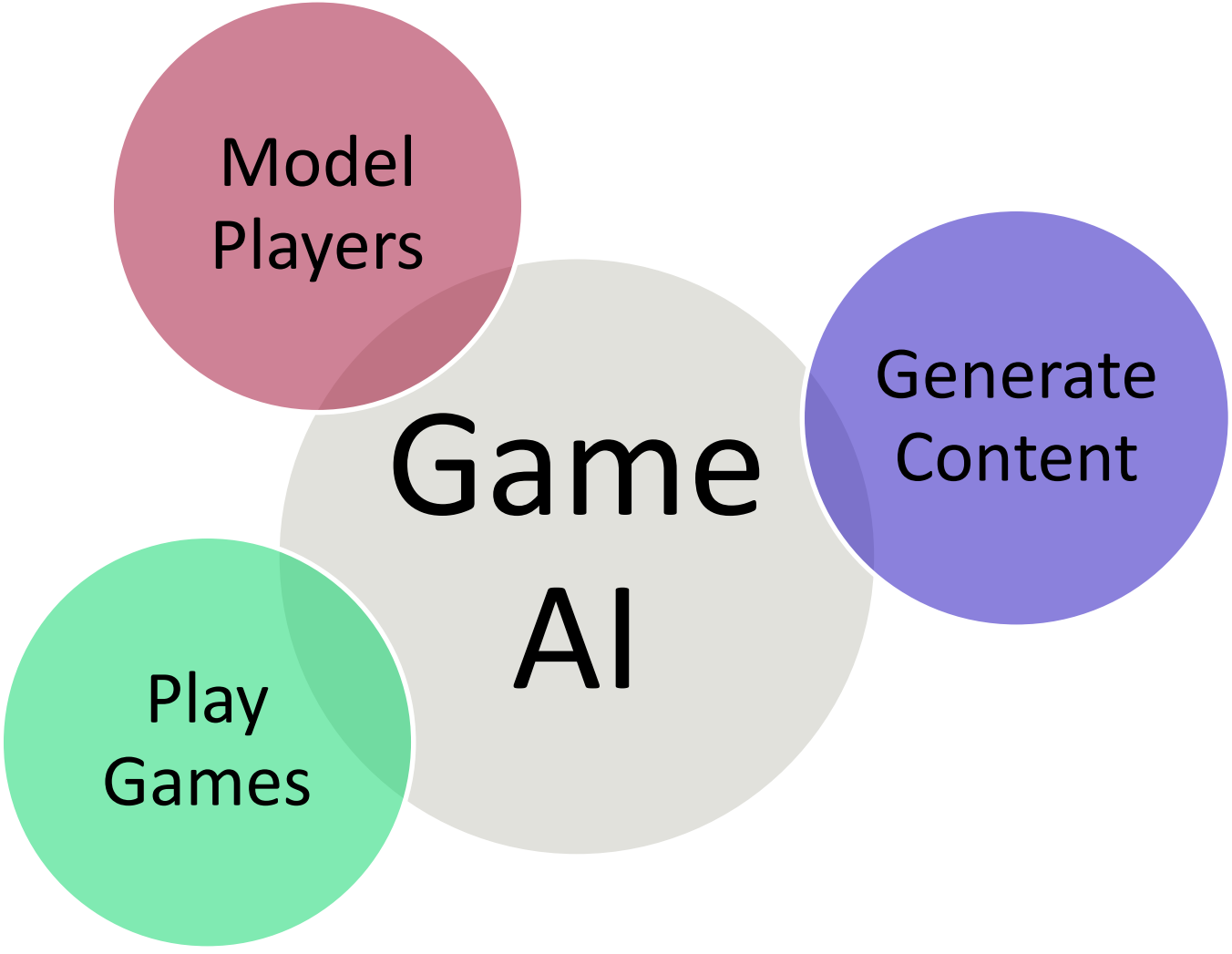
Artificial Intelligence and Games

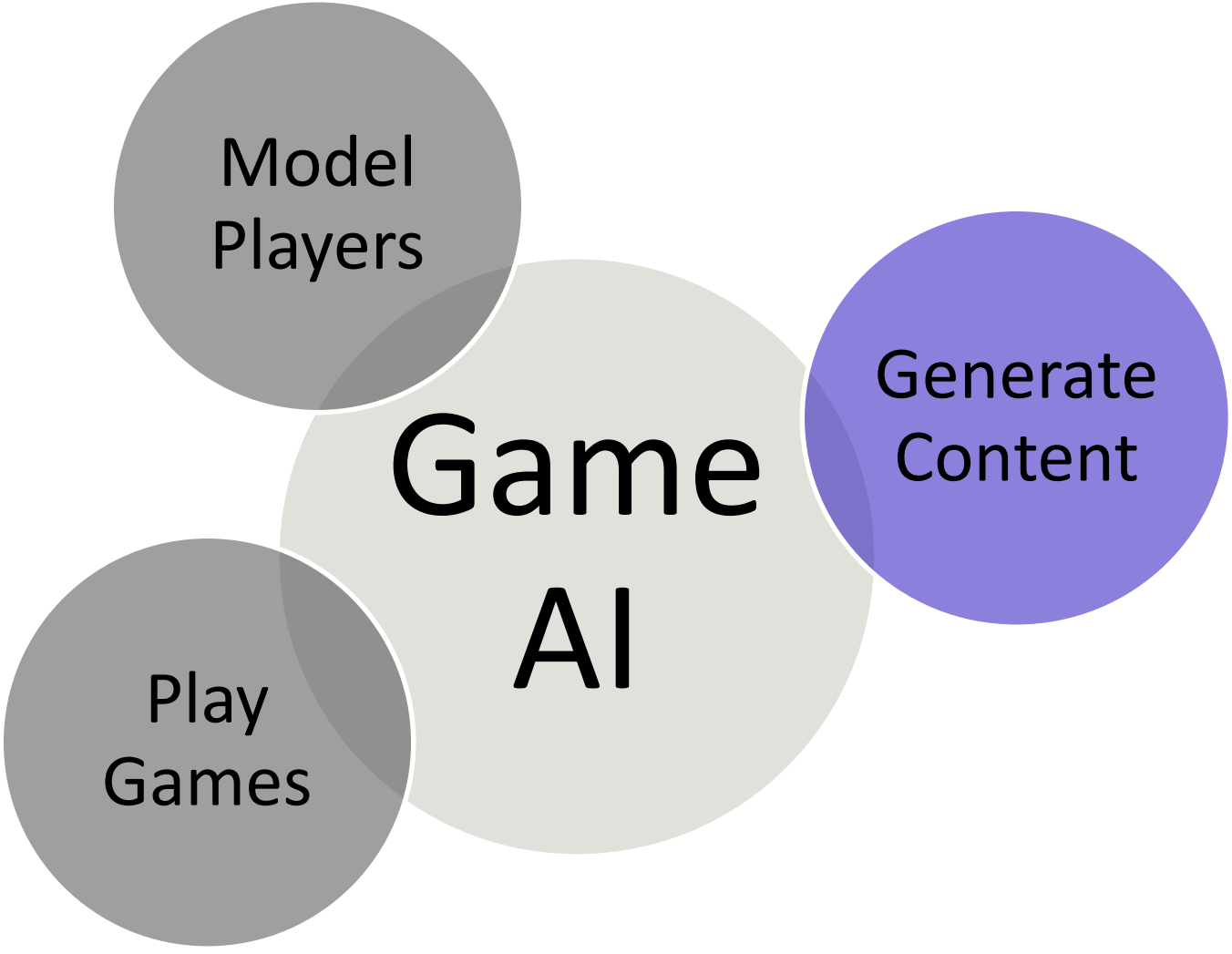
Generating Content

Georgios N. Yannakakis
@yannakakis

Julian Togelius
@togelius



G. N. Yannakakis and J. Togelius, "**Artificial Intelligence and Games**," Springer Nature, 2018.



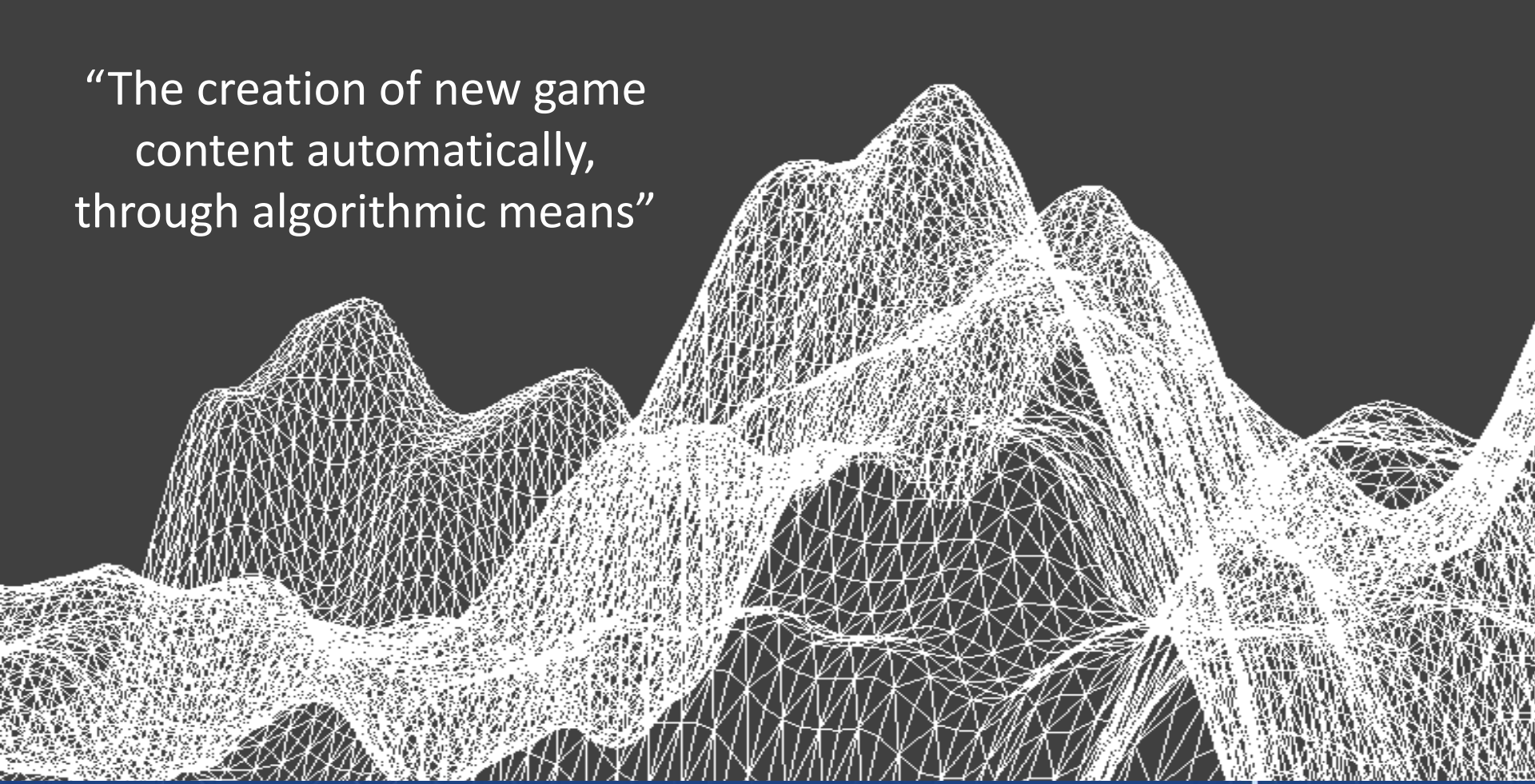
G. N. Yannakakis and J. Togelius, "**Artificial Intelligence and Games**," Springer Nature, 2018.

Your readings from **gameaibook.org**

Chapter: 4

Overview

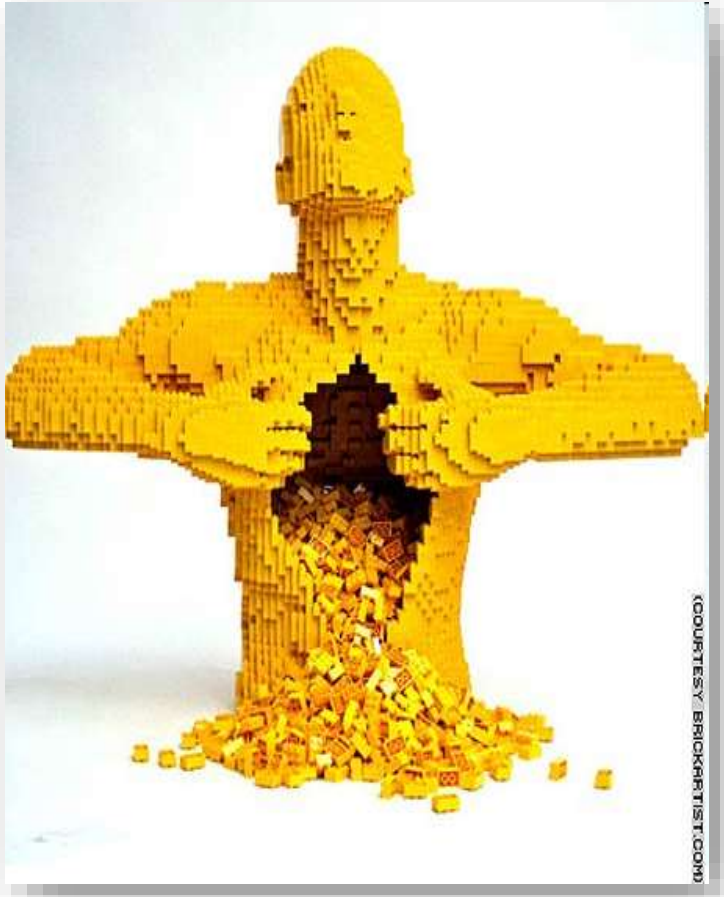
- Procedural Content Generation (PCG)
  - What is it?
  - Why we need it?
- Constructive Approaches
- Search-Based PCG
- Machine Learning PCG
- Mixed-Initiative PCG
- Experience-driven PCG



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)



Content Types

- Maps
- Levels
- Weapons / items
- Game rules
- Stories
- …
- Everything together?

