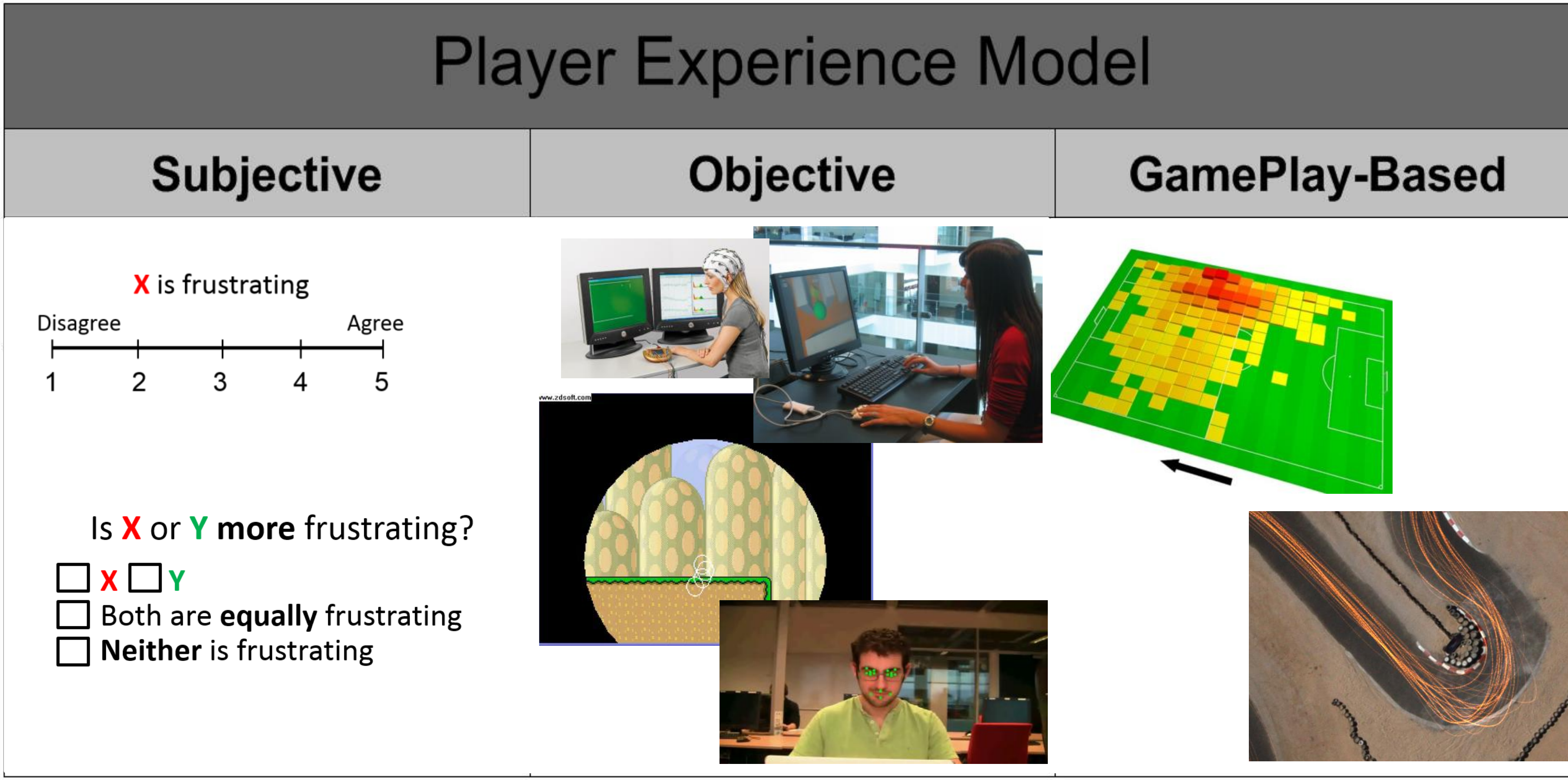

Content Representation
Content Generator
Player Experience Model
Content Quality


Content Representation
Content Generator
Player Experience Model
Content Quality

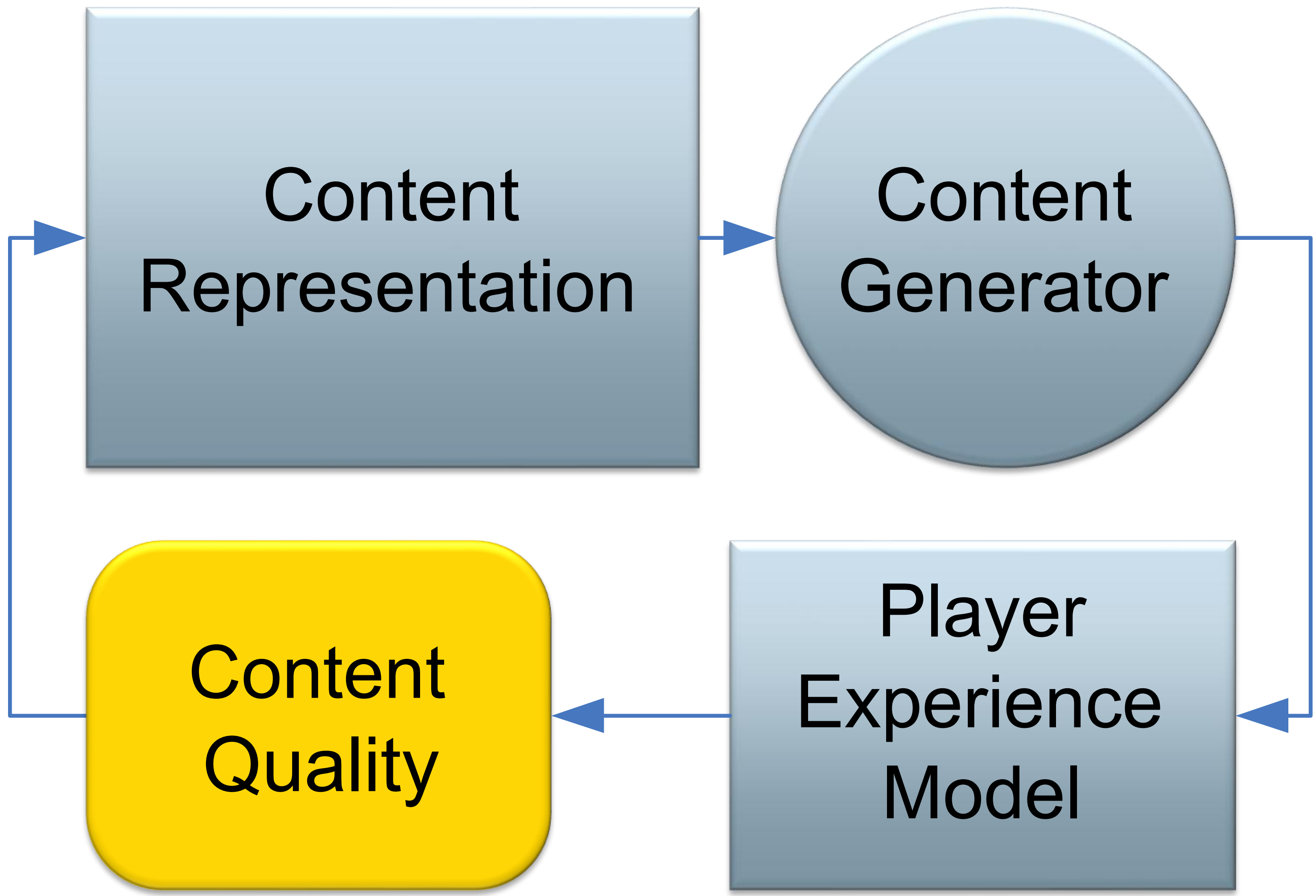
Player Experience Modeling

- Content evaluation functions could (should?) be based on player experience models
- Predict player experience from behavior/cognition/affect and/or game content
- Derived from empirical measurements of player experience

Yannakakis, G. N., & Togelius, J. (2011). **Experience-driven procedural content generation**. *Affective Computing, IEEE Transactions on*, *2*(3), 147-161.

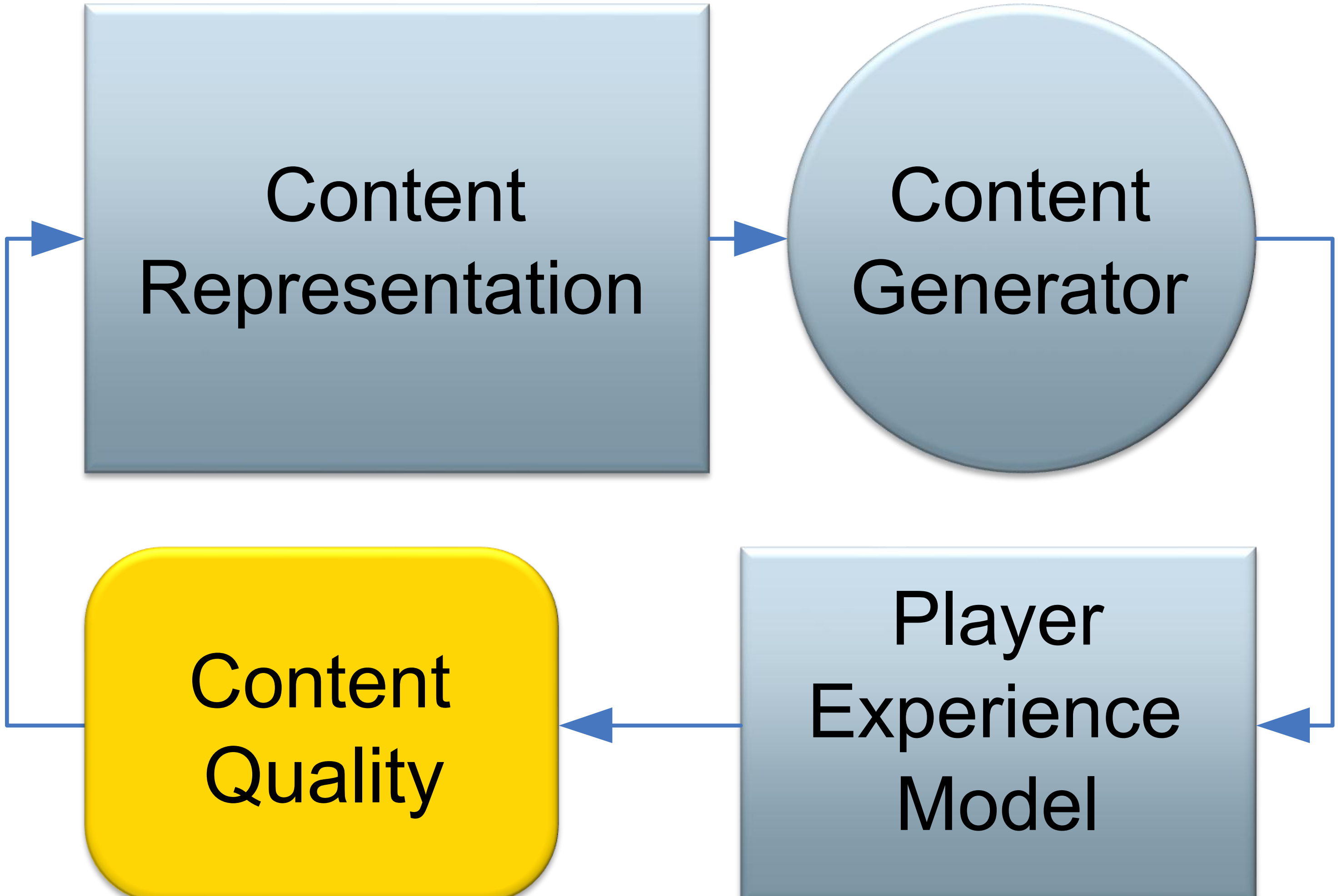

Player Experience Model
Subjective
Objective
GamePlay-Based
X is frustrating
Disagree
Agree
1
2
3
4
5
Is X or Y more frustrating?
X
Y
Both are equally frustrating
Neither is frustrating

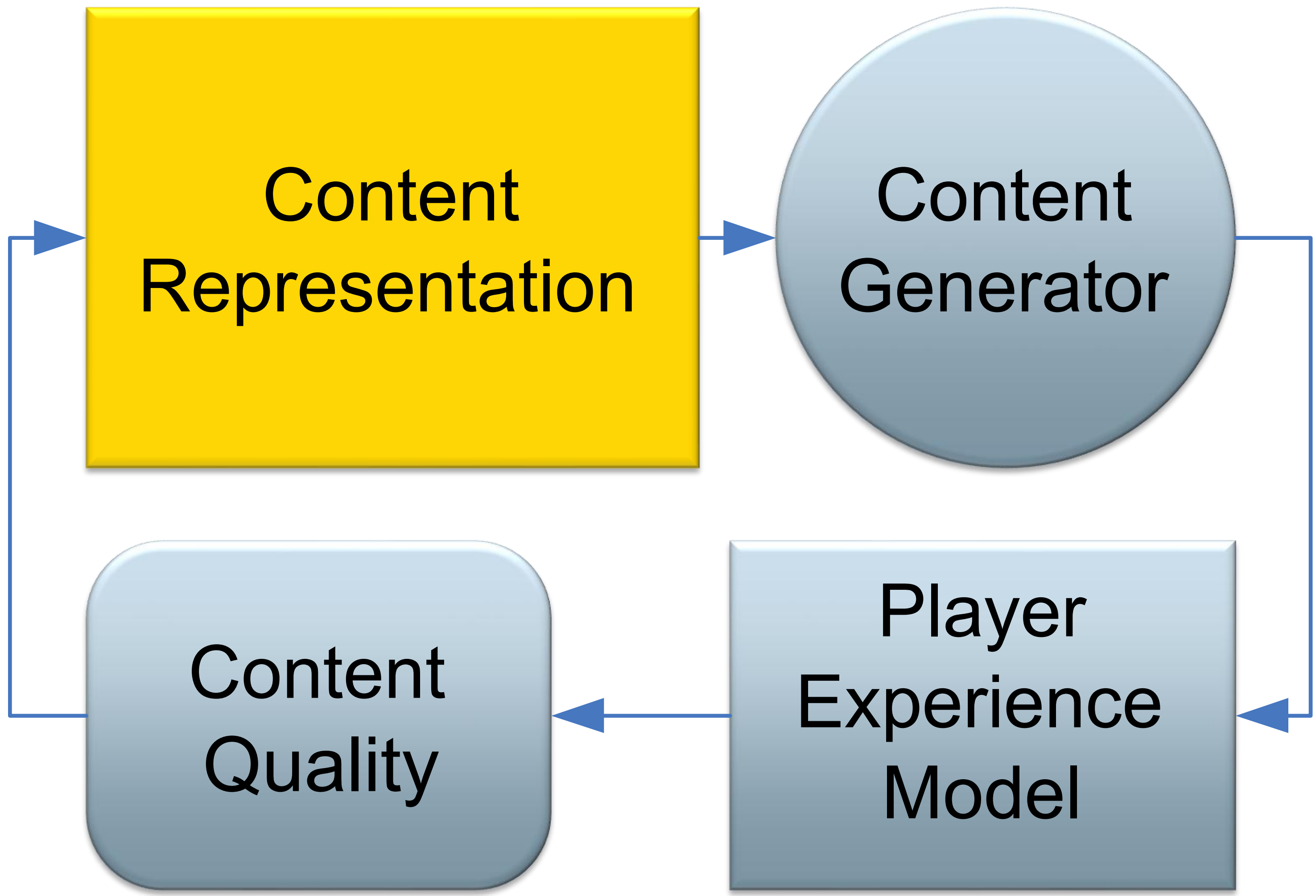
Player Experience Model		
Subjective	**Objective**	**GamePlay-Based**
Free-Response	**Model-based** (e.g. arousal-valence dimensions of emotion)	**Model-based** (e.g. OCC, BDI)
Forced Rating vs. Preference	**Model-free** (e.g. facial expression annotation)	**Model-free** (e.g. player modeling)

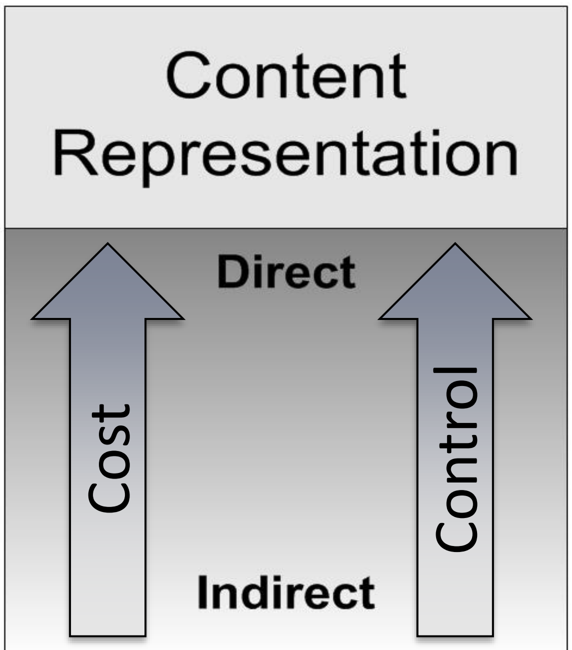

Content Representation
Content Generator
Player Experience Model
Content Quality

Content Quality

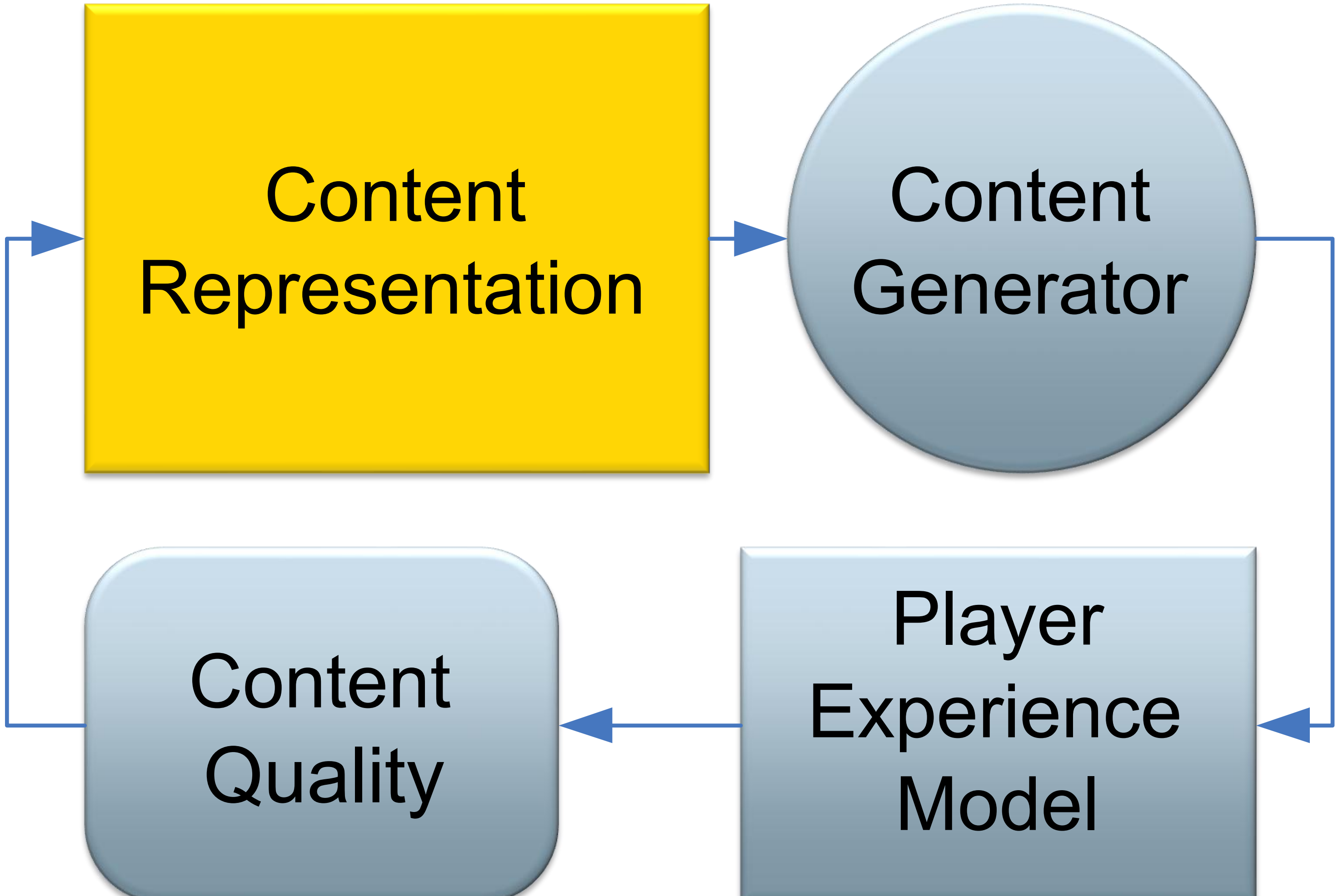
Direct	**Simulation-based**	**Interactive**
Theory-driven vs. Data-driven	Static vs. Dynamic	Implicit vs. Explicit

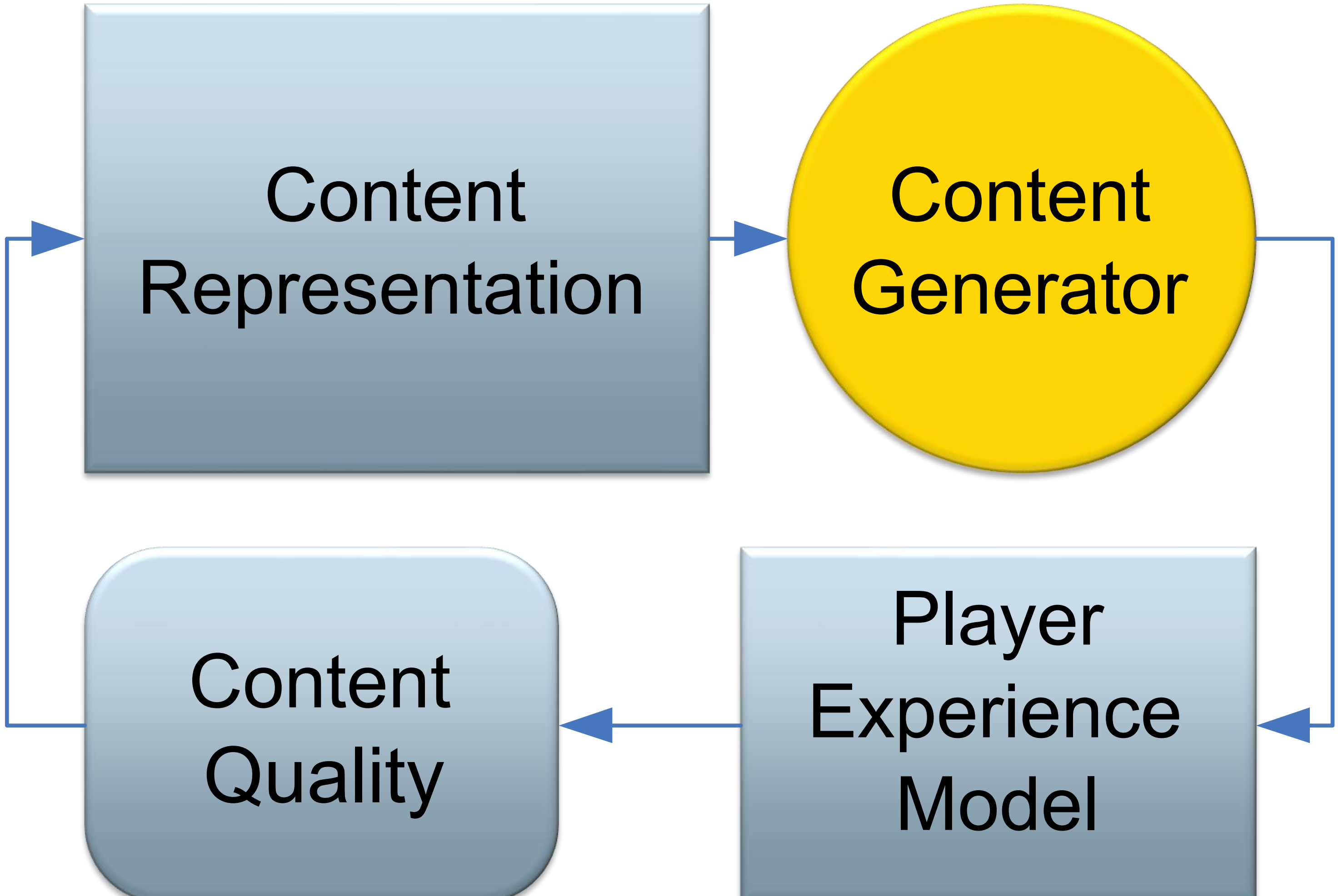

Content Representation
Content Generator
Player Experience Model
Content Quality

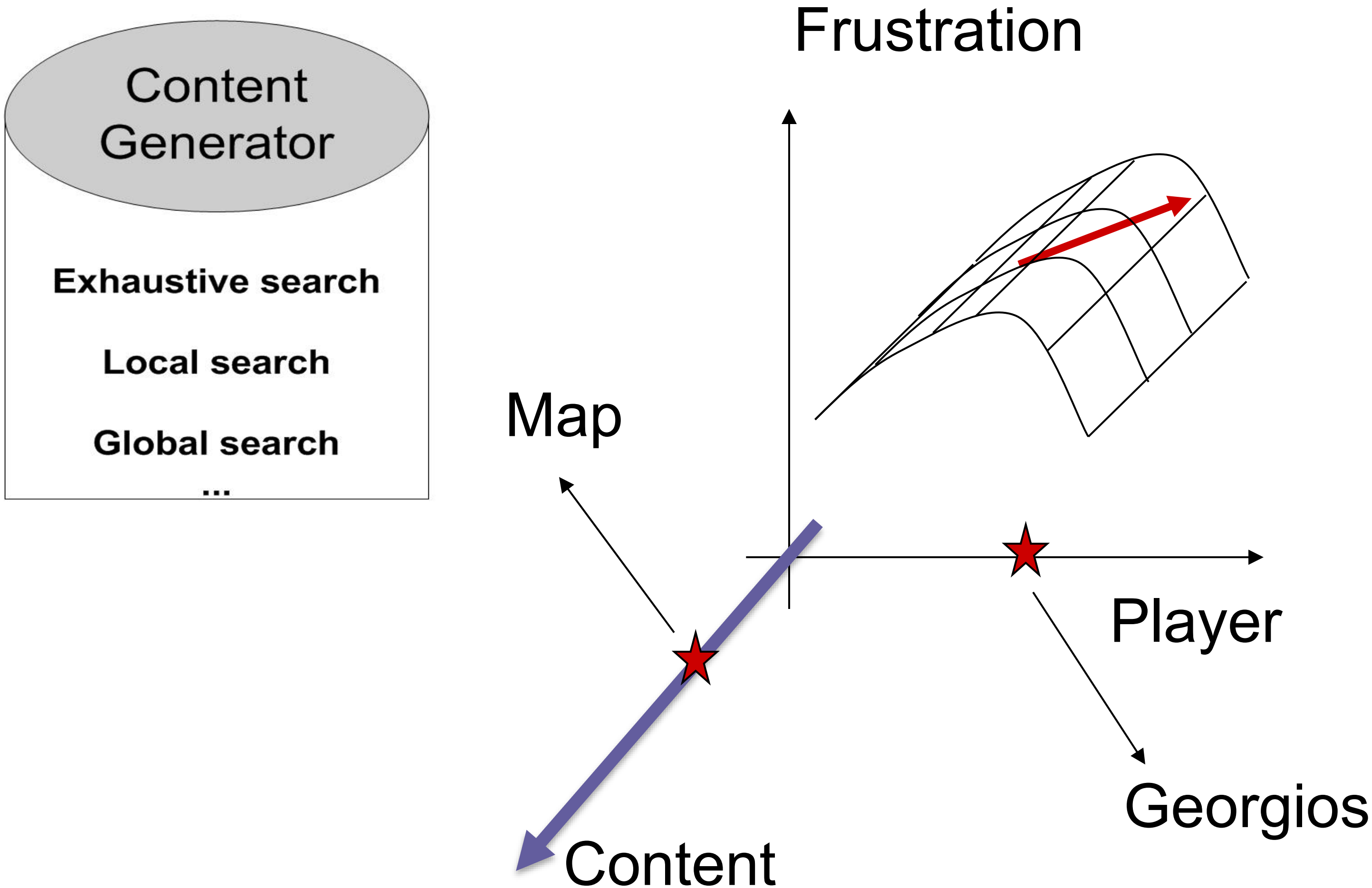

Content Representation
Content Generator
Player Experience Model
Content Quality



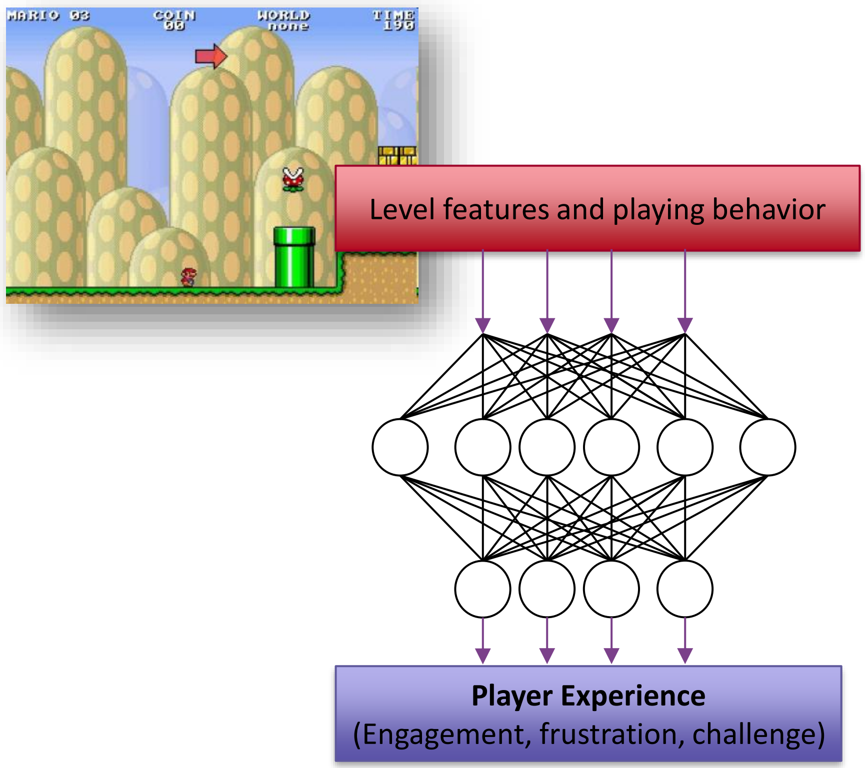
→ Grid
→ Position and orientation of walls
→ Patterns of walls and floor
→ Number of rooms and doors
→ Random seed


Content Representation
Content Generator
Player Experience Model
Content Quality


Content Representation
Content Generator
Player Experience Model
Content Quality

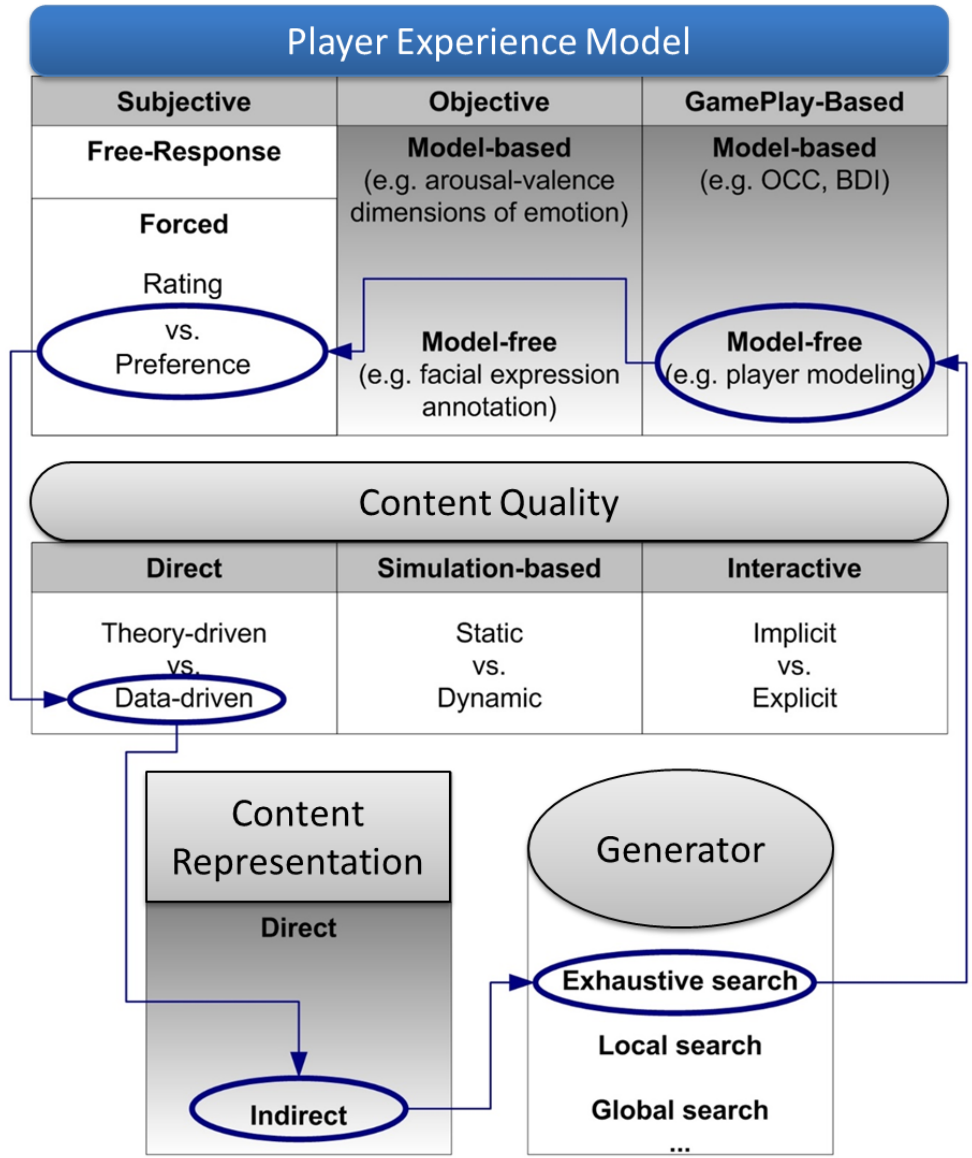

Content
Generator
Exhaustive search
Local search
Global search
...
Frustration
Map
Player
Georgios
Content

A Super Mario Bros Example



- **Model-free,** gameplay-based PEM (pairwise preferences as outputs)
- **Direct** (data-driven) evaluation function
- **Indirect** content representation (a few parameters)
- Search for good content via **exhaustive search**!

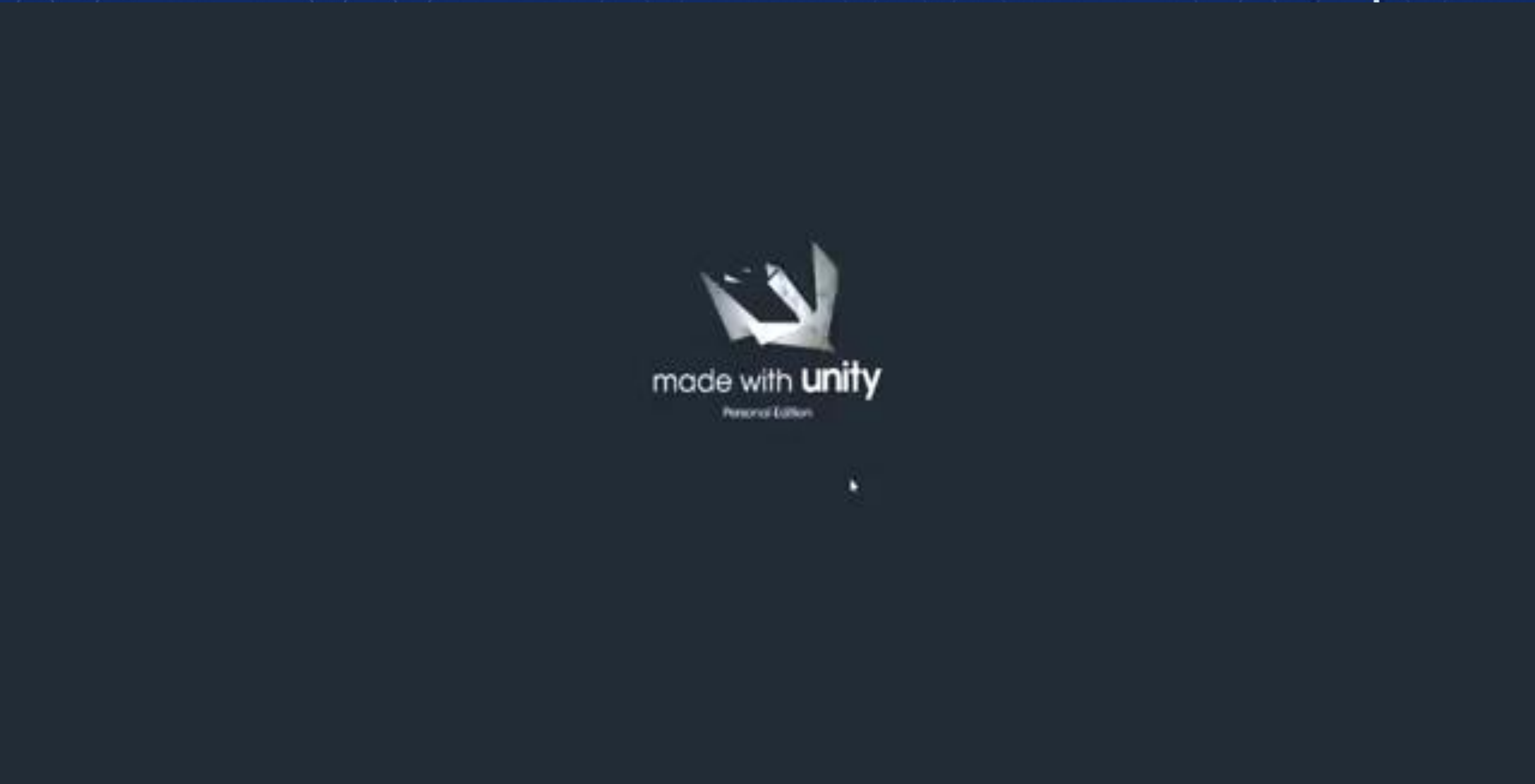
Shaker, N., Yannakakis, G. N., & Togelius, J. (2010, October). Towards Automatic Personalized Content Generation for Platform Games. In *AIIDE*.


Player Experience Model
Subjective
Objective
GamePlay-Based
Free-Response
Forced
Rating
vs.
Preference
Model-based
(e.g. arousal-valence dimensions of emotion)
Model-free
(e.g. facial expression annotation)
Model-based
(e.g. OCC, BDI)
Model-free
(e.g. player modeling)
Content Quality
Direct
Simulation-based
Interactive
Theory-driven
vs.
Data-driven
Static
vs.
Dynamic
Implicit
vs.
Explicit
Content Representation
Direct
Indirect
Generator
Exhaustive search
Local search
Global search
...