PCG in Industry

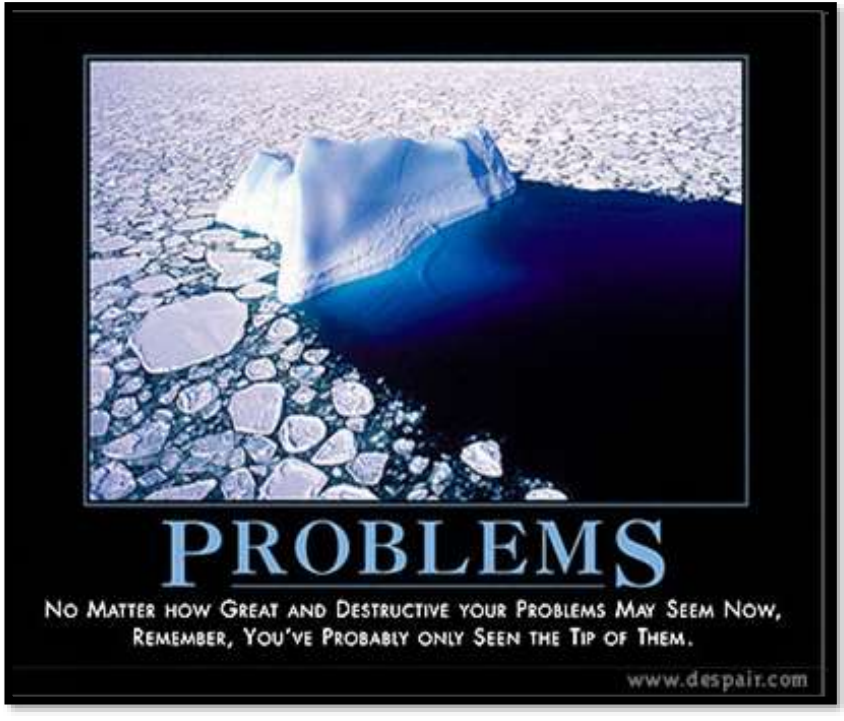
PCG in Academic

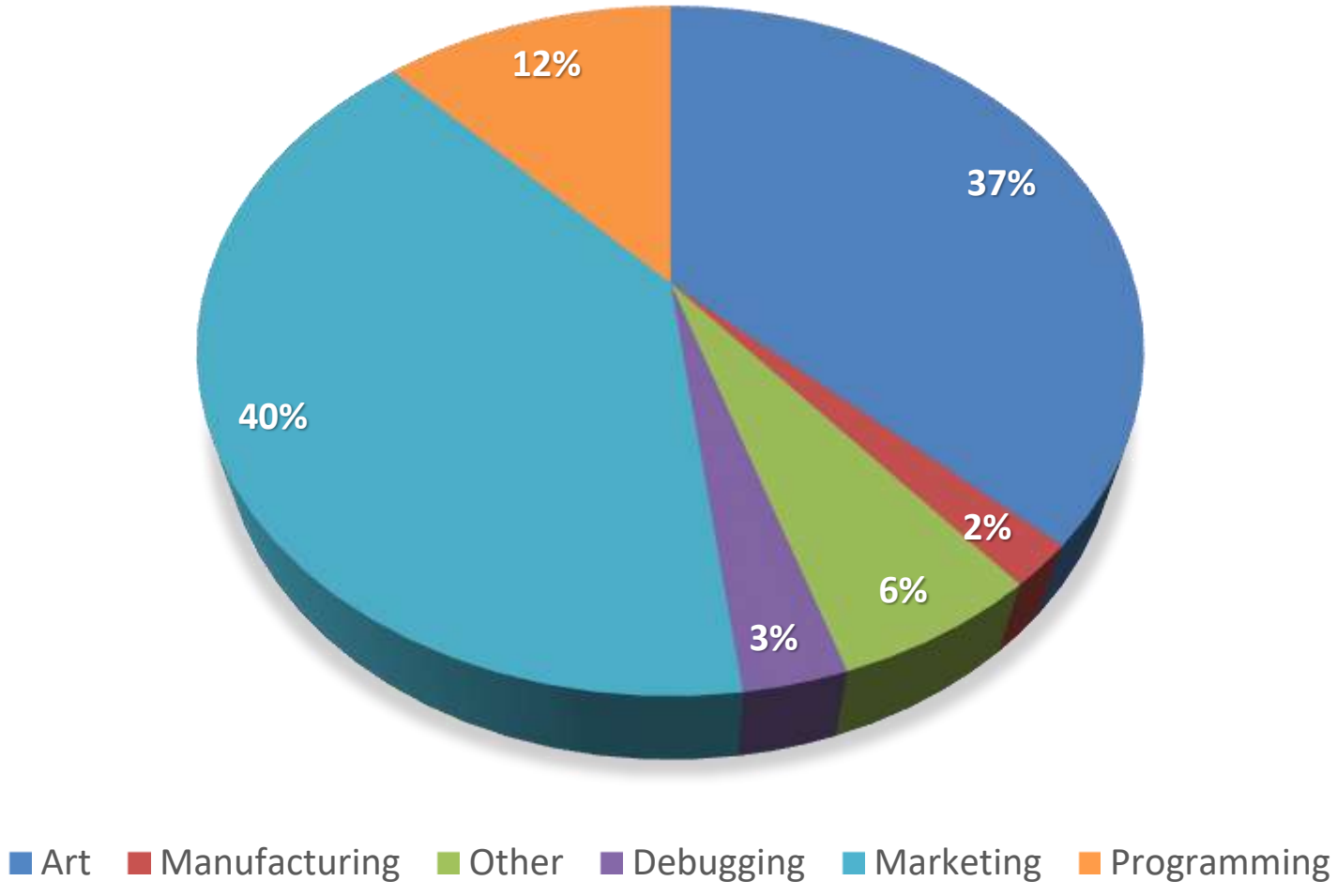
- An IEEE Task Force
- A book: pcgbook.com
- A number of PCG dedicated sessions/workshops since 2010: FDG, IEEE CIG, AIIDE, AAAI, IJCAI, ….
- A paradigm shift:
  - ML-based PCG
  - Mixed-Initiative PCG
  - Computational Game Creativity
  - Experience-driven PCG

What is the Problem?

- Content costs! (money and time)
- Games end when finished
- Game worlds have bounds
- Human imagination (creativity) is limited
- Designer is not always present

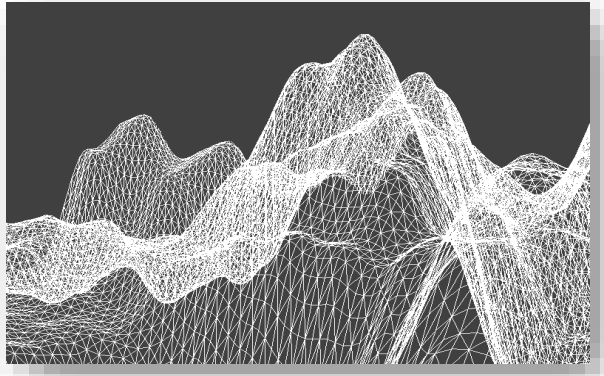
In sum, there is **content shortage**




12%
37%
40%
2%
6%
3%
Art
Manufacturing
Other
Debugging
Marketing
Programming

What Can PCG Do?

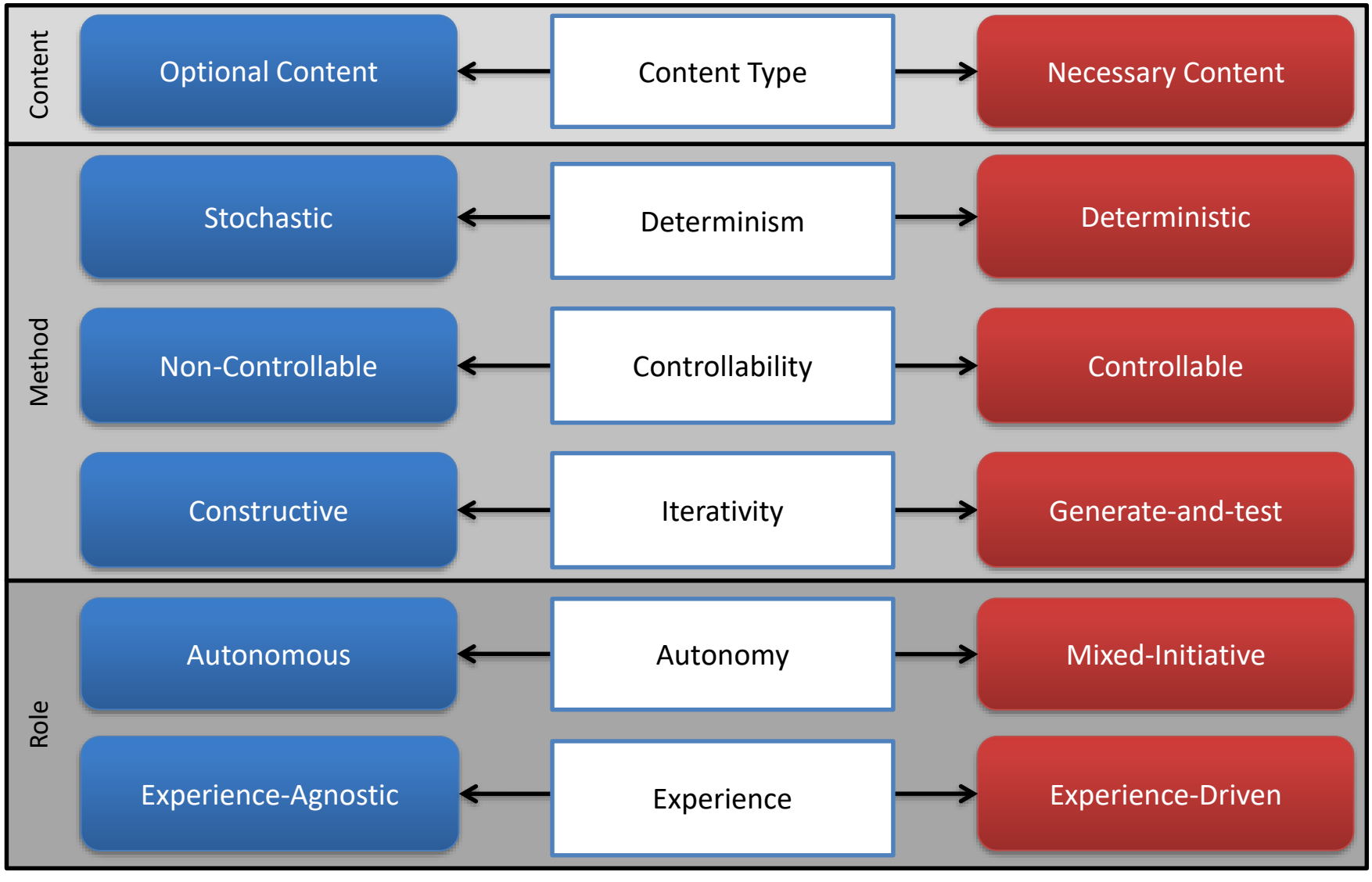
- Can we drastically cut game development costs by creating content automatically from designers' intentions?
- Can we create games that adapt their game worlds to the preferences of the player?
- Can we create endless games?
- Can the computer circumvent or augment limited human creativity and create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

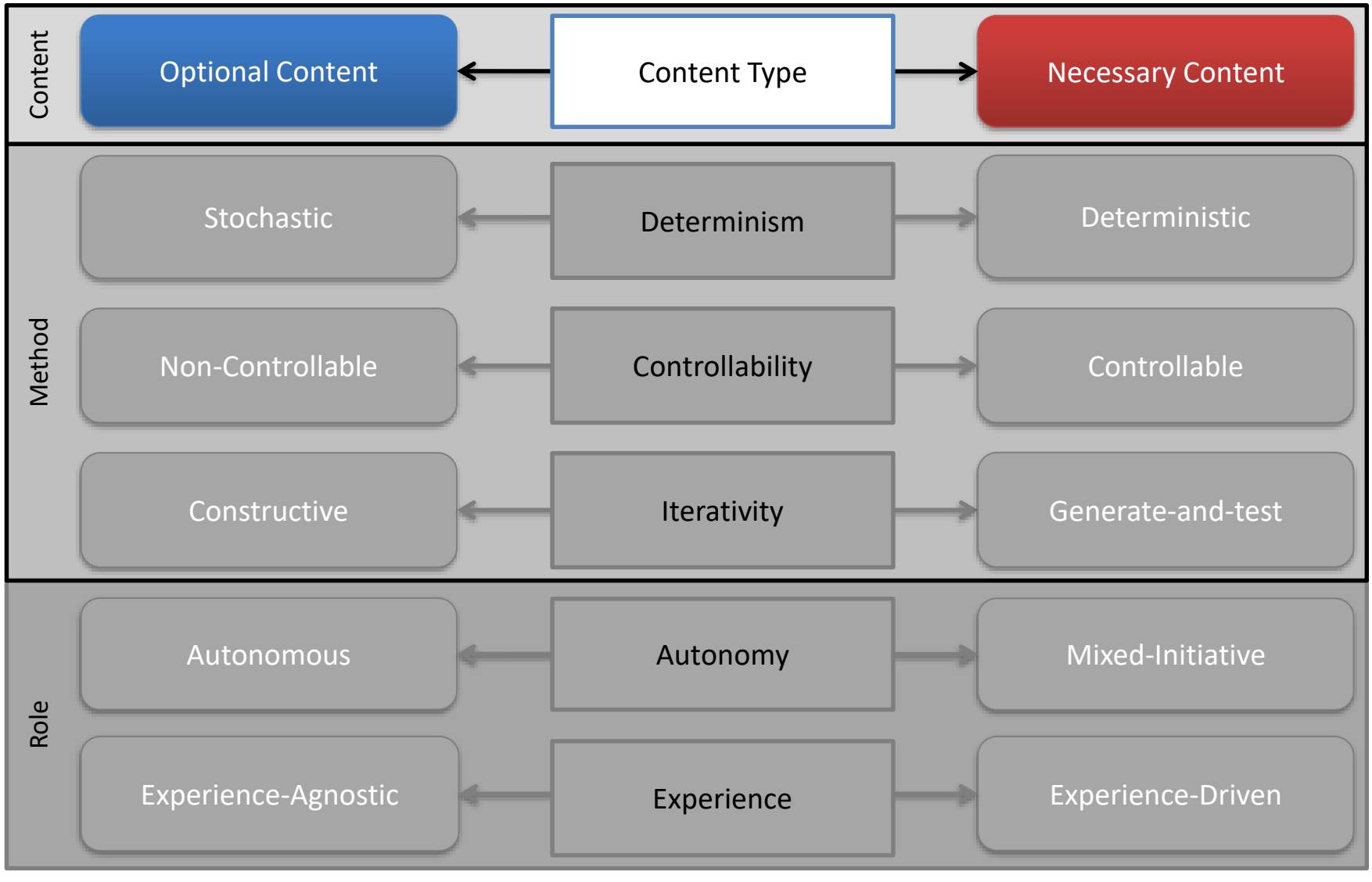


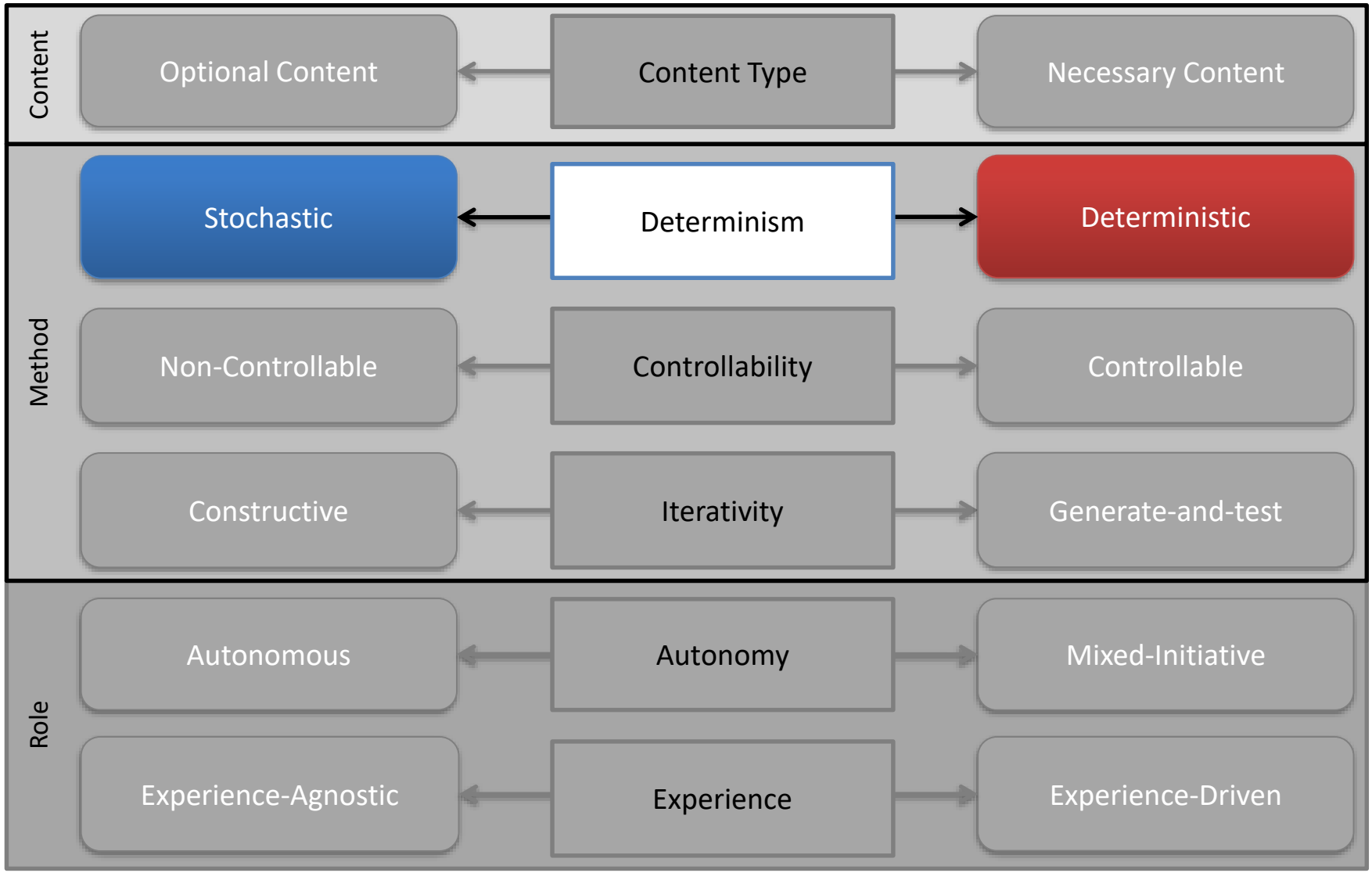
What Are the Trade-offs

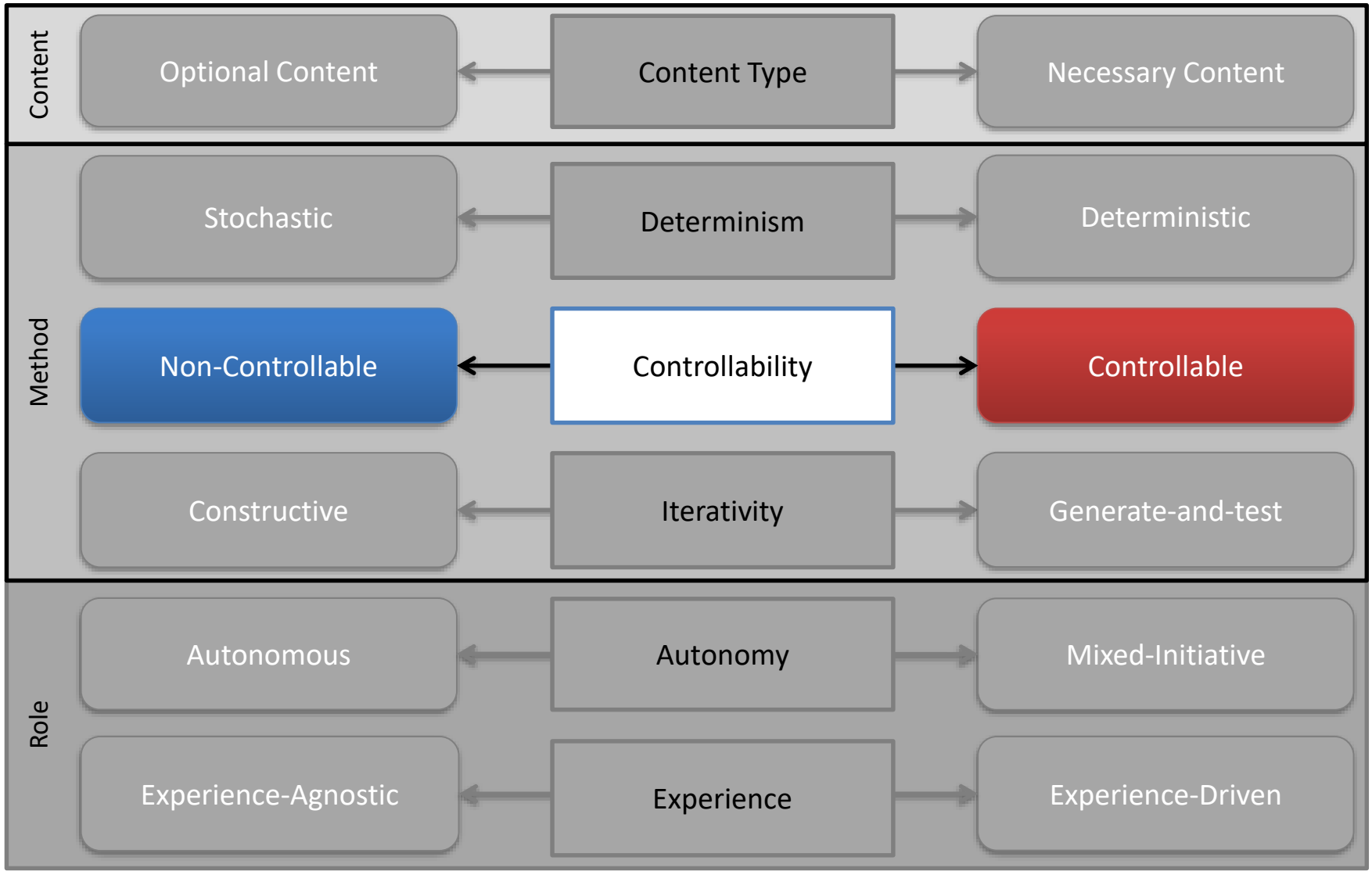
- *Speed*
  Real-time? Or design-time?
- *Reliability*
  Catastrophic failures break gameplay
- *Controllability*
  Allow specification of constraints and goals
- *Diversity*
  Content looks like variations on a theme
- *Creativity*
  Content looks “computer-generated”

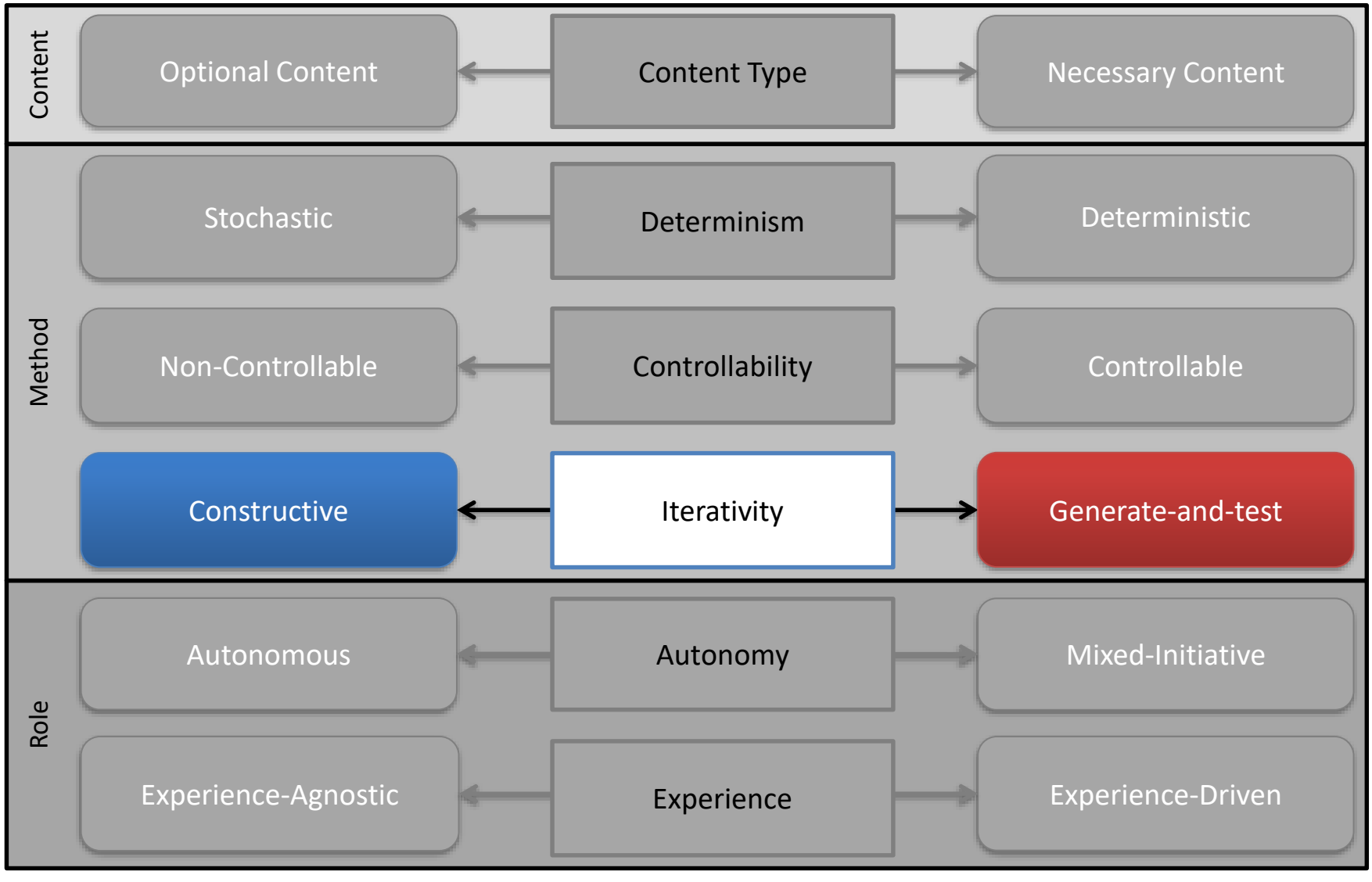
A PCG Taxonomy


Content
Optional Content
Content Type
Necessary Content
Method
Stochastic
Determinism
Deterministic
Non-Controllable
Controllability
Controllable
Constructive
Iterativity
Generate-and-test
Role
Autonomous
Autonomy
Mixed-Initiative
Experience-Agnostic
Experience
Experience-Driven