Other EDPCG Examples

Sonancia

Lopes, Liapis, and Yannakakis: **"Sonancia: Sonification of Procedurally Generated Game Levels,"** in Proceedings of the ICCC workshop on Computational Creativity & Games, 2015

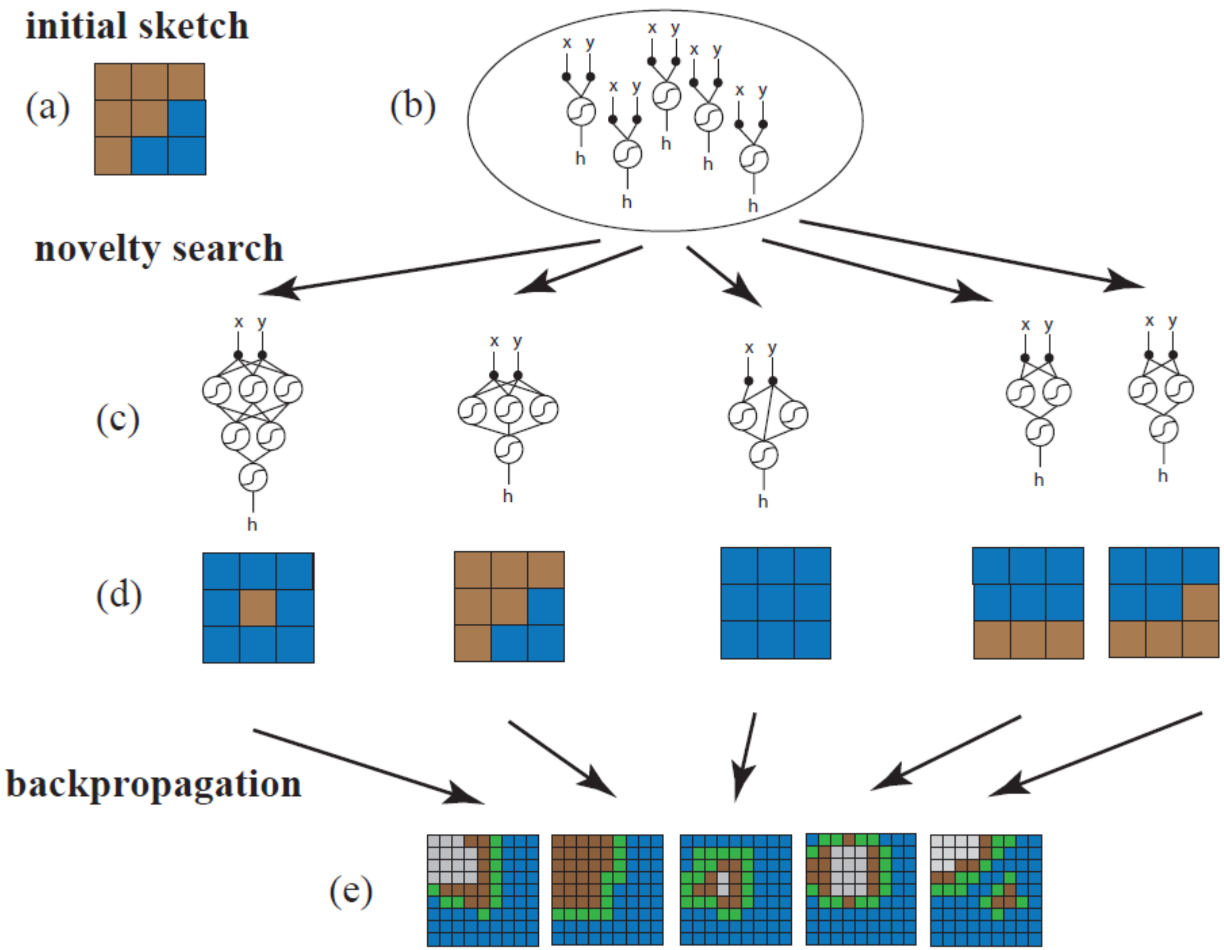


EDPCG for Serious Games: *Village Voices*

Khaled and Yannakakis "Village Voices: An adaptive game for conflict resolution", in Proc. of FDG, pp. 425-426

Sentient World:
Hybridizing Evolution and Gradient Search

Liapis et al. **"Sentient World: Human-Based Procedural Cartography,"** *EvoMusArt*, 2013.

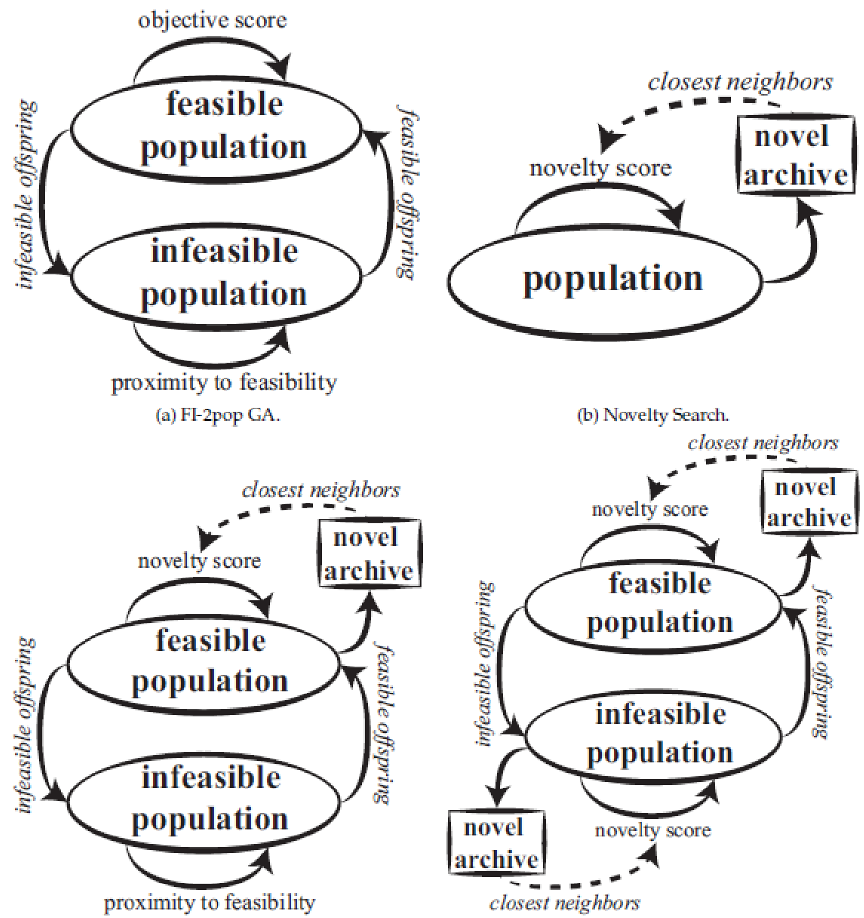


Sentient World:
Hybridizing Evolution and Gradient Search

Liapis et al. **"Sentient World: Human-Based Procedural Cartography,"** *EvoMusArt*, 2013.

Constrained Novelty Search for PCG

Liapis, Yannakakis and Togelius, **Constrained Novelty Search: A Study on Game Content Generation**, Evolutionary Computation, 21(1), 2015, pp. 101-129



(a) FI-2pop GA.

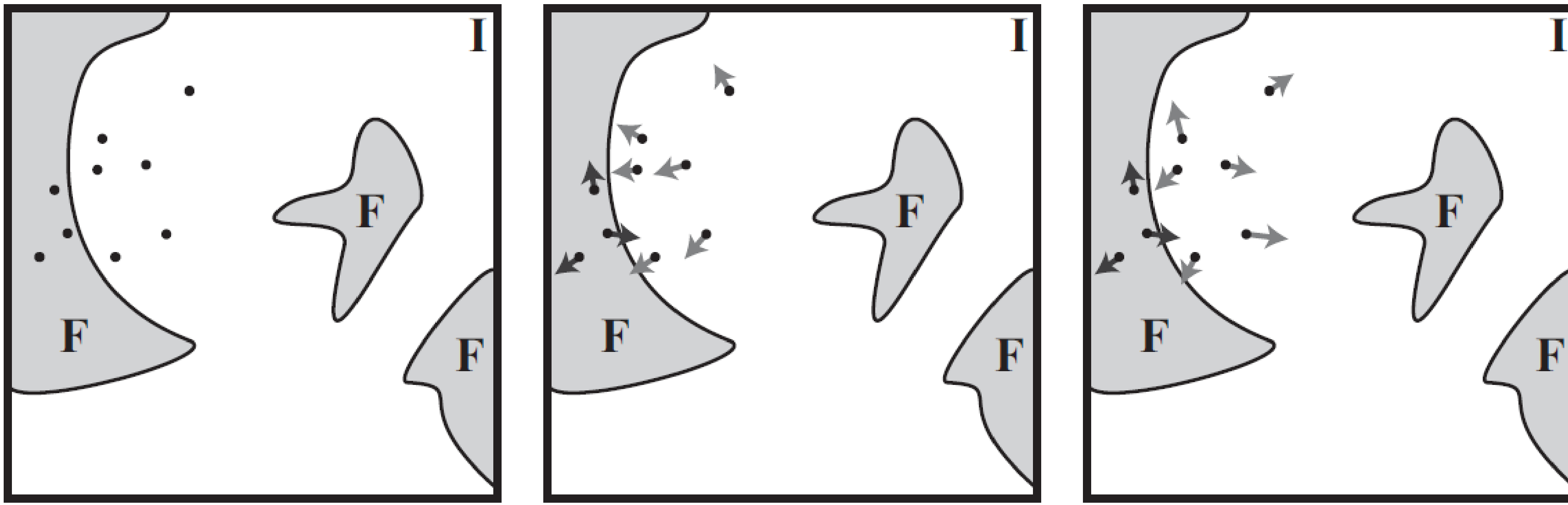
(b) Novelty Search.

(c) FINS.

(d) FI2NS.

Constrained Novelty Search for PCG

Liapis, Yannakakis and Togelius, **Constrained Novelty Search: A Study on Game Content Generation**, Evolutionary Computation, 21(1), 2015, pp. 101-129

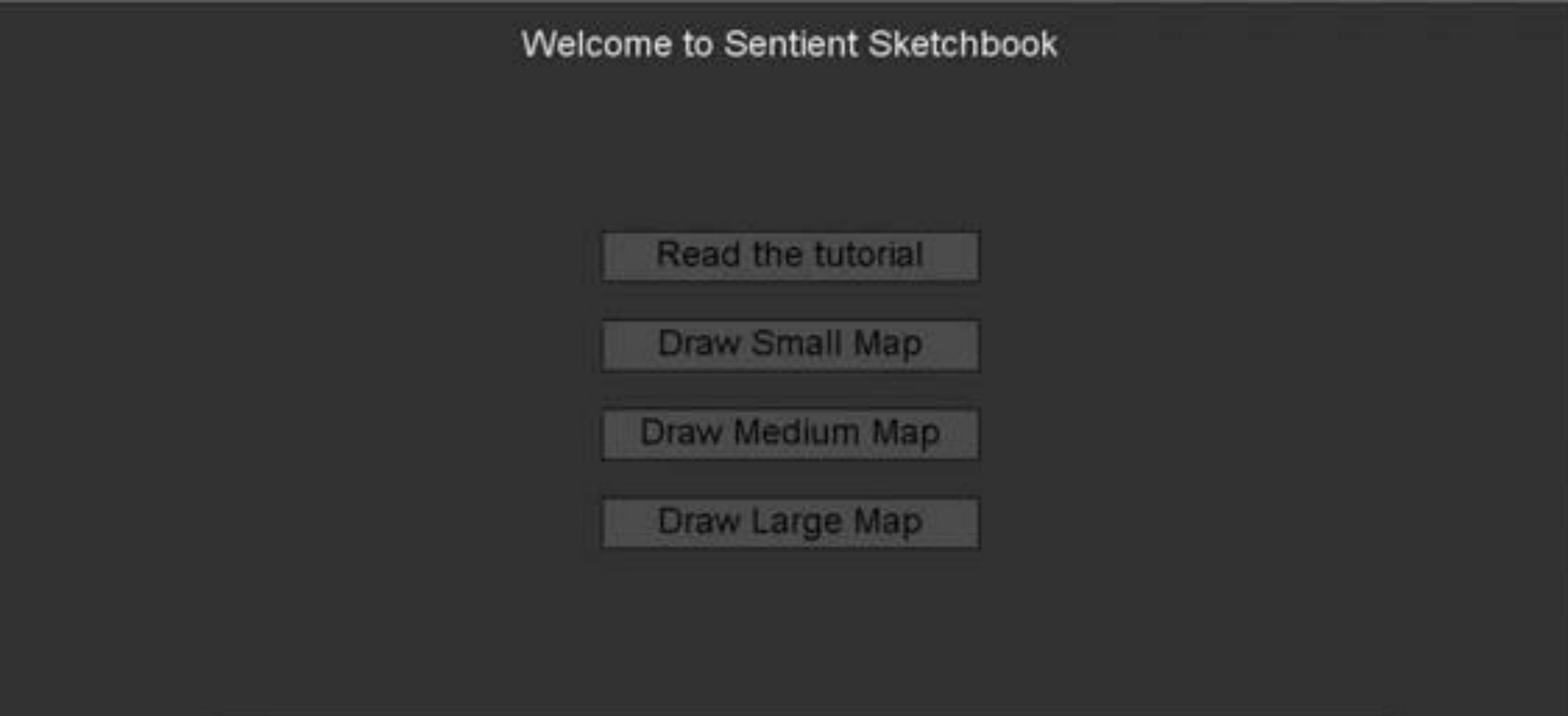


(a) Feasible and Infeasible Spaces. (b) FINS search process. (c) FI2NS search process.

Sentient Sketchbook

Georgios N. Yannakakis, Antonios Liapis and Constantine Alexopoulos:
"Mixed-Initiative Co-Creativity," in Proc. of the ACM Conference on Foundations of Digital Games, 2014.

- Map Sketches (strategy game, dungeon, FPS level)
- Multiple solutions evolved & shown in real-time
- Fitnesses on area influence, exploration and balance… and novelty
- Constraints on playability handled with FI-2pop GA

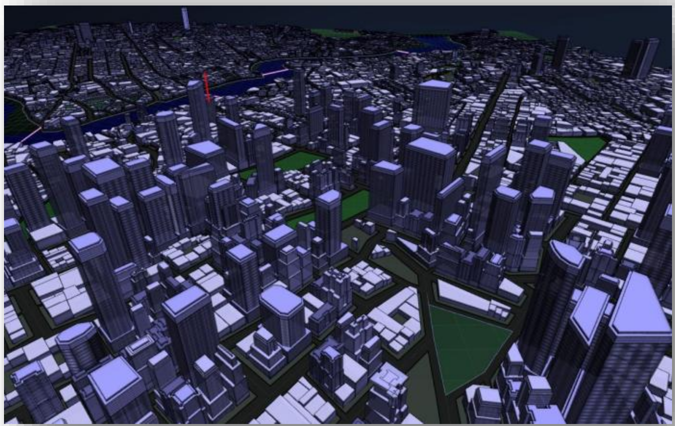


A user can select among a predefined set of map sizes.
Map size determines the number of allowed bases and resources.

Georgios N. Yannakakis, Antonios Liapis and Constantine Alexopoulos: **"Mixed-Initiative Co-Creativity,"** in Proc. of the ACM Conference on Foundations of Digital Games, 2014.

"Classical" PCG Approaches

- Mostly constructive, stochastic, non-controllable, random seed, online
- Mostly optional (non-vital) content such as decoration and redundant items
- Interesting examples from classic games:
  - Elite (deterministic, offline)
  - Rogue (online, crucial content)
- Techniques: L-trees, Fractals, …



Chapter 8: Methods For Generating Content

PCG Methods

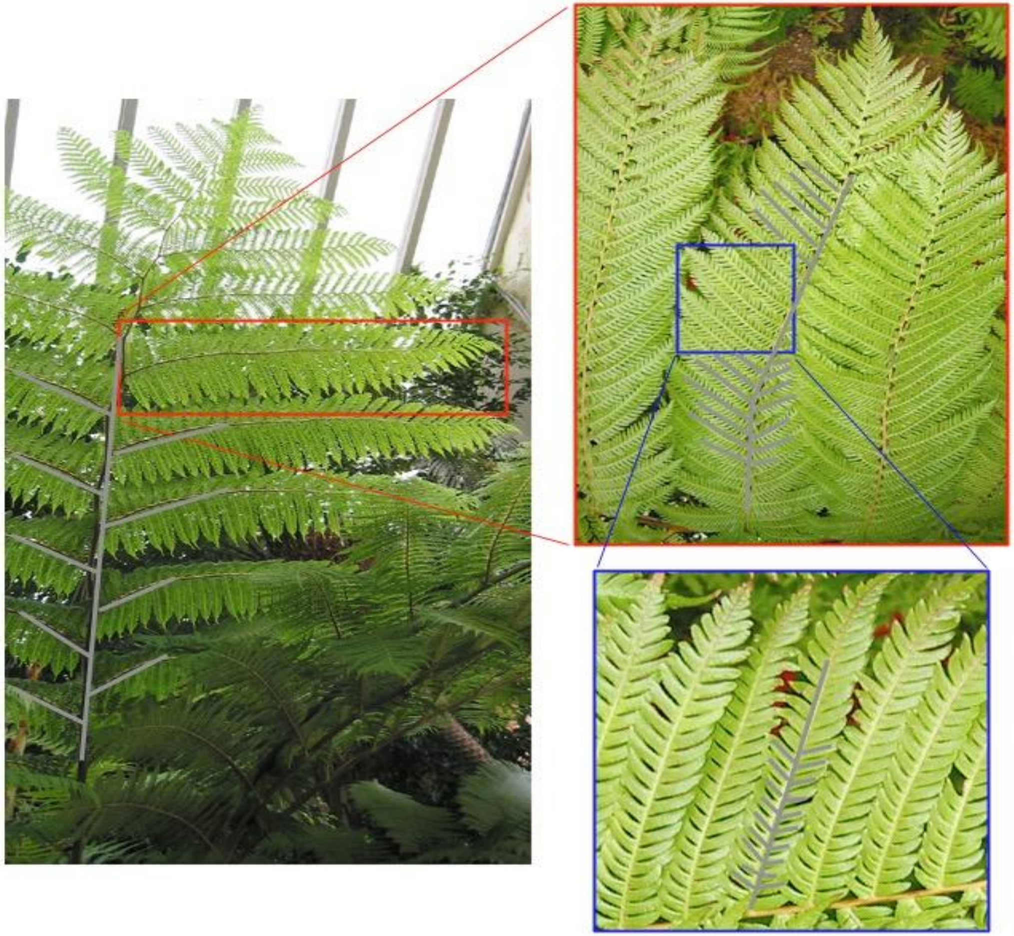
- Good old PCG:
  - Constructive methods (cellular automata, noise, fractals, templates, ad-hoc methods)
  - Solver-based
  - Grammar-based
  - Search-based
- “Generative AI”
  - Self-supervised learning on pixels/tiles
  - Self-supervised learning on strings (LLMs)
  - Reinforcement learning

Constructive Methods

Grammar-Based Methods

Grammar-Based Methods

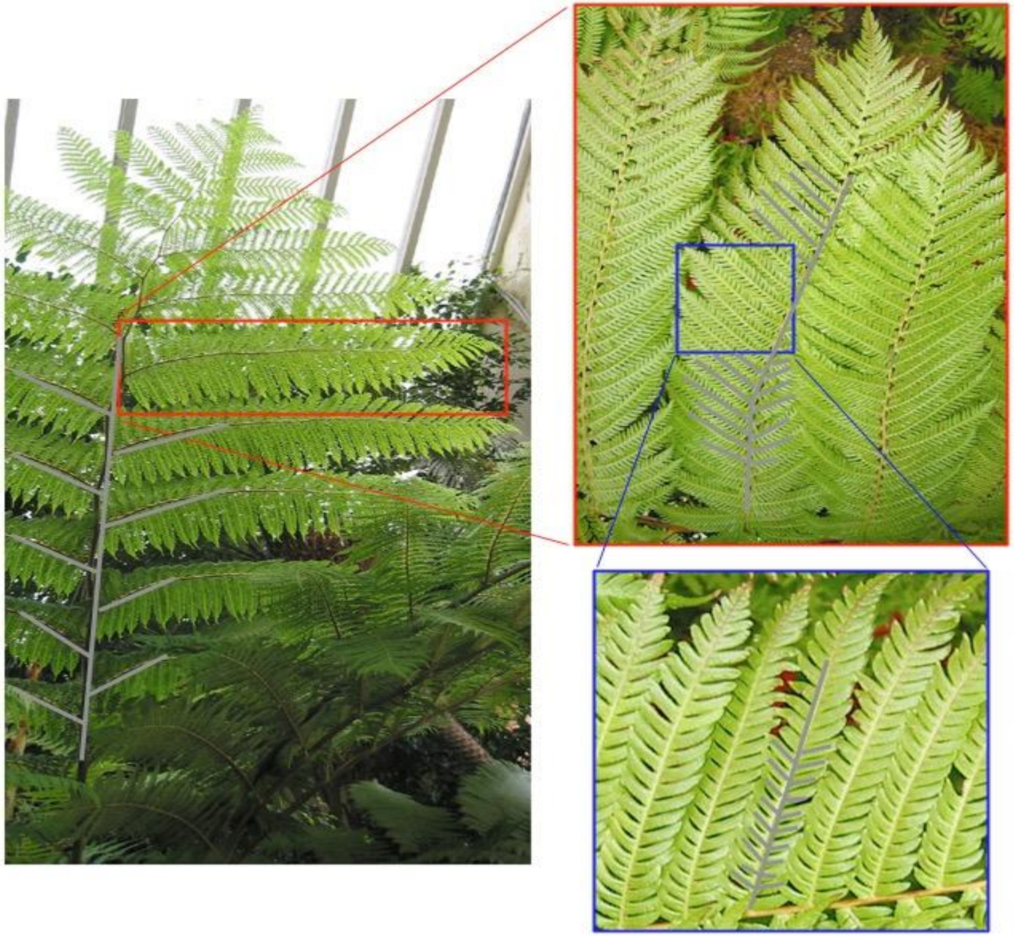
- Basic computer science concept, used in theory of computation
- Define a grammar and an alphabet and then watch an axiom unfold into ever more complex strings
- Commonly used for generating e.g. plants



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?

Self-Similarity



Self-Similarity

Grammars for Adventure Level Design

1. Dungeon → Obstacle + treasure
2. Obstacle → key + Obstacle + lock + Obstacle
3. Obstacle → monster + Obstacle
4. Obstacle → room

could give...

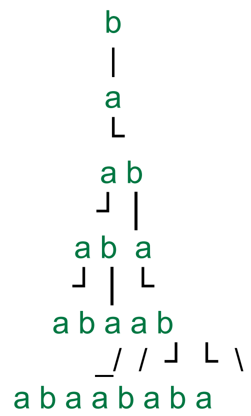
key + monster + room + lock + monster + room + treasure
key + monster + key + room + lock + monster + room + lock + room + treasure
room + treasure
monster + monster + monster + monster + room + treasure

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

Types of L-systems

Context

- **Context-free**: production rules refer only to an individual symbol
- **Context-sensitive**: productions can depend on the symbol's neighbors

Determinism

- **Deterministic**: there is exactly one production for each symbol
- **Non-deterministic**: several productions for a symbol

A Graphical Interpretation of L-systems

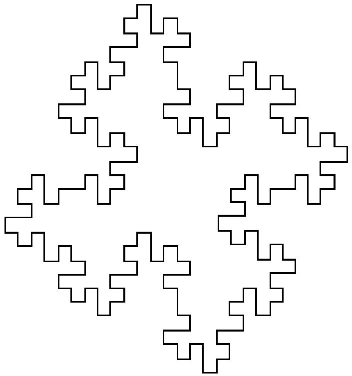
- Invented/popularized by Prusinkiewicz in 1986
- Core idea: interpret generated strings as instructions for a turtle in **turtle graphics**
- Read the string from left to right, changing the state of the turtle (x, y, heading)

Example Graphical Systems

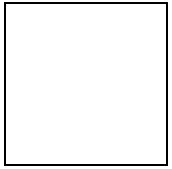
- Alphabet: {F, f, +, -}
- F: move the turtle forward (drawing a line)
- f: move the turtle forward (don’t draw)
- +/-: turn right/left (by some angle)

Graphical L-system

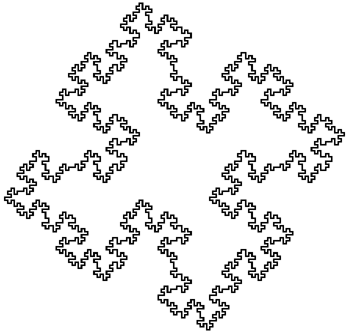
- Axiom: F+F+F+F
- Grammar: F>F+F-F-FF+F+F-F
- Turning angle: 90°



n=1



n=0



n=2

Graphical L-systems

What's the limit of these systems?

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!

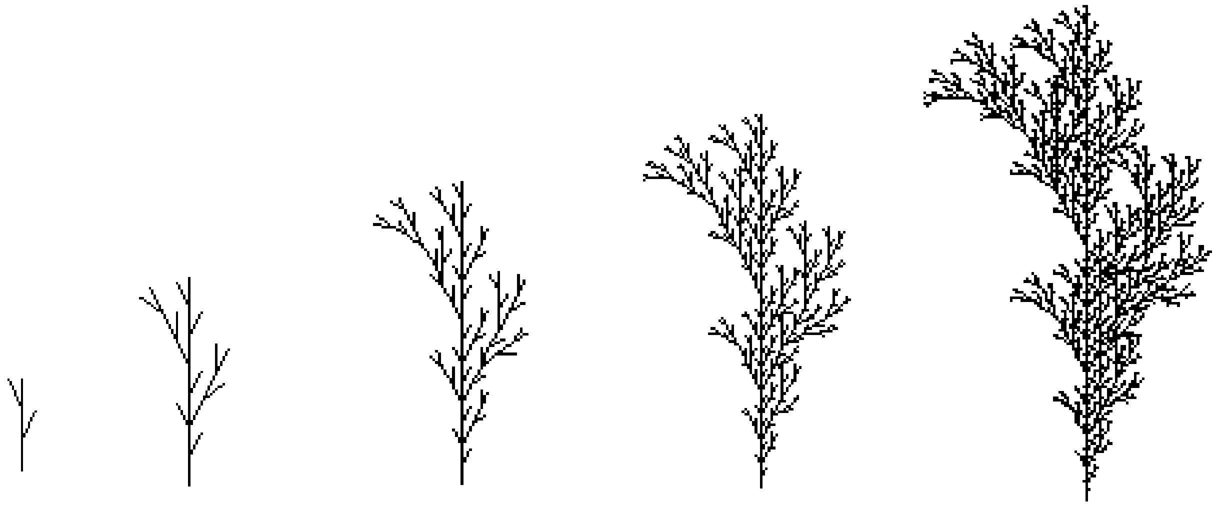
Bracketed L-systems

Axiom: F

Grammar: F>F[-F]F[+F][F]

Turning angle: 30°

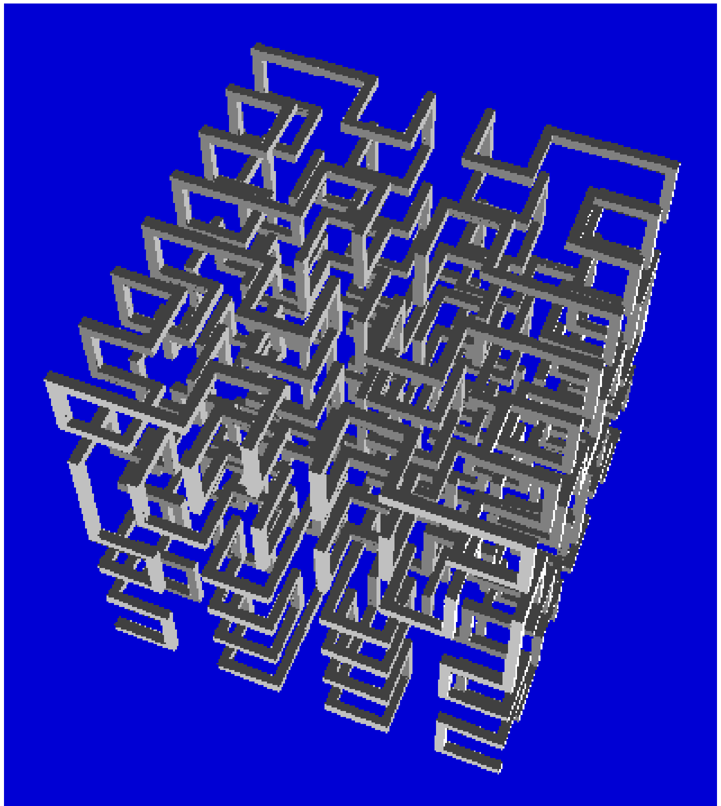
n=1,…,5

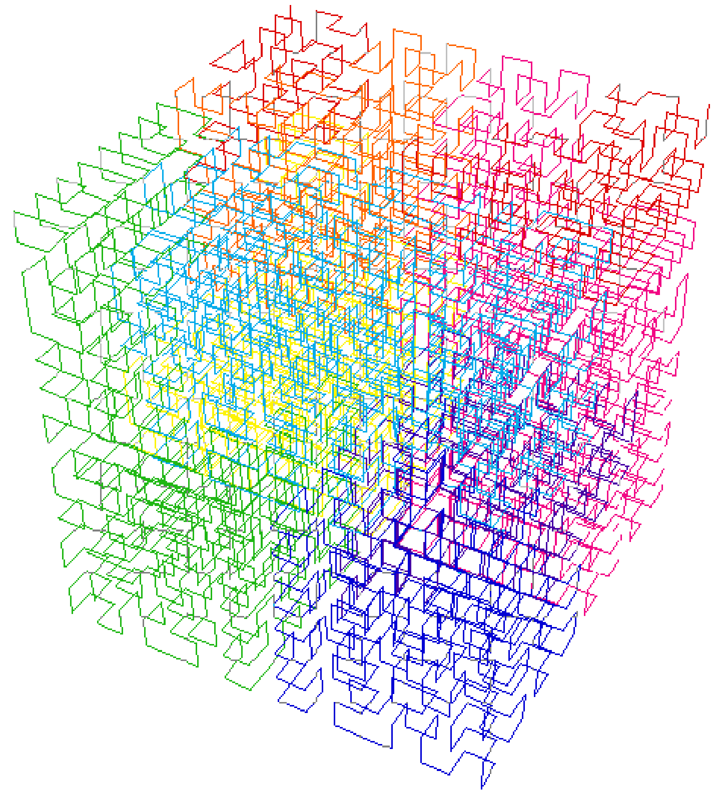


3D Graphics

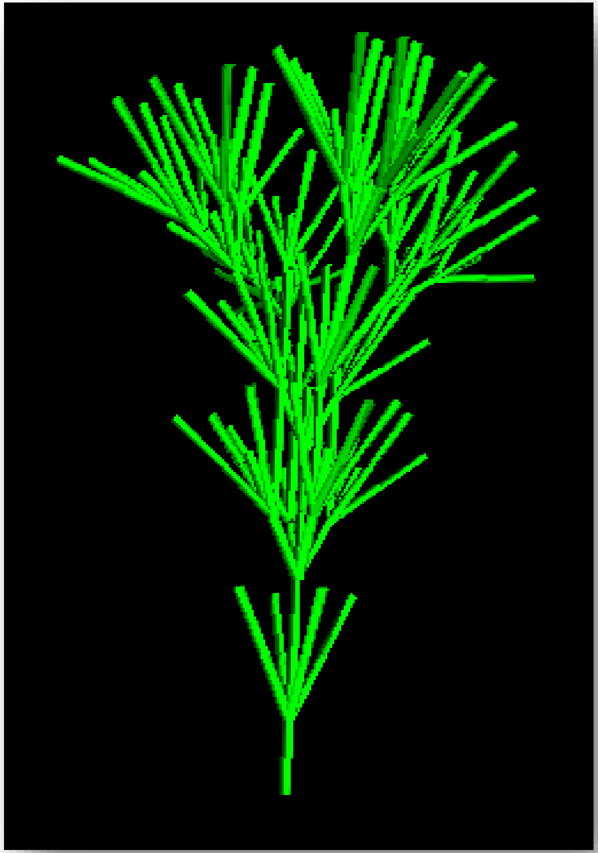
- Turtle graphics L-system interpretation can be extended to 3D space:
- Represent state as x, y, z and pitch, roll, yaw
- +, -: turn (yaw) left/right
- &, ^: pitch down/up
- \, /: roll left/right (counterclockwise/clockwise)

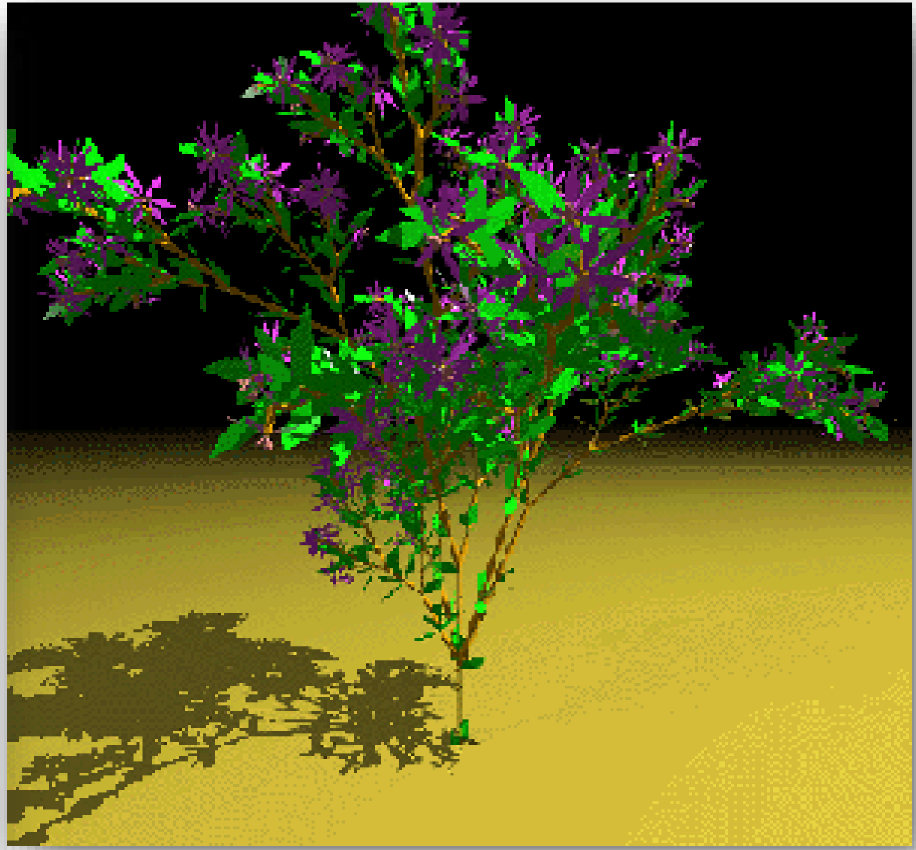
3D Interpretation of L-systems





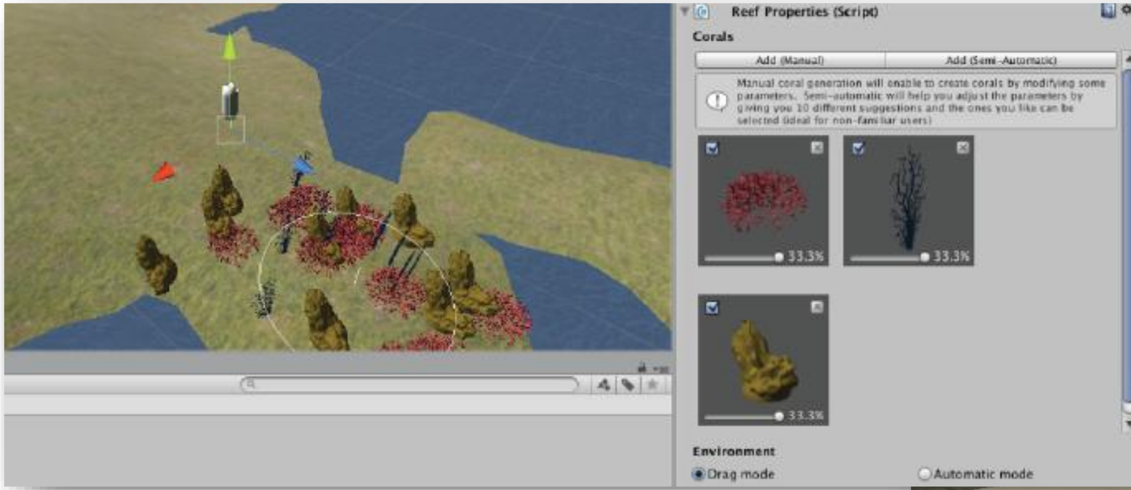
3D Interpretation of Bracketed L-systems



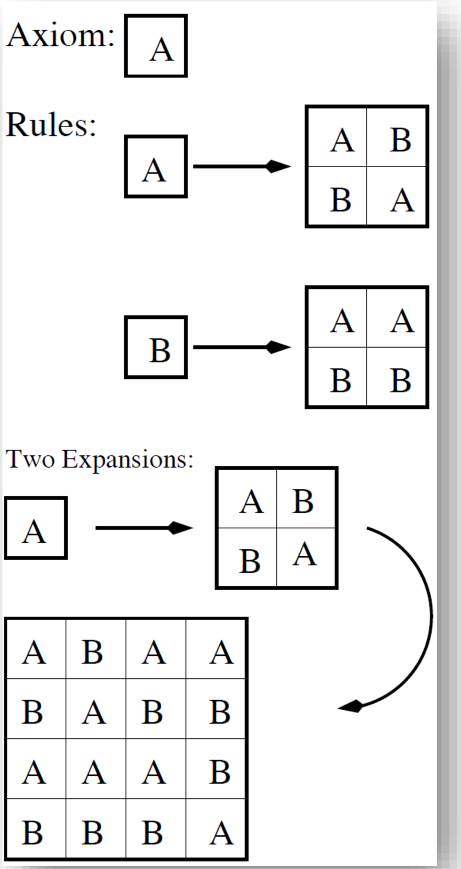


Coralize: 3D Corals in Unity

Abela, R., Liapis, A. and Yannakakis, G.N., 2015. **A constructive approach for the generation of underwater environments**. In Proceedings of the FDG workshop on Procedural Content Generation in Games.