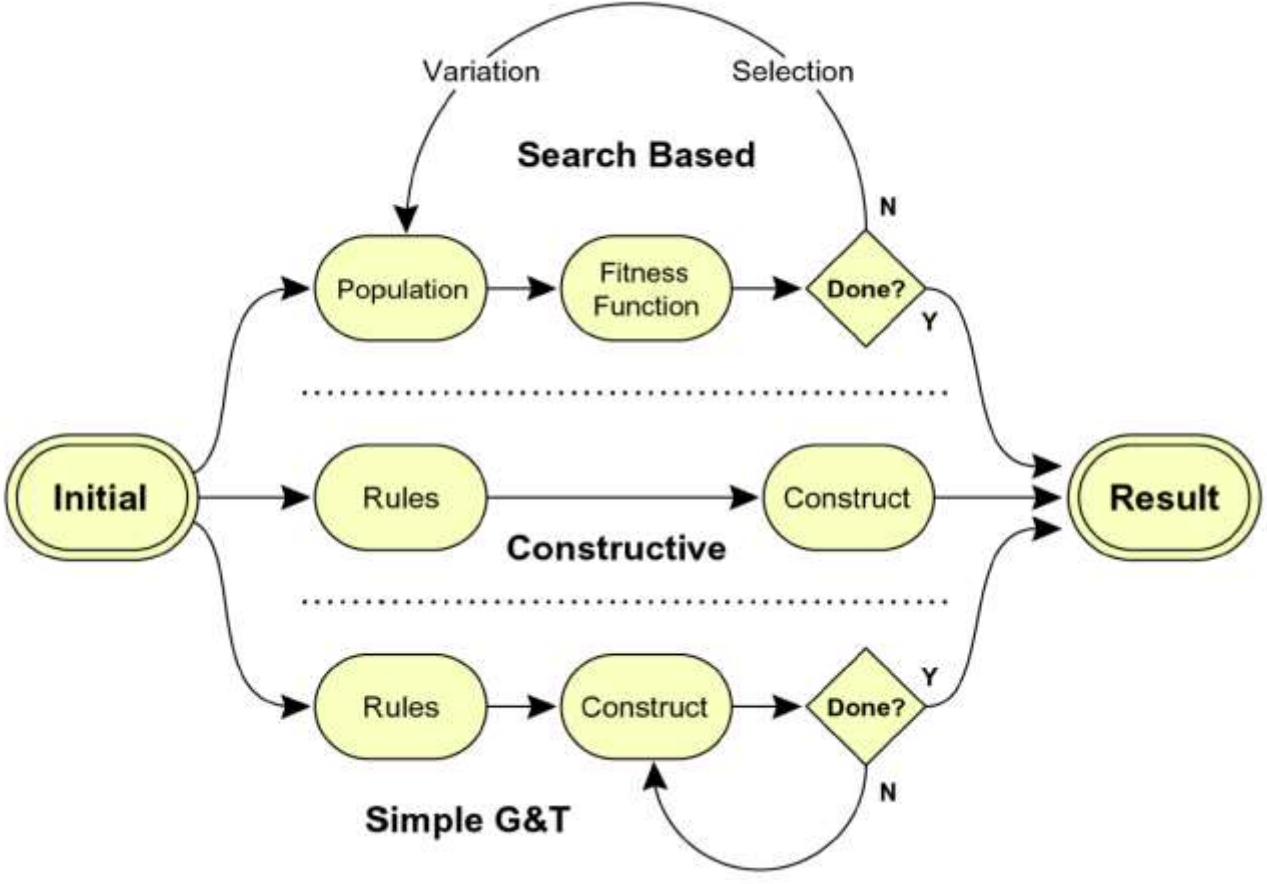

Content
Optional Content
Content Type
Necessary Content
Method
Stochastic
Determinism
Deterministic
Non-Controllable
Controllability
Controllable
Constructive
Iterativity
Generate-and-test
Role
Autonomous
Autonomy
Mixed-Initiative
Experience-Agnostic
Experience
Experience-Driven


Content
Optional Content
Content Type
Necessary Content
Method
Stochastic
Determinism
Deterministic
Non-Controllable
Controllability
Controllable
Constructive
Iterativity
Generate-and-test
Role
Autonomous
Autonomy
Mixed-Initiative
Experience-Agnostic
Experience
Experience-Driven


Content
Optional Content
Content Type
Necessary Content
Method
Stochastic
Determinism
Deterministic
Non-Controllable
Controllability
Controllable
Constructive
Iterativity
Generate-and-test
Role
Autonomous
Autonomy
Mixed-Initiative
Experience-Agnostic
Experience
Experience-Driven

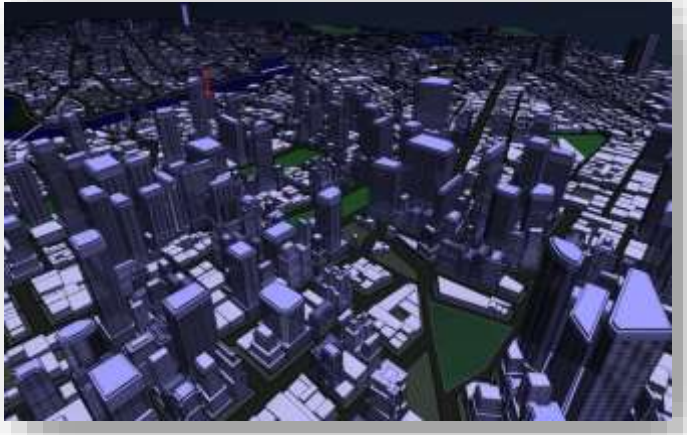

Content
Optional Content
Content Type
Necessary Content
Method
Stochastic
Determinism
Deterministic
Non-Controllable
Controllability
Controllable
Constructive
Iterativity
Generate-and-test
Role
Autonomous
Autonomy
Mixed-Initiative
Experience-Agnostic
Experience
Experience-Driven

Three Main PCG Approaches



“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, …



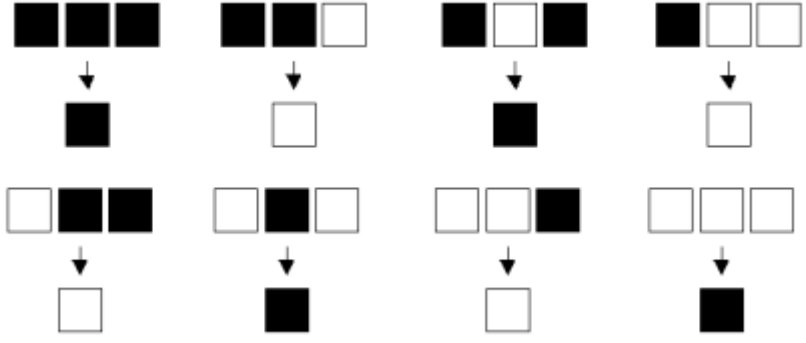
How Could we Generate Content?

PCG Methods

- Cellular automata
- Solver-based
- Grammar-based
- Search-based
- Machine learning
- Noise and fractals
- Ad-hoc constructive methods

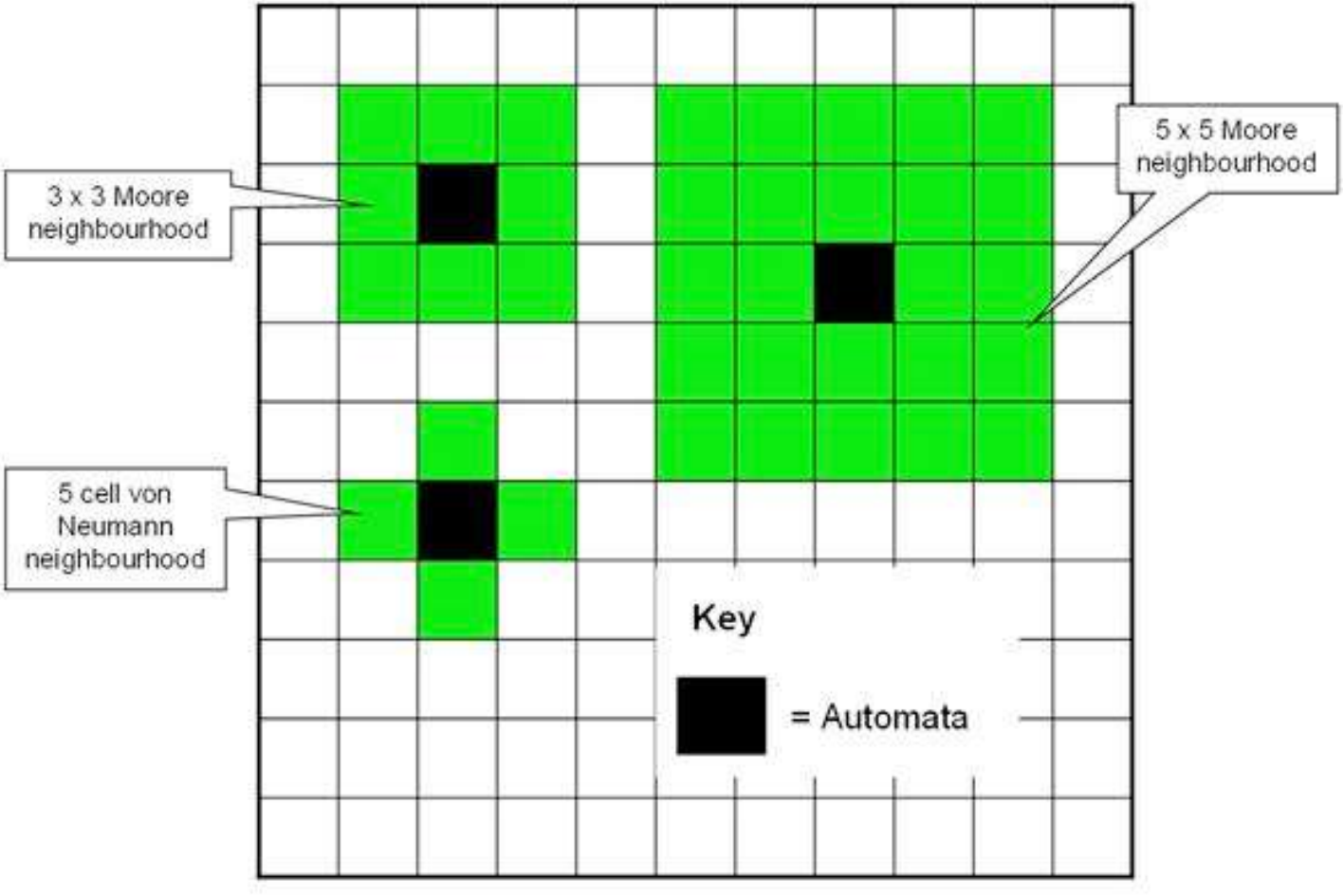
Cellular Automata

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

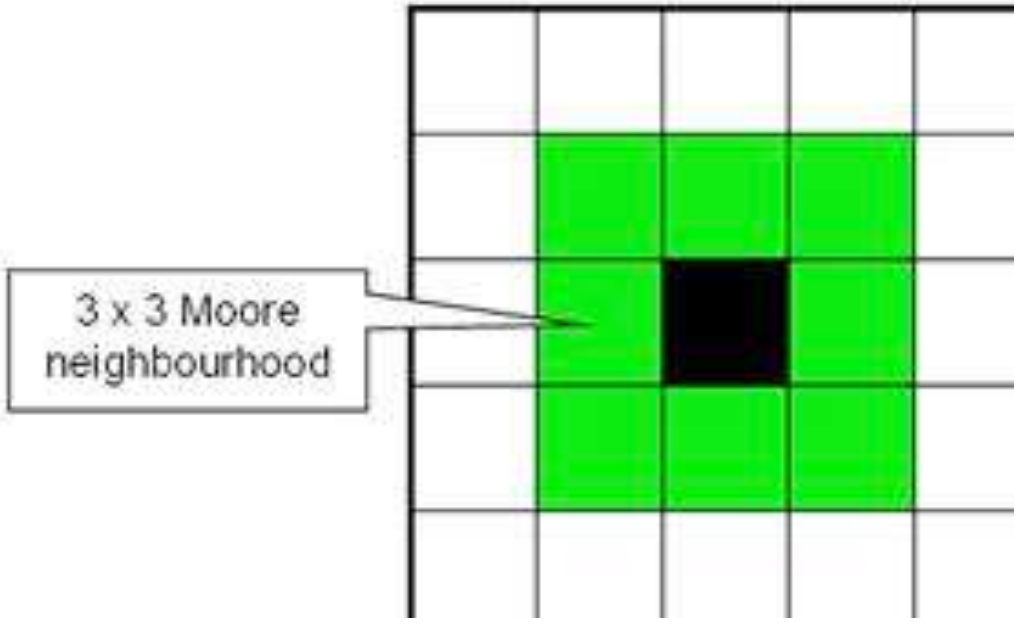
The Motivation

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

CA Cave Generation

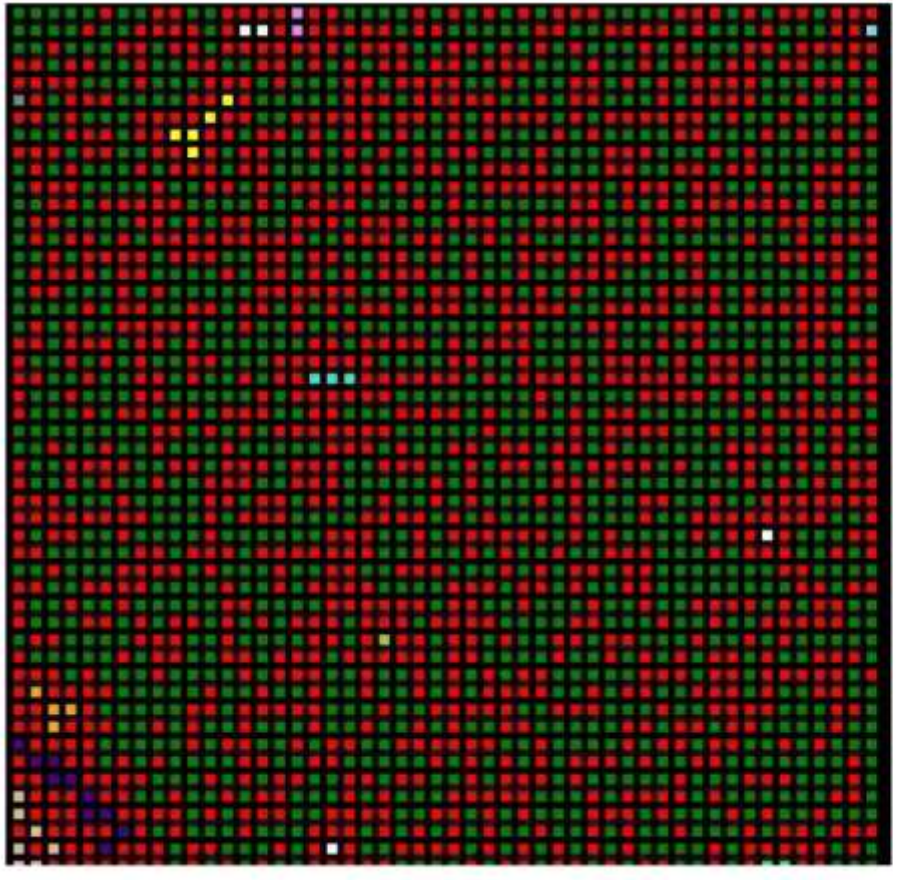
- Start with a square grid (e.g. 50*50) – all floor
- Randomly switch a proportion of cells from floor to rock
- Run a CA *n* times, where each cell is set to:
  - Rock: if at least *T* neighbors are rock
  - Floor: otherwise
- Fill in the interior of rock formations

Core CA Mechanic

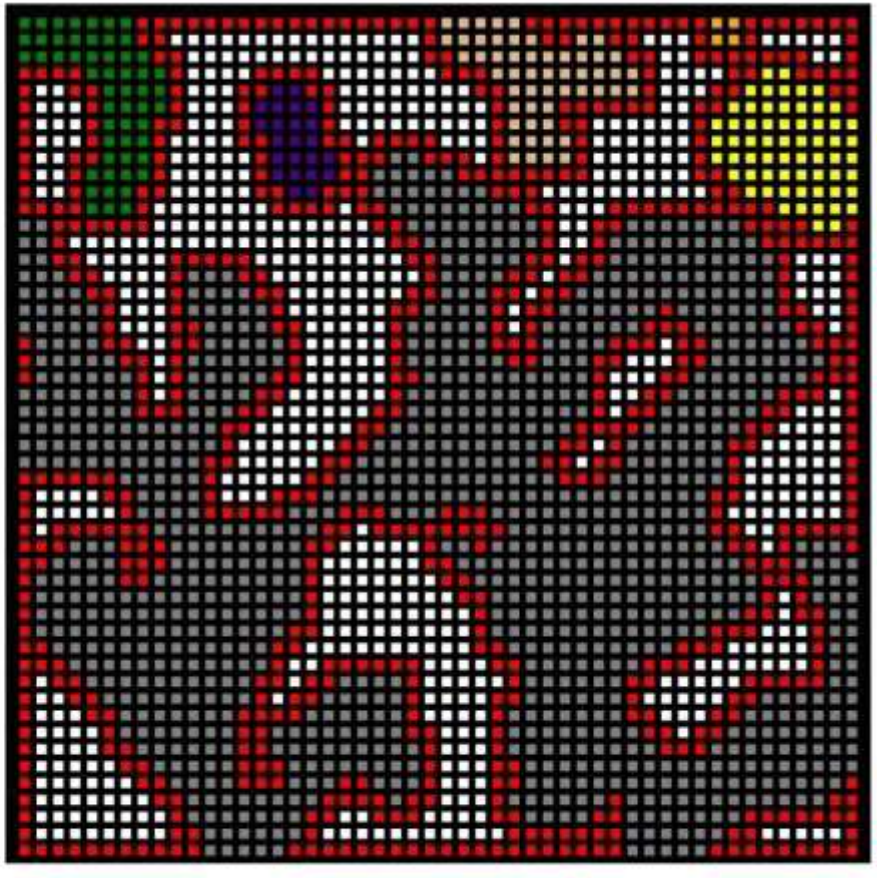


CA Parameters

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



(a) Random map



(b) CA map

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

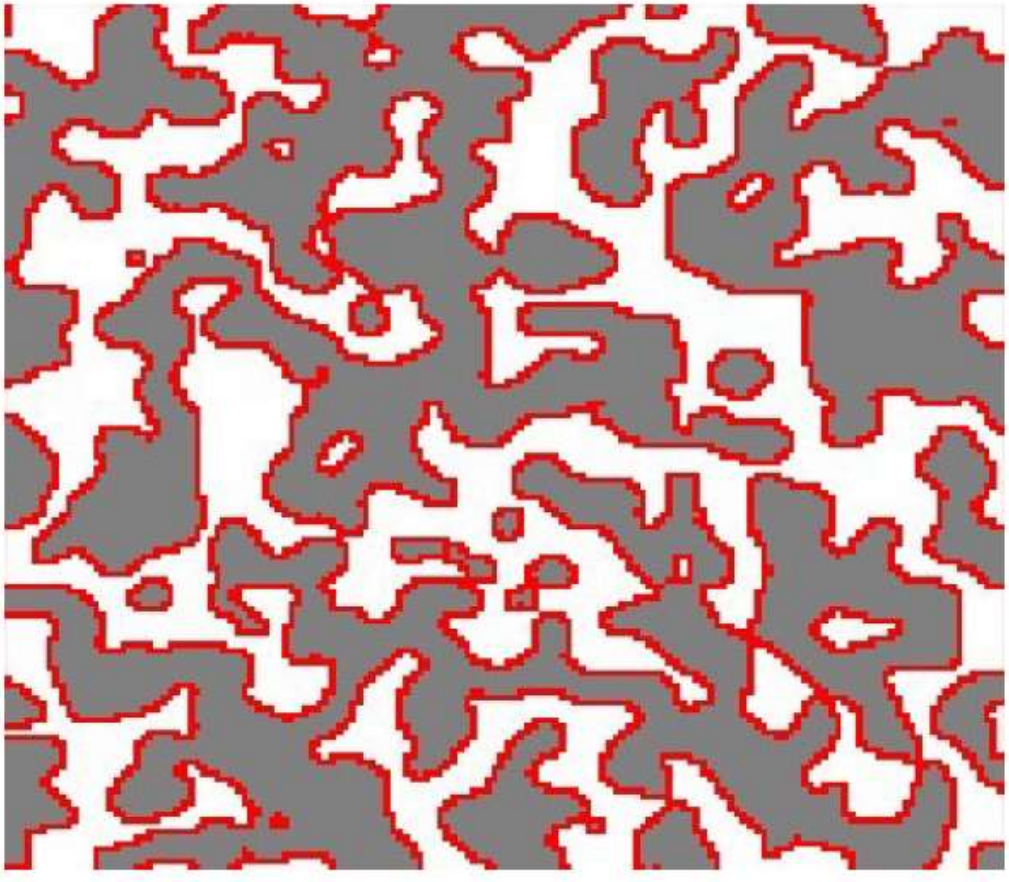
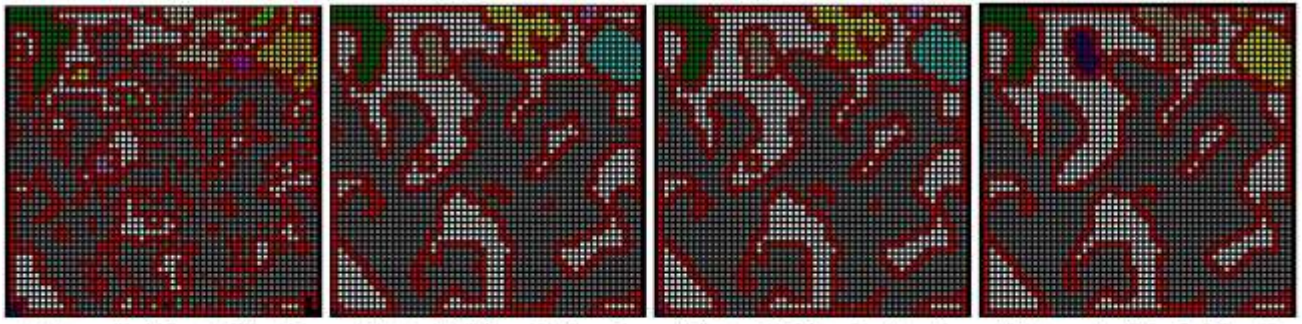


Figure 3: A 3×3 base grid map generated with CA. Rock and wall cells are represented by red and white color respectively. Grey areas represent floor. ($M = 2; T = 13; n = 4; r = 50\%$)

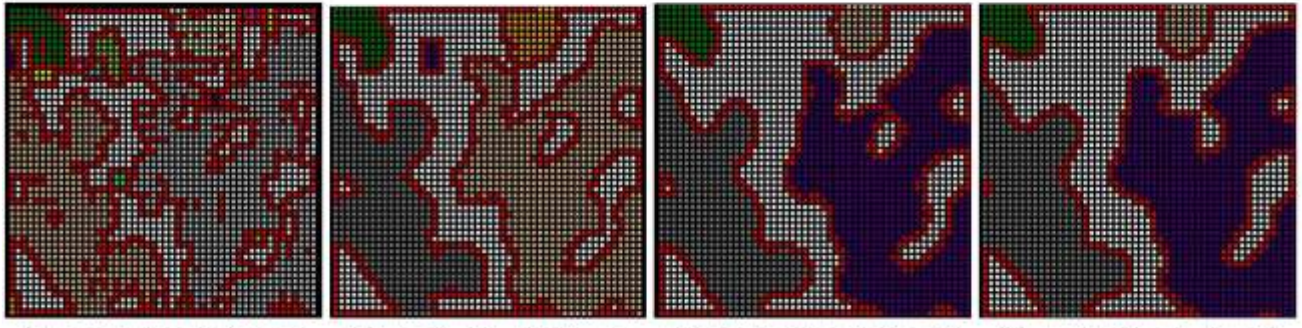
Controllable?

Parameters can be varied…

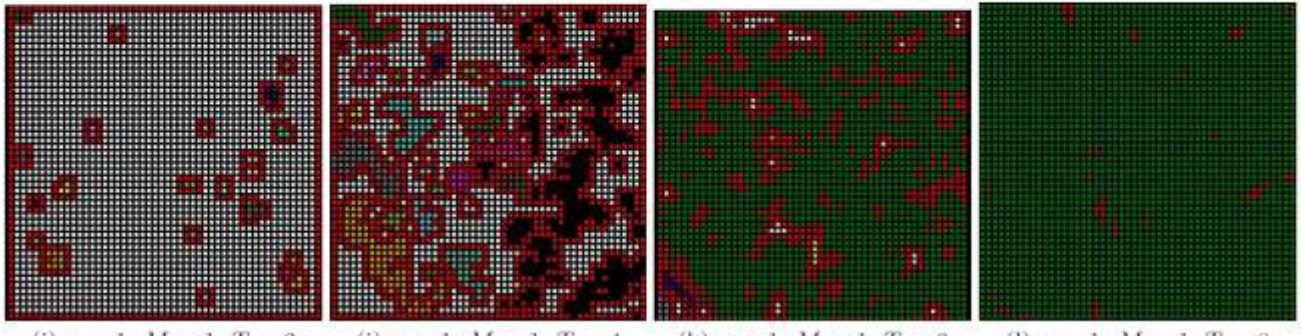
…but what do they mean?



(a) $n = 1$, $M = 1$, $T = 5$ (b) $n = 2$, $M = 1$, $T = 5$ (c) $n = 3$, $M = 1$, $T = 5$ (d) $n = 4$, $M = 1$, $T = 5$

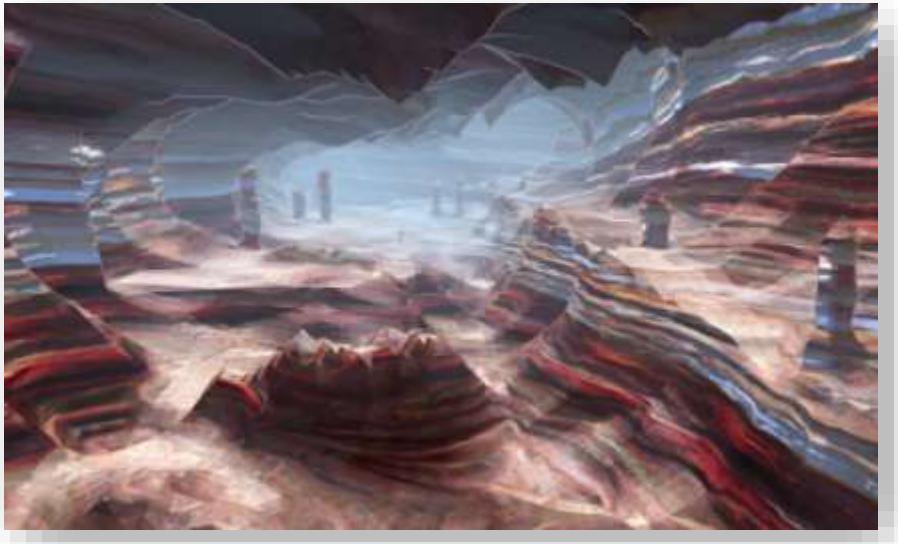


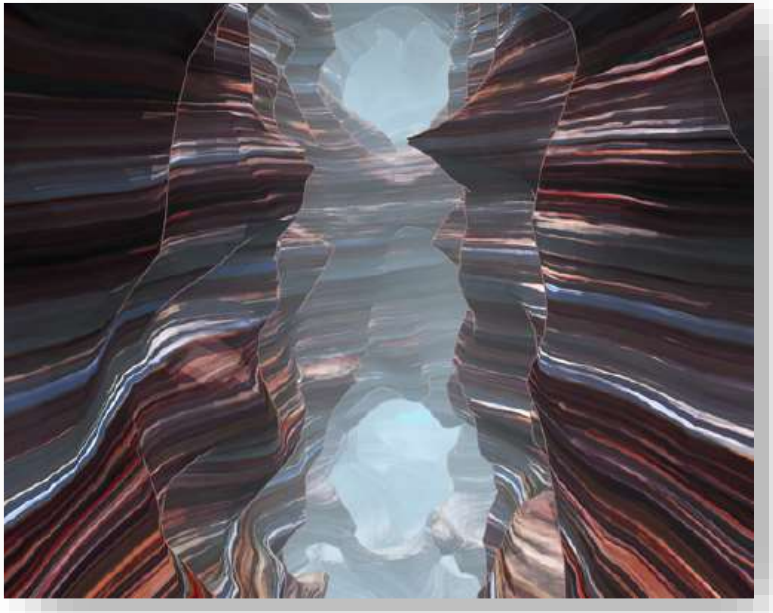
(e) $n = 1$, $M = 2$, $T = 13$ (f) $n = 2$, $M = 2$, $T = 13$ (g) $n = 3$, $M = 2$, $T = 13$ (h) $n = 4$, $M = 2$, $T = 13$