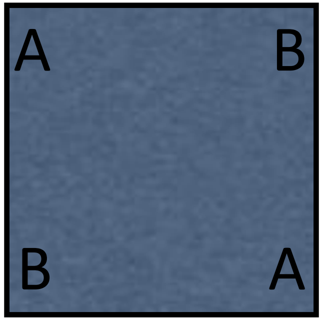
2D L-systems



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

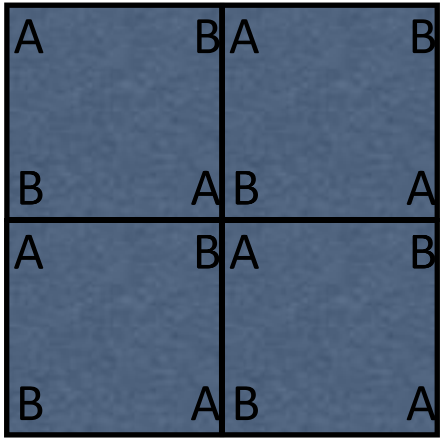
Terrain Interpretation of 2D L-systems

- 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

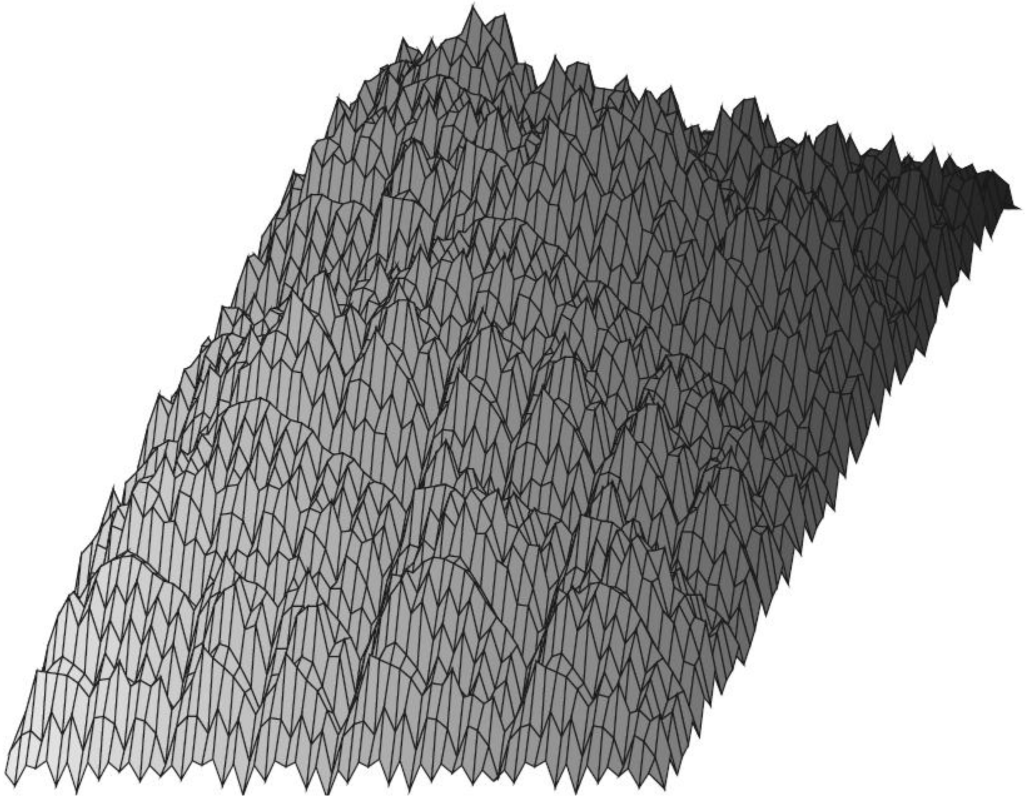


Terrain Interpretation of 2D L-systems

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



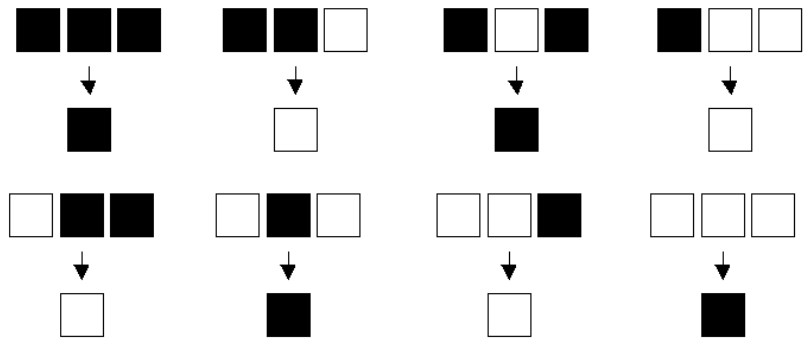
Terrain Interpretation of 2D L-systems



Six rewritings of A>ABBA, B>AABB

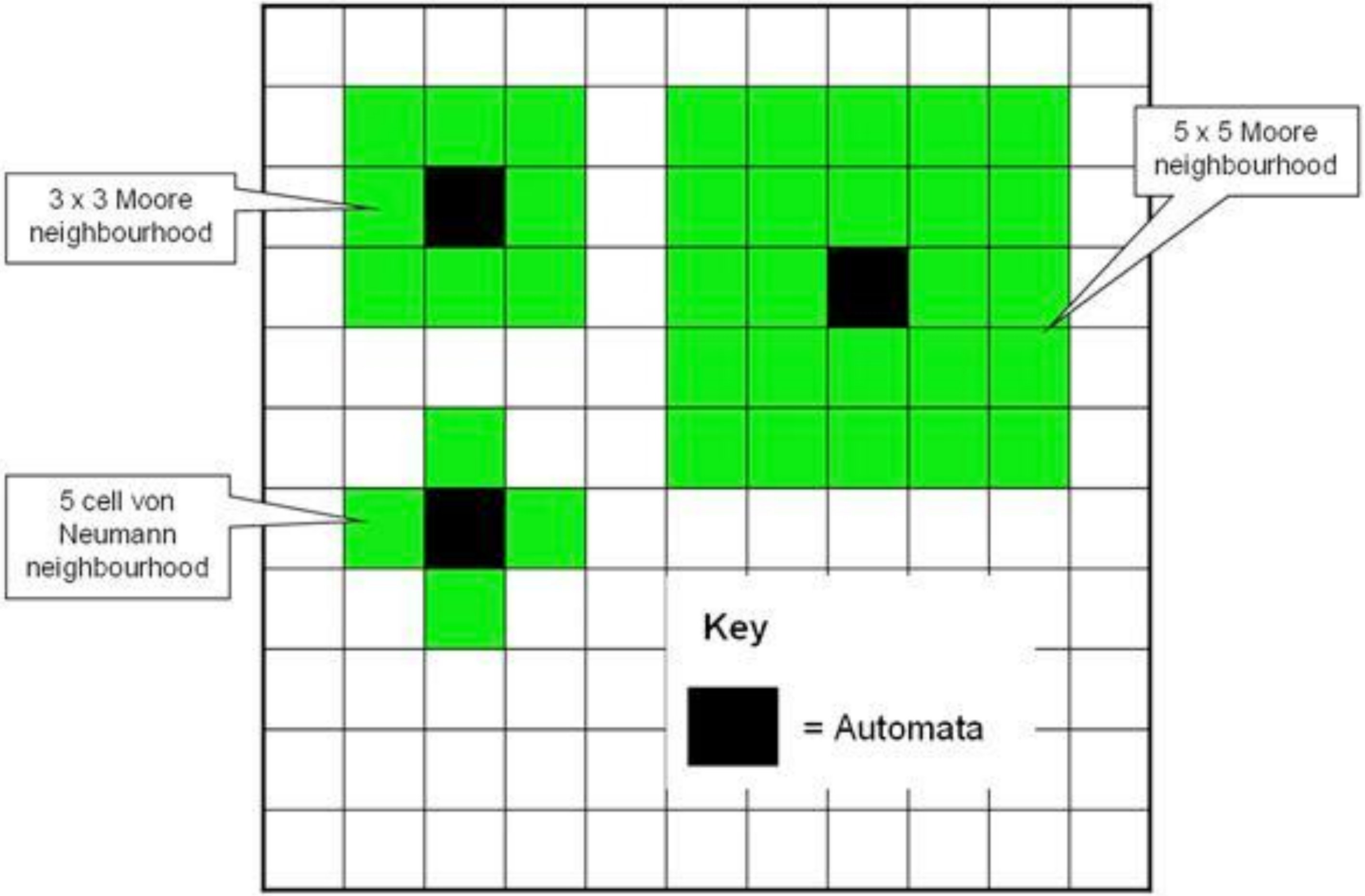
Cellular Automata

Cellular Automata



- Computational paradigm based on local interaction
- Used in artificial life and complexity studies
- The value of each cell in iteration *n+1* is based on the value of neighboring cells in iteration *n* and some rule

2D Cellular Automata



An Example

Cellular automata for real-time generation of infinite cave levels

Lawrence Johnson, Georgios N. Yannakakis and Julian Togelius

FDG PCG Workshop 2010

This…

A CA-based algorithm for generating infinite 2D caves

- simple
- real-time
- looks good
- *somewhat* controllable

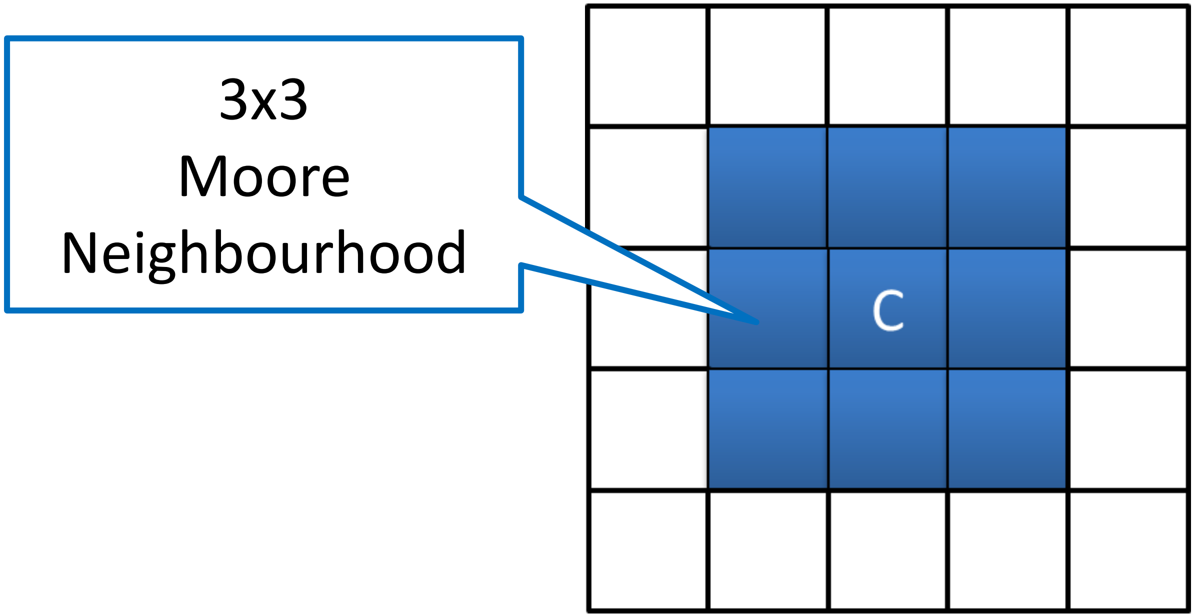
The Motivation

- *Cave Crawler* : a cooperative *abusive* dungeon crawler
- Never ends – therefore needs to produce infinite caves...

CA Cave Generation

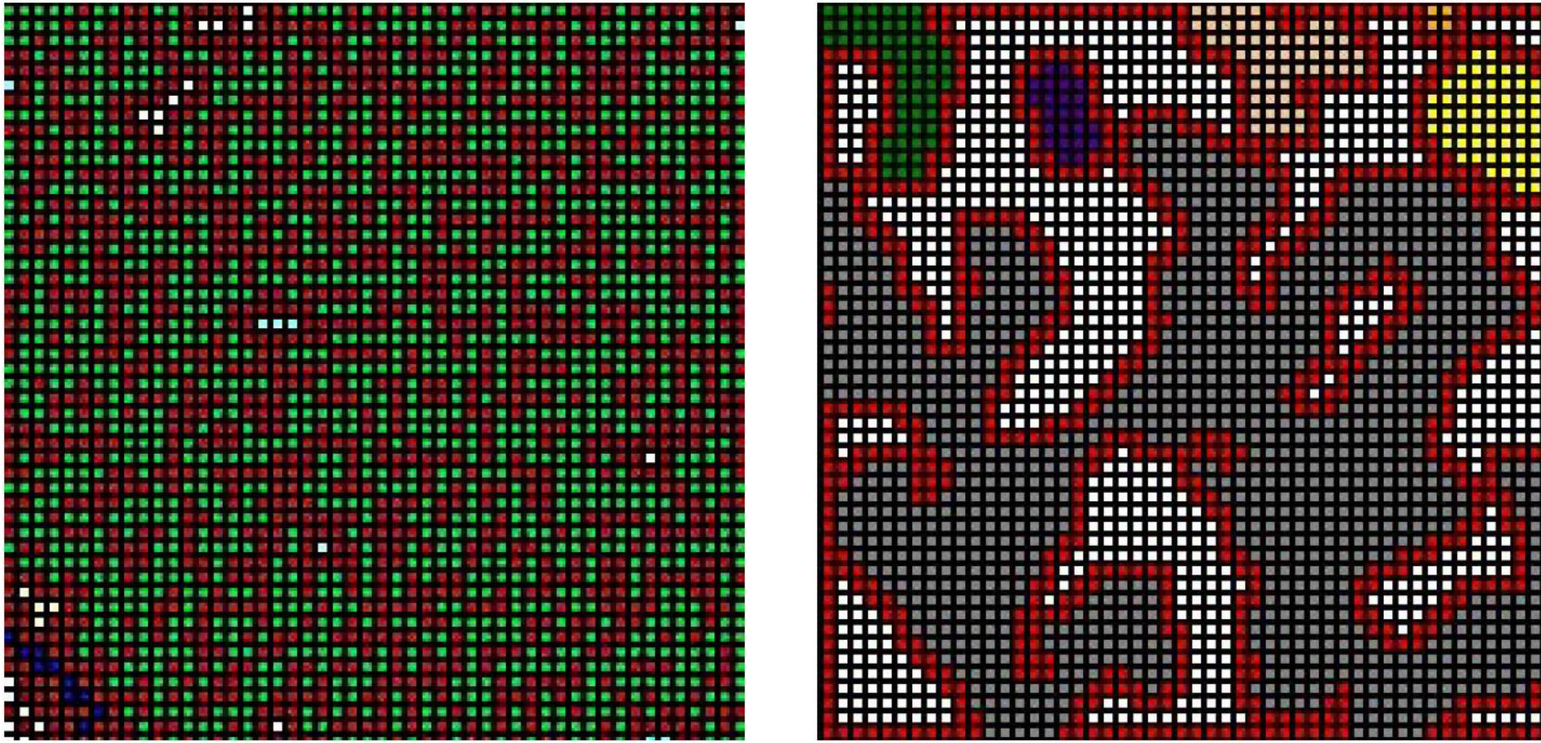
- Start with a square grid (e.g. 50*50) – all floor
- Randomly switch a proportion of cells from floor to rock
- Run a CA *n* times, where each cell is set to:
  - Rock: if at least *T* neighbors are rock
  - Floor: otherwise
- Fill in the interior of rock formations

Core CA Mechanic



CA Parameters

- *r* : initial proportion of rock cells (0.5)
- *n* : CA iterations (4)
- *T* : neighborhood value threshold that defines a rock (5)
- *M* : Moore neighbourhood size (1)



(a) A random map (b) A map generated with cellular automata

Fig. 8.3 Cave generation: Comparison between a CA and a randomly generated map. The CA parameters used are as follows: the CA runs for four generations; the size of the Moore neighborhood considered is 1; the threshold value for the CA rule is 5 ($T = 5$); and the percentage of rock cells at the beginning of the process is 50% (for both maps). Rock and wall cells are represented by white and red color respectively. Colored areas represent different tunnels (floor clusters). Images adapted from [505] with permission.

Adjacent Rooms

- The infinite cave needs to be contiguous - and you need to be able to turn back! (Visited rooms stored as random seeds)
- Generate all four neighbors of a new room
- Dig tunnels from the central room to the new rooms at the shortest points
- Run the CA *m* times (2) on *all five rooms together* to smooth out edges

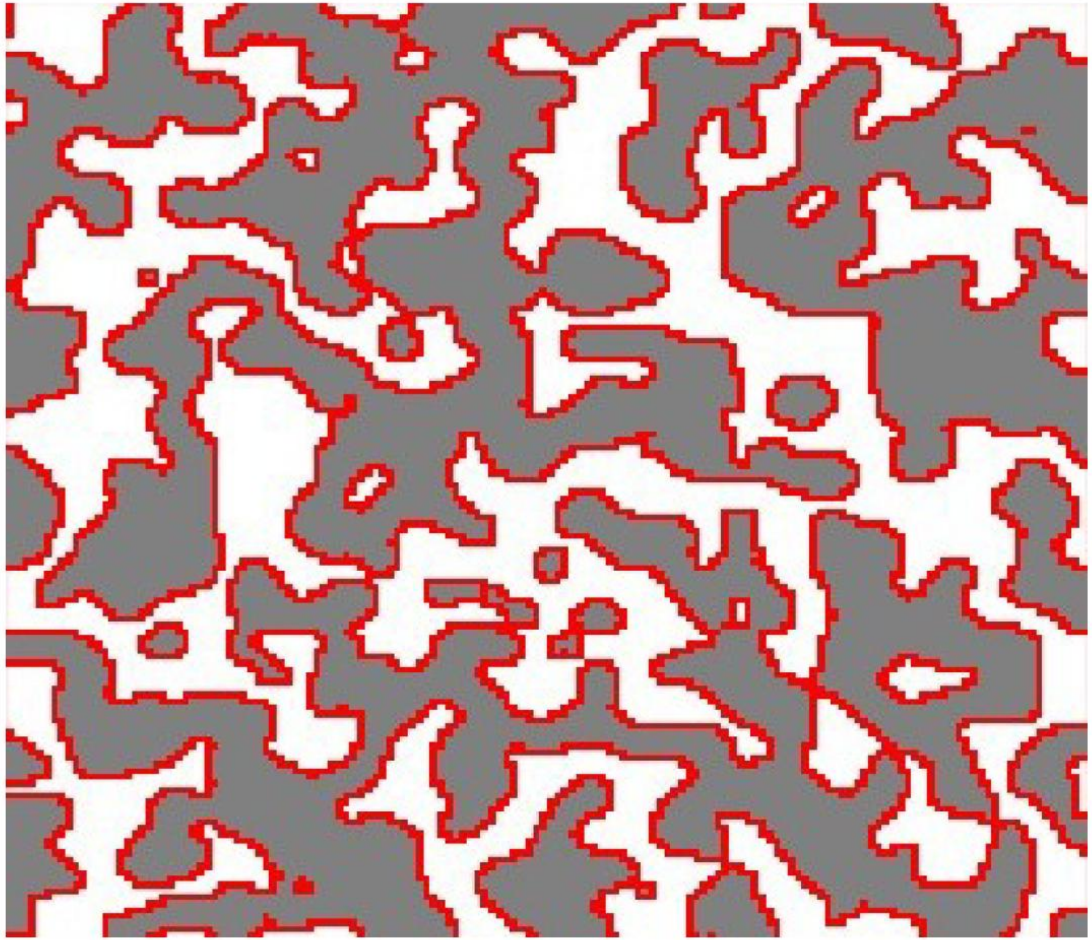
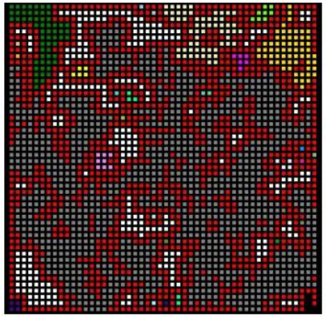


Fig. 8.4 Cave generation: a 3×3 base grid map generated with CA. Rock and wall cells are represented by white and red color respectively; gray areas represent floor. Moore neighborhood size is 2, T is 13, number of CA iterations is 4, and the percentage of rock cells at the initialization phase is 50%. Image adapted from [505].

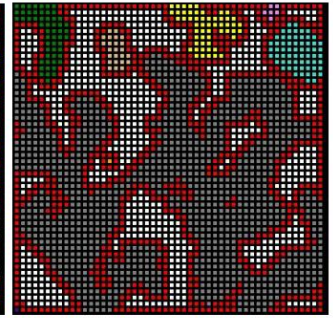
Controllable?

Parameters can be varied…

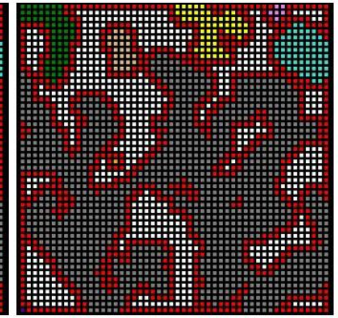
…but what do they mean?



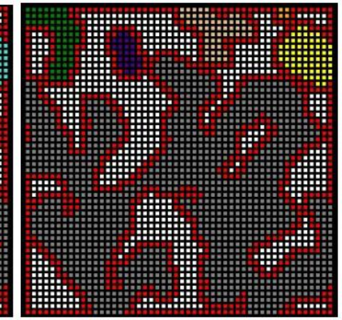
(a) $n = 1,\ M = 1,\ T = 5$



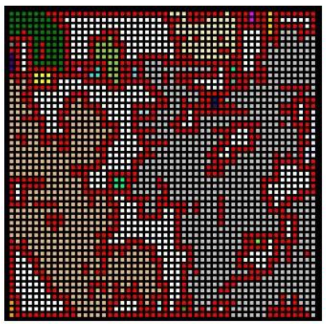
(b) $n = 2,\ M = 1,\ T = 5$



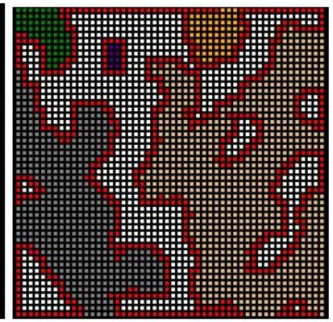
(c) $n = 3,\ M = 1,\ T = 5$



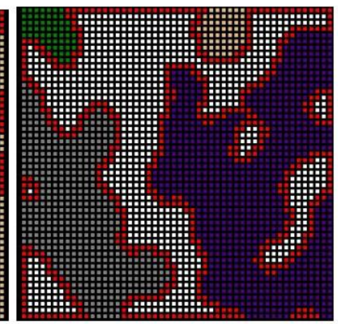
(d) $n = 4,\ M = 1,\ T = 5$



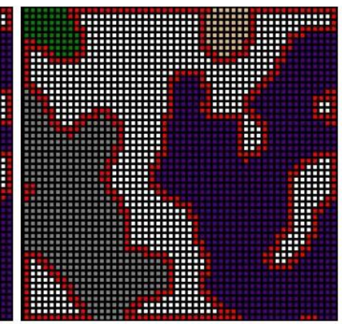
(e) $n = 1,\ M = 2,\ T = 13$



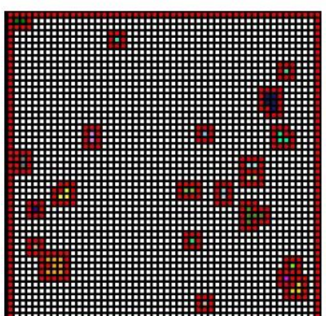
(f) $n = 2,\ M = 2,\ T = 13$



(g) $n = 3,\ M = 2,\ T = 13$

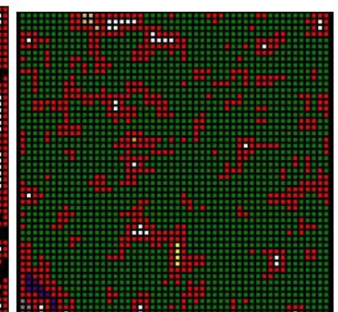


(h) $n = 4,\ M = 2,\ T = 13$

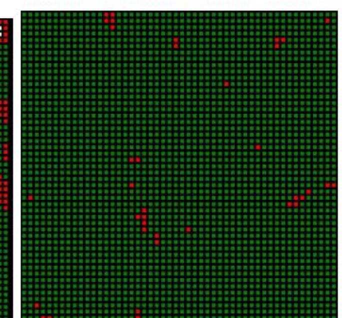


(i) $n = 1,\ M = 1,\ T = 2$

(j) $n = 1,\ M = 1,\ T = 4$

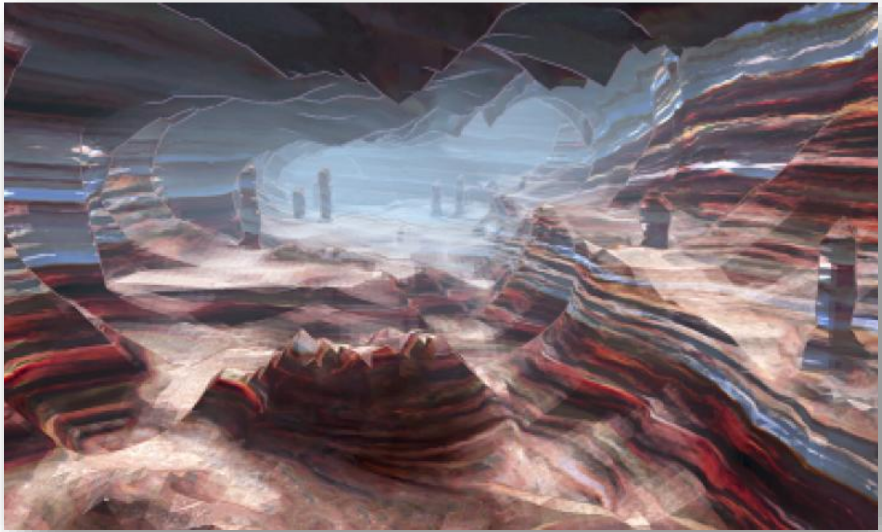


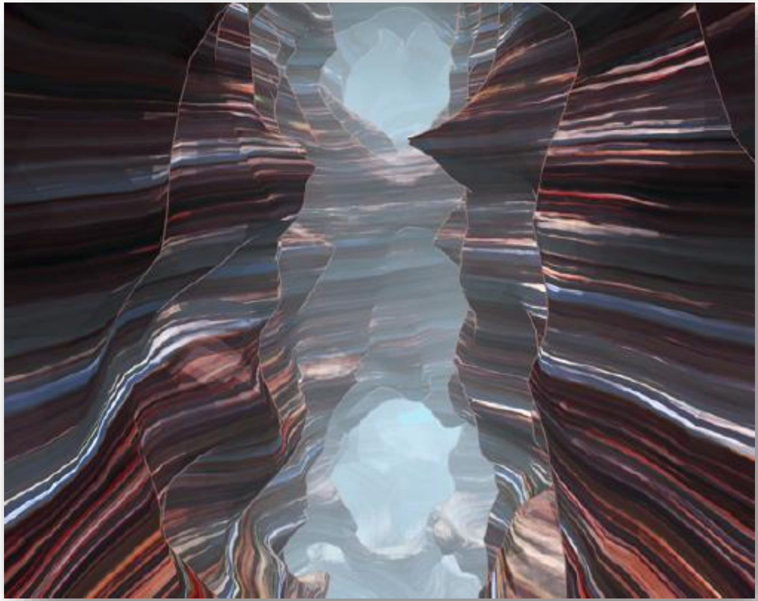
(k) $n = 1,\ M = 1,\ T = 6$



(l) $n = 1,\ M = 1,\ T = 8$

3D L-systems + CA

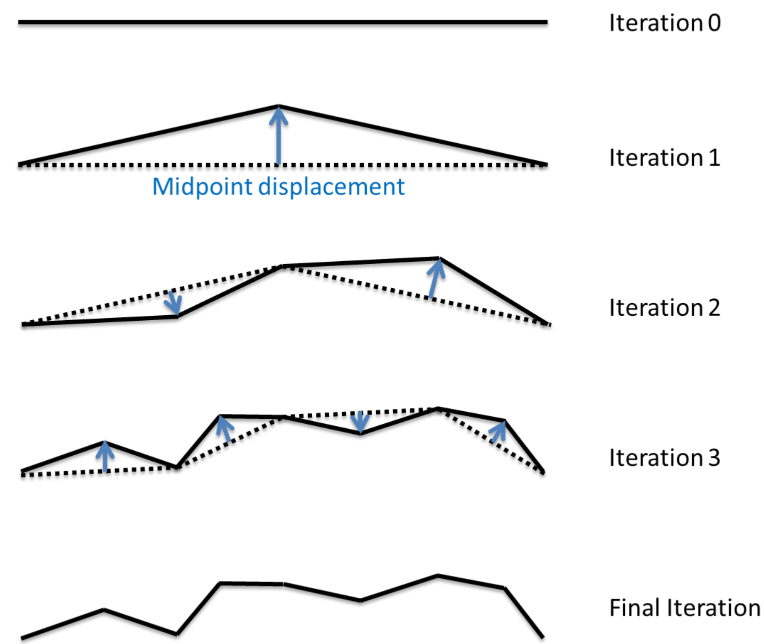




Noise and Fractals

Noise and Fractals

- **Noise algorithms** (often fractal) → fast, easy, scale-invariant; limited controllability.
- **Common representation**: 2D matrix of real numbers.
  - **Textures** → intensity maps (values = pixel brightness).
  - **Terrains** → heightmaps (values = elevation over baseline).
- **Interpolation**: fills in details when render resolution → heightmap resolution.
- **Voxel representation**: 3D grid (e.g., *Minecraft*) → allows caves, overhangs; needs more storage.
- **Midpoint displacement algorithm (figure):** generates landscapes via recursive subdivision & random displacement, reducing variation each step.



Noise and Fractals

Diamond-Square Algorithm

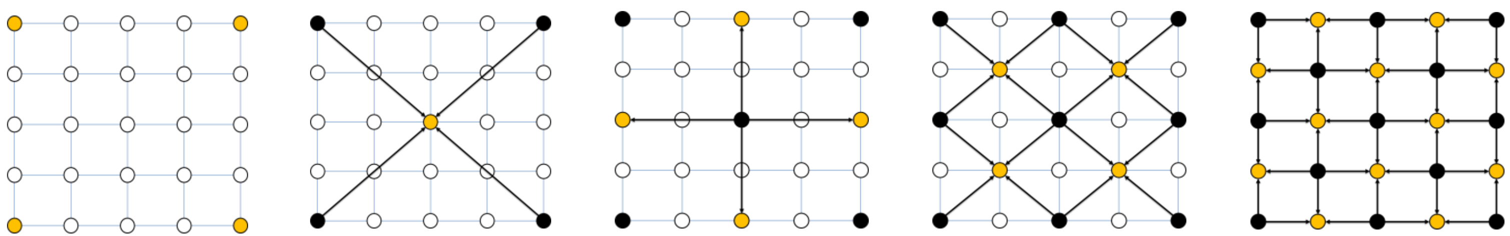
- Extension of **midpoint displacement** to 2D (aka cloud fractal or plasma fractal)
- Works on a **square matrix**

Initialization: all cells = 0; corners = random values (e.g., [−1,1]).

Algorithmic steps:

1. **Diamond step** – midpoint of four corners = average + random offset.
2. **Square step** – edge midpoints = average of neighbours + random offset.

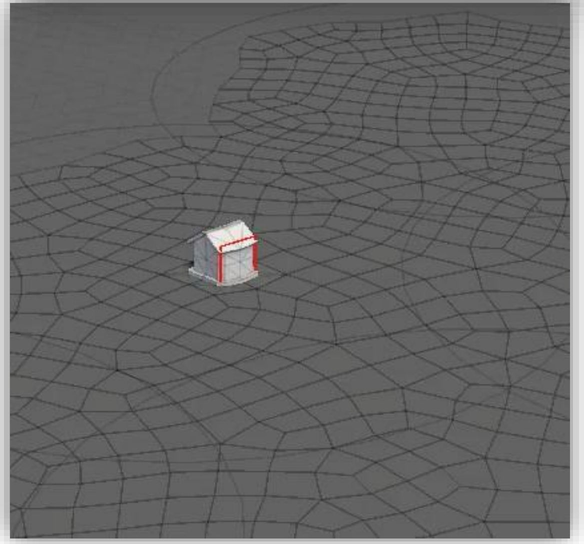
Recursion: repeat for 4 sub-squares until minimum size (3×3); reduce randomness each recursion for smoother terrain.



Wave Function Collapse

Wave Function Collapse

- Popular in the indie space
- A constraint-solving method that learns constraints from limited examples
- Alternative interpretation: self-supervised learning from small samples



Evolutionary Algorithms and Search-Based PCG

Togelius, J., Yannakakis, G.N., Stanley, K.O. and Browne, C., 2011. **Search-based procedural content generation: A taxonomy and survey**. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3), pp.172-186.

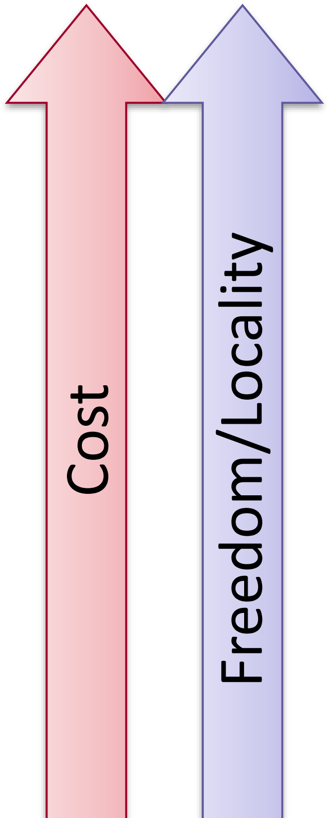
Search-Based PCG

- Use evolutionary computation to search the design space for good artifacts (e.g. levels)
  - Technically, we could use other stochastic search / optimization algorithms
- Major issues:
  - Representing the content
  - Devising a good evaluation / fitness function

The Algorithm

- Lots of different types of evolutionary algorithms: Genetic Algorithms, Evolution Strategies, Evolutionary Programming
- And evolution-like algorithms: Particle Swarm Optimization, Differential Evolution
- Keep It Simple, Stupid!
  - Often, simple $\mu+\lambda$ ES with no crossover and no self-adaptation works well enough

Representing Content (e.g. a dungeon)



- *Directly*: grid
- *More indirectly*: position and orientation of walls
- *Even more indirectly*: patterns of walls and floor
- *Very Indirectly*: number of rooms and doors
- *Indirectly*: random seed

How to Evaluate Content Quality

- **Directly**
  - A direct mapping between content and quality; e.g. number of jumps in a platform game
- **Simulation-based**
  - An AI (maybe a human imitator) plays the game for a while and content is evaluated
- **Interactively**
  - Real-time evaluation via a player (or players)

Togelius, J., Yannakakis, G. N., Stanley, K. O., & Browne, C. (2011). Search-based procedural content generation: A taxonomy and survey. *Computational Intelligence and AI in Games, IEEE Transactions on*, *3*(3), 172-186.

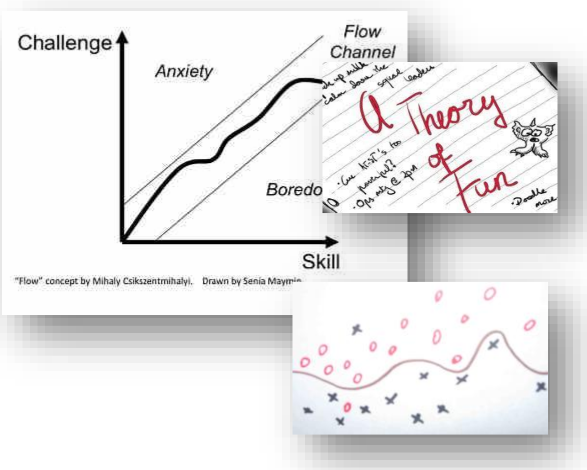
How to Evaluate Content Quality

- **Directly**
  - **Theory-driven**: evaluation function is based on a theoretical model – e.g. Koster's theory of fun
  - **Data-driven**: evaluation function is derived via gameplay (or other modalities of) data
- **Simulation-based**
  - **Static**: evaluation function does not change over time
  - **Dynamic**: evaluation function is affected as time goes by
- **Interactively**
  - **Implicit**: game behavior gives value to content (e.g. preference over a weapon)
  - **Explicit**: ask players to score content

Togelius, J., Yannakakis, G. N., Stanley, K. O., & Browne, C. (2011). Search-based procedural content generation: A taxonomy and survey. *Computational Intelligence and AI in Games, IEEE Transactions on*, *3*(3), 172-186.

Content Quality

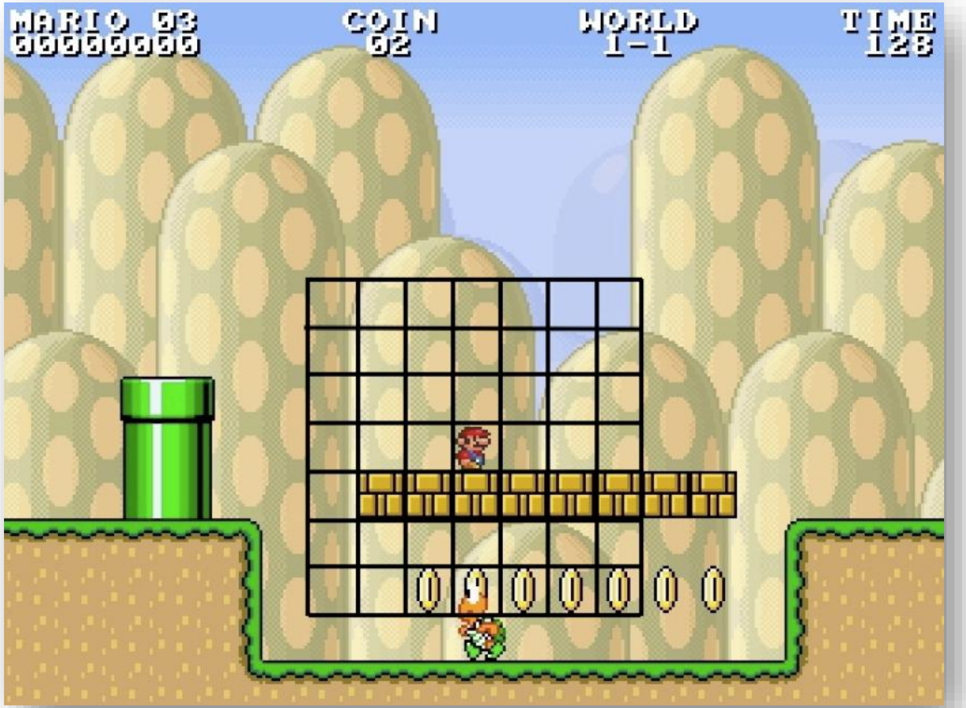
Direct	Simulation-based	Interactive



Search-Based PCG Example #1

How would we generate levels for Super Mario Bros?