(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





Solver-based Methods

Solver-based Methods

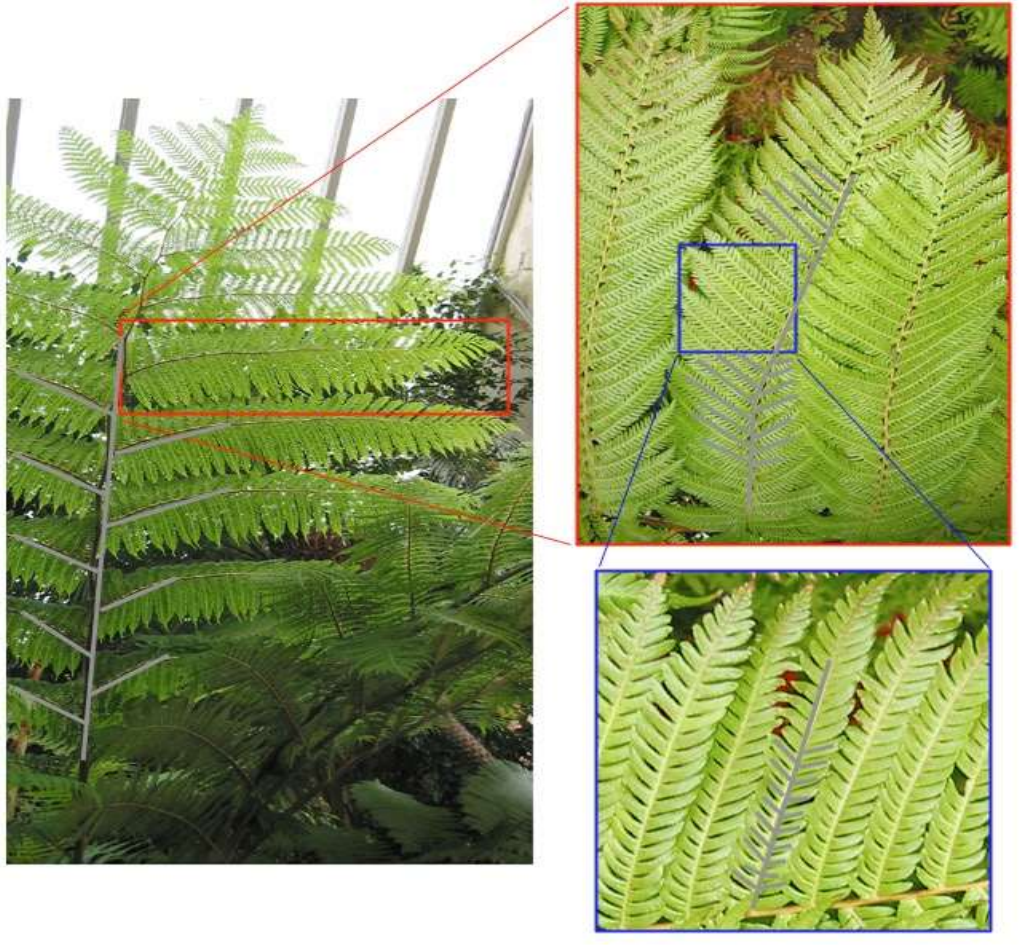
- Instead of using evolutionary algorithms, use constraint solvers and specify constraints
- Example: Answer Set Programming (ASP)
  - Declarative programming for search problems
  - Based on logic programming (syntax very similar to Prolog)
  - Finding an answer set is equivalent to solving a satisfiability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

Grammars

Grammars

- 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



Self-Similarity

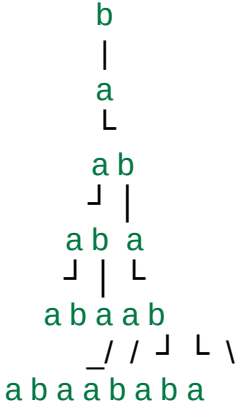
- Nature has obviously thought out some clever way of representing complex organisms using a compact description...
- ...permitting individual variation...
- ...why is this relevant for us?

L-systems

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

An Example L-system

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



Example of a derivation in a DOL-System

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

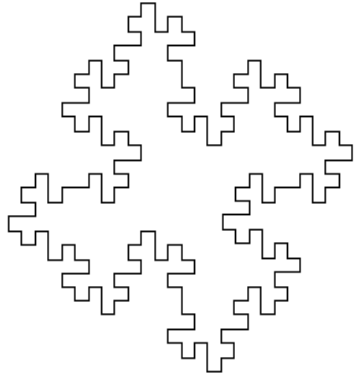
- Invented/popularized by Prusinkiewicz in 1986
- Core idea: interpret generated strings as instructions for a turtle in **turtle graphics**
- Read the string from left to right, changing the state of the turtle (x, y, heading)

Example Graphical Systems

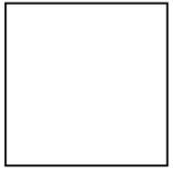
- Alphabet: {F, f, +, -}
- F: move the turtle forward (drawing a line)
- f: move the turtle forward (don’t draw)
- +/-: turn right/left (by some angle)

Graphical L-system

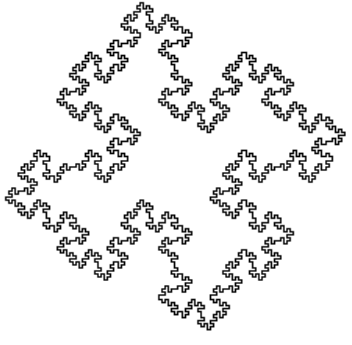
- Axiom: F+F+F+F
- Grammar:
  F>F+F-F-FF+F+F-F
- Turning angle: 90º



n=1



n=0



n=2

Graphical L-systems

What’s the limit of these systems?

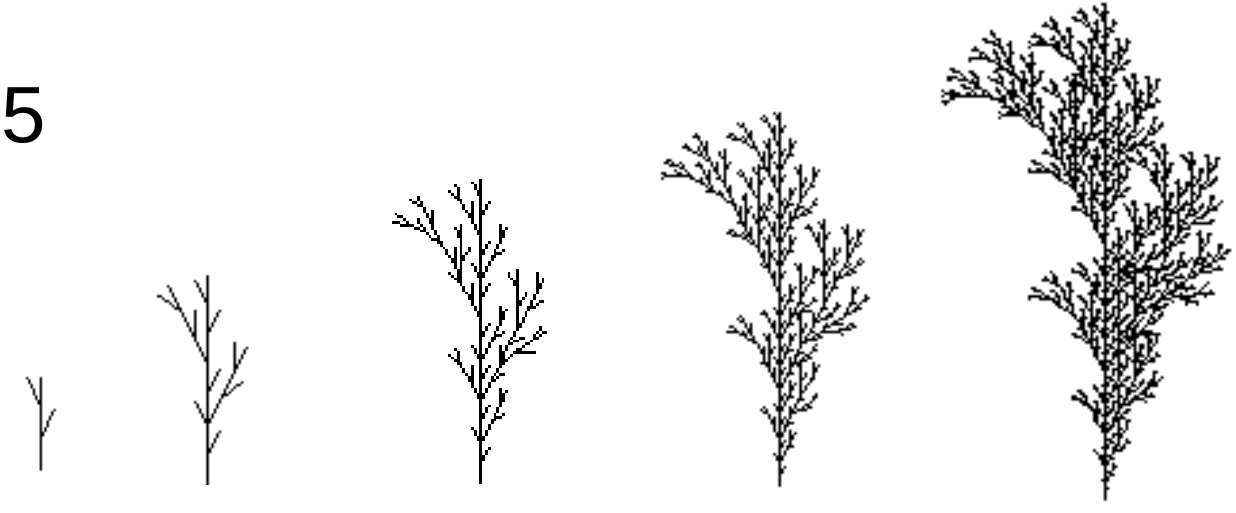
Bracketed L-systems

- 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: 30º

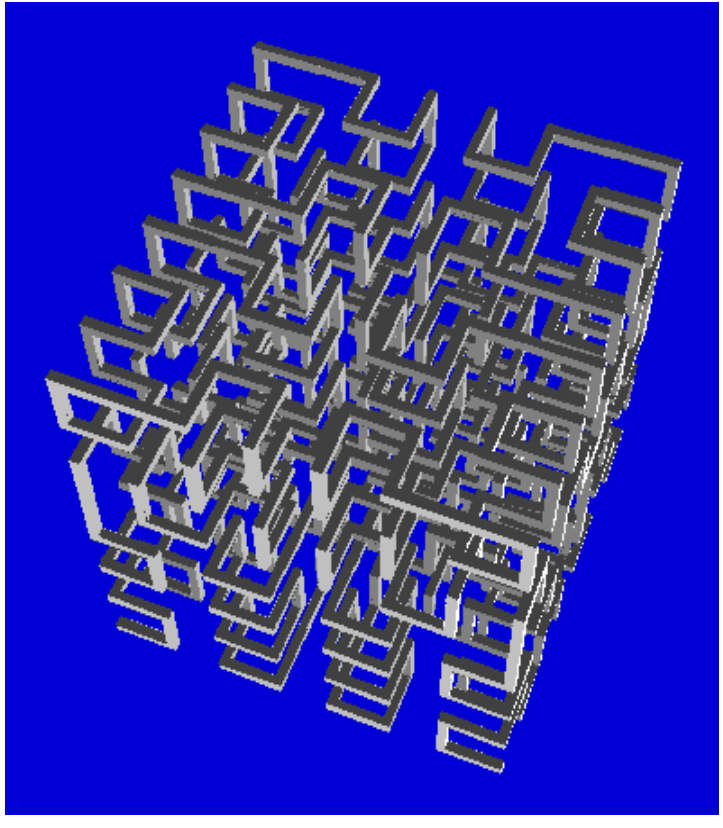
n=1,…,5

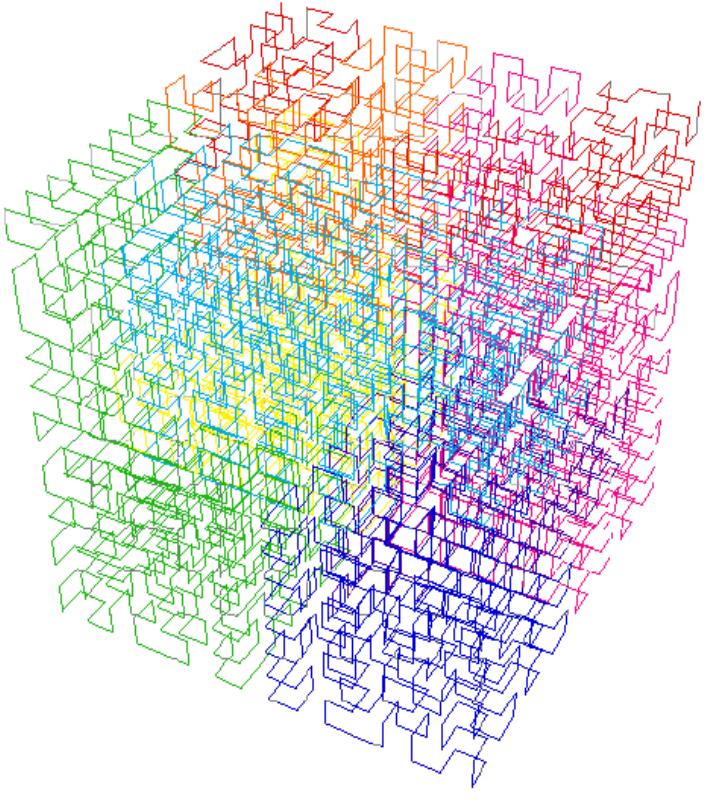


3D Graphics

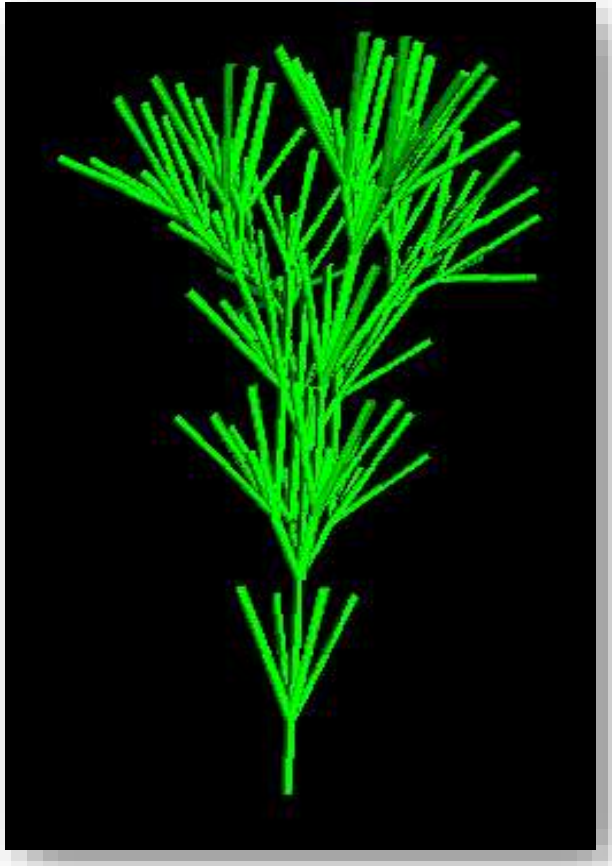
- Turtle graphics L-system interpretation can be extended to 3D space:
- Represent state as x, y, z and pitch, roll, yaw
- +, -: turn (yaw) left/right
- &, ^: pitch down/up
- \, /: roll left/right (counterclockwise/clockwise)

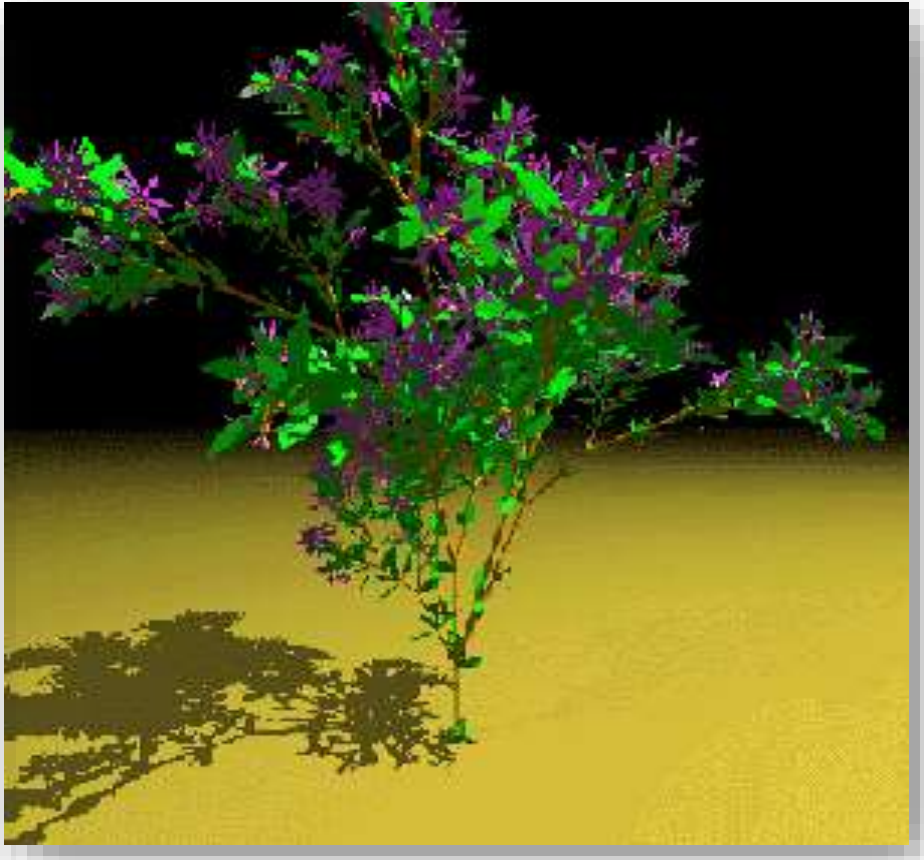
3D Interpretation of L-systems





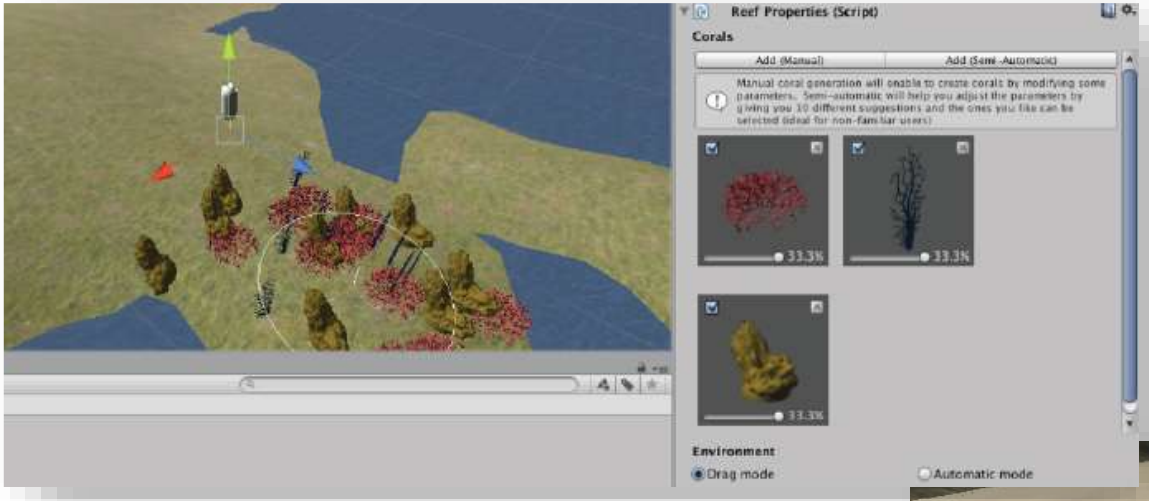
3D Interpretation of Bracketed L-systems

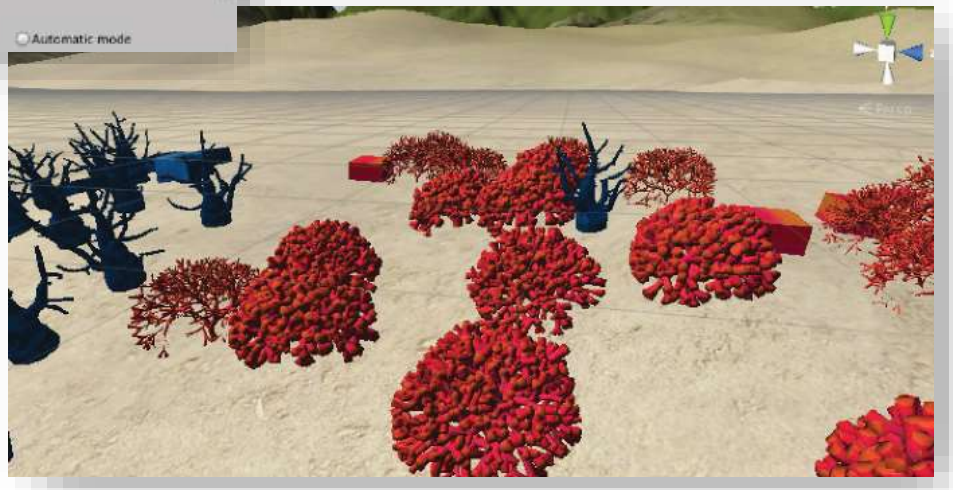




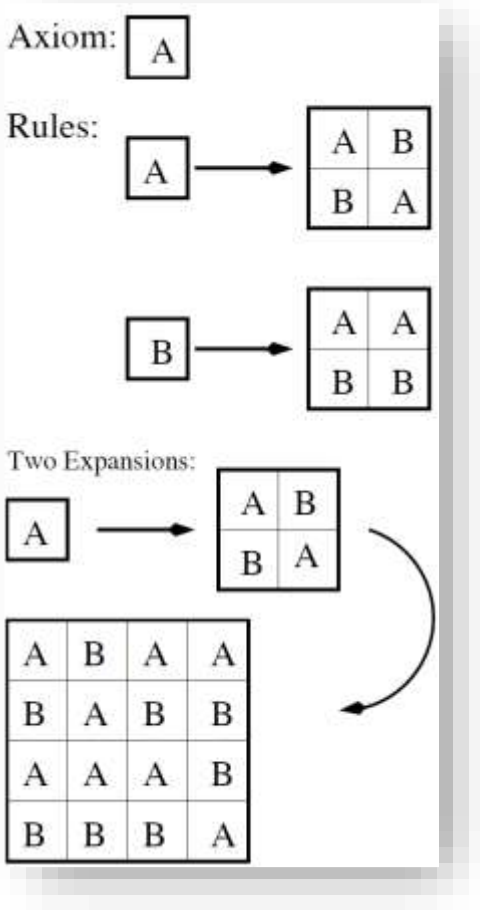
Coralize: 3D Corals in Unity

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.



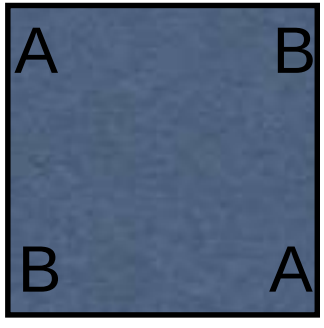


2D L-systems


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

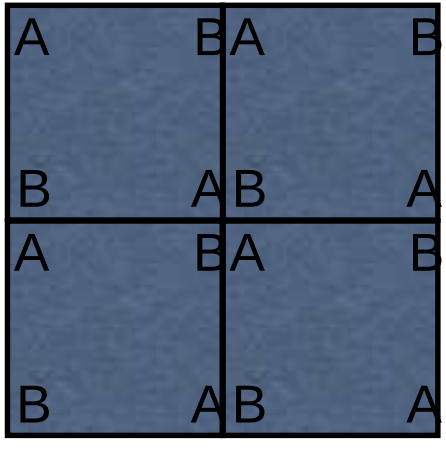
Terrain Interpretation of 2D L-systems

- 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

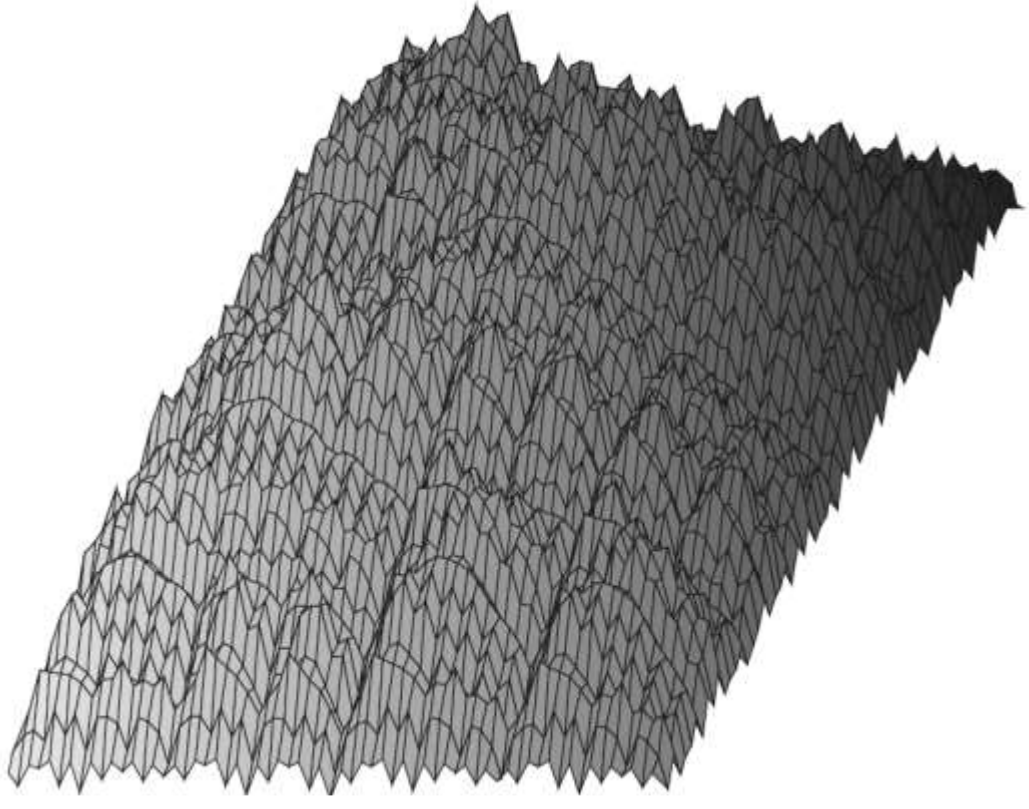


Terrain Interpretation of 2D L-systems

- 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

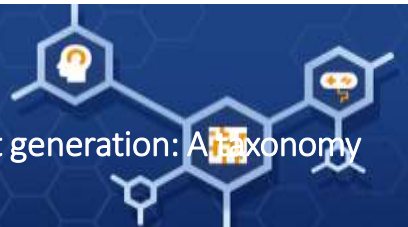
Grammars for Adventure Level Design

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

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.

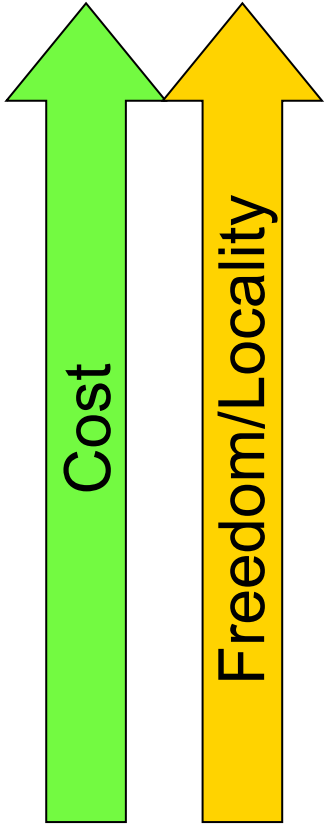
Search-Based PCG

- 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 μ+λ 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)

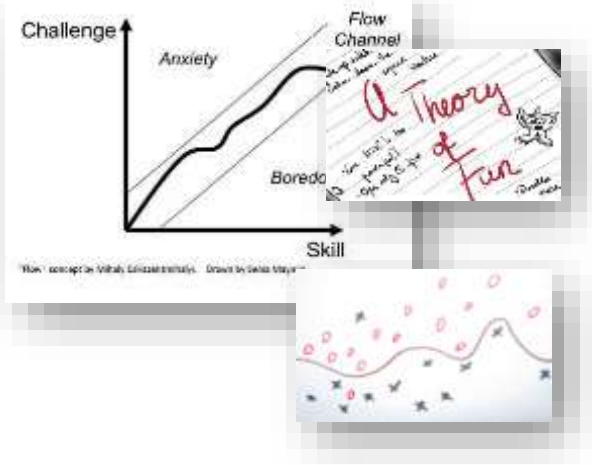
Togelius, J., Yannakakis, G. N., Stanley, K. O., & Browne, C. (2011). Search-based procedural content generation: A taxonomy and survey. *Computational Intelligence and AI in Games, IEEE Transactions on*, *3*(3), 172-186.

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.

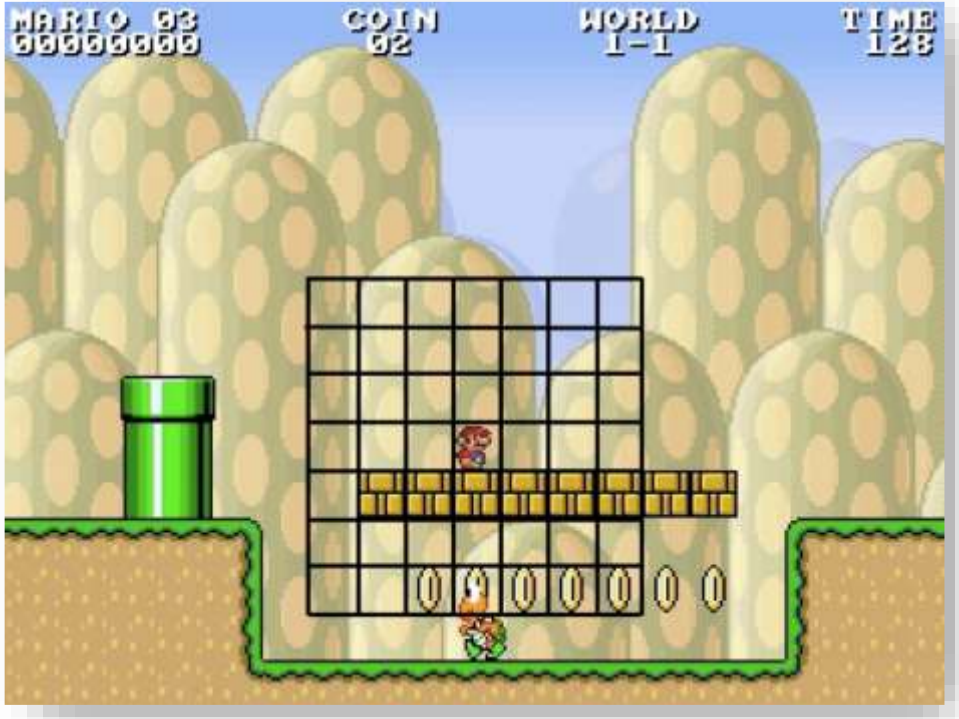
Content Quality

Direct	Simulation-based	Interactive
		

Search-Based PCG Example #1

How would we generate levels for Super Mario Bros?