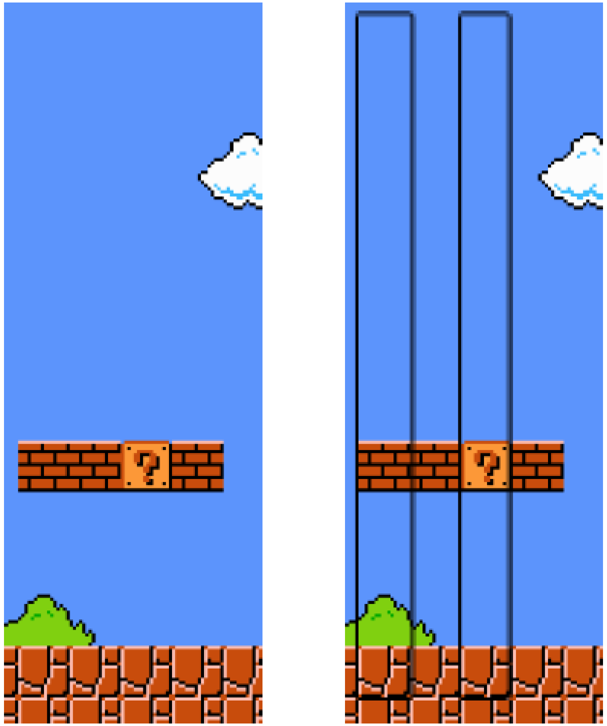
The Mario AI Benchmark



- Reasonably faithful clone of SMB 1/3
- APIs for level generators and AI controllers

Representation

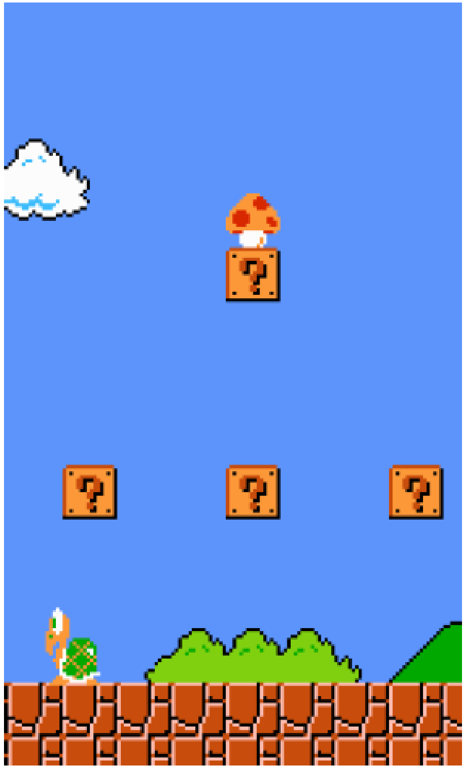
- A number of “vertical slices” are identified from the original SMB levels
- Levels are represented as strings, where each character correspond to a pattern



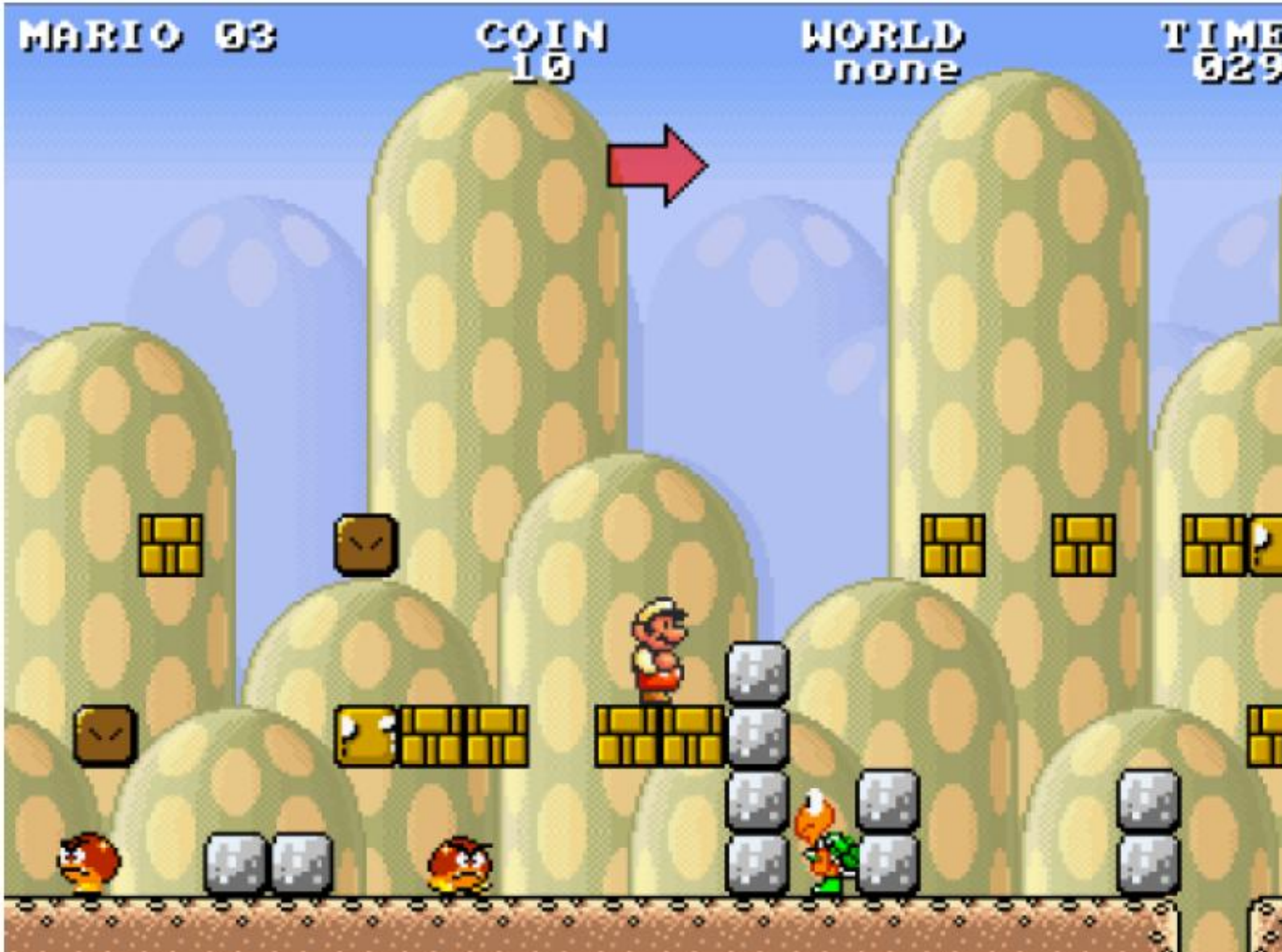
Steve Dahlskog and Julian Togelius: **Patterns as Objectives for Level Generation**. PCG Workshop 2013

Evaluation

- 25 patterns are identified in the original SMB levels
- e.g. enemy hordes, pipe valleys, 3-paths…
- The fitness function counts the number of patterns found in the level



Steve Dahlskog and Julian Togelius: **Patterns as Objectives for Level Generation**. PCG Workshop 2013



Steve Dahlskog and Julian Togelius: **Patterns as Objectives for Level Generation**. PCG Workshop 2013

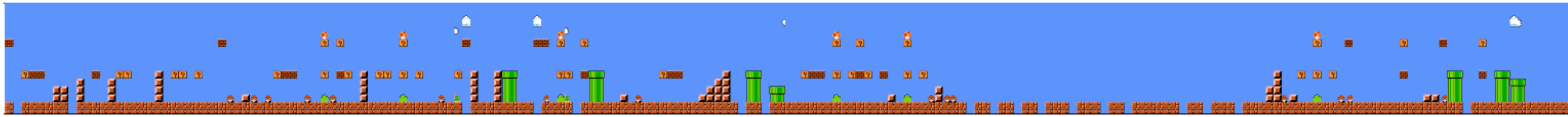


Fig. 15. FFMacro #98 MC: 6, fitness value: 2332 (highest).

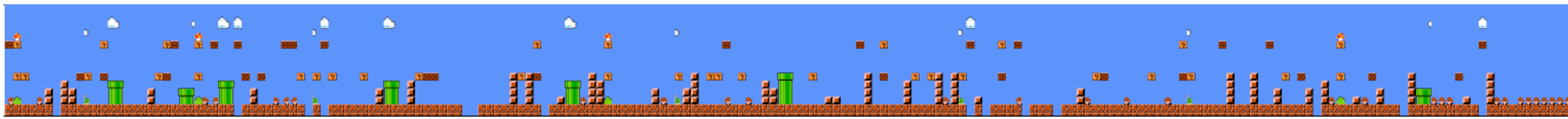


Fig. 16. FFMesoB #6 MC: 0, fitness value: 545.

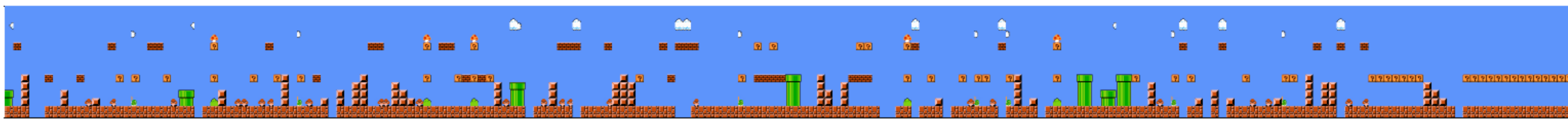


Fig. 17. FFMesoB #30 MC: 3, fitness value: 409 (lowest).

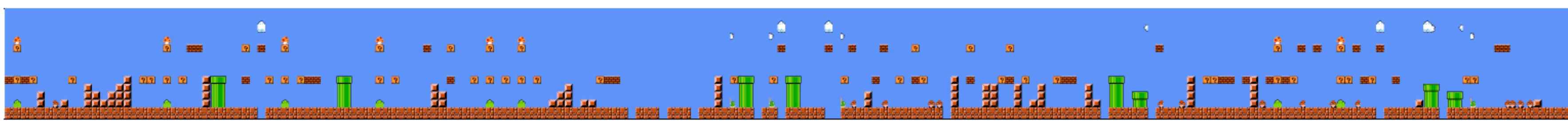


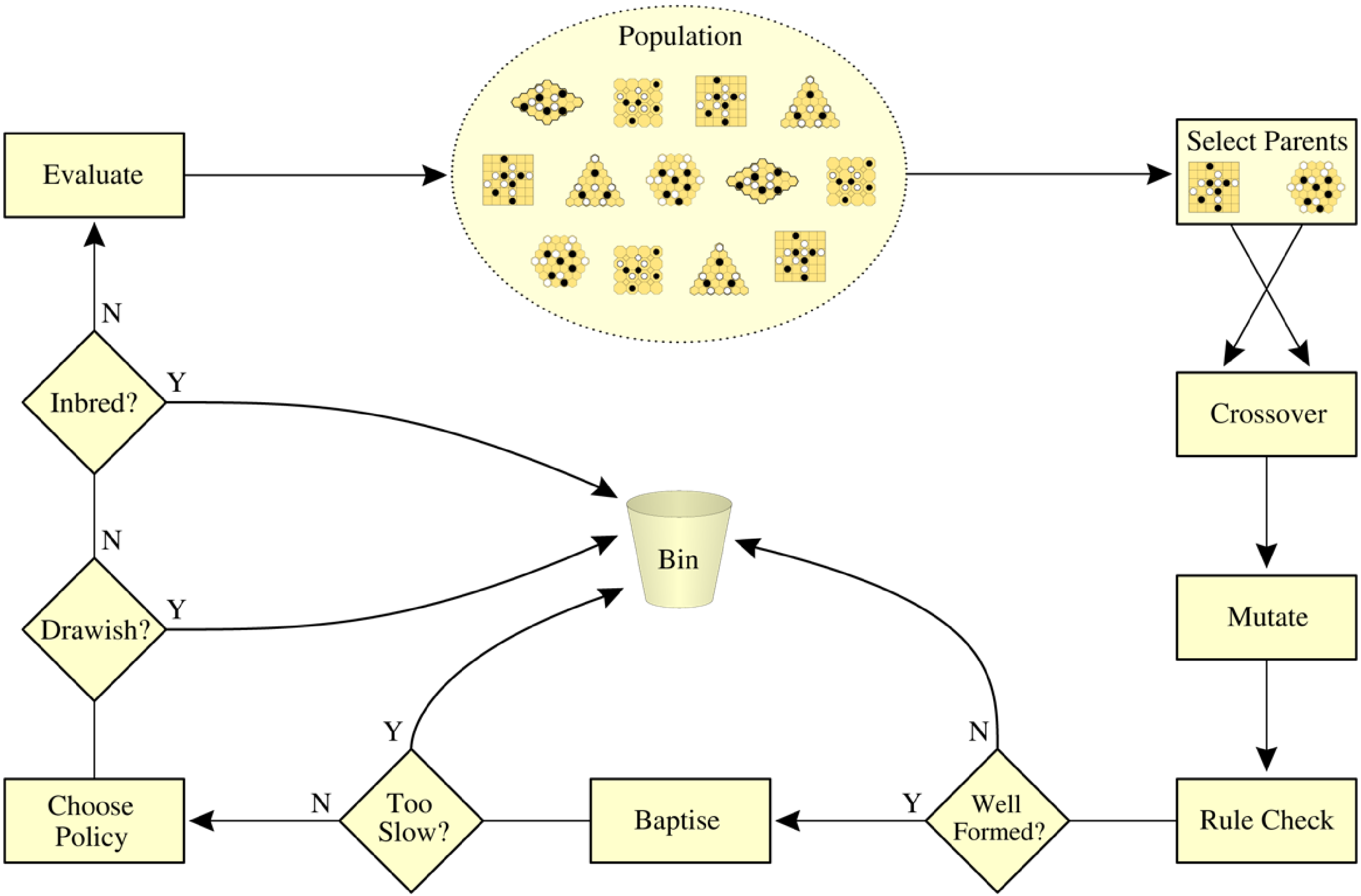
Fig. 18. FFMesoB #64 MC: 6, fitness value: 2065 (highest).

Search-Based PCG Example #2

How would we create new game rules?

Creating Game Rules

- Rules are also content…
- Will need simulation-based evaluation - you can only judge game rules by playing the game
- Has been attempted for simple Pac-Man-like games (Togelius 2008), GVGAI games (Nielsen et al 2015)
- Perhaps most convincingly for board games (Browne 2008)

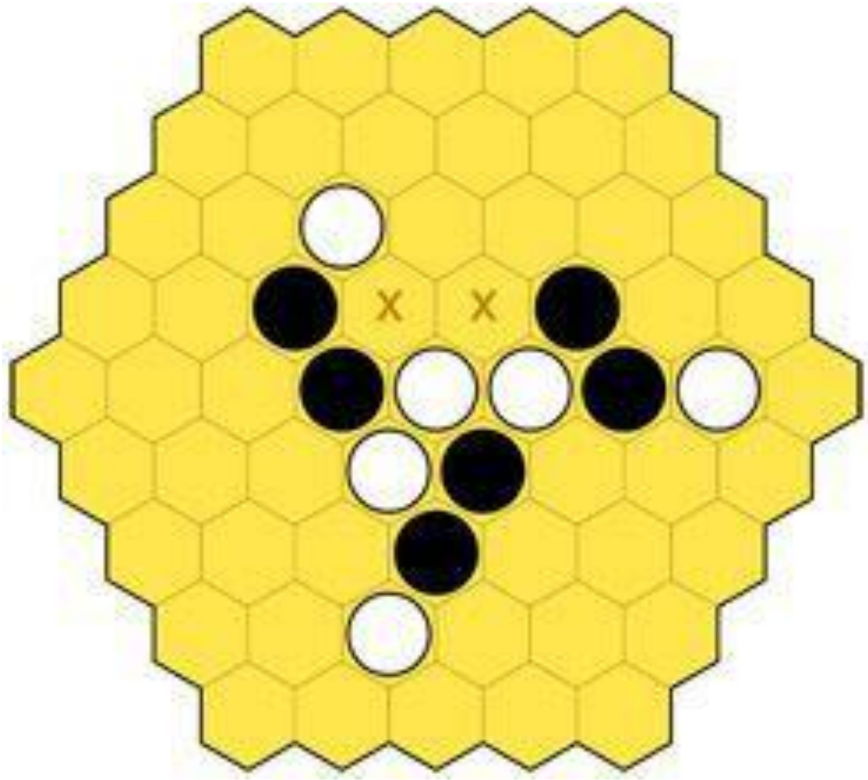

Population
Evaluate
Select Parents
Crossover
Mutate
Rule Check
Well Formed?
Y
N
Baptise
Too Slow?
Y
N
Choose Policy
Drawish?
Y
N
Inbred?
Y
N
Bin

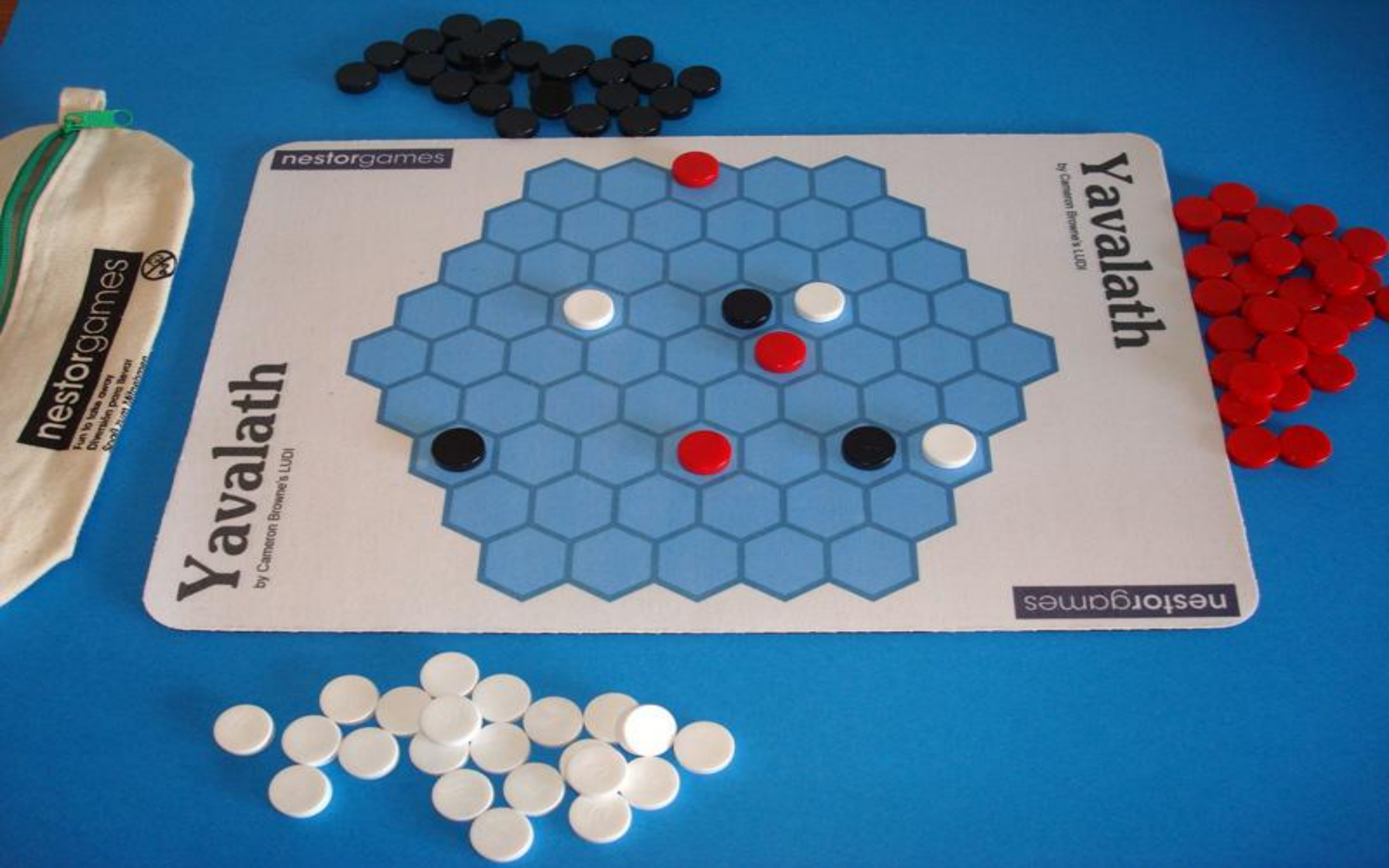
Yavalath

YAVALATH (#2)

```
(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

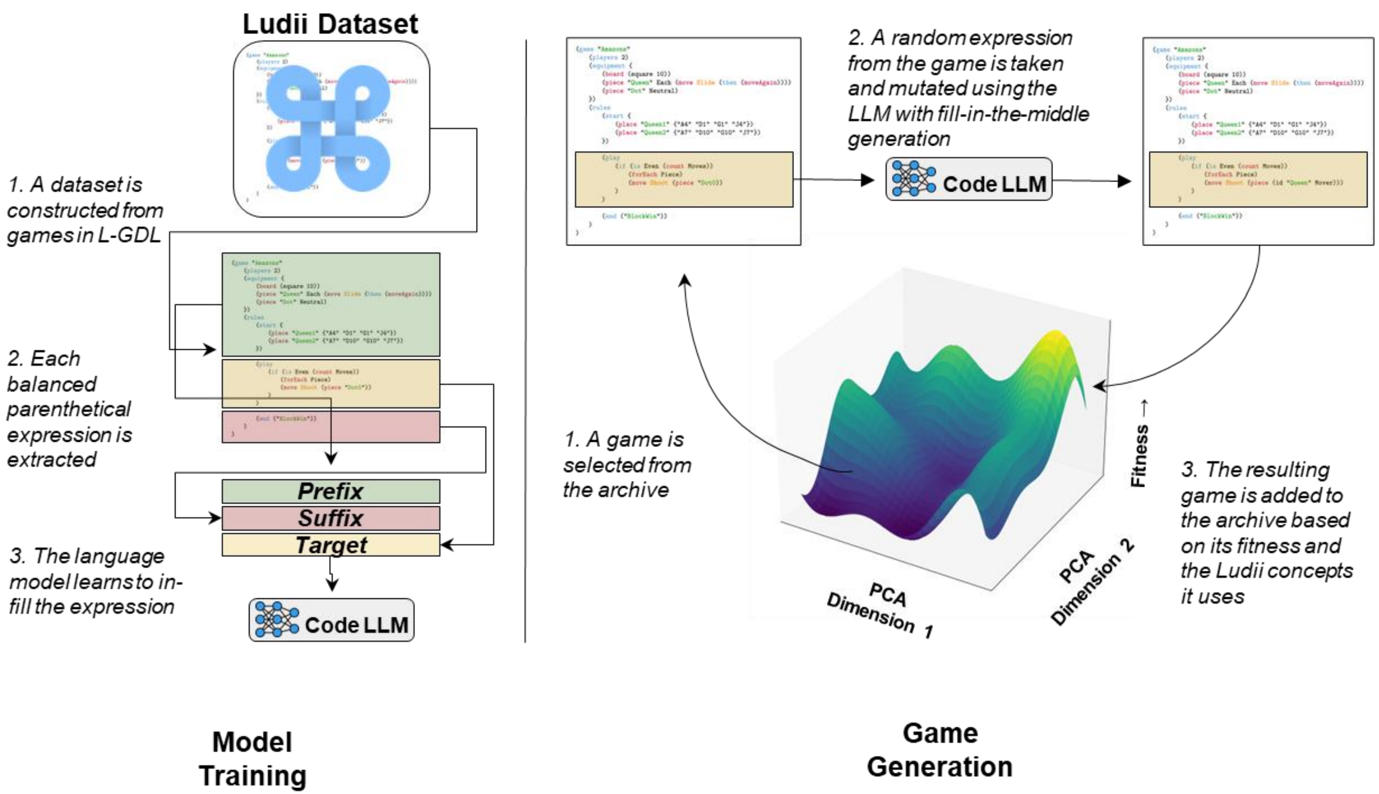



nestorgames
Yavalath
by Cameron Browne's LUDI
Yavalath
by Cameron Browne's LUDI
nestorgames
nestorgames

From Ludi to Ludii

GAVEL: Generating Games via Evolution and Language Models

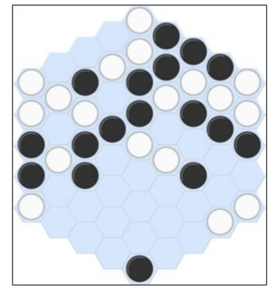
Graham Todd, Alexander Padula, Matthew Stephenson, Éric Piette, Dennis J.N.J. Soemers, Julian Togelius



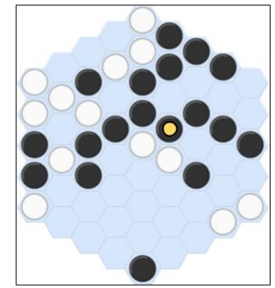
From Ludi to Ludii

```
(game "YavaGo"
    (players 2)

    (equipment {
            (board (rotate 90 (hex 5)))
            (piece "Marker" Each)
    })

    (rules
        (meta
            (no Repeat)
        )
        (play
            (move Add
                (to (sites Empty))
                (then
                    (enclose
                        (from (last To)) Orthogonal
                        (between if:(is Enemy (who at:(between)))
                            (apply (remove (between)))
                        )
                    )
                )
            )
        )

        (end {
                (if (is Line 5) (result Next Loss) )
                (if (is Line 4) (result Next Win))
        })
    )
)
```

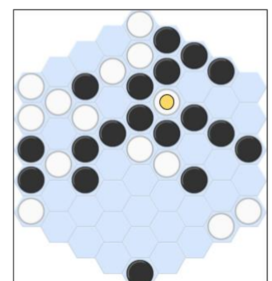
Players take turns placing markers on a hexagonal board. Getting 5 pieces in a row ***wins****, but getting 4 in a row* ***loses****. Surrounding your opponent's pieces removes them.*



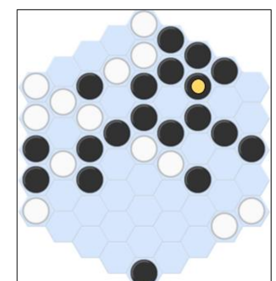
1. *What seems to be a relatively balanced position…*



2. *Black moves, capturing many pieces and setting up a lethal threat…*



3. *White makes the only possible play to not lose on the spot…*



4. *But in a single move, black removes white's defense and sets up a second threat for the win…*

Search-Based PCG Example #3

How would we design L-systems?

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
- Evolving the interpretation:
  - Evolve production probabilities
  - Evolve other aspects (e.g. turning angles)

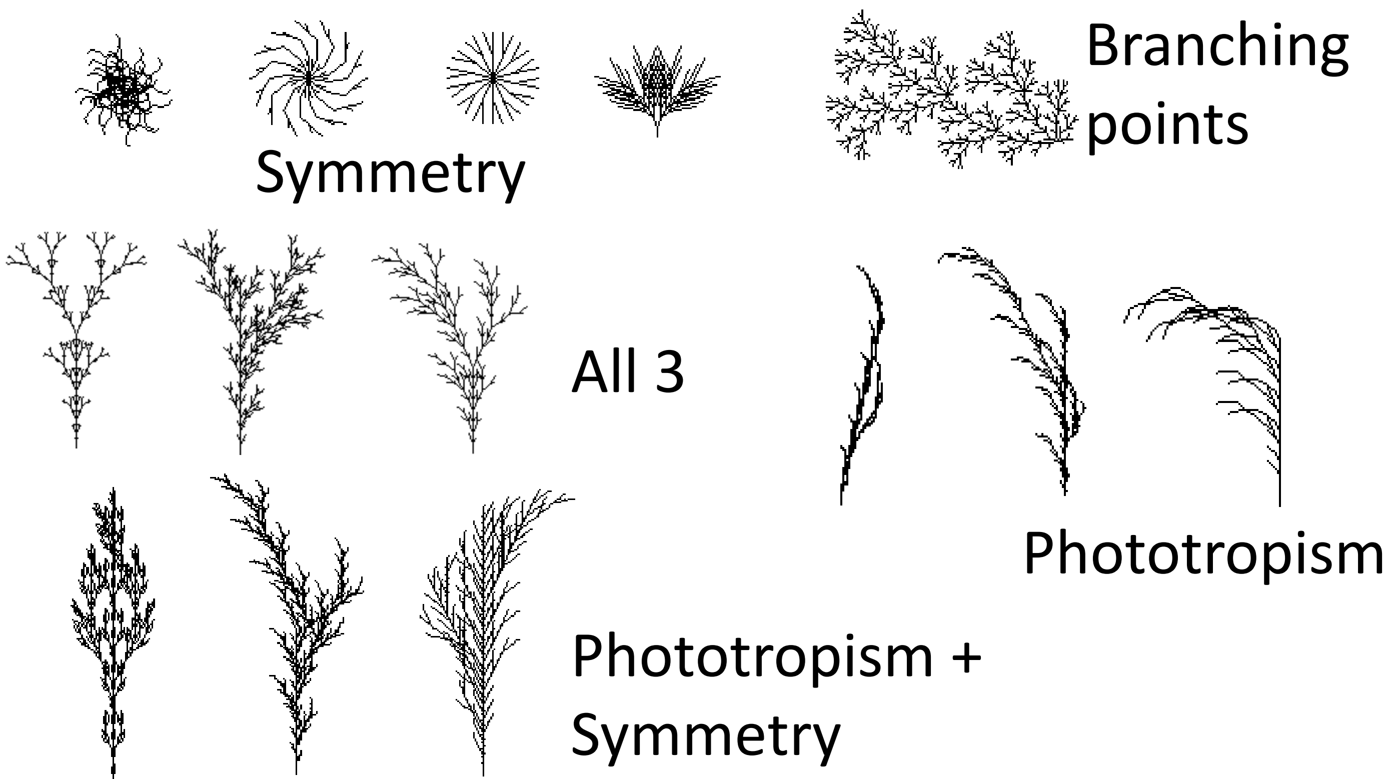
Evolving L-systems

- One example: Ochoa evolved the consequent of a single production rule
  - starting from F>F[-F]F[+F]F
- Mutation: replace single symbols, or blocks of a few symbols
- Crossover: swap complete “sub-trees” (like in genetic programming)

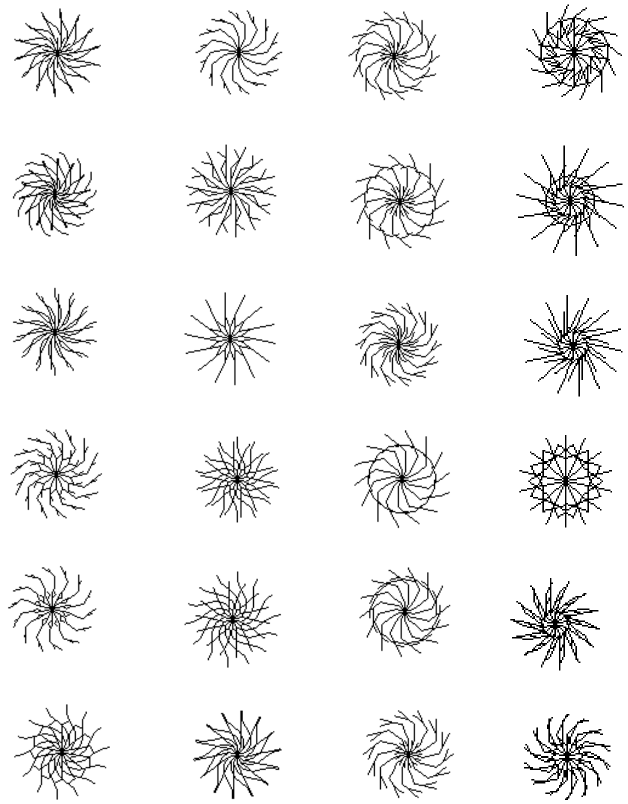
Fitness Functions

- Phototropism
- Bilateral symmetry
- Proportion of branching points

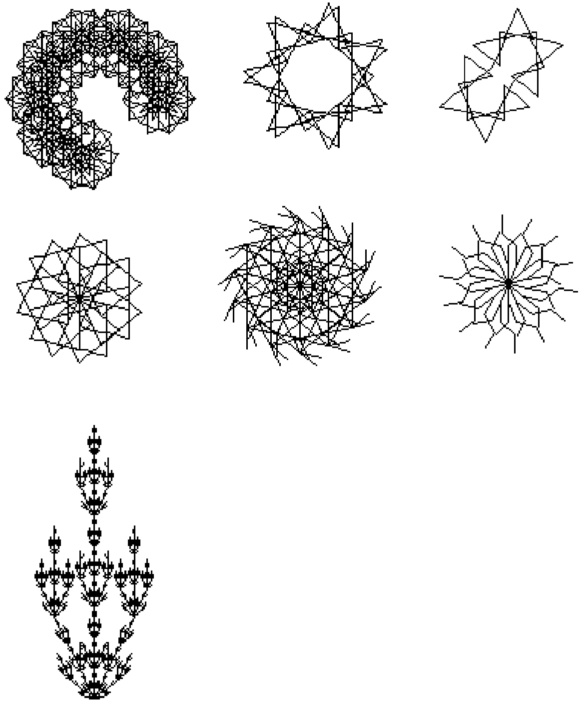
Evolved Systems

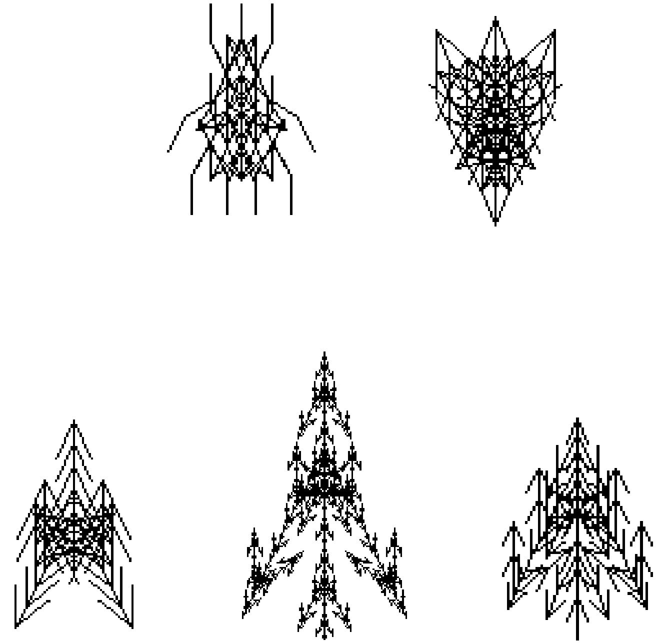


Evolved Systems



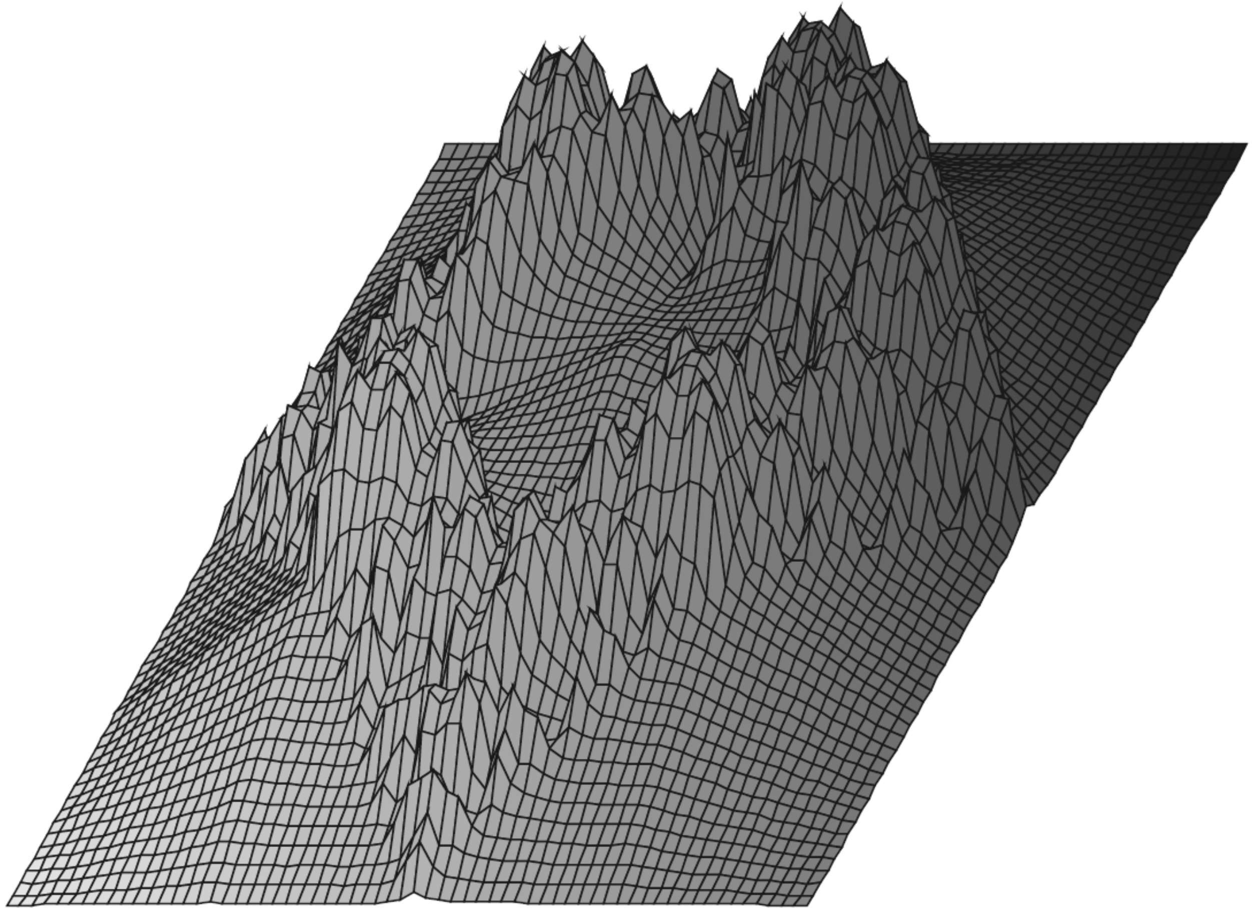
Evolved Systems



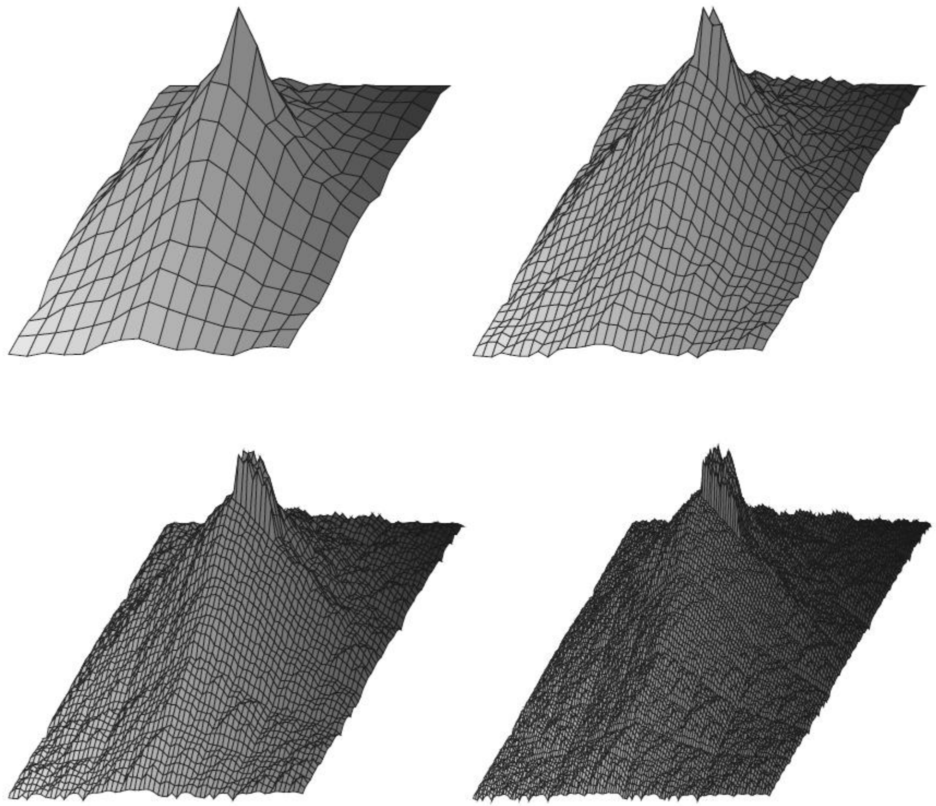


…and this was an extremely simple L-system!

Evolved 2D L-system Terrains



Evolved 2D L-system Terrains



Very short specification, yet infinite resolution!

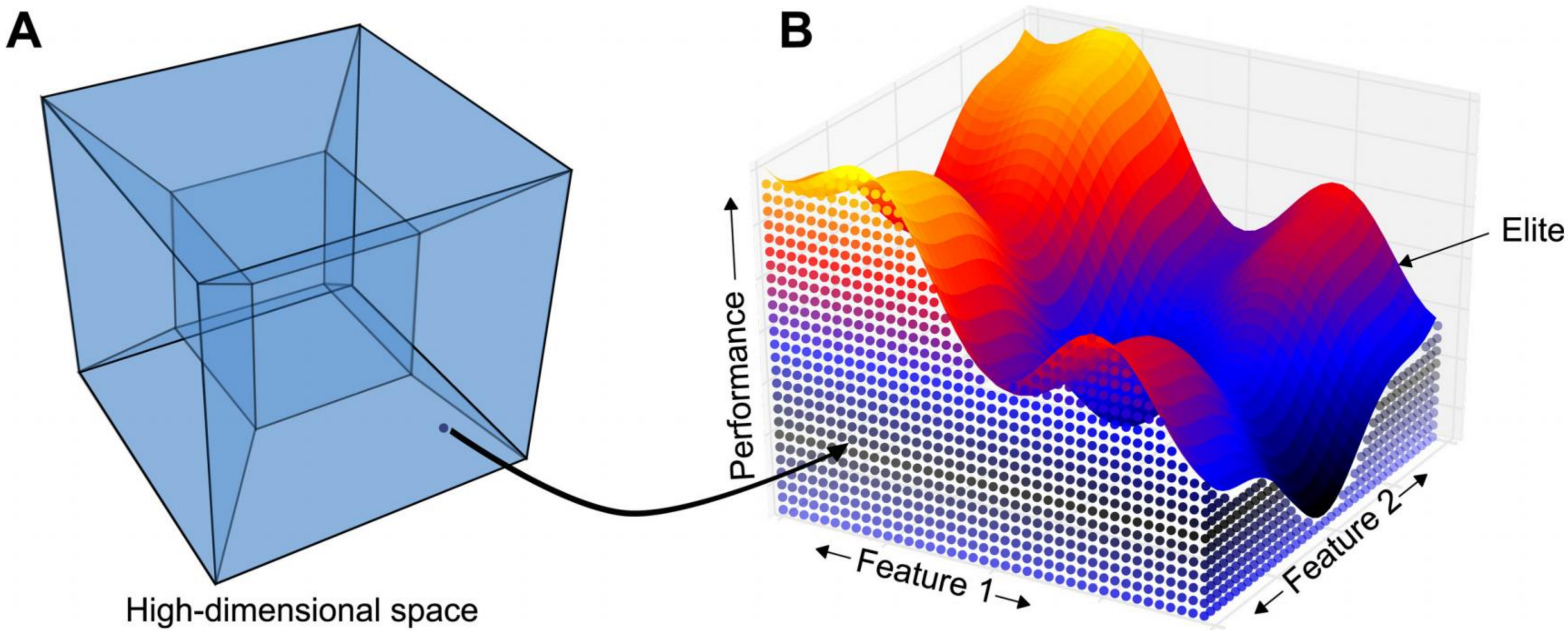
Quality-Diversity Methods

PCG through Quality Diversity

- Generating many diverse good solutions
- Needs a fitness function, and one or several behavioral characteristics (or simply metrics)
- Very useful for search-based PCG!

Daniele Gravina, Ahmed Khalifa, Antonios Liapis, Julian Togelius and Georgios N. Yannakakis. **Procedural Content Generation through Quality Diversity**. IEEE CoG 2019

Map-Elites


A
High-dimensional space
B
Performance
Elite
Feature 1
Feature 2

PCGQD Examples: CMA-ME

Covariance Matrix Adaptation for the Rapid Illumination of Behavior Space

Matthew C. Fontaine
Viterbi School of Engineering
University of Southern California
Los Angeles, CA
mfontain@usc.edu

Julian Togelius
Tandon School of Engineering
New York University
New York City, NY
julian@togelius.com

Stefanos Nikolaidis
Viterbi School of Engineering
University of Southern California
Los Angeles, CA
nikolaid@usc.edu

Amy K. Hoover
Ying Wu College of Computing
New Jersey Institute of Technology
Newark, NJ
ahoover@njit.edu

ABSTRACT

We focus on the challenge of finding a diverse collection of quality solutions on complex continuous domains. While quality diversity (QD) algorithms like Novelty Search with Local Competition (NSLC) and MAP-Elites are designed to generate a diverse range of solutions, these algorithms require a large number of evaluations for exploration of continuous spaces. Meanwhile, variants of the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) are among the best-performing derivative-free optimizers in single-objective continuous domains. This paper proposes a new QD algorithm called Covariance Matrix Adaptation MAP-Elites (CMA-ME). Our new algorithm combines the self-adaptation techniques of CMA-ES with archiving and mapping techniques for maintaining diversity in QD. Results from experiments based on standard continuous optimization benchmarks show that CMA-ME finds better-quality solutions than MAP-Elites; similarly, results on the strategic game Hearthstone show that CMA-ME finds both a higher overall

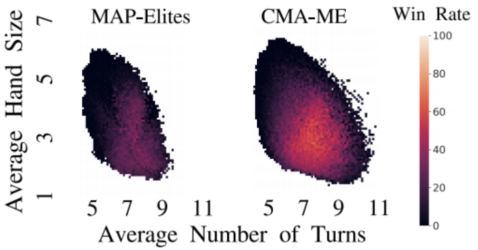
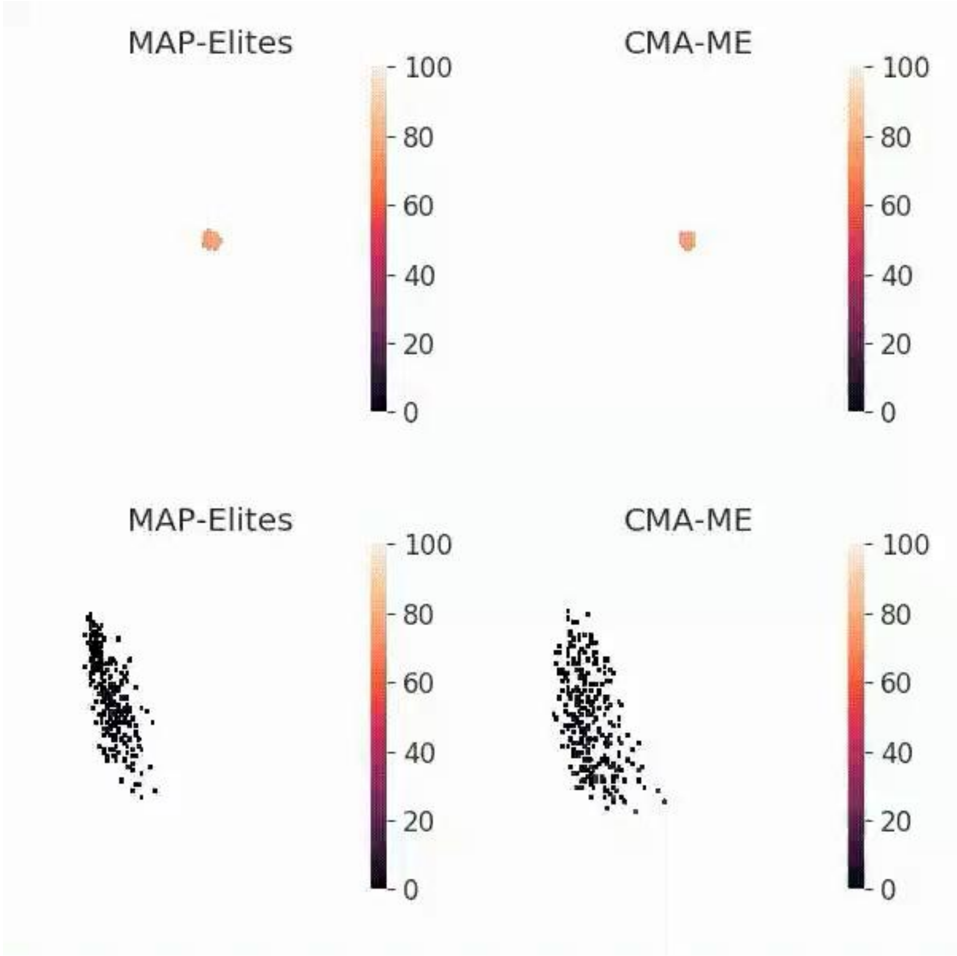


Figure 1: *Comparing Hearthstone Archives.* Sample archives for both MAP-Elites and CMA-ME from the Hearthstone experiment. Our new method, CMA-ME, both fills more cells in behavior space and finds higher quality policies to play Hearthstone than MAP-Elites. Each grid cell is an elite (high performing policy) and the intensity value represent the win rate across 200 games against difficult opponents.

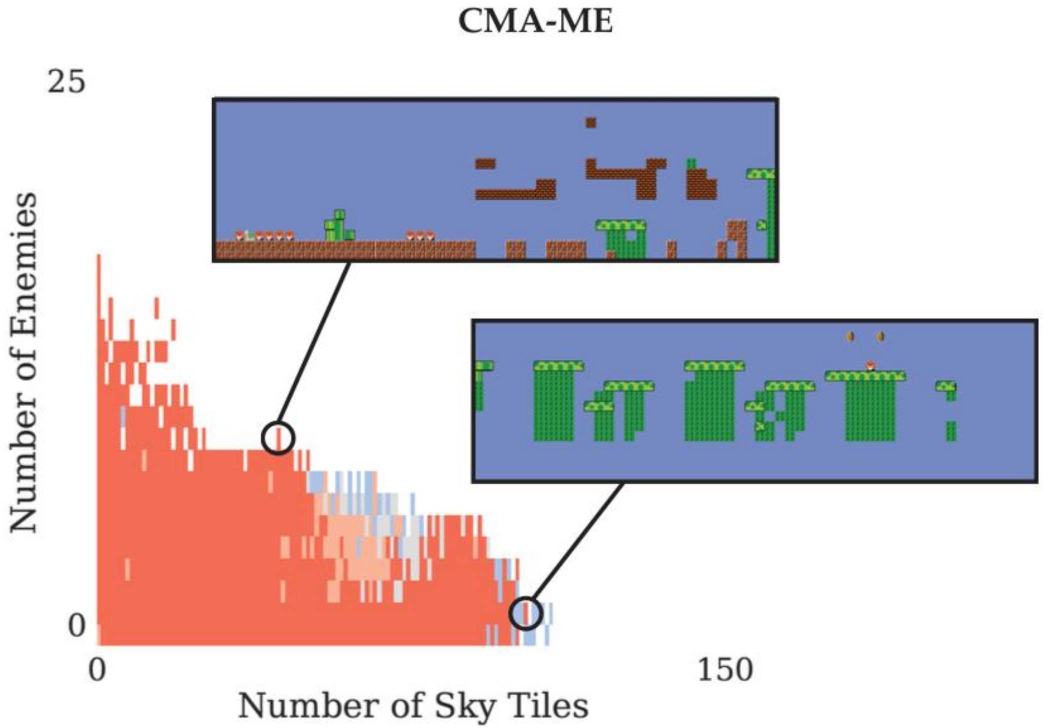
PCGQD Examples: CMA-ME



PCGQD Examples: Super Mario

Illuminating Mario Scenes in the Latent Space of a Generative Adversarial Network

Matthew C. Fontaine[1], Ruilin Liu[1], Ahmed Khalifa[2], Jignesh Modi[1], Julian Togelius[2], Amy K. Hoover[3], Stefanos Nikolaidis[1]



PCGQD Examples: Super Mario

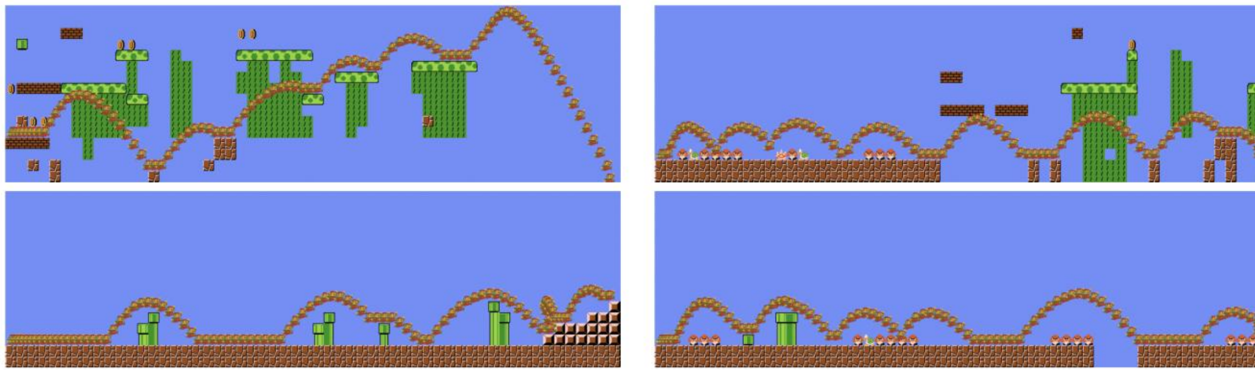


Figure 5: Generated scenes using CMA-ME for small and large values of sky tiles and number of enemies.

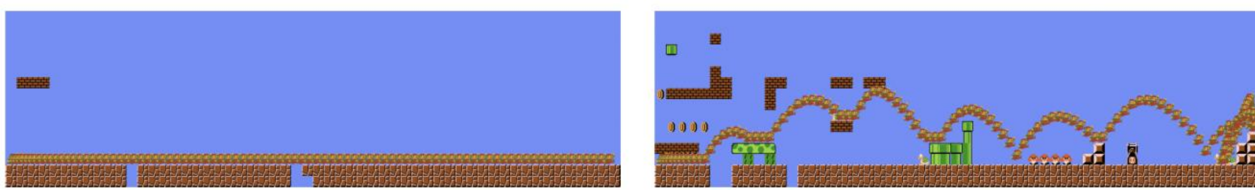


Figure 6: Playable scenes with minimum (left) and maximum (right) sum value (6) of the 8 binary agent-based BCs.

PCGQD Examples: Talakat

Talakat: Bullet Hell Generation through Constrained Map-Elites

Ahmed Khalifa
New York University
New York City, New York
ahmed.khalifa@nyu.edu

Scott Lee
New York University
New York City, New York
sl3998@nyu.edu

Andy Nealen
New York University
New York City, New York
andy@nealen.net

Julian Togelius
New York University
New York City, New York
julian@togelius.com

ABSTRACT

We describe a search-based approach to generating new levels for bullet hell games, which are action games characterized by and requiring avoidance of a very large amount of projectiles. Levels are represented using a domain-specific description language, and search in the space defined by this language is performed by a novel variant of the Map-Elites algorithm which incorporates a feasible-infeasible approach to constraint satisfaction. Simulation-based evaluation is used to gauge the fitness of levels, using an agent based on best-first search. The performance of the agent can be tuned according to the two dimensions of strategy and dexterity, making it possible to search for level configurations that require a specific combination of both. As far as we know, this paper describes the first generator for this game genre, and includes several algorithmic innovations.

As such, games sometimes are differentiated based on the quality and quantity of their content. Some genres and series exhibit this emphasis on content more strongly than others, and it is in this space that Procedural Content Generation (PCG) [20] can be particularly useful.

Content in the bullet hell genre is often measured by its challenge, so developers often attempt to make their games punishingly hard. However, players come with a wide variety of skill levels, and so many bullet hell games add multiple difficulty levels in the hope that the game can present challenging content to a wider variety of players. It is entirely possible for designers to miss the mark on easier content, making it too close in difficulty to harder content or making it too easy to be interesting. This is a fundamental challenge that can arise in game design, and one that may be lightened with PCG techniques.

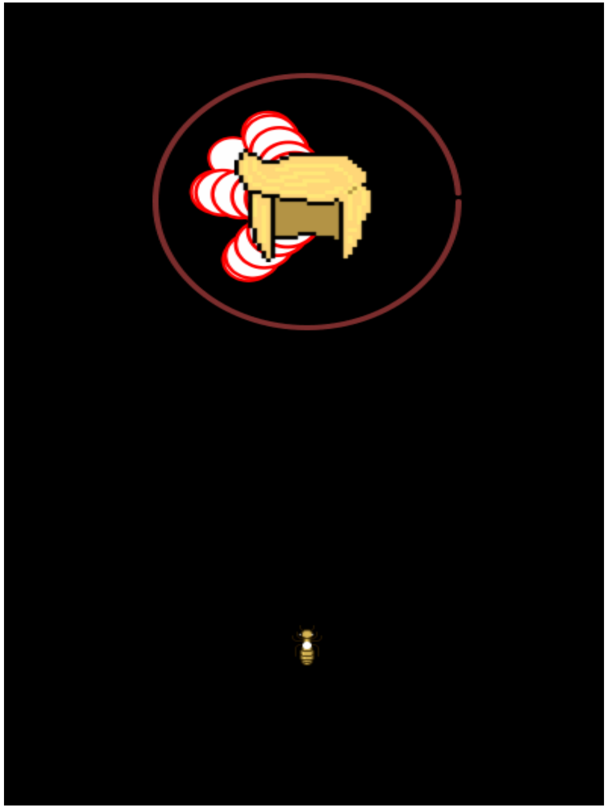
PCGQD Examples: Talakat

PCGQD Examples: Talakat

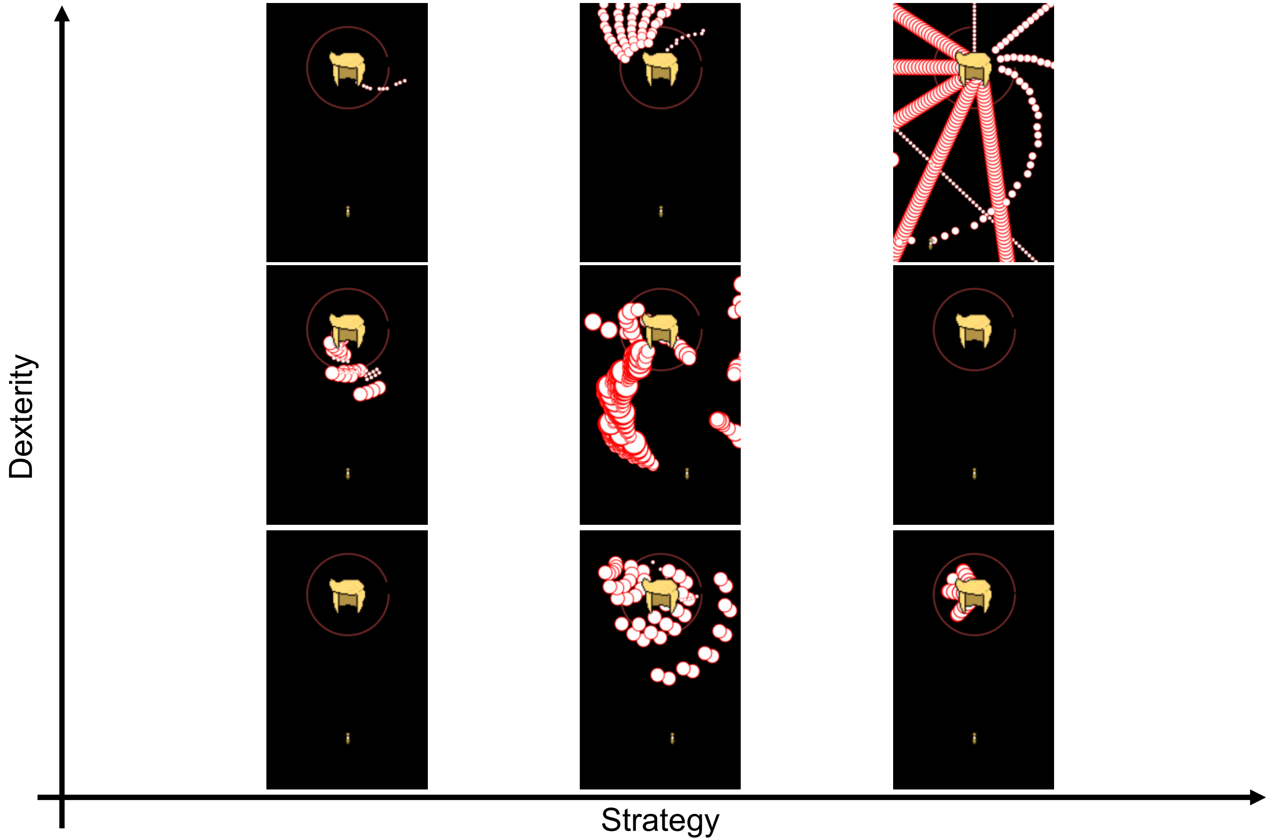
- A* agent

$$f(x) = 0.5 \cdot progress - lose + 0.5 \cdot safety - 0.25 \cdot future$$

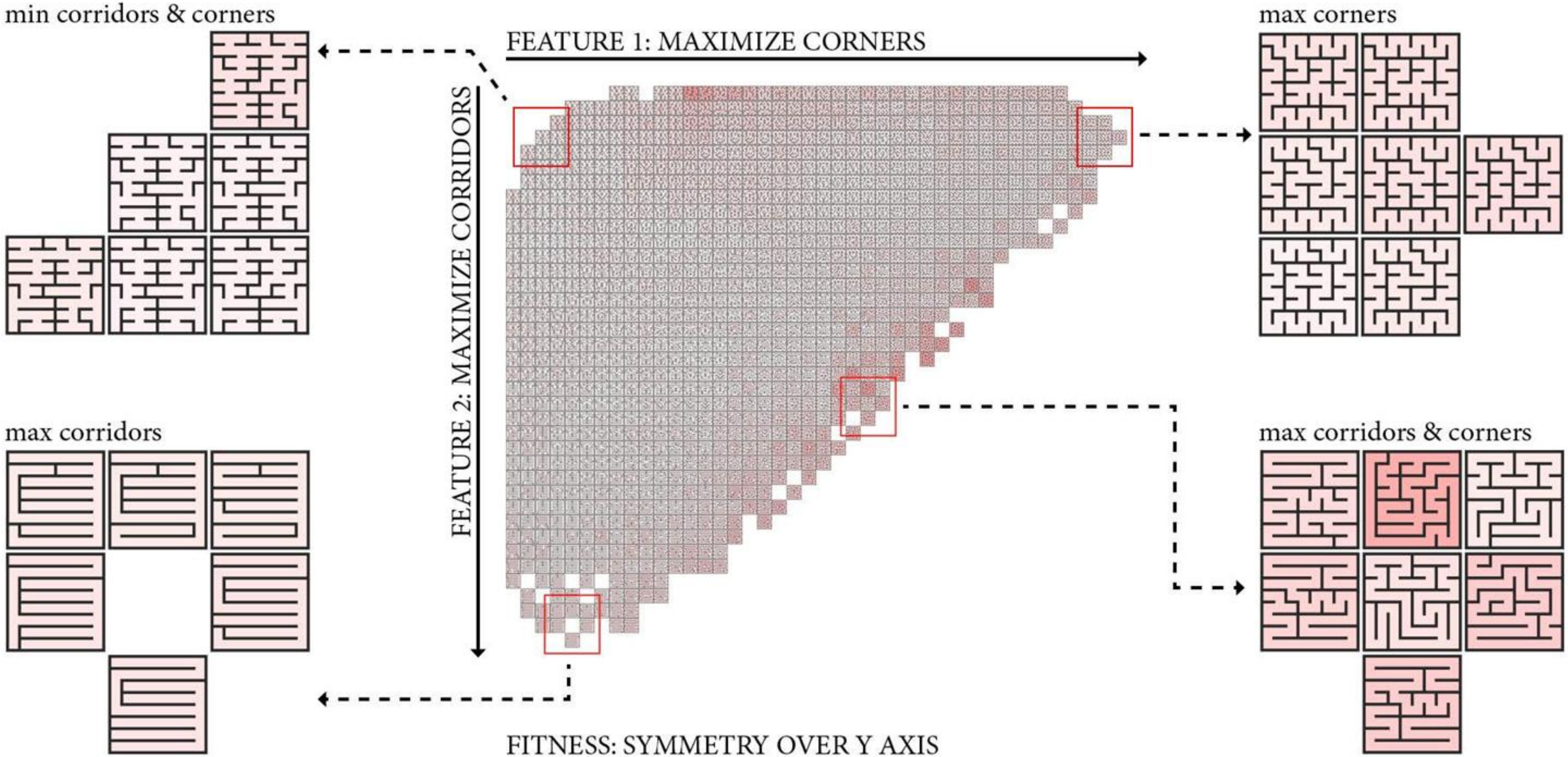
- Dexterity Metric: measured with an agent with impaired maneuvering
- Strategy Metric: measured with an agent with limited planning ahead



PCGQD Examples: Talakat

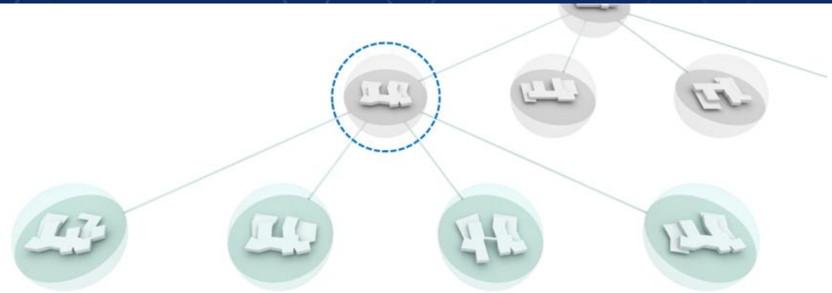


PCGQD Examples: Maze Generation



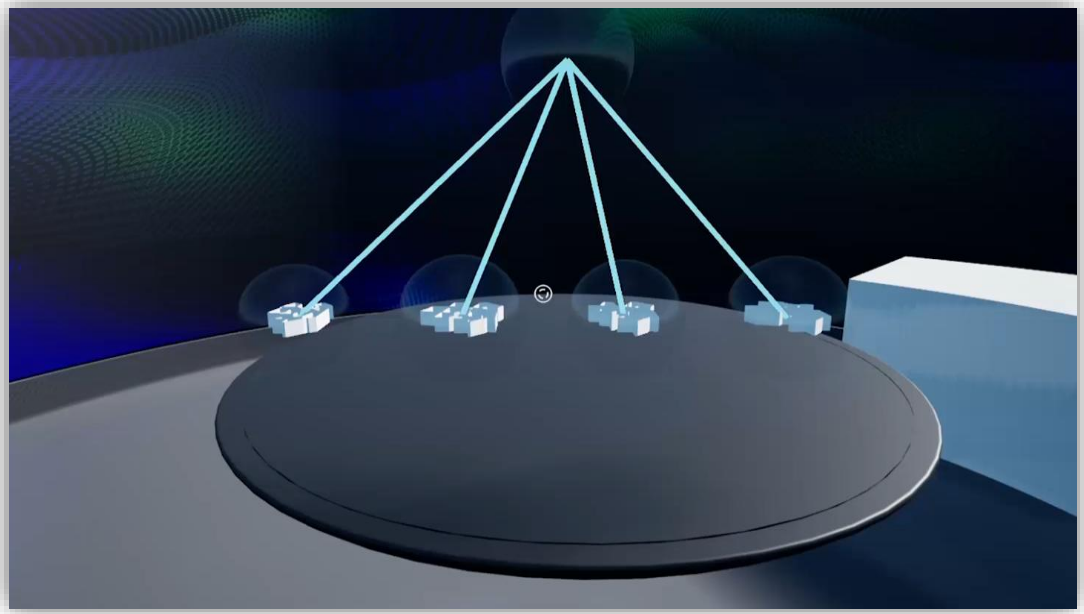
Sfikas, Liapis and Yannakakis: **“Monte Carlo Elites: Quality-Diversity Selection as a Multi-Armed Bandit Problem”** in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2021

PCGQD on Architecture & Urban Design



Sfikas, Liapis, Yannakakis **A General-Purpose Expressive Algorithm for Room-based Environments**, in *Proceedings of FDG*, 2022

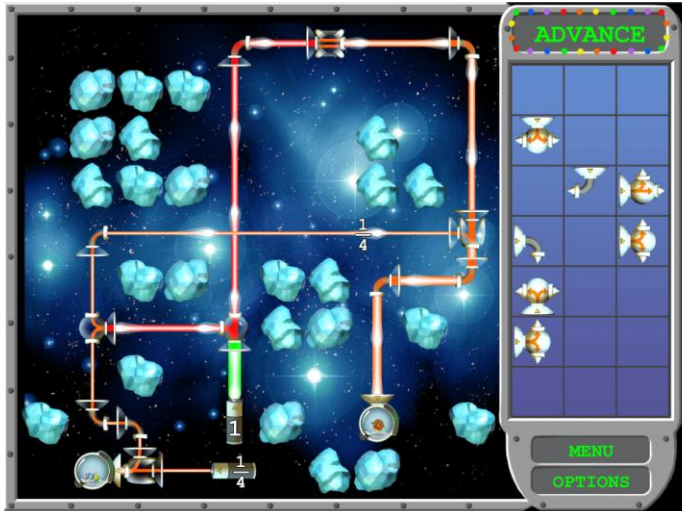
Galanos, Liapis, Yannakakis, **ARCH-Elites: Quality-Diversity for Urban Design**, *GECCO*, 2021



Solver-Based Methods

Solver-Based Methods

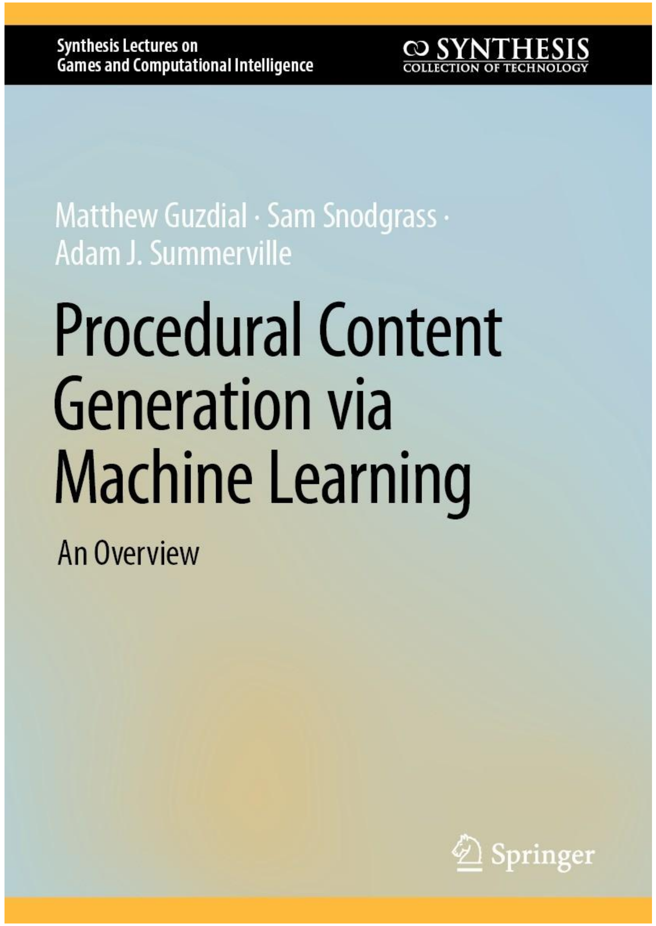
- Instead of using evolutionary algorithms, use constraint solvers and specify constraints
- Example: Answer Set Programming (ASP)
  - Declarative programming for search problems
  - Based on logic programming (syntax very similar to Prolog)
  - Finding an answer set is equivalent to solving a satisfiability problem



Procedural Content Generation via Machine Learning (PCGML)

Adam Summerville[1], Sam Snodgrass[2], Matthew Guzdial[3], Christoffer Holmgård[4], Amy K. Hoover[5], Aaron Isaksen[6], Andy Nealen[6], and Julian Togelius[6],
[1]Department of Computational Media, University of California, Santa Cruz, CA 95064, USA
[2]College of Computing and Informatics, Drexel University, Philadelphia, PA 19104, USA
[3]School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA
[4]Duck and Cover Games ApS, 1311 Copenhagen K, Denmark
[5]College of Arts, Media and Design, Northeastern University, Boston, MA 02115, USA
[6]Department of Computer Science and Engineering, New York University, Brooklyn, NY 11201, USA
emails: asummerv@ucsc.edu, sps74@drexel.edu, mguzdial3@gatech.edu, christoffer@holmgard.org, amy.hoover@gmail.com, aisaksen@appabove.com, nealen@nyu.edu, julian@togelius.com

This survey explores *Procedural Content Generation via Machine Learning* (PCGML), defined as the generation of game content using machine learning models trained on existing content. As the importance of PCG for game development increases, researchers explore new avenues for generating high-quality content with or without human involvement; this paper addresses the relatively new paradigm of using machine learning (in contrast with search-based, solver-based, and constructive methods). We focus on what is most often considered functional game content such as platformer levels, game maps, interactive fiction stories, and cards in collectible card games, as opposed to cosmetic content such as sprites and sound effects. In addition



PCG via Machine Learning

PCG via Machine Learning

- Basic idea of Procedural Content Generation via Machine Learning (PCGML): train machine learning models on corpuses of existing content, then generate new content
- Many methods useful, including n-grams, Markov chains, and… yes, even deep learning and LLMs
- Reference: Summerville et al. (2018): Procedural Content Generation via Machine Learning (PCGML)

N-Grams

N-Grams

- Capture the statistic co-occurrence of sequential characters
- n: number of previous characters a new character depends on
- Commonly used for predictive text
- Can also be used for linear game content

N-Grams

- 1-D generation technique
- Learn to generate next character based on the previous ones
- Used in Language Modeling

NULL > O (0.2), A (0.2), B (0.4), E (0.1), END (0.1)

Unigram Model

A > B (1.0)
B > E (0.5), O (0.1), A (0.1), END (0.3)
E > B (0.3), END (0.7)
O > O (0.5), B (0.5)

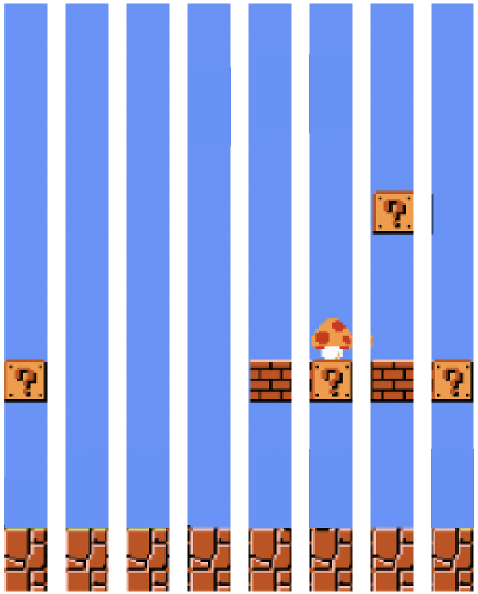
Bigram Model

AB > B (1.0)
BE > E (0.5), O (0.1), A (0.1), END (0.3)

Trigram Model

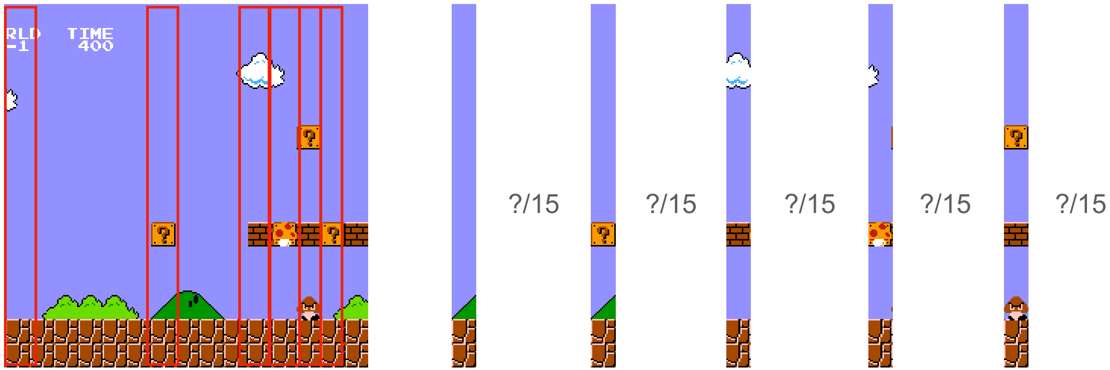
N-Grams in Super Mario

Mario level viewed as a group of slices



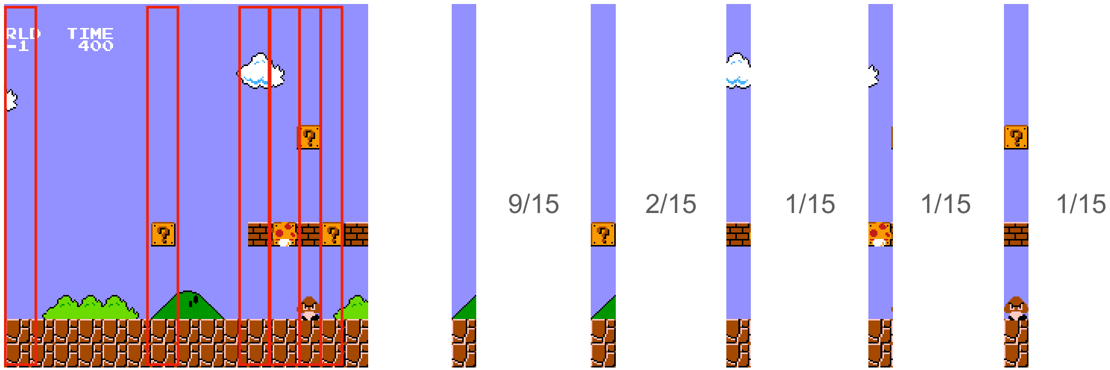
N-Grams in Super Mario

Unigram: Count each slice and convert to probability



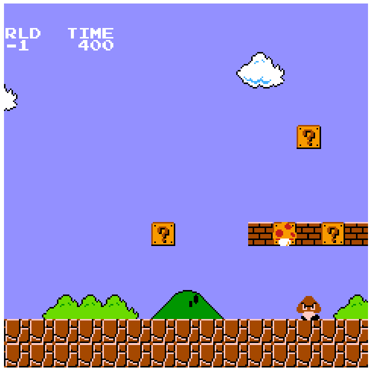
N-Grams in Super Mario

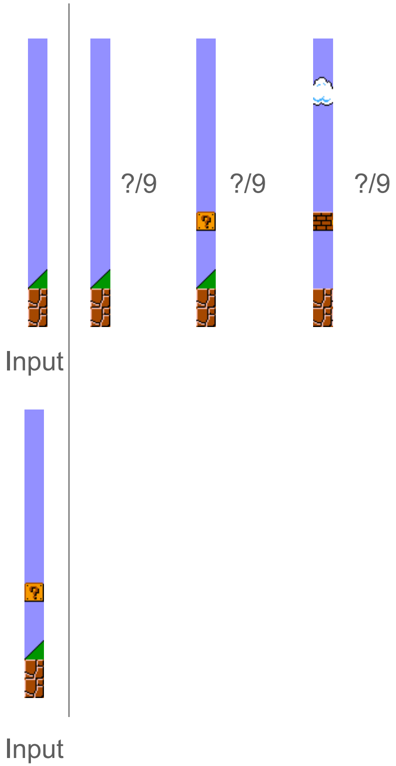
Unigram: Count each slice and convert to probability



N-Grams in Super Mario

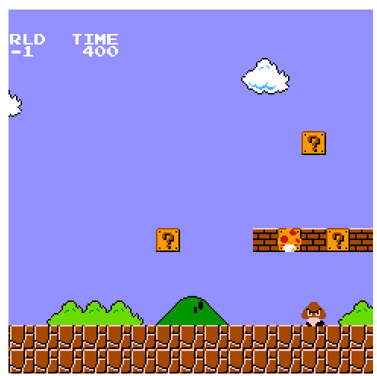
Bigram: For every slice, count the next upcoming different slice and convert to probability

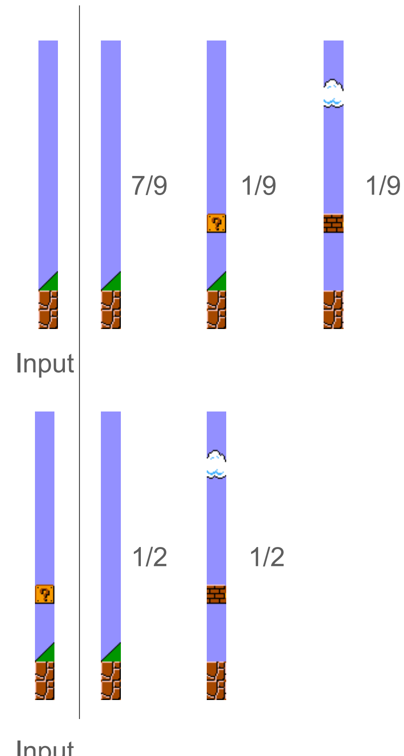




N-Grams in Super Mario

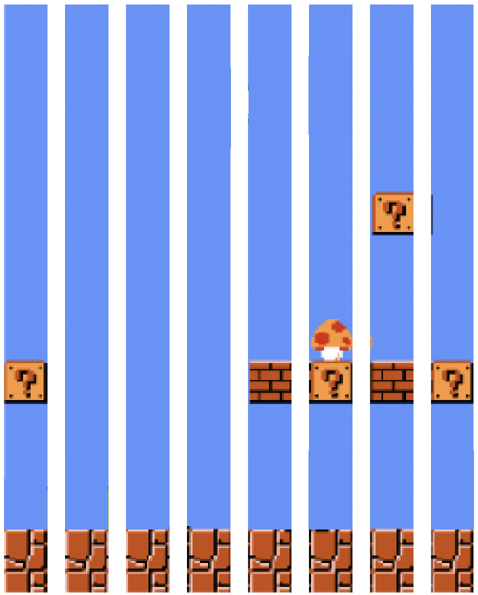
Bigram: For every slice, count the next upcoming different slice and convert to probability



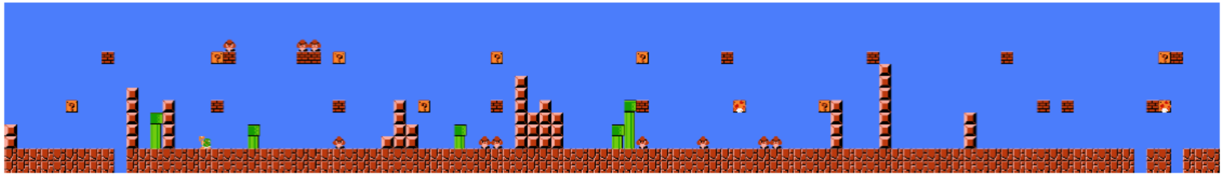


N-Grams in Super Mario

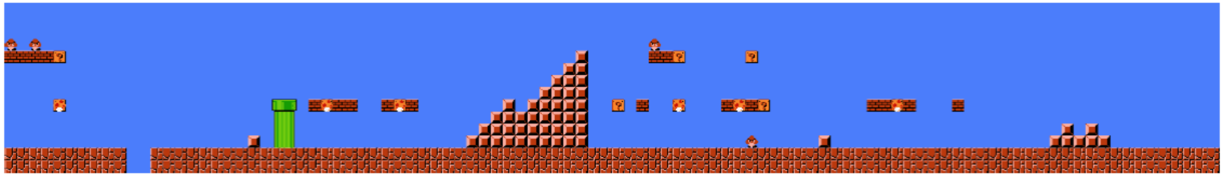
- Mario level is a group of slices
- Train unigram, bigram, and trigram over all 15 Mario levels
- Use trained N-Grams to generate new levels



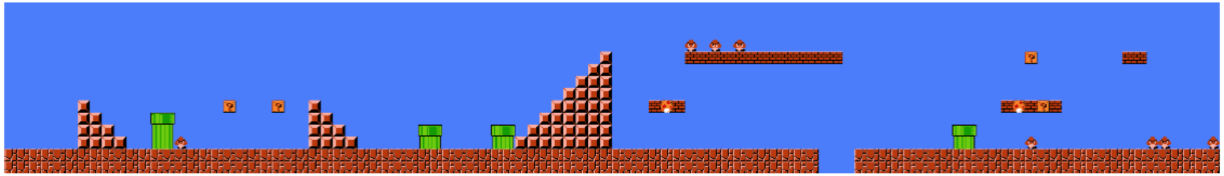
Why?



(a) $n = 1$



(b) $n = 2$



(c) $n = 3$

Figure 6: Mario levels reconstructed by n-grams with n set to 1, 2, and 3 respectively.

Steve Dahlskog, Julian Togelius, and Mark J. Nelson (2014): **Linear levels through n-grams.** In proceedings of MindTrek

Neural Networks

Neuroevolution

- Evolving the weights of neural networks, either interactively or according to some automatic fitness function
- Can be used for PCG: predicting some aspect of a level (e.g. enemies in Super Mario Bros) from other aspects (e.g. bricks and question mark blocks)
- Analogically: The different layers of a level are akin to different instruments in a song

Neuroevolution for Mario

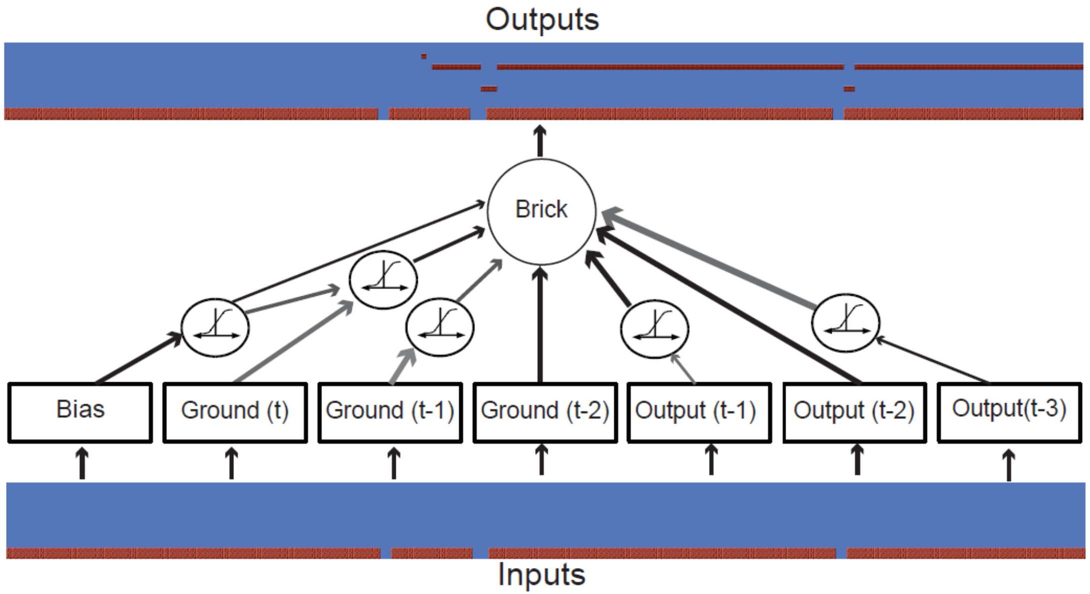
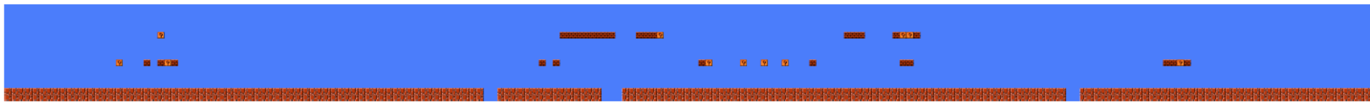


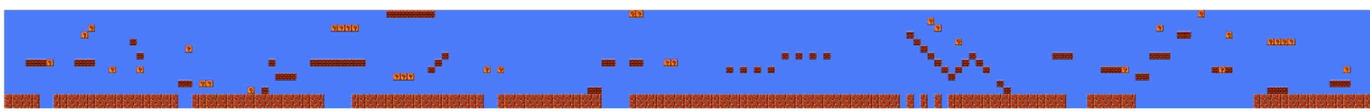
Figure 1: **The ANN Representation**. At each column in a given level, an ANN inputs visual assets (i.e., musical voices) from at least one tile type of *Super Mario Bros.* and outputs a different type of visual asset. This example figure shows ground tiles being input to the ANN while the brick tile placements are output from predictions made through NeuroEvolution of Augmenting Topologies [39]. To best capture the regularities in a level, each column is also provided information about the most recently encountered inputs (i.e., input values for the three previous columns). The inputs and outputs are then fed back into the ANN at each subsequent tick. Once trained, ANNs can potentially suggest reasonable placements for new human composed in-game tiles.

(a) Original World One Level One in Super Mario Bros.

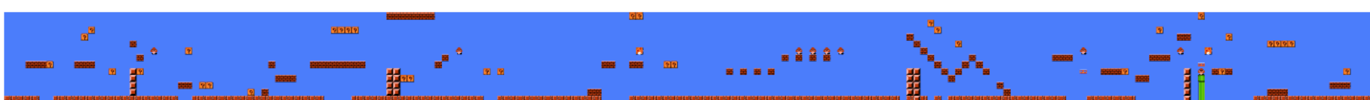


(b) The Ground, Brick, and Question Mark Voices in World One Level One

(c) The remaining voices regenerated from several trained networks, using the three-voice level in (b) as input



(d) Author-created Level with Ground, Brick, and Question Mark Voices



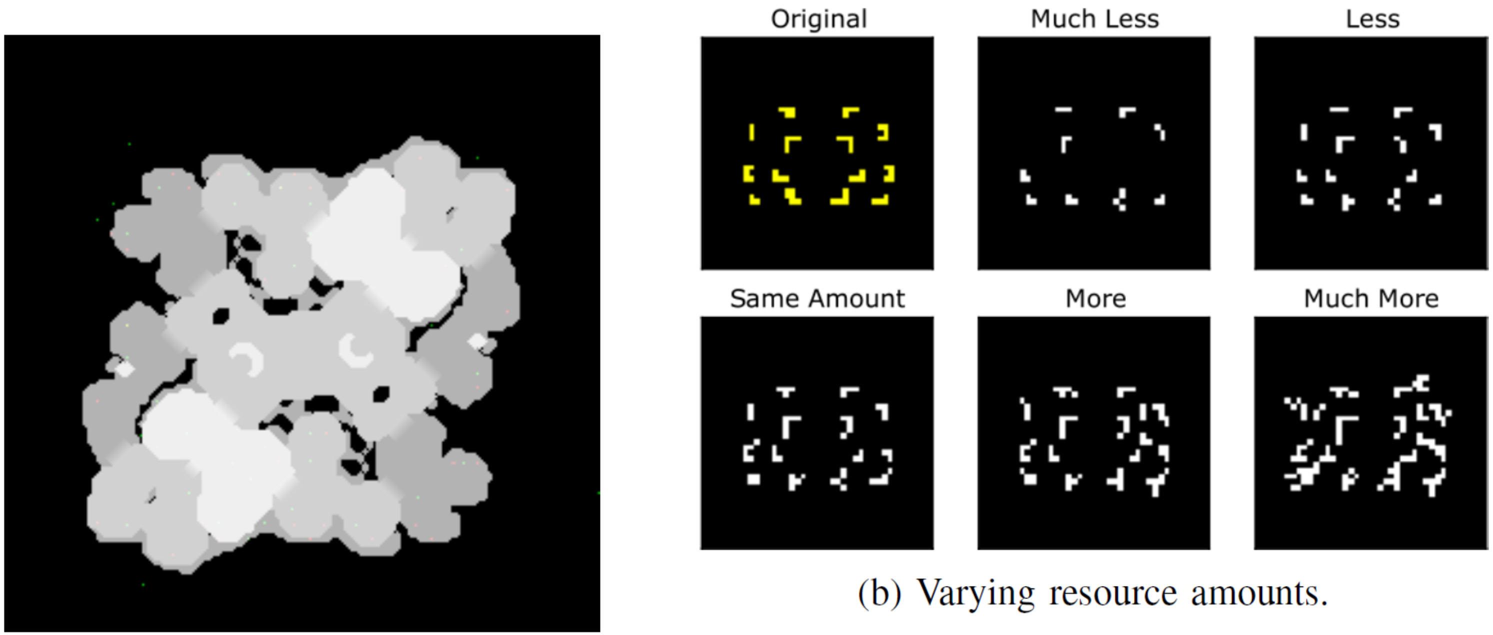
(e) Additional voices generated by trained networks for the author-created level.

Amy K. Hoover, Julian Togelius and Georgios N. Yannakakis (2015): **Composing Video Game Levels with Music Metaphors through Functional Scaffolding**. ICCC Workshop on Computational Creativity and Games.

Convolutional Neural Nets

- Commonly used for classification and prediction from high dimensional matrices, such as images
- Makes use of the convolution operation to find recurring features in images while minimizing the number of parameters
- Can be used for PCG: predicting level features from other features (e.g. resources from base locations)

CNNs for *StarCraft* Resources



(a) A *StarCraft II* heightmap

(b) Varying resource amounts.

Figure 4: A heightmap and generated resource locations for *StarCraft II*

Scott Lee, Aaron Isaksen, Christoffer Holmgård and Julian Togelius (2016): **Predicting Resource Locations in Game Maps Using Deep Convolutional Neural Networks**. EXAG.

Long Short-Term Memory

- Commonly used for sequence recognition and prediction, for example for audio or text
- Often used for generative text, through predicting the next word or letter based on some training set
- Can be used for PCG: predicting the next element in a sequence, e.g. a level segment or a word in a description

LSTM for Mario

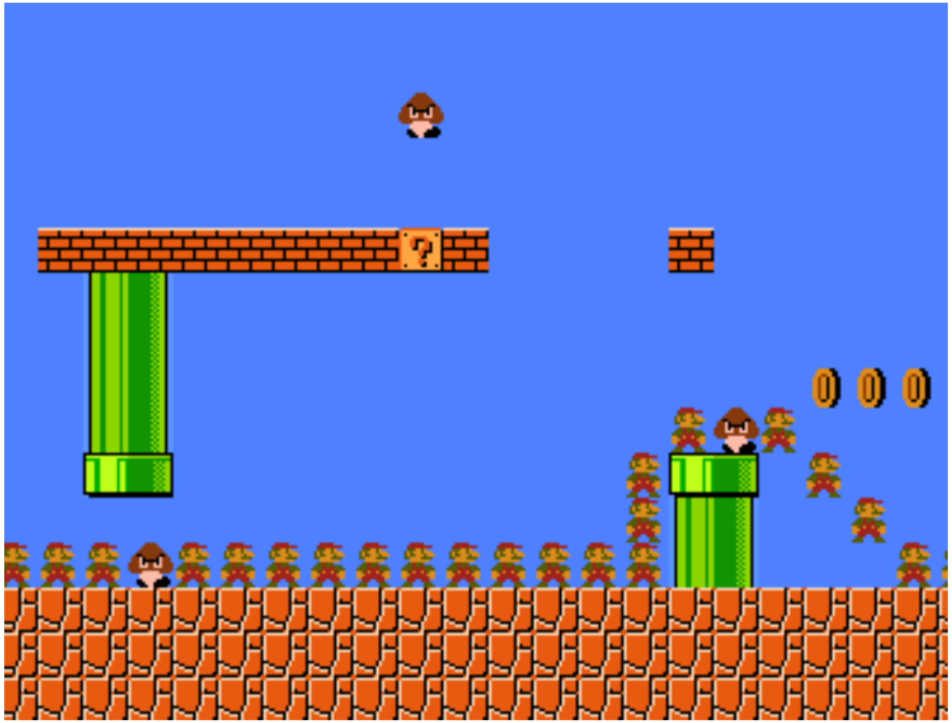


Figure 2: Example output of the LSTM approach, including generated exemplar player path.

LSTM for Magic Cards

Figure 5: Example partial card specification and its resulting output.

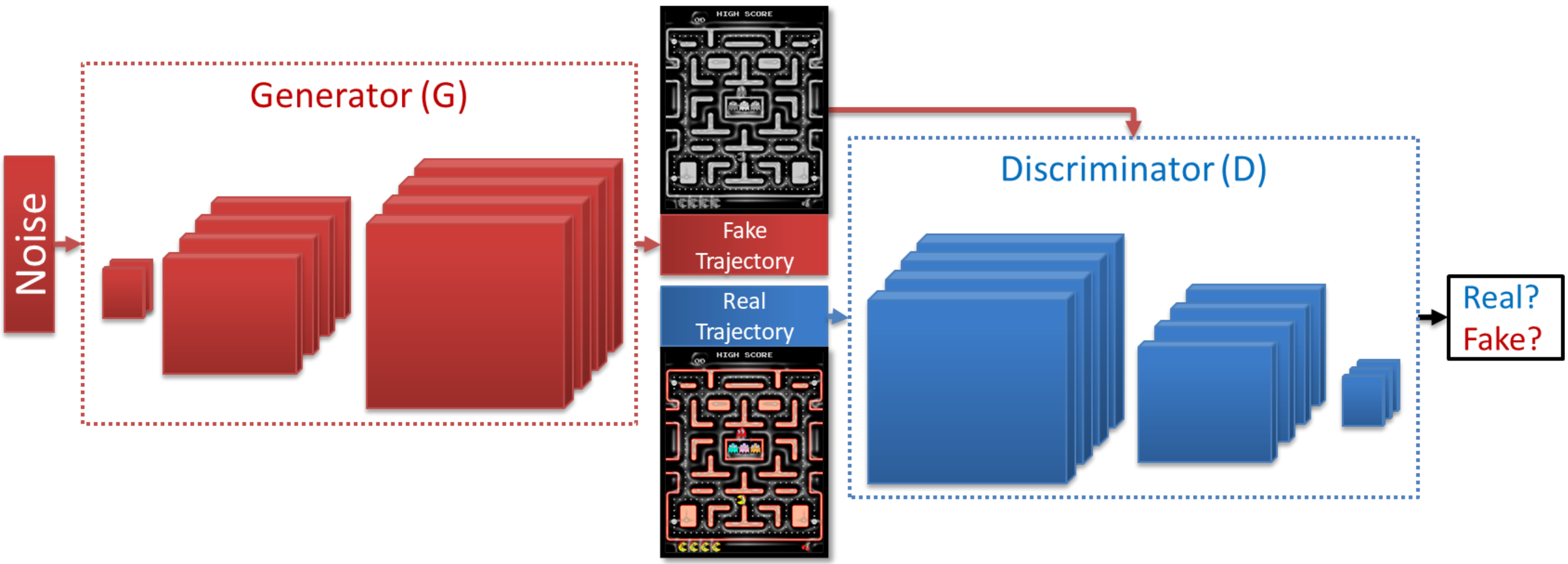
Summerville and Mateas 2016: **Mystical tutor: A magic: The gathering design assistant via denoising sequence-to-sequence learning**

Generative Adversarial Networks

Generative Adversarial Networks

- Train two networks intermittently: a generator and a discriminator
- The discriminator is trained to distinguish real artifacts from fake (generated) ones
- The generator is trained to generate artifacts that fool the discriminator
- Essentially the same idea as competitive coevolution

Generative Adversarial Networks



GANs for Mario Level Generation

Evolving Mario Levels in the Latent Space of a Deep Convolutional Generative Adversarial Network

Vanessa Volz
TU Dortmund University
Dortmund, Germany
vanessa.volz@tu-dortmund.de

Jacob Schrum
Southwestern University
Georgetown, TX USA
schrum2@southwestern.edu

Jialin Liu
Queen Mary University of London
London, UK
jialin.liu@qmul.ac.uk

Simon M. Lucas
Queen Mary University of London
London, UK
simon.lucas@qmul.ac.uk

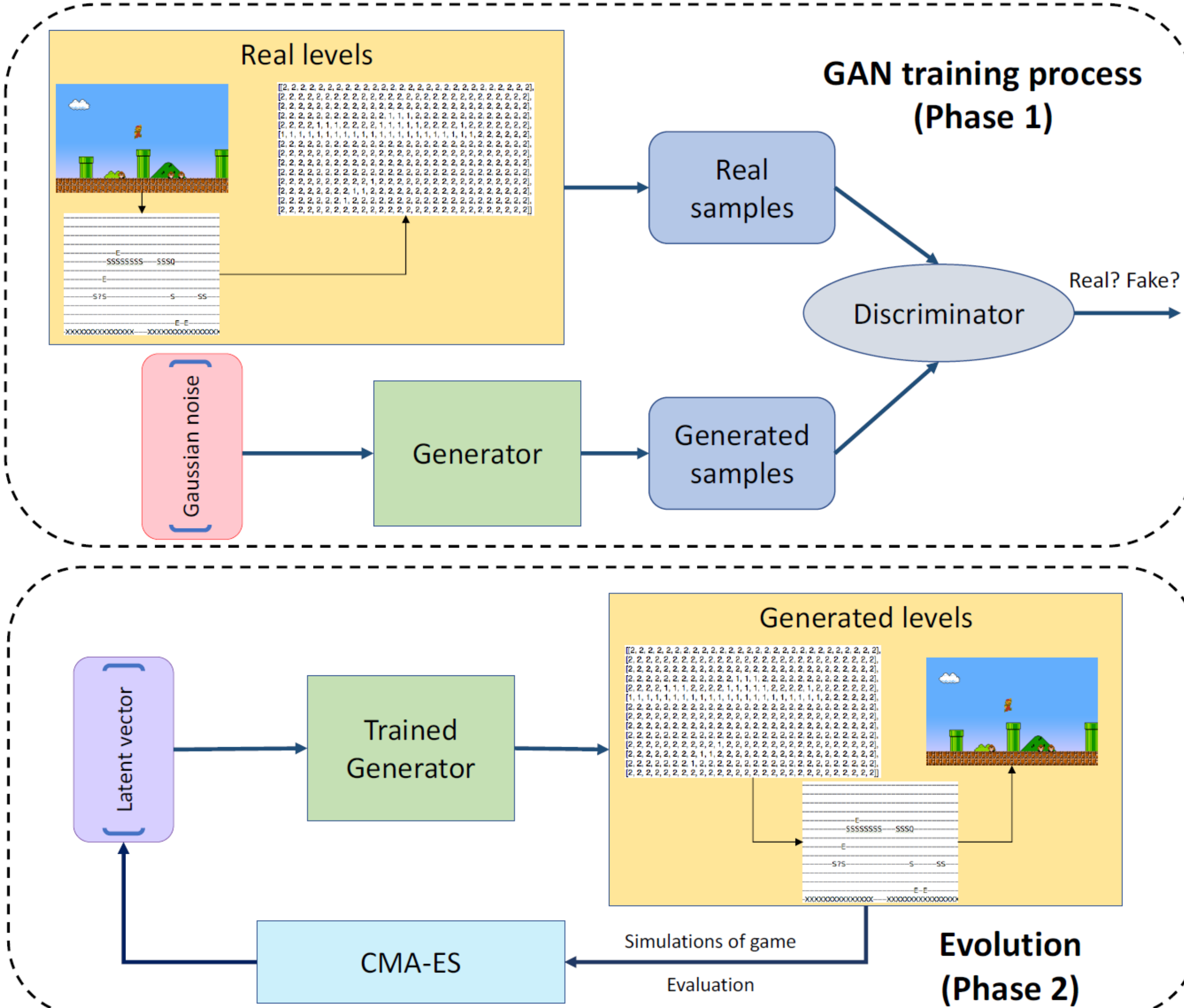
Adam Smith
University of California
Santa Cruz, CA USA
amsmith@ucsc.edu

Sebastian Risi
IT University of Copenhagen
Copenhagen, Denmark
sebr@itu.dk

- Train a GAN on a Mario level, so that it can produce level segments

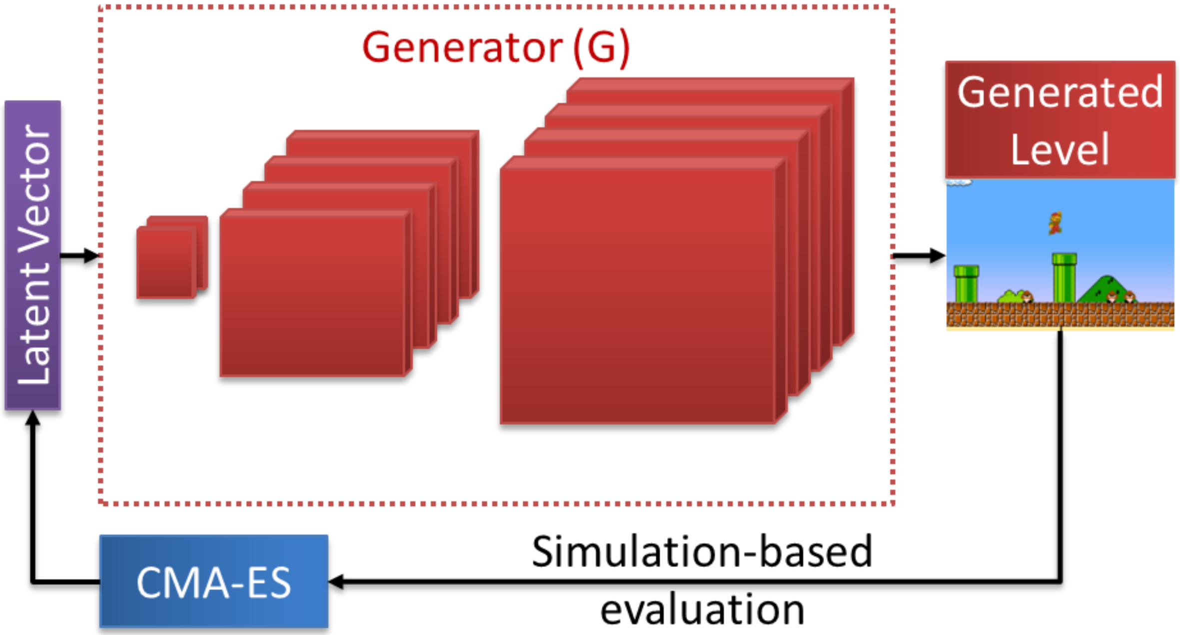
- Search the latent space of the trained GAN for level segments with particular properties

  - Approach called *Latent Variable Evolution*



Real levels
GAN training process
(Phase 1)
Real samples
Discriminator
Real? Fake?
Gaussian noise
Generator
Generated samples
Generated levels
Latent vector
Trained Generator
CMA-ES
Simulations of game
Evaluation
Evolution
(Phase 2)

GANs for Mario Level Generation



GANs for Mario Level Generation

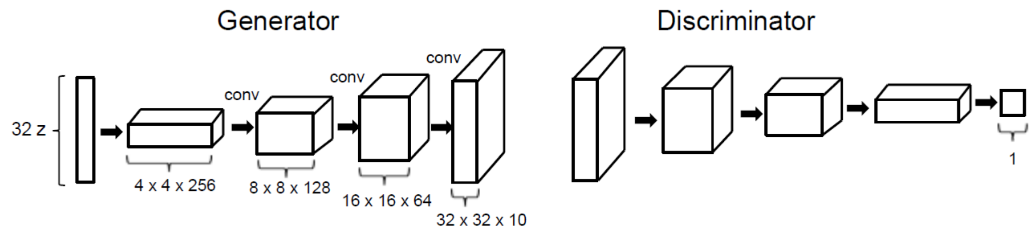


Figure 3: The Mario DCGAN architecture.

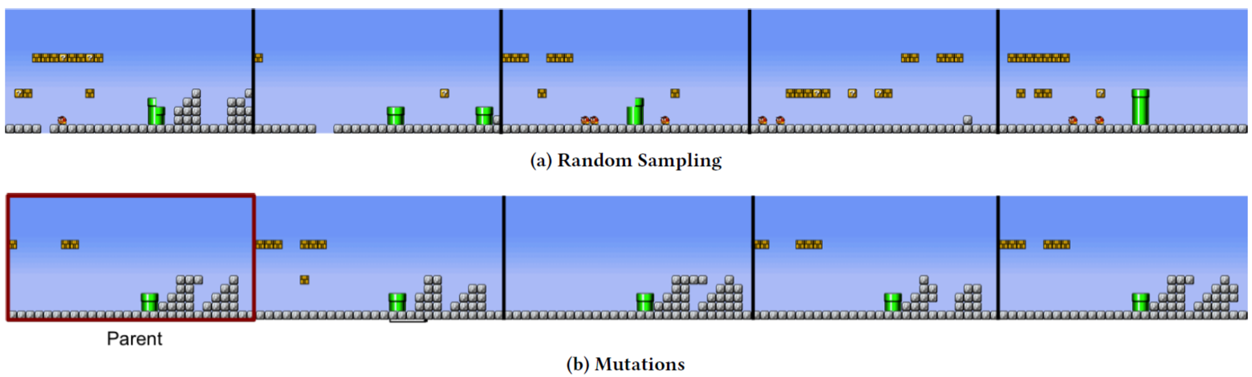
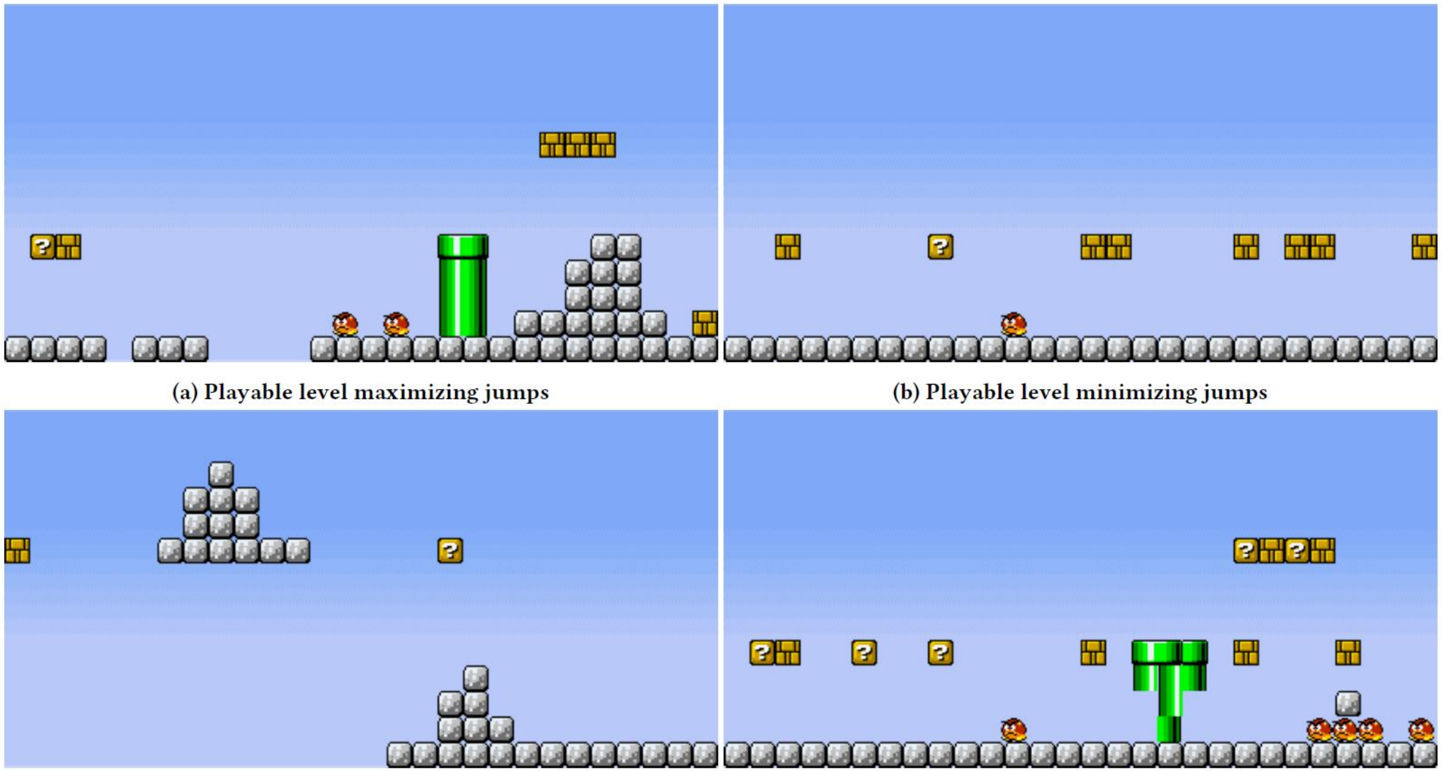


Figure 4: Generated Examples. Shown are samples produced by the GAN by (a) sampling random latent vectors, and (b) randomly mutating a specific latent vector. The main result is that the generator is able to produce a wide variety of different level layouts, but varied offspring still resemble their parent.

GANs for Mario Level Generation



(a) Playable level maximizing jumps

(b) Playable level minimizing jumps

(c) Unplayable level

(d) Broken titles

Figure 7: Agent-based optimization examples. (a) and (b) show good examples of levels in which the number of jumps is maximized (F_1), and minimized (F_2), respectively. (c) shows an example of one of the worst individuals found (not playable, F_1). An example of an individual that reaches high fitness (maximizing jumps, F_1) but has broken titles is shown in (d).

GANs for Mario Level Generation

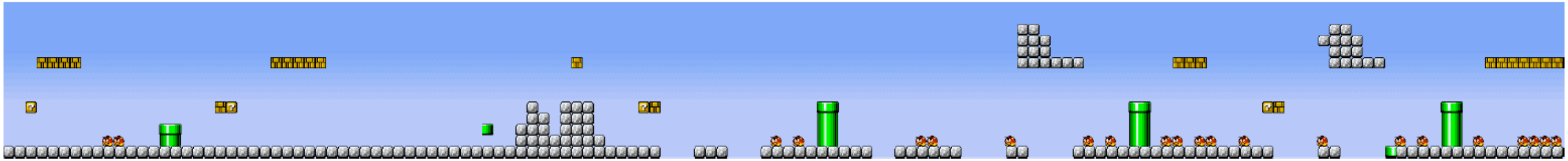


Figure 6: Level with increasing difficulty. Our LVE approach can create levels composed of multiple parts that gradually increase in difficulty (less ground tiles, more enemies). In the future this approach could be used to create a level in real-time that is tailored to the particular skill of the player (dynamic difficulty adaptation).

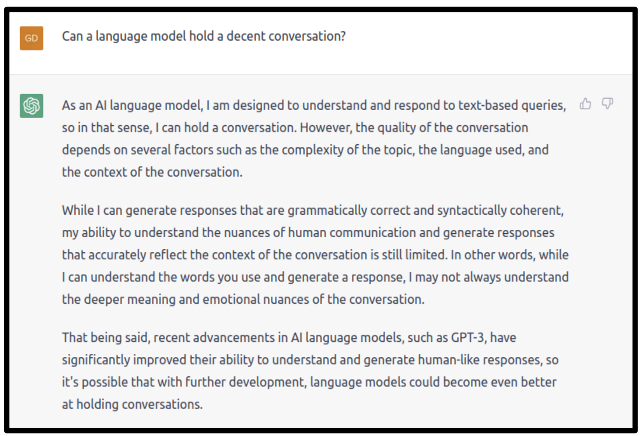
LLMs

LLMs

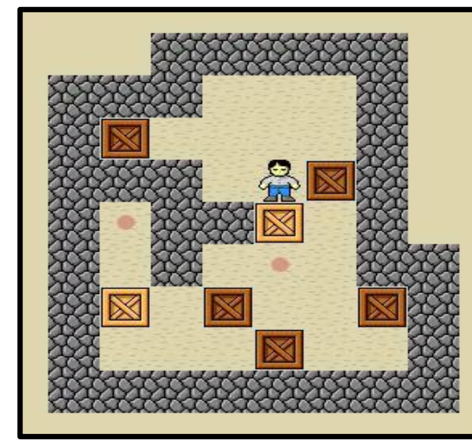
- Text-to-image, text-to-text, text-to-music, text-to-code etc works well…
- Can we use text-conditioning in our content generation?
- Could we simply ask a model to create a new game for us?

LLMs

Large Language Models (LLMs) have demonstrated impressive capabilities for a variety of tasks, from chatting to generating computer code. But what about generating working game levels?




```
def longest_consecutive_char(string):
    longest = 0
    current = 0
    for i in range(len(string)):
        if i == 0:
            current = 1
        elif string[i] == string[i-1]:
            current += 1
        else:
            current = 1
        if current > longest:
            longest = current
    return longest
```



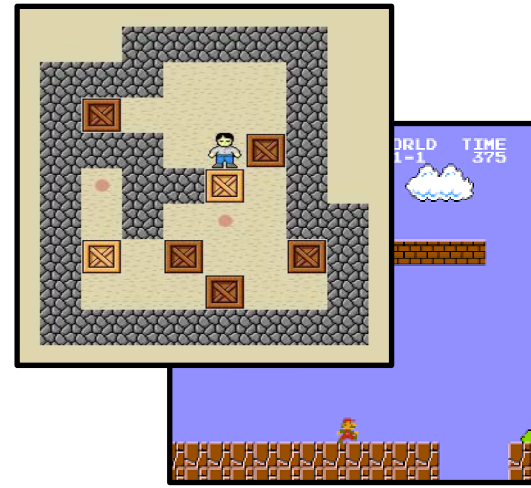
Todd, Earle, Nasir, Togelius (2023): **Level Generation Through Large Language Models**. PCG Workshop at FDG.

Opportunities and Challenges

"Give me a **challenging** level with **7 boxes**"

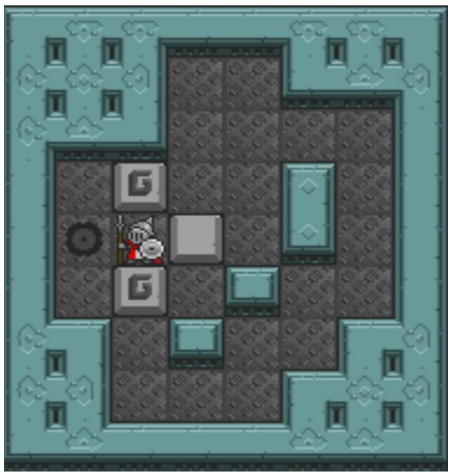
Controllability

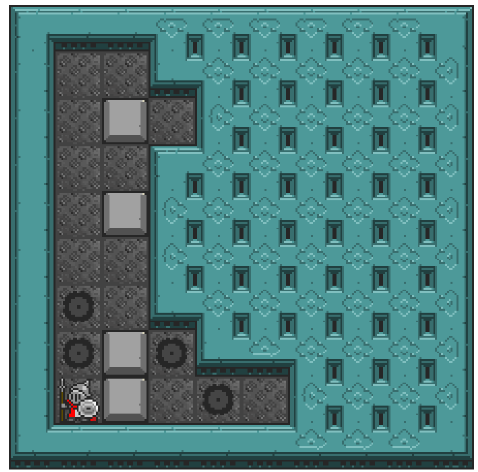
"Give me a level for **Sokoban** and another level for **Super Mario Bros**"

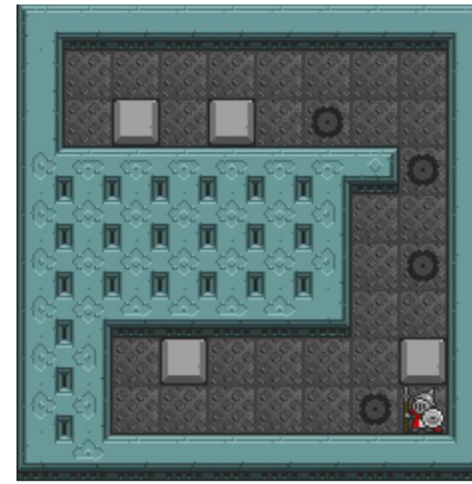


Generalizability

(Ancient) LLMs Can Make Sokoban Levels





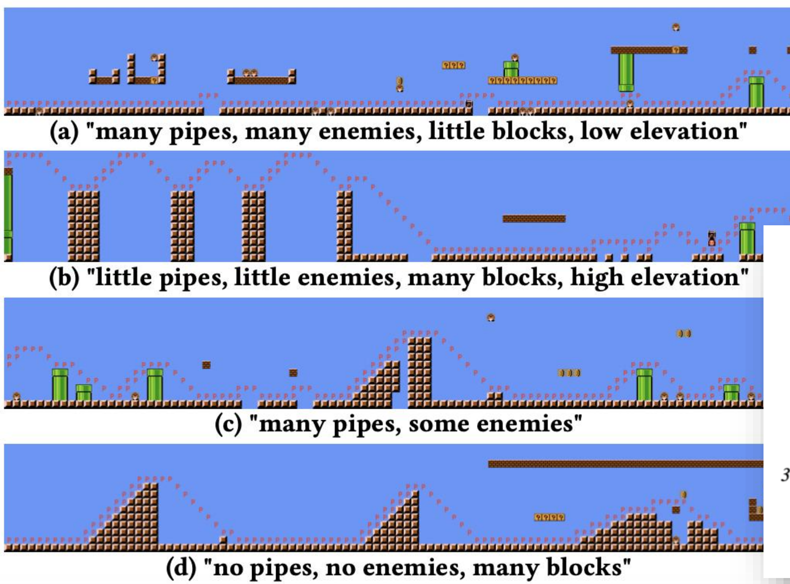


Valid, playable, novel Sokoban levels sampled from GPT-2

MarioGPT

MarioGPT: Open-Ended Text2Level Generation through Large Language Models

Shyam Sudhakaran[1], Miguel González-Duque[1], Claire Glanois[1],
Matthias Freiberger[1], Elias Najarro[1], Sebastian Risi[1,2]

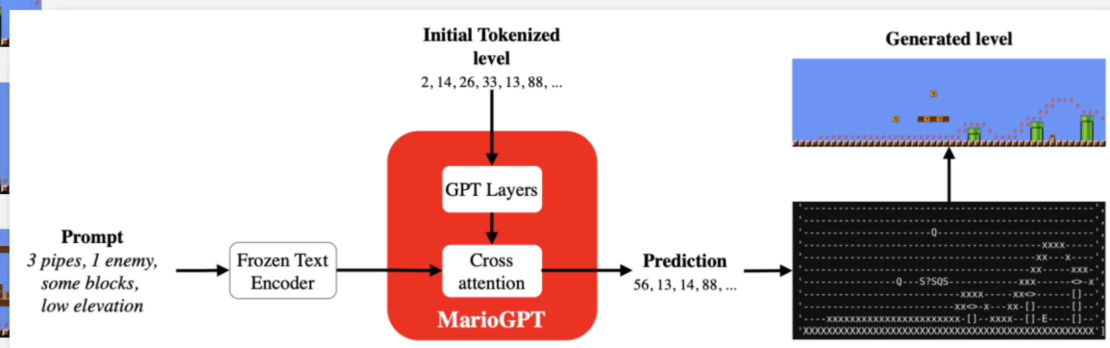


(a) "many pipes, many enemies, little blocks, low elevation"

(b) "little pipes, little enemies, many blocks, high elevation"

(c) "many pipes, some enemies"

(d) "no pipes, no enemies, many blocks"



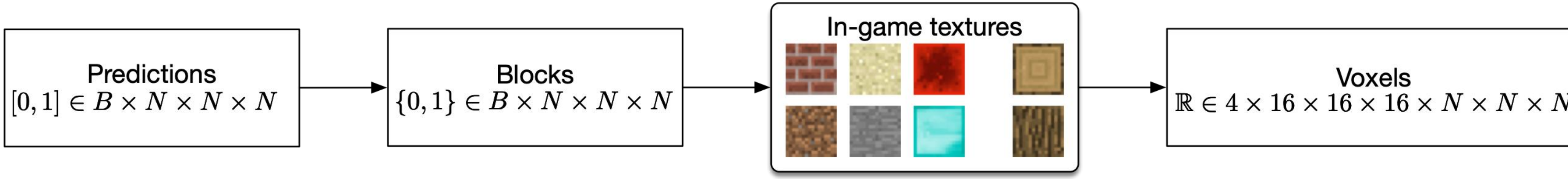
Text-Guided 3D *Minecraft* Generation

- Previous controllable level generators are controlled via **domain-specific computable metrics** (e.g. solution length, number of enemies, symmetry).
- Could we allow for more open-ended control via **natural language**?
- Can we leverage a model trained with knowledge over **many domains**?

generative model

Earle et al. (2024). Dreamcraft: **Text-guided generation of functional 3D environments in Minecraft.** In *Proceedings of the 19th International Conference on the Foundations of Digital Games* (pp. 1-15).

Text-Guided 3D *Minecraft* Generation



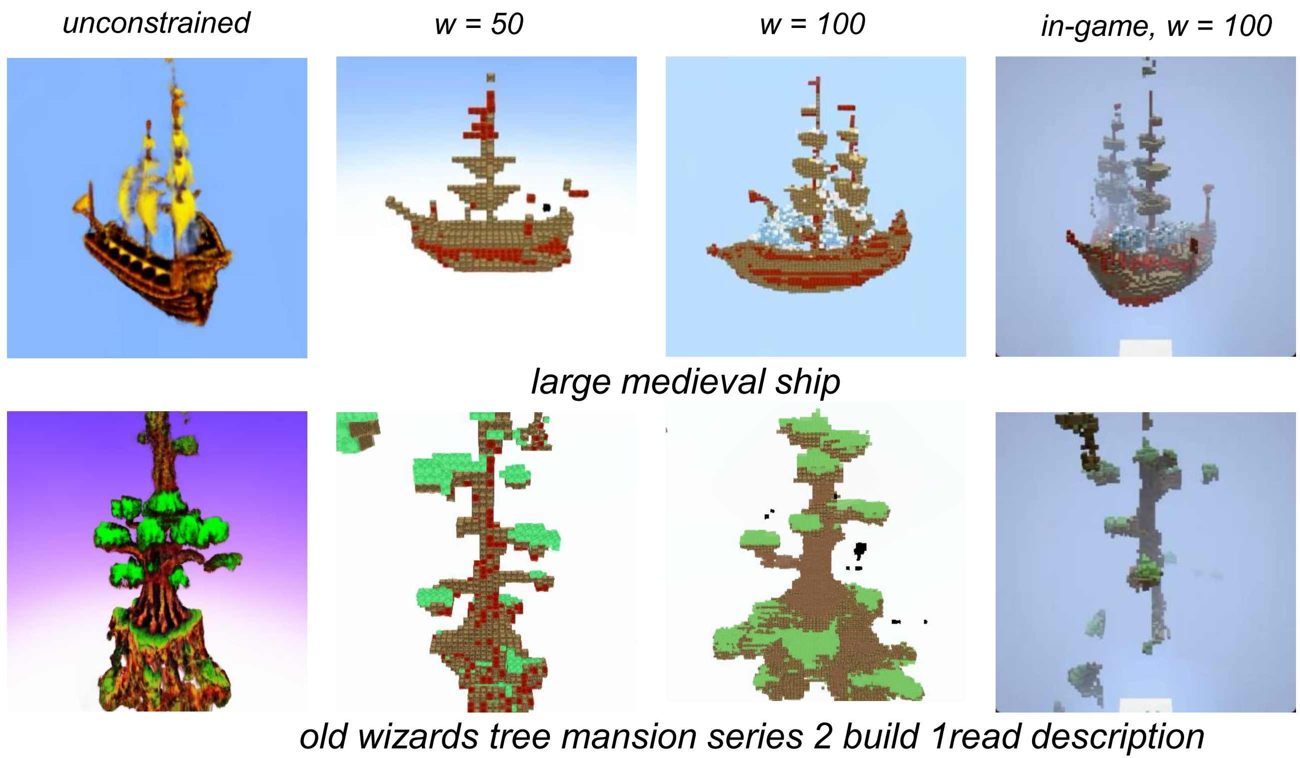
NeRF parametrizes a 3D grid, **predicting block types**

Discretize grid using "straight through" trick → **discrete blocks**

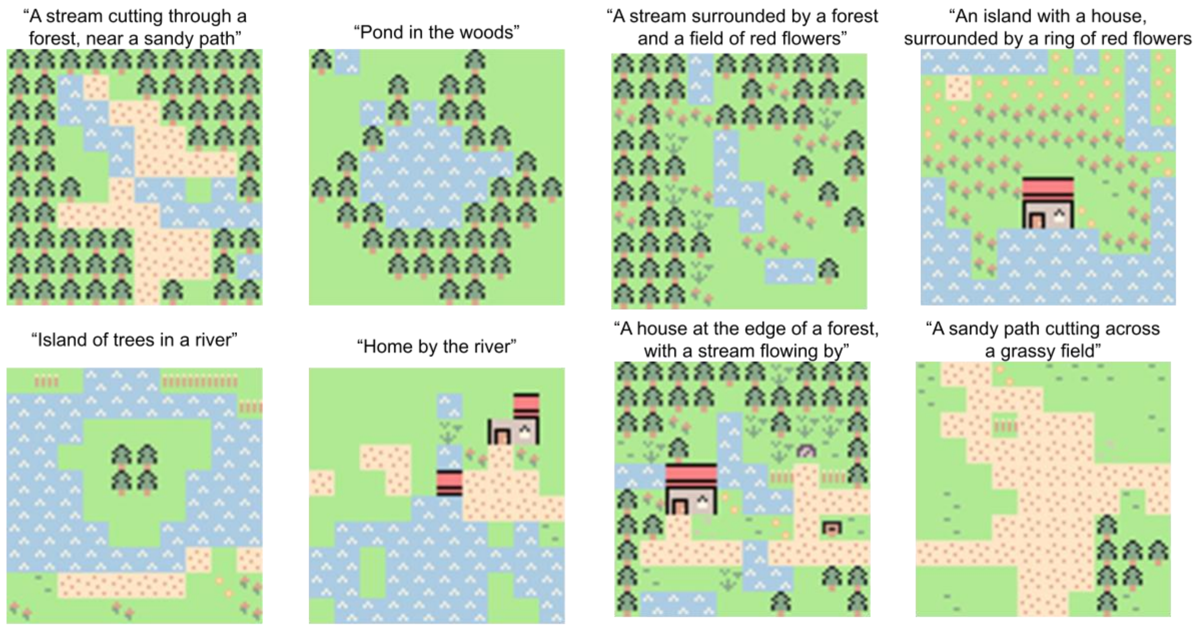
Project the one-hot grid into a point cloud comprising in-game textures

Differentiably render resultant **voxel grid**.

Text-Guided 3D *Minecraft* Generation

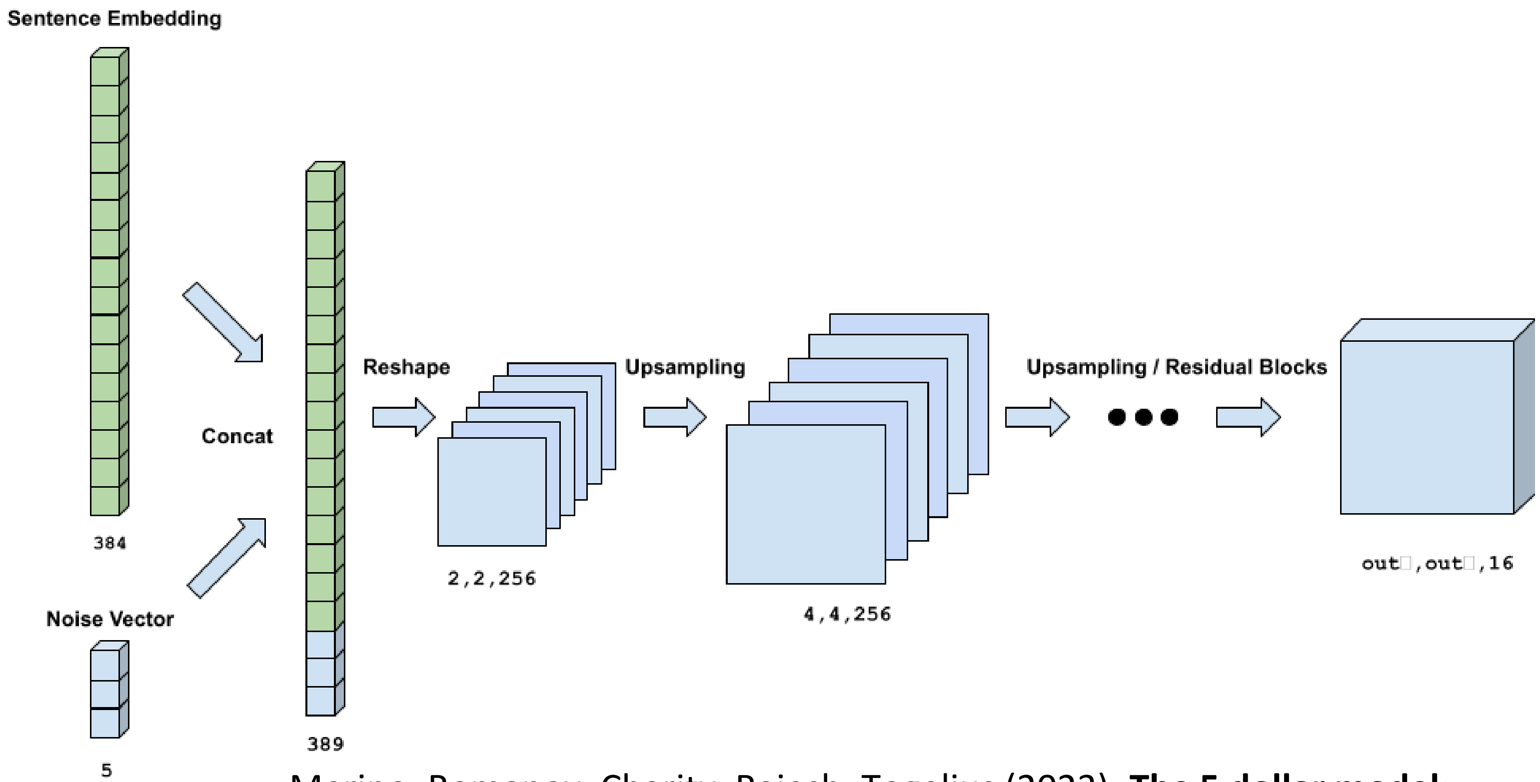


The $5 Model



Merino, Romanov, Charity, Rajesh, Togelius (2023): **The 5 dollar model: Generating Game Maps and Sprites from Sentence Embeddings**

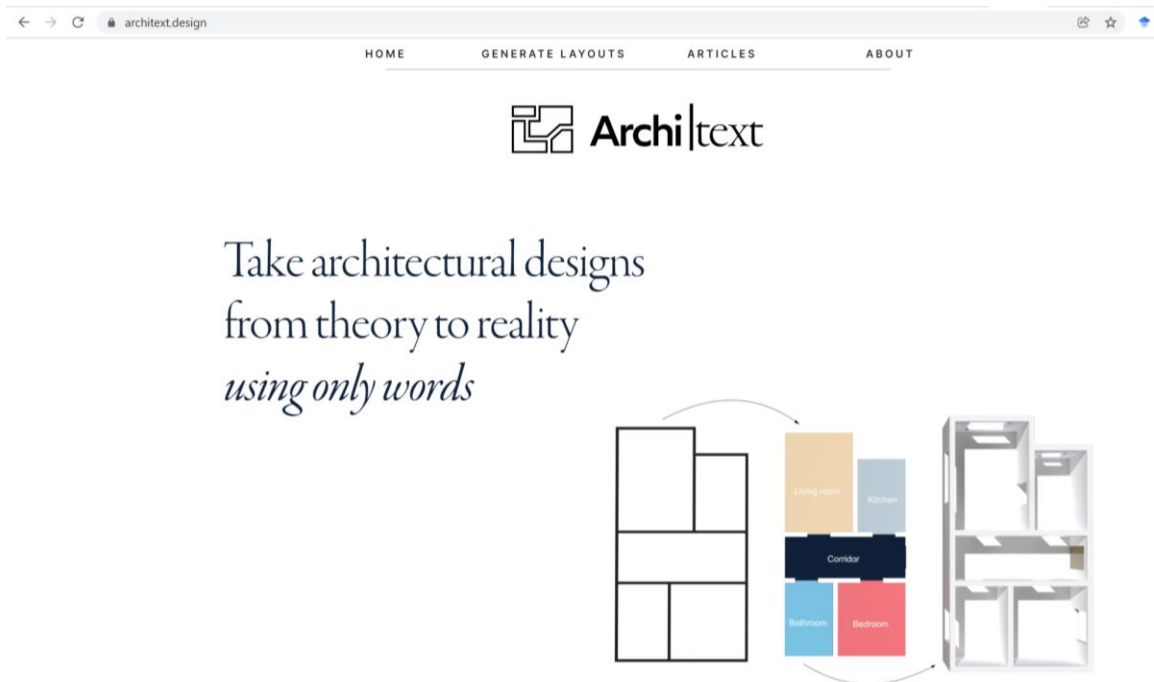
The $5 Model



Merino, Romanov, Charity, Rajesh, Togelius (2023): **The 5 dollar model: Generating Game Maps and Sprites from Sentence Embeddings**

The $5 Model

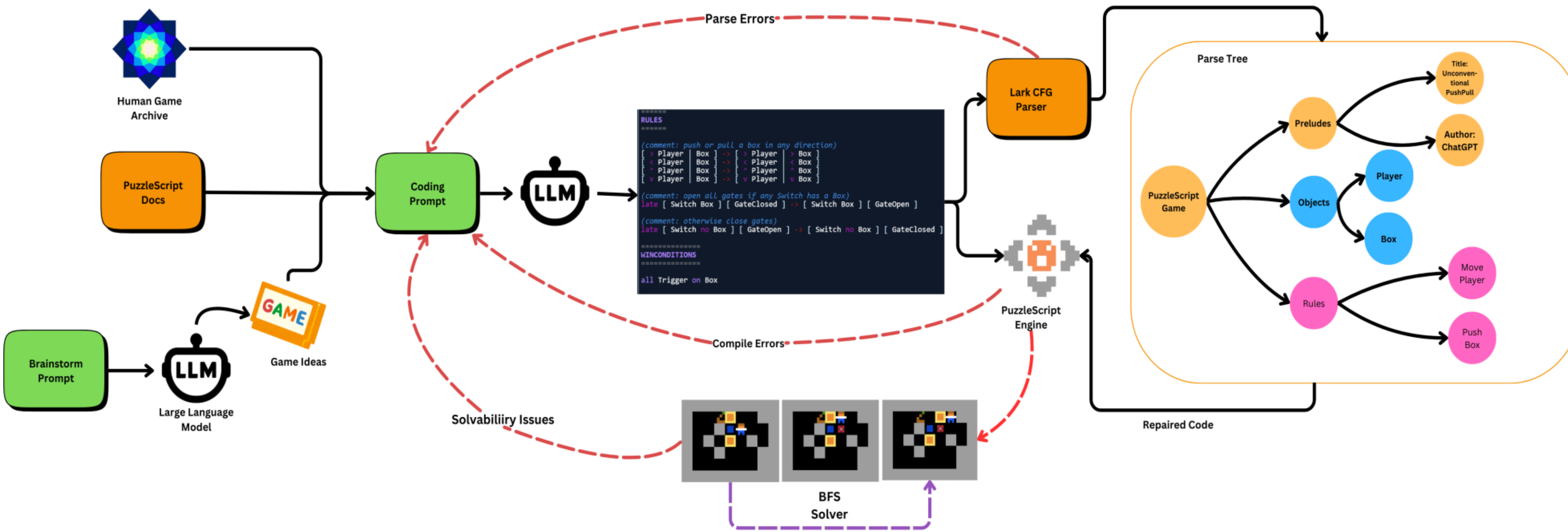
ArchiText: Text to Architecture



Galanos, T., Liapis, A., & Yannakakis, G. N. (2023). **Architext: Language-Driven Generative Architecture Design**. *arXiv preprint arXiv:2303.07519*.

ScriptDoctor

Earle et al, "Automatic Generation of PuzzleScript Games via Large Language Models and Tree Search", arxiv 2025



Fewshot-prompt LLM with existing games, feed back parse/compile issues.

Feed back results from tree search-based solver agent.

ScriptDoctor

Earle et al, “Automatic Generation of PuzzleScript Games via Large Language Models and Tree Search”, arxiv 2025

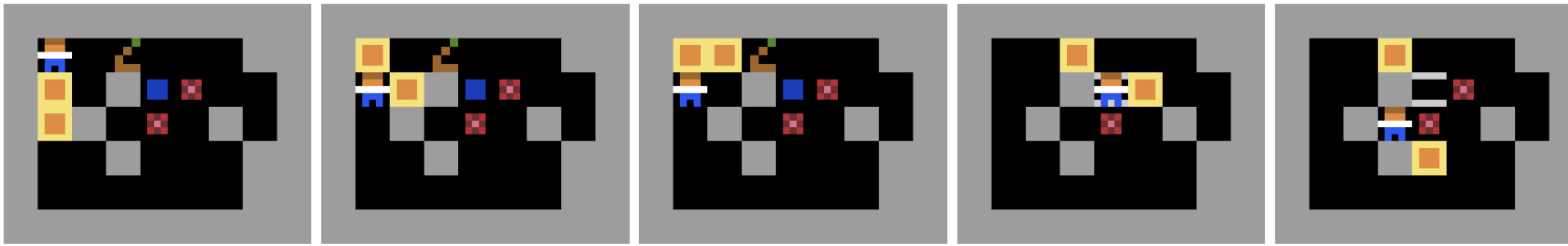


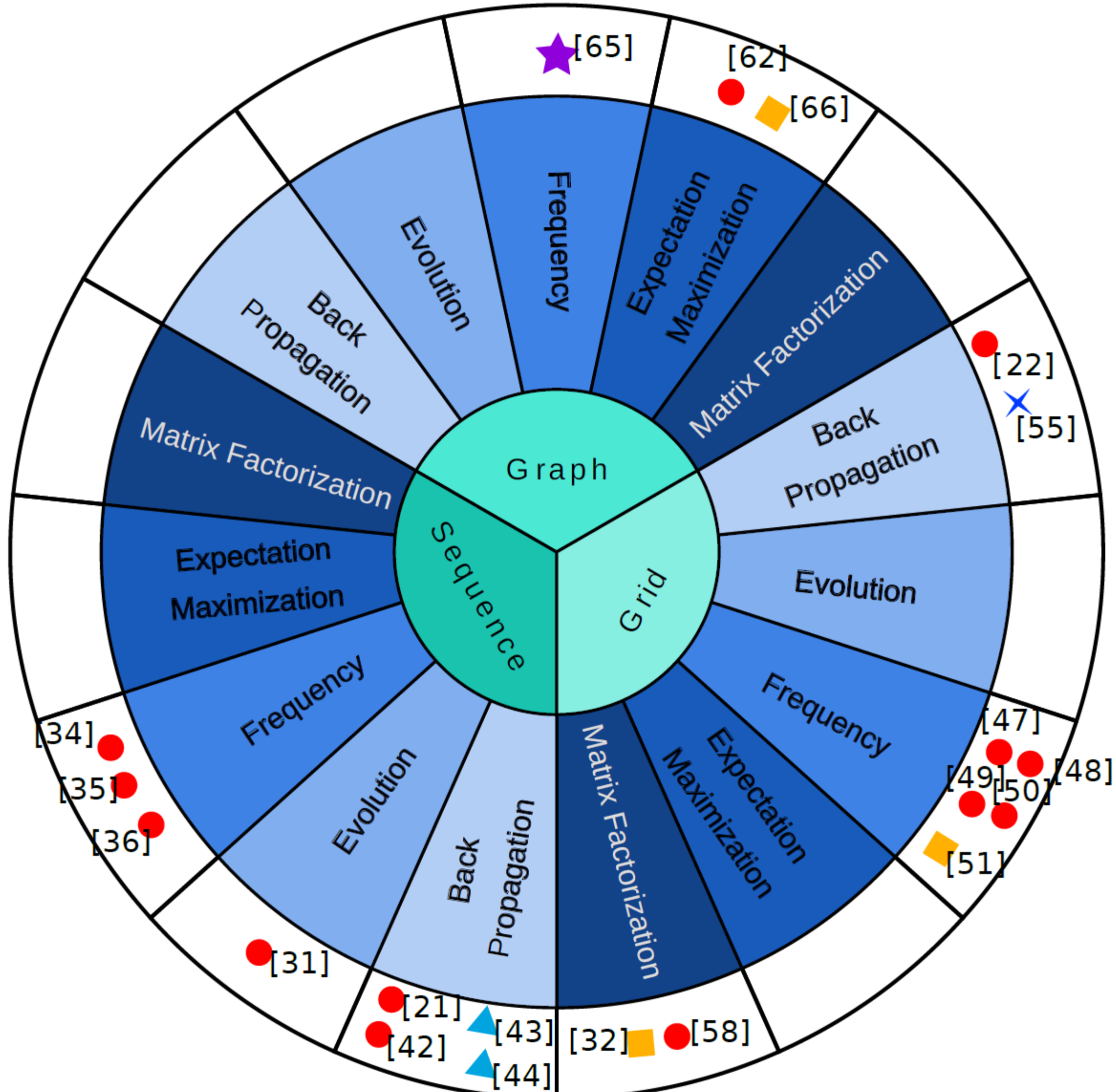
Fig. 2: Select frames from a solution to the *Unconventional PushPull* puzzle generated by ScriptDoctor with o1. The player

- Most generated games were mazes, or sokoban-likes.
- A few were actually interesting and (seemingly) novel.

```
RULES

[ > Player  |  Box ] -> [ > Player  |  > Box ]
[ < Player  |  Box ] -> [ < Player  |  < Box ]
[ ^ Player  |  Box ] -> [ ^ Player  |  ^ Box ]
[ V Player  |  Box ] -> [ V Player  |  V Box ]
late [ Switch Box ] [ GateClosed ]
-> [ Switch Box ] [ GateOpen ]
late [ Switch no Box ] [ GateOpen ]
-> [ Switch no Box ] [ GateClosed ]

WINCONDITIONS

all Trigger on Box
```

AI People

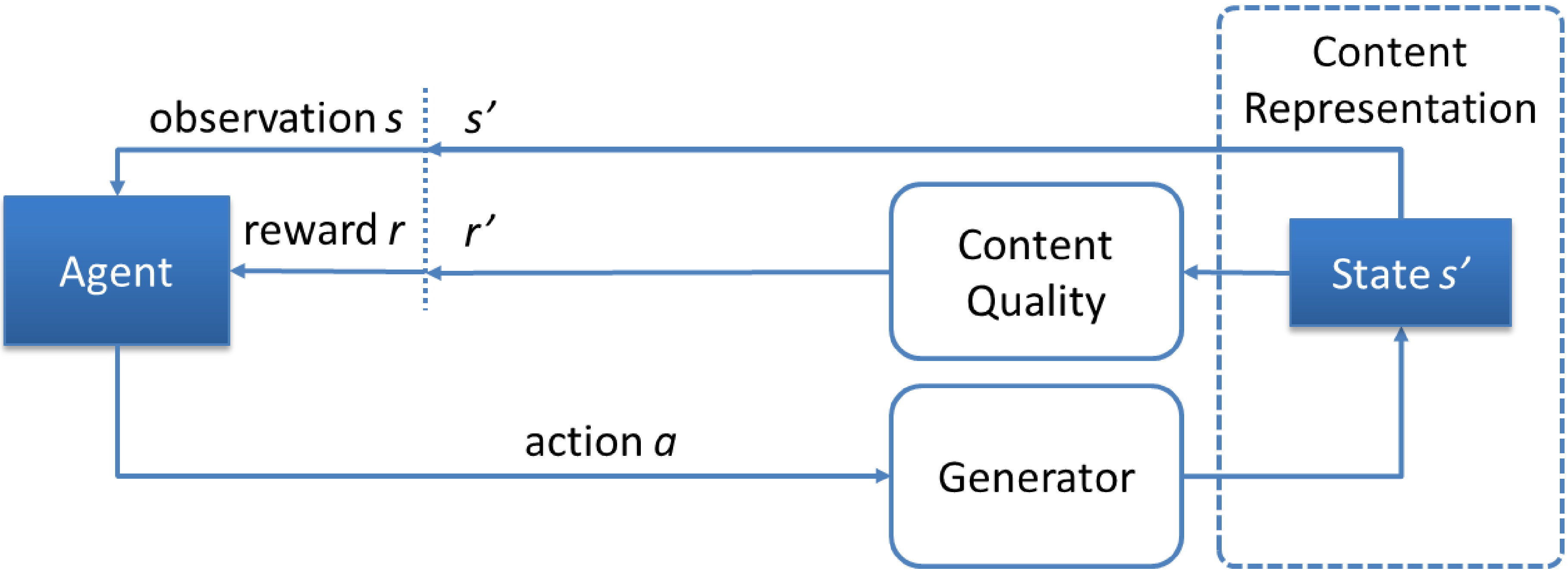
Gallotta, Todd, Zammit, Earle, Liapis, Togelius & Yannakakis.
Large Language Models and Games: A Survey and Roadmap
IEEE Transactions on Games, 2024.



Graph
Grid
Sequence
Frequency
Expectation Maximization
Matrix Factorization
Back Propagation
Evolution
Back Propagation
Evolution
Frequency
Expectation Maximization
Matrix Factorization
Back Propagation
Evolution
Frequency
Expectation Maximization
Matrix Factorization
[65]
[62]
[66]
[22]
[55]
[47]
[48]
[49]
[50]
[51]
[32]
[58]
[21]
[43]
[42]
[44]
[31]
[34]
[35]
[36]

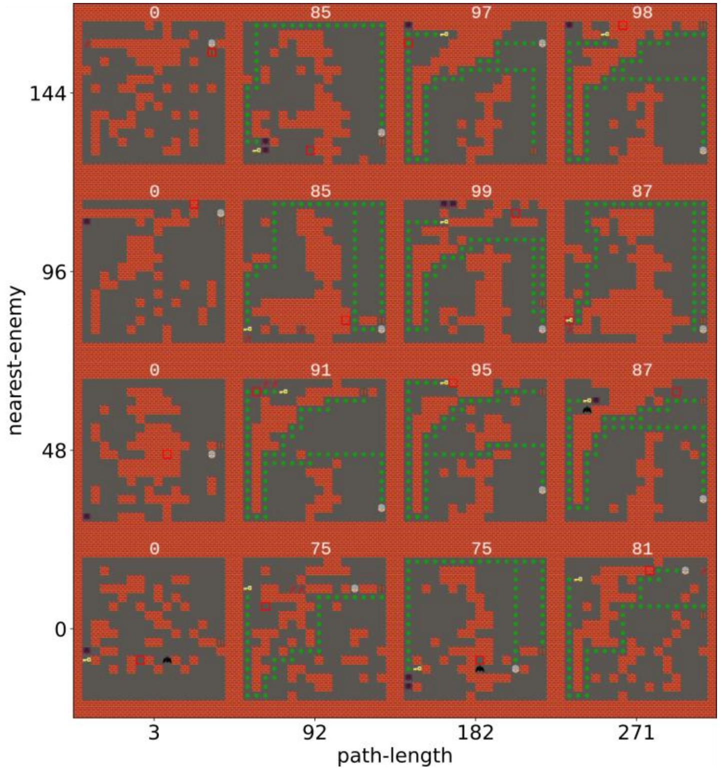
Reinforcement Learning

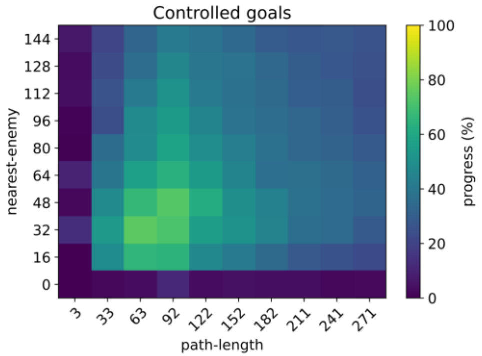
Procedural Content Generation via Reinforcement Learning (PCGRL)



Khalifa, A., Bontrager, P., Earle, S., & Togelius, J. (2020). **PCGRL: Procedural content generation via reinforcement learning**. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

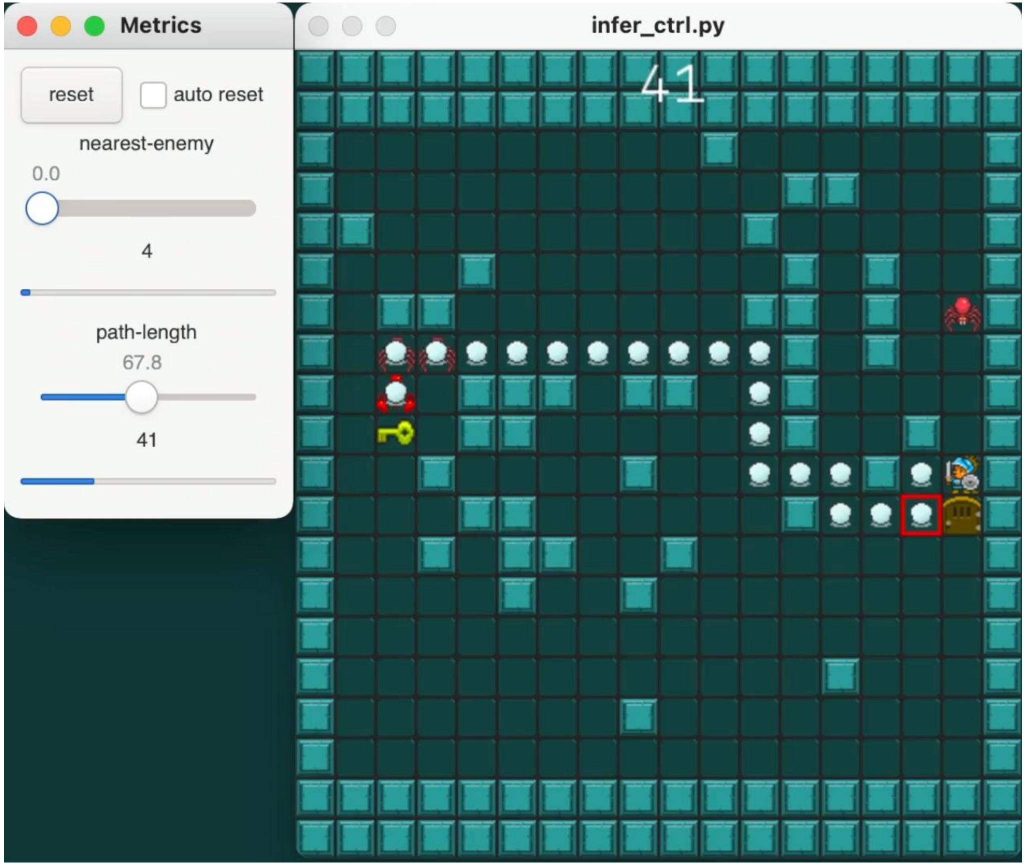
PCGRL



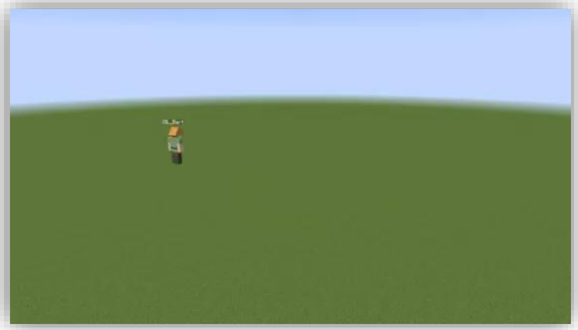


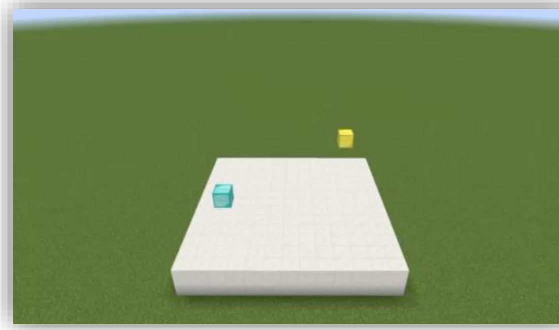
Earle, S., Edwards, M., Khalifa, A., Bontrager, P., & Togelius, J. (2021). Learning controllable content generators. IEEE CoG

Controllable PCGRL



3D PCGRL


Maximize diameter

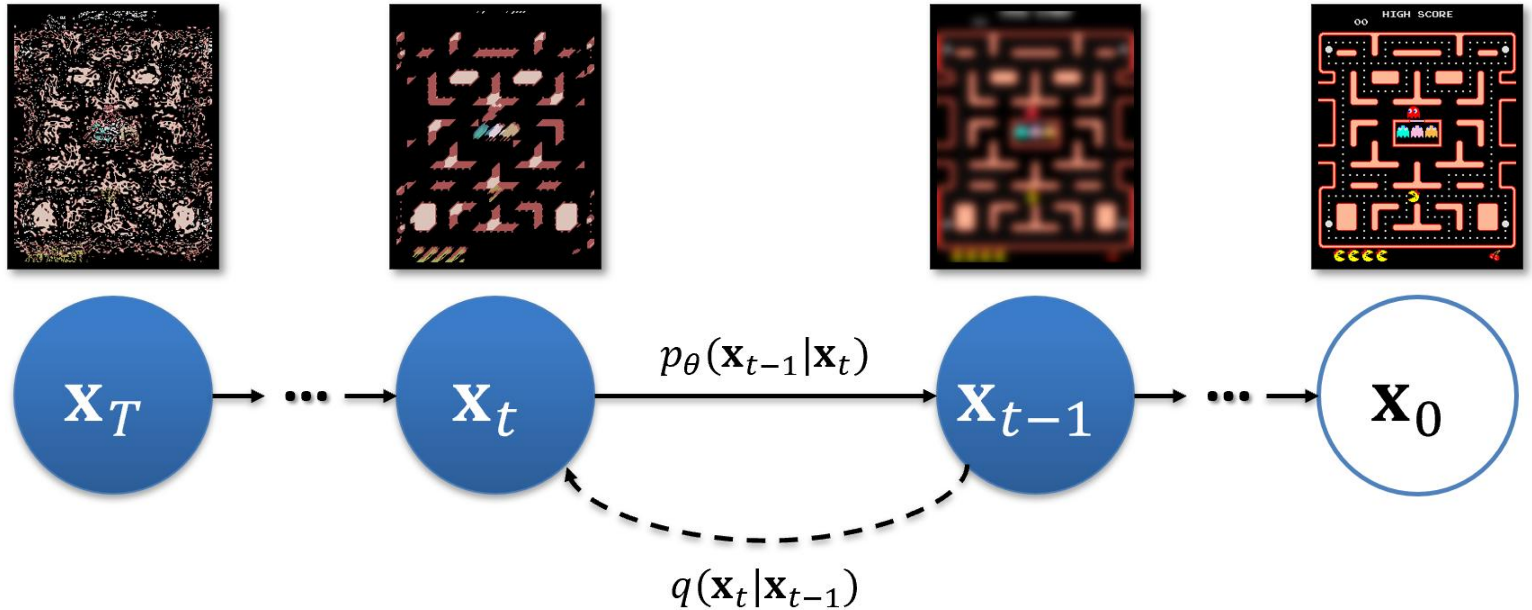

Maximize path-length


Dungeon: maximize path-length


Max. diameter w/ NCAs

Jiang, Z., Earle, S., Green, M., & Togelius, J. (2022). **Learning Controllable 3D Level Generators**. In *Proceedings of the 17th International Conference on the Foundations of Digital Games*.

Diffusion Models

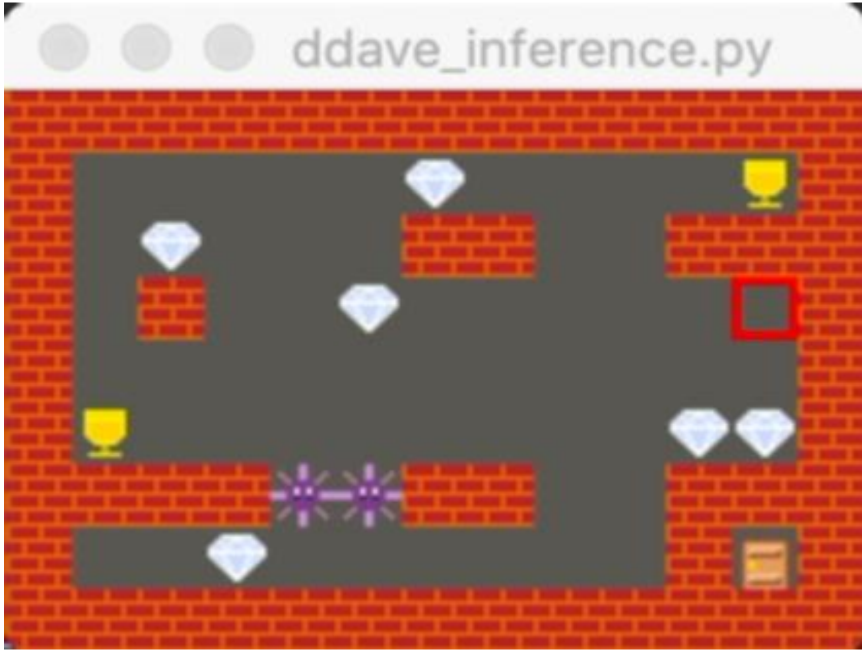


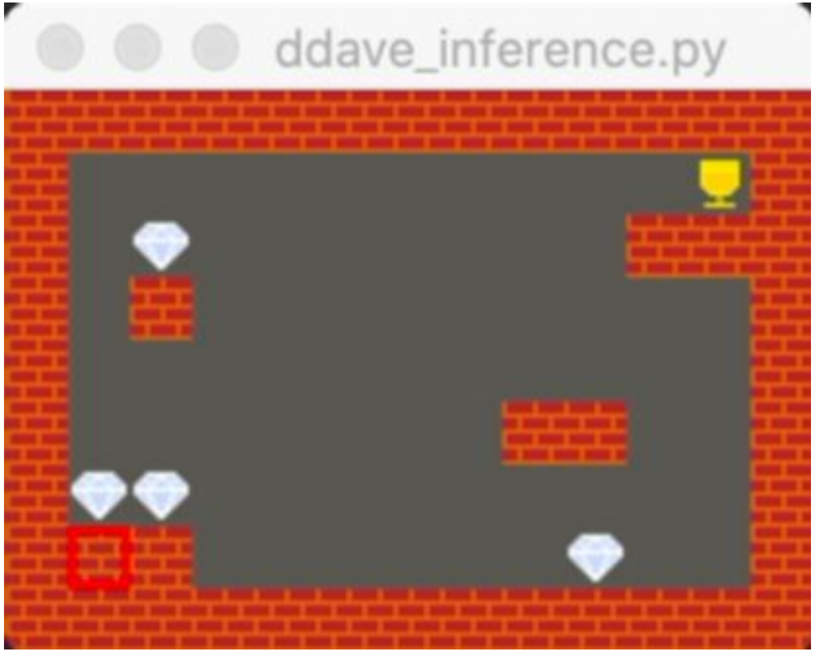
The Path of Destruction

- Can we learn iterative generators from data?
- Approach: destroy the level in a myriad ways, use the reverse of the destruction steps as a dataset to train on
- Basically, learn to repair, and see creativity as repair from a random state

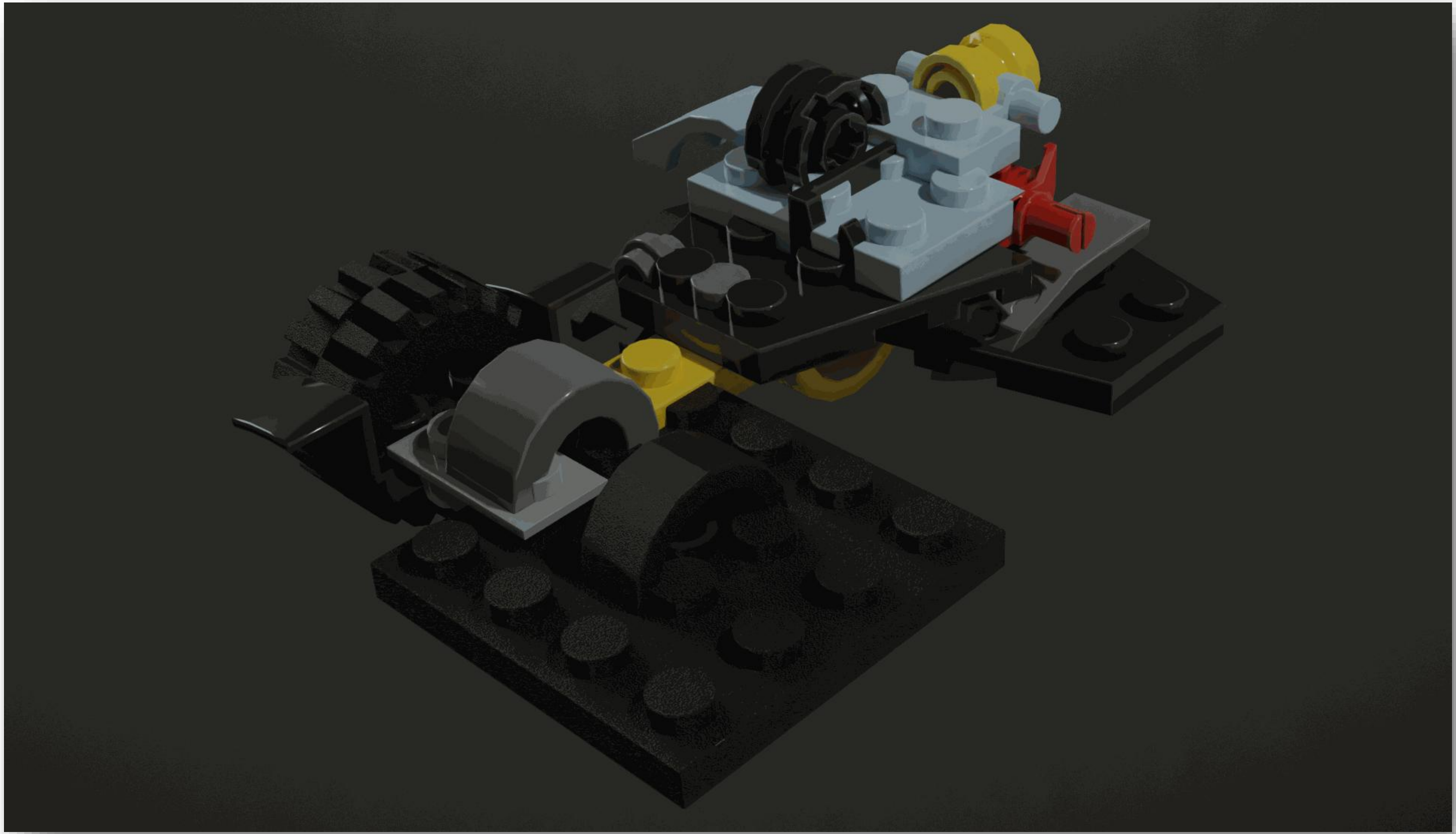
Matthew Siper, Ahmed Khalifa, Julian Togelius (2022): **Path of Destruction: Learning an Iterative Level Generator from a Small Dataset**. IEEE SSCI

PoD Agents Generating Levels

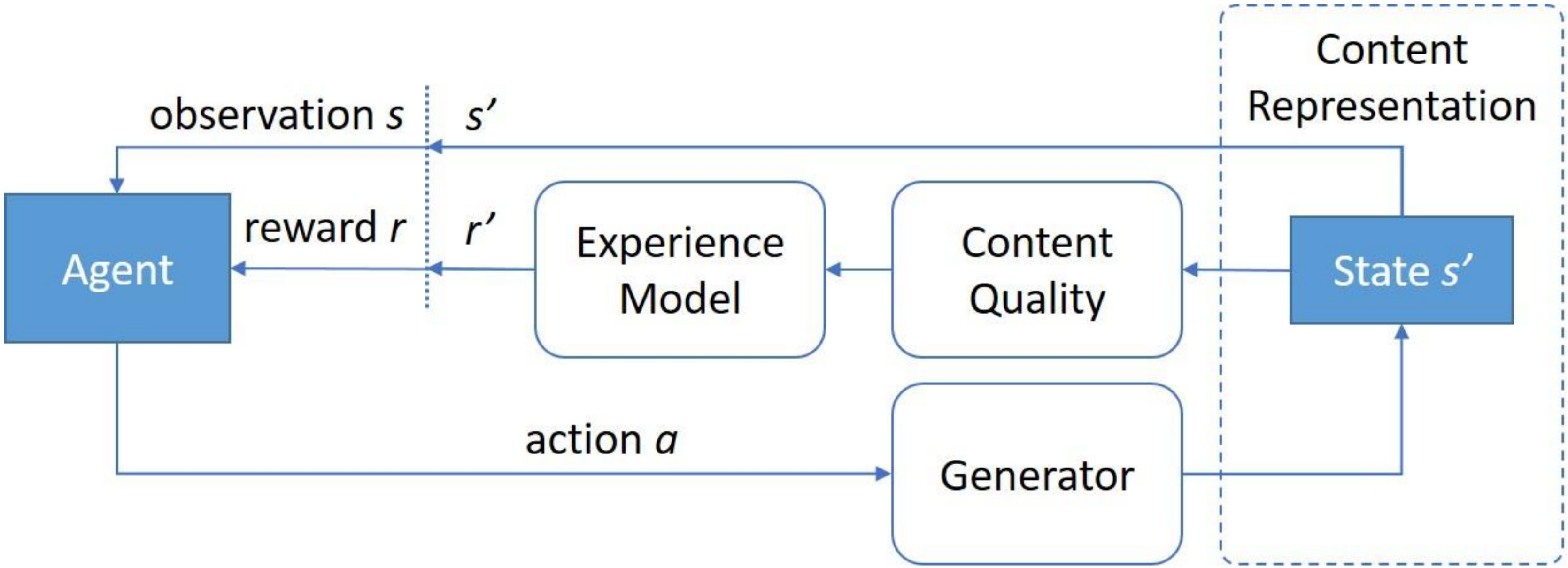




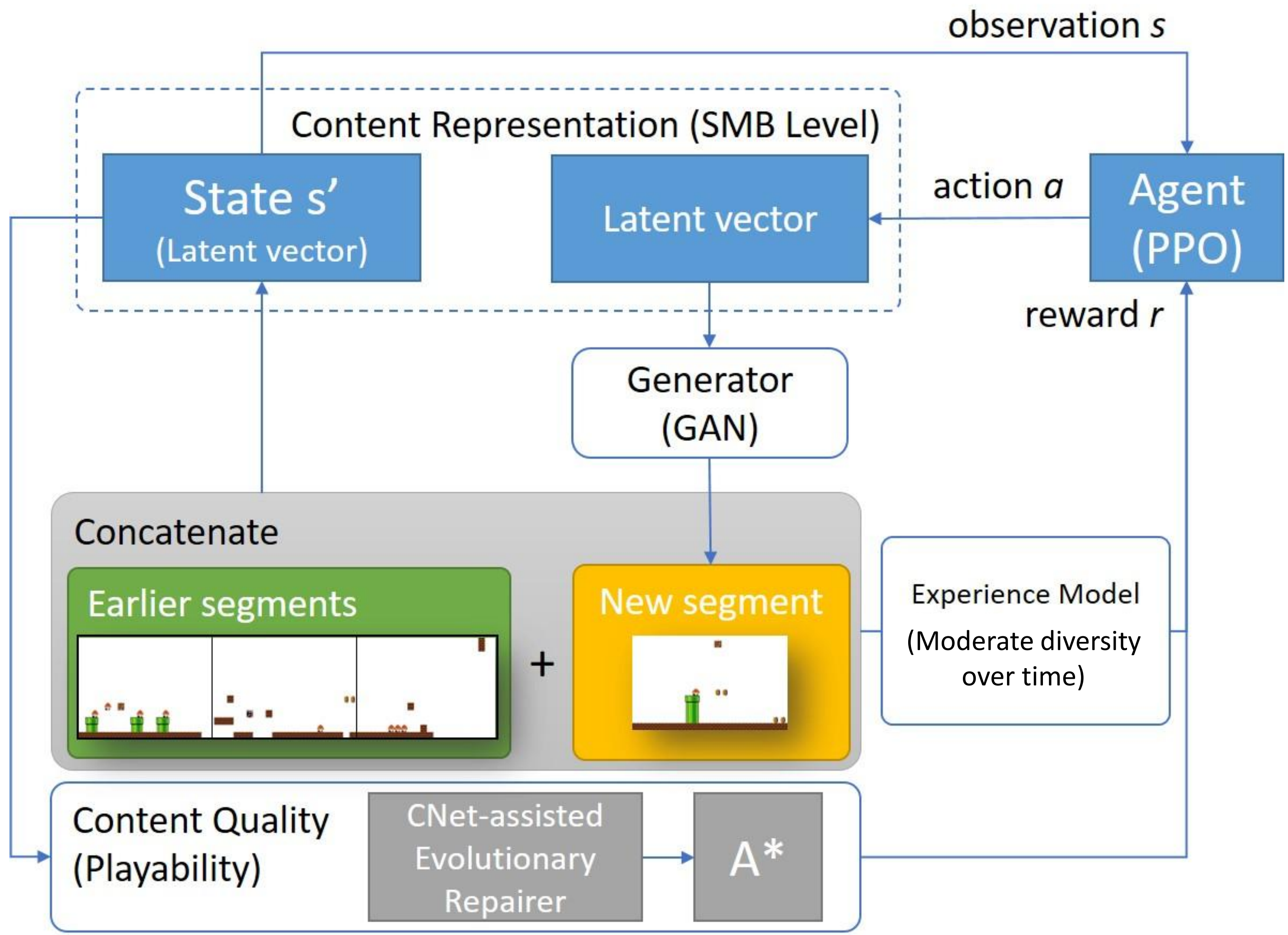
A PoD Agent Building a Car in Lego



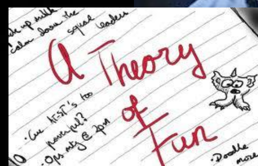
Shu, Liu and Yannakakis, **Experience-Driven PCG via Reinforcement Learning: A Super Mario Bros Study**, *IEEE Conference on Games*, 2021



Combining ED(PCG)RL: EDRL

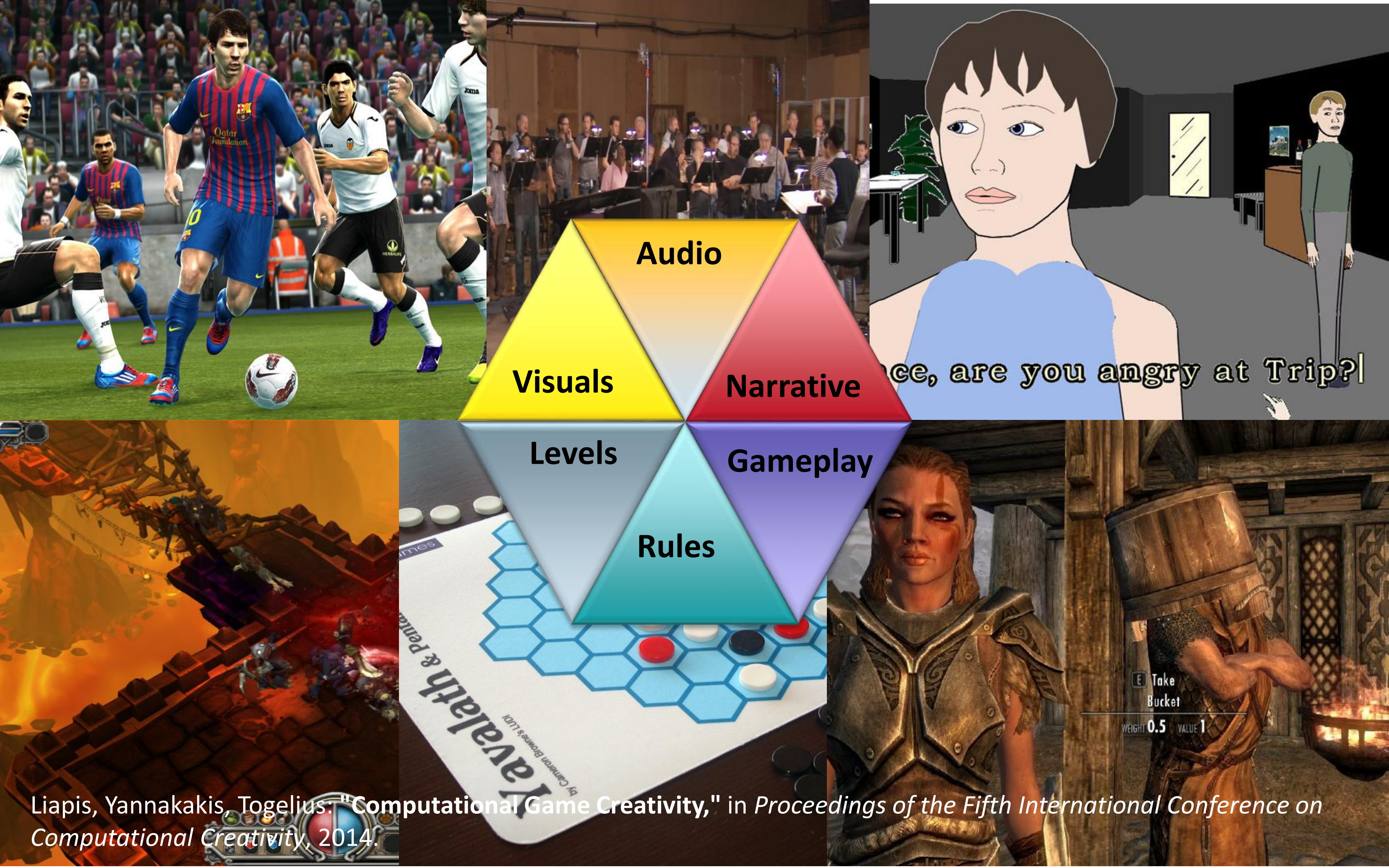


observation s
Content Representation (SMB Level)
State s'
(Latent vector)
Latent vector
action a
Agent
(PPO)
reward r
Generator
(GAN)
Concatenate
Earlier segments
+
New segment
Experience Model
(Moderate diversity over time)
Content Quality
(Playability)
CNet-assisted
Evolutionary
Repairer
A*



Wang, Liu and Yannakakis, **The Fun Facets of Mario: Multifaceted Experience-Driven PCG via Reinforcement Learning**, *In Proc. of FDG,* 2022

Chapter 9: PCG By Content Type

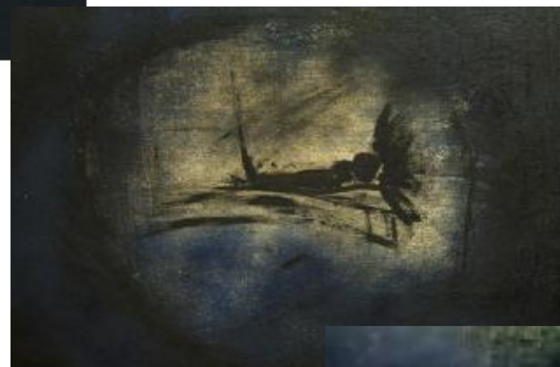


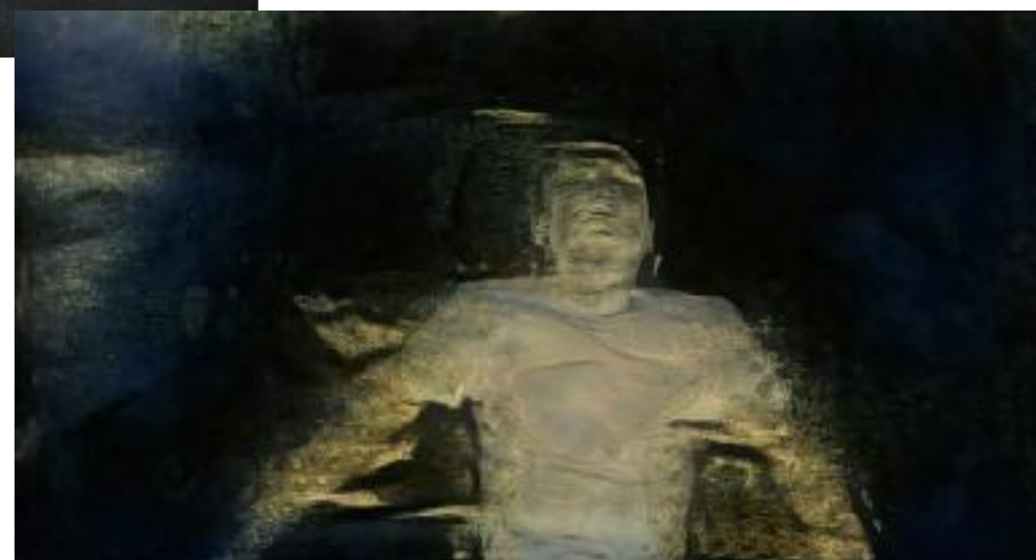
Liapis, Yannakakis, Togelius: "**Computational Game Creativity,**" in *Proceedings of the Fifth International Conference on Computational Creativity*, 2014.


Level
design

Complete Game Generation – Orchestration

Antonios Liapis, Georgios N. Yannakakis, Mark J. Nelson, Mike Preuss and Rafael Bidarra: **"Orchestrating Game Generation"** in Transactions on Games, 2019.





Visuals

All of the music you hear in this video was composed in real time by the program.

Brown, Daniel. "**Mezzo: An adaptive, real-time composition program for game soundtracks.**" *Proceedings of the AIIDE Workshop on Musical Metacreativity*. 2012.


Narrative
are you angry at Trip?

Orkin, J., and Roy, D. 2007. **The restaurant game: Learning social behavior and language from thousands of players online**. Journal of Game Development 3(1):39–60.

YAVALATH (#2)

```
(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))))
  )
)
```

Rules

Browne, Cameron. "Yavalath." *Evolutionary Game Design*. Springer London, 2011. 75-85.

Gameplay

Gameplay

Gameplay

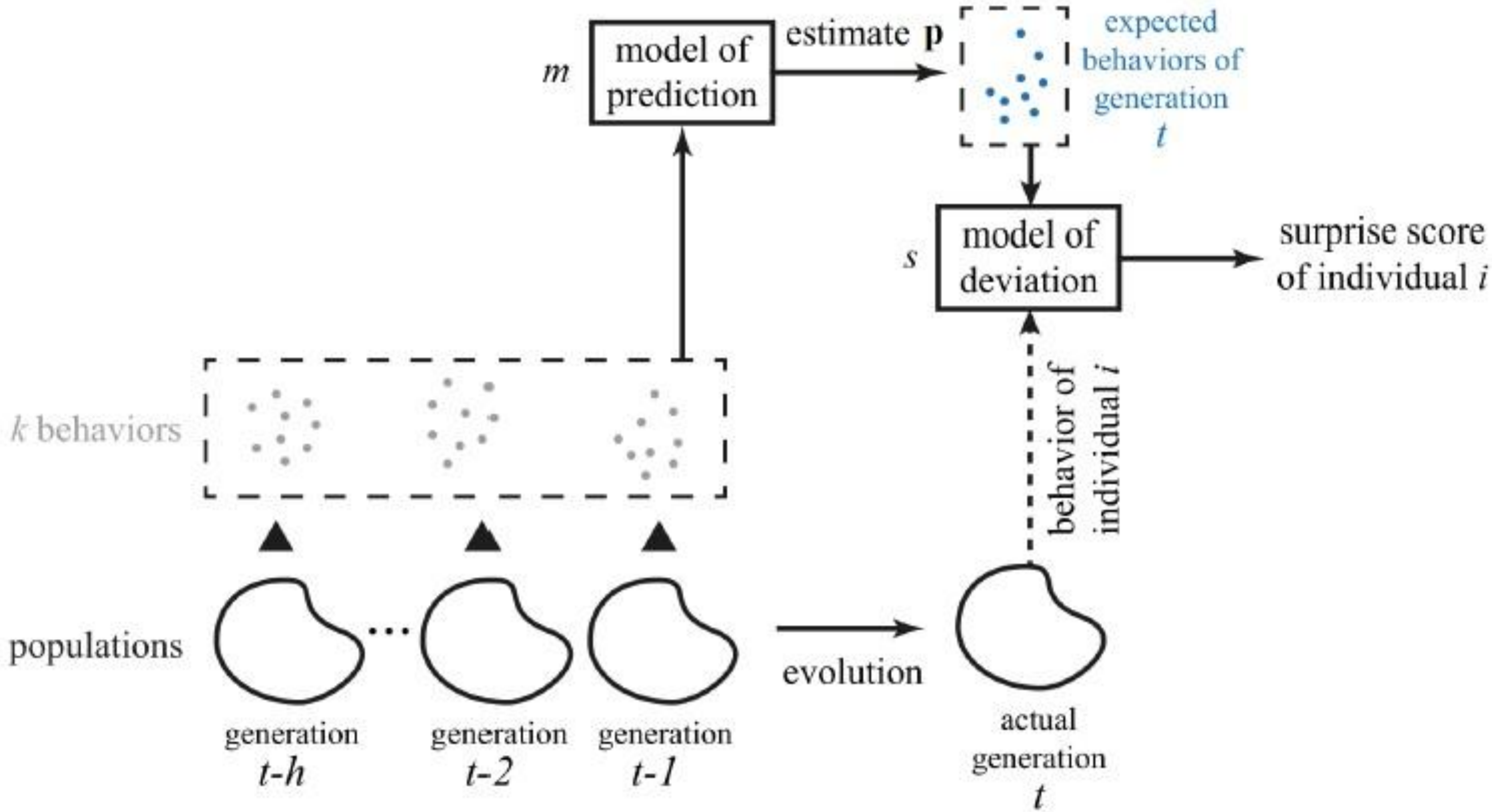
Denzinger, J.; Loose, K.; Gates, D.; and Buchanan, J. 2005. **Dealing with parameterized actions in behavior testing of commercial computer games**. In Proc. of the IEEE Symposium on Computational Intelligence and Games , 37–43.

Gameplay

G. N. Yannakakis, and J. Hallam, "**Evolving Opponents for Interesting Interactive Computer Games**," in *Proceedings of the 8th International Conference on the Simulation of Adaptive Behavior (SAB'04); From Animals to Animats 8*, pp. 499-508, Los Angeles, CA, USA, July 13-17, 2004. The MIT Press.

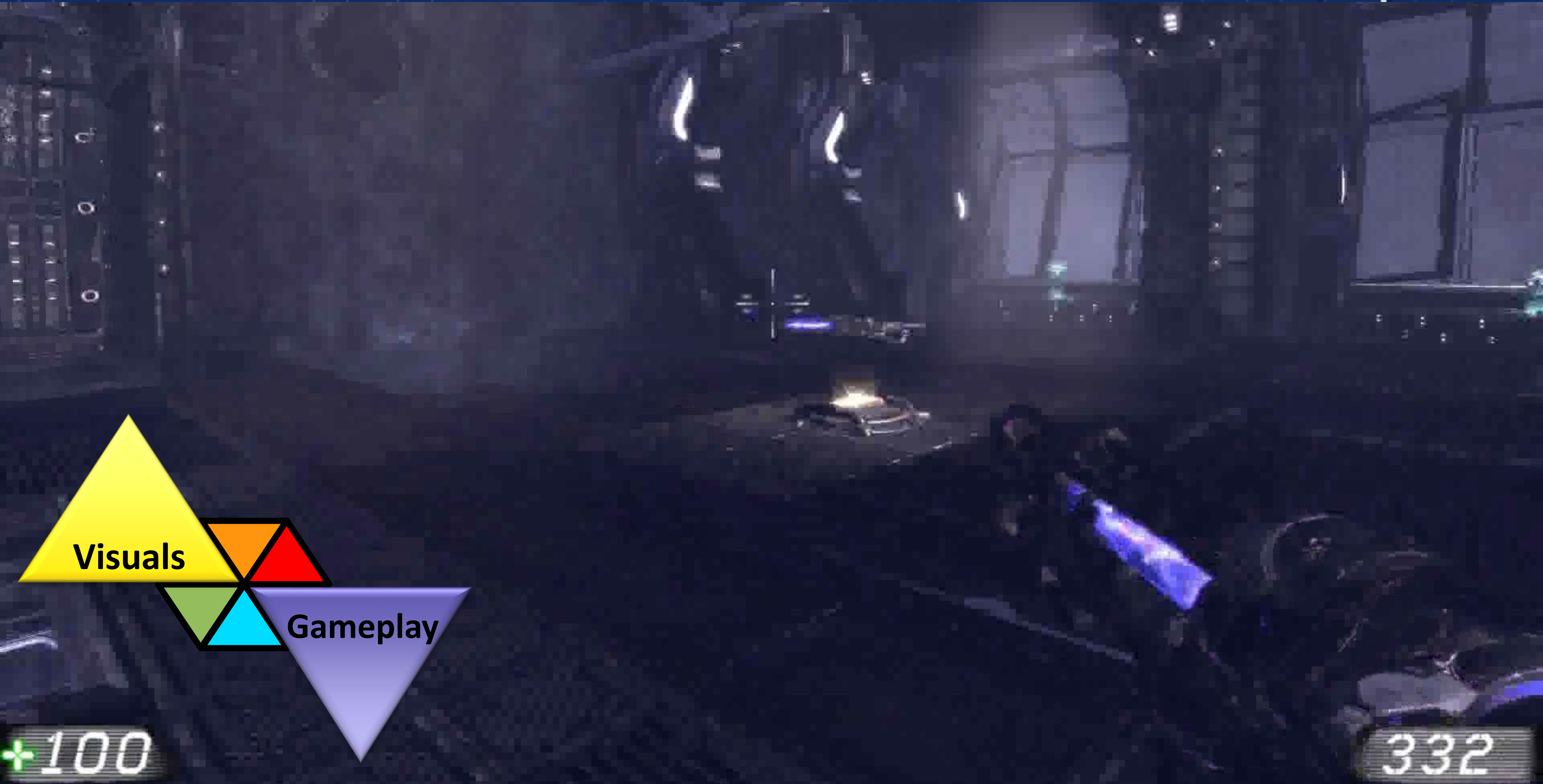
From Novelty Search to Surprise Search

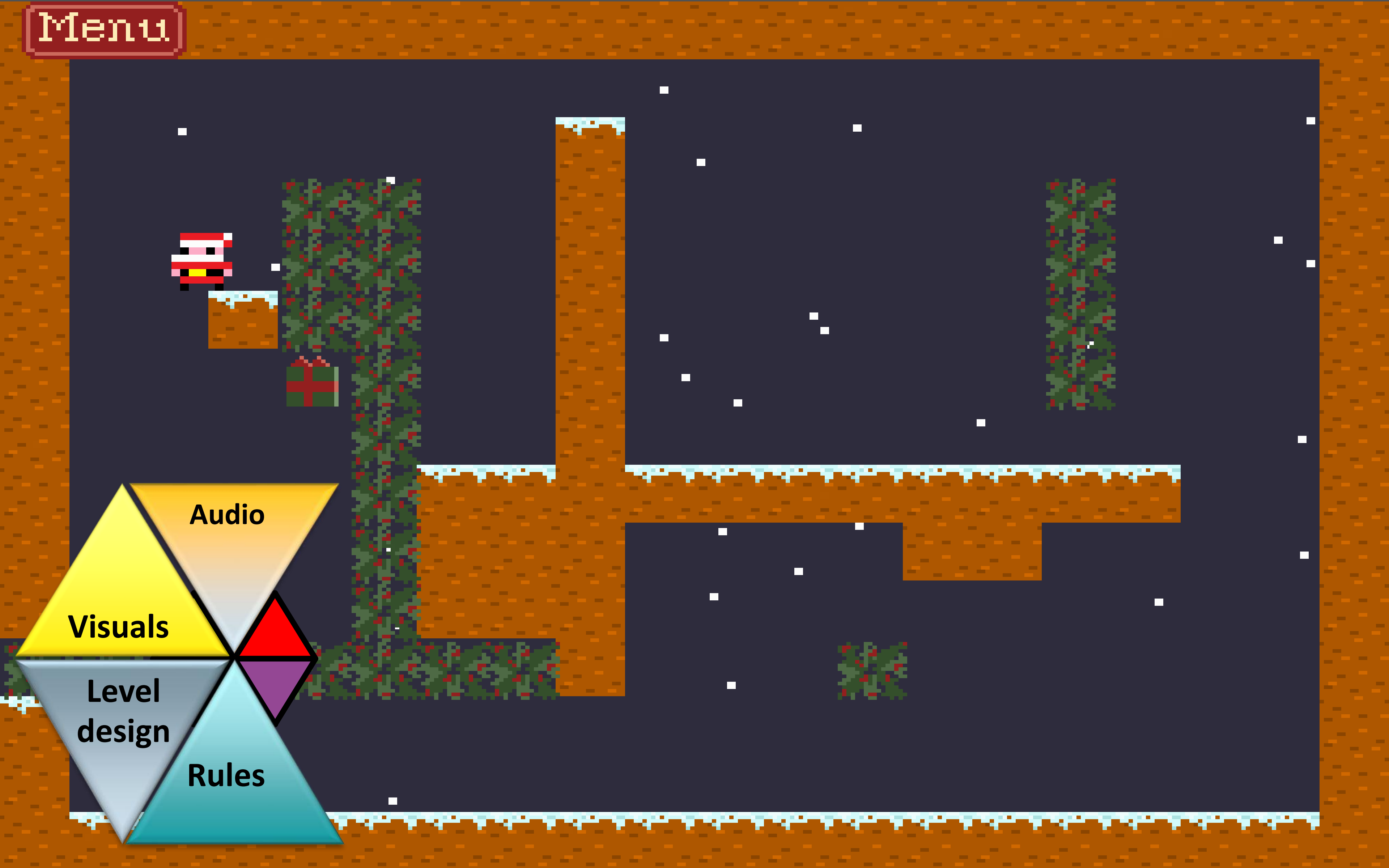
Gravina, Liapis, and Yannakakis: **"Surprise Search: beyond Novelty and Objectives"** in Proceedings of GECCO, 2016



Surprising Weapons!

Gravina, Liapis and Yannakakis: **"Constrained Surprise Search for Content Generation,"** in Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG). 2016.



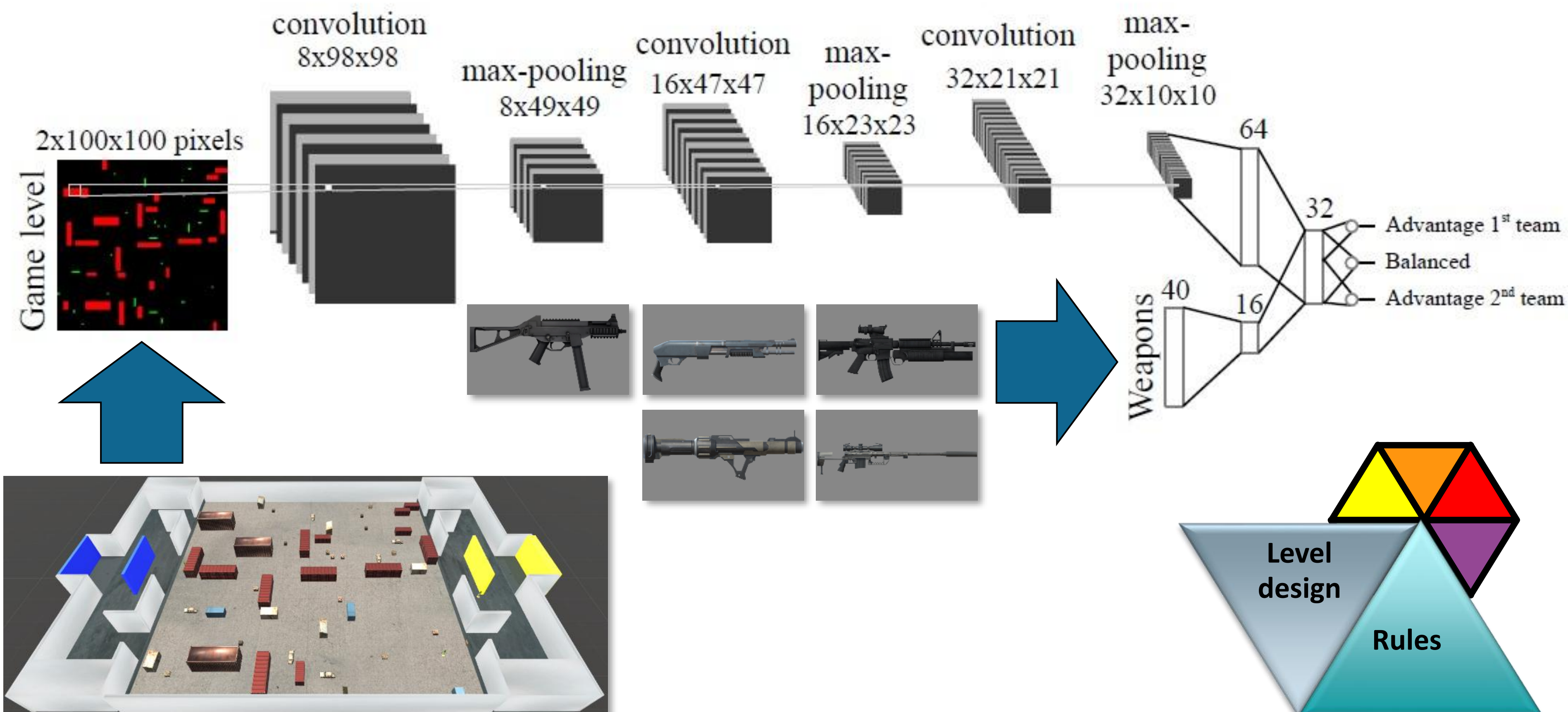

Menu
Audio
Visuals
Level design
Rules

AudioInSpace: From Music to Weapons!

Hoover, Cachia, Liapis, Yannakakis, **"AudioInSpace: A Proof-of-Concept Exploring the Creative Fusion of Generative Audio, Visuals and Gameplay,"** in EvoMusArt, 2015

Orchestrating Level and Game Design

Karavolos, Liapis, and Yannakakis: **“Learning Patterns of Balance in a Multi-Player Shooter Game,"** in *Proceedings of the Foundations on Digital Games*, 2017.

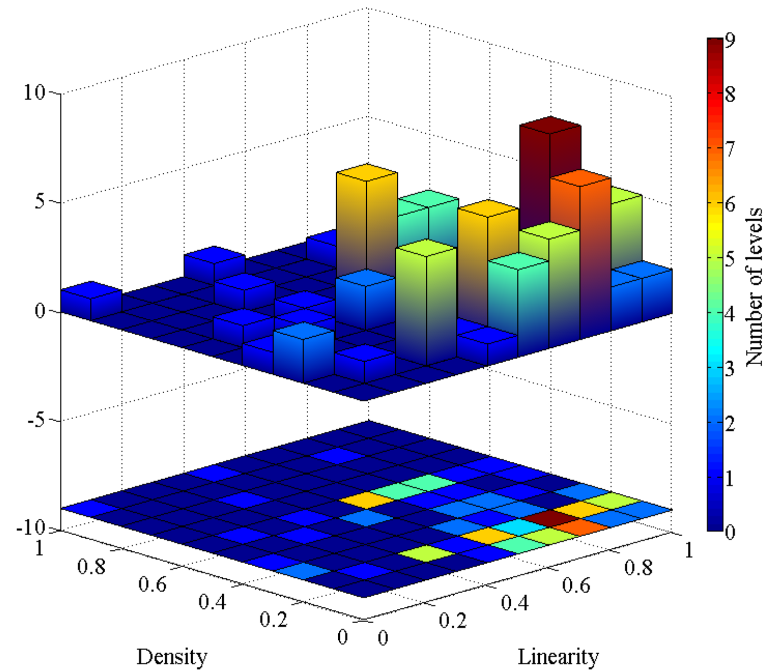


Evaluating Content Generators

How Can we Evaluate a Content Generator?

Generally speaking, there are three ways:

- Visualization (e.g. expressive range)
- AI players (playtesting / personas)
- Human players (testing, QA, annotations)



Artificial Intelligence and Games

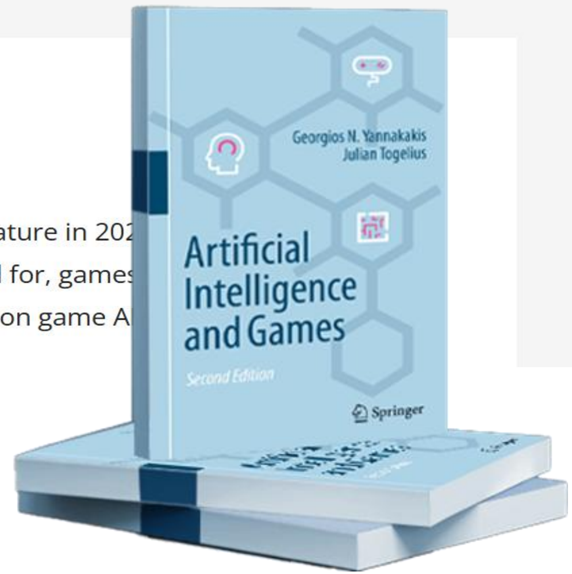
A Springer Textbook | By Georgios N. Yannakakis and Julian Togelius

About the Book   Table of Contents   Lectures   Exercises   Resources

About the Book

Second Edition Published!

Welcome to the Artificial Intelligence and Games book (2nd edition) published with Springer Nature in 202
the first comprehensive textbook on the application and use of artificial intelligence (AI) in, and for, games
book will be used by educators and students of graduate or advanced undergraduate courses on game A
practitioners at large.



Readings: Part III

gameaibook.org