The Mario AI Benchmark

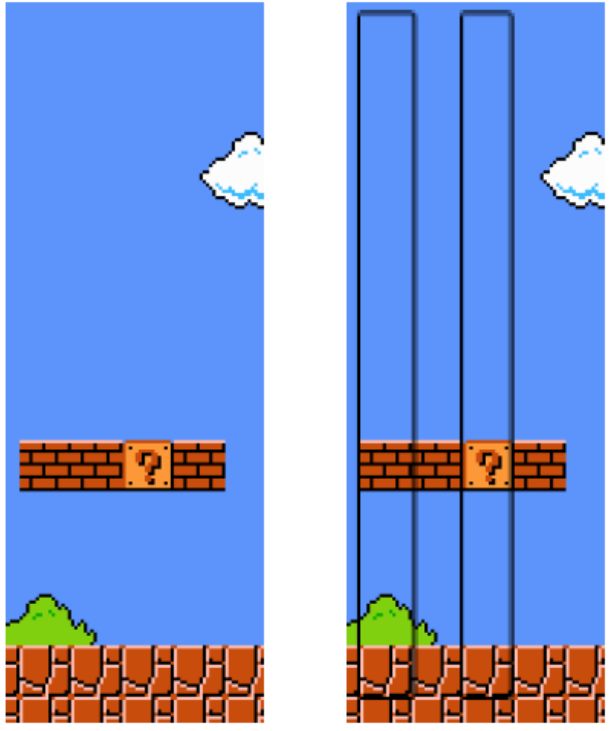


- Reasonably faithful clone of SMB 1/3
- APIs for level generators and AI controllers

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

Representation

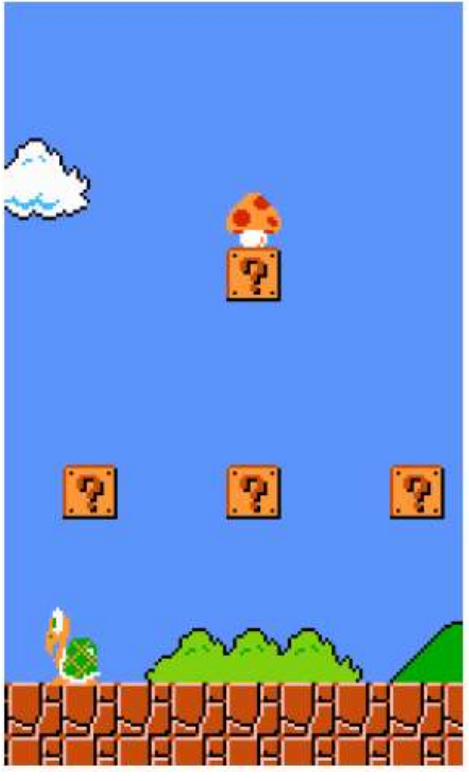
- A number of “vertical slices” are identified from the original SMB levels
- Levels are represented as strings, where each character correspond to a pattern



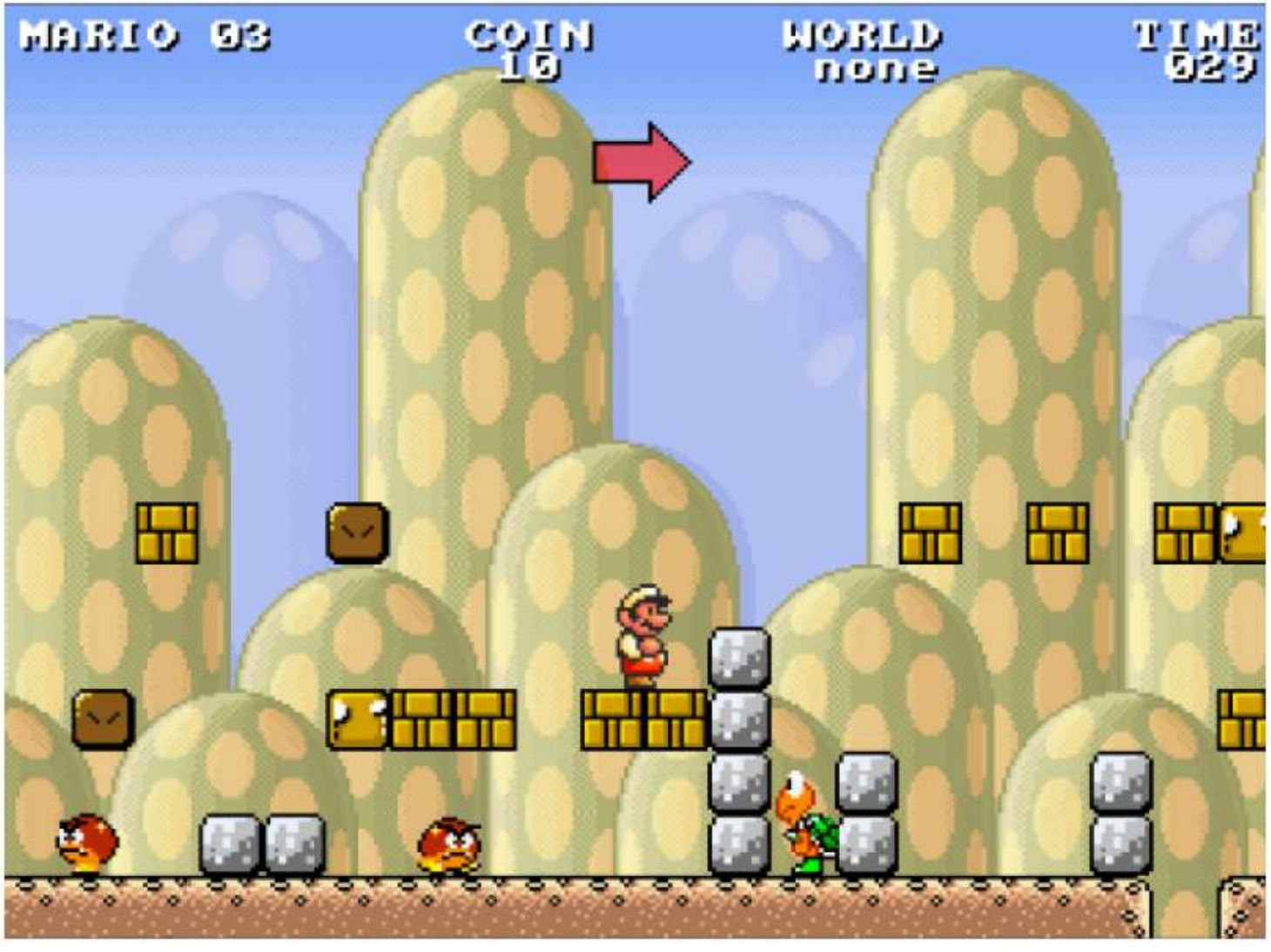
Steve Dahlskog and Julian Togelius: **Patterns as Objectives for Level Generation**. PCG Workshop 2013

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).

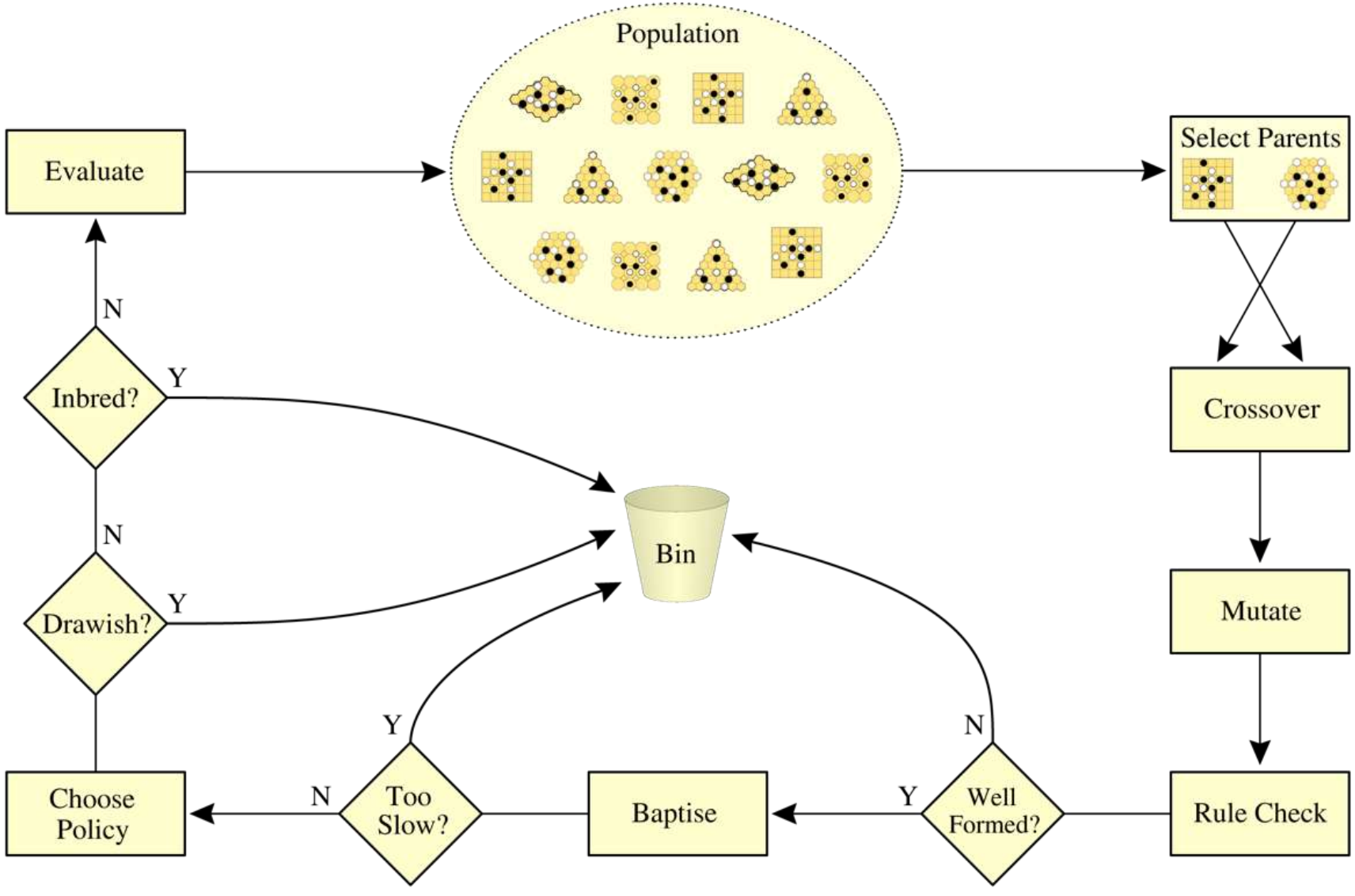
Fig. 18. FFMesoB #64 MC: 6, fitness value: 2065 (highest).

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)

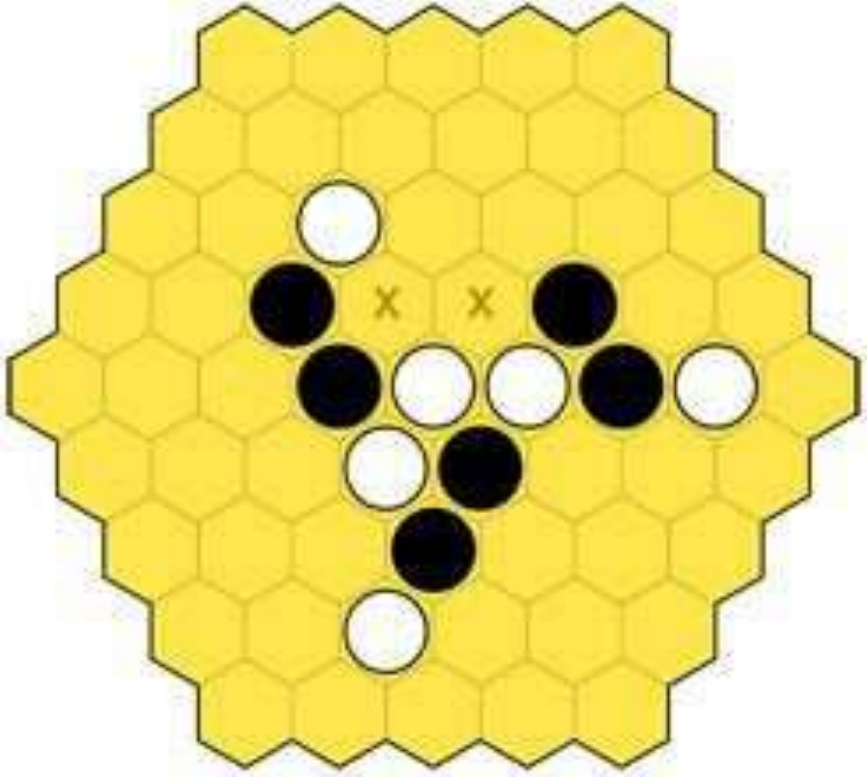

Population
Evaluate
Select Parents
Crossover
Mutate
Rule Check
Well Formed?
Baptise
Too Slow?
Choose Policy
Drawish?
Inbred?
Bin
N
Y
N
Y
N
Y
Y
N
N
Y

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

nestorgames

Yavalath

by Cameron Browne's LUDI

nestorgames

Search-Based PCG Example #2

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

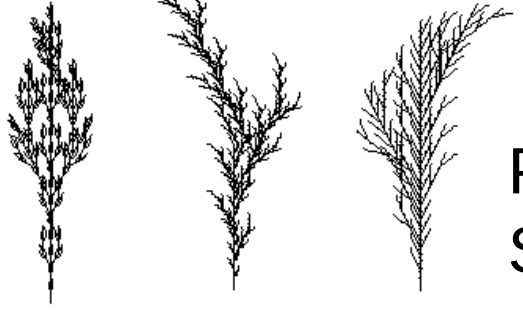
Evolved Systems

Symmetry

Branching points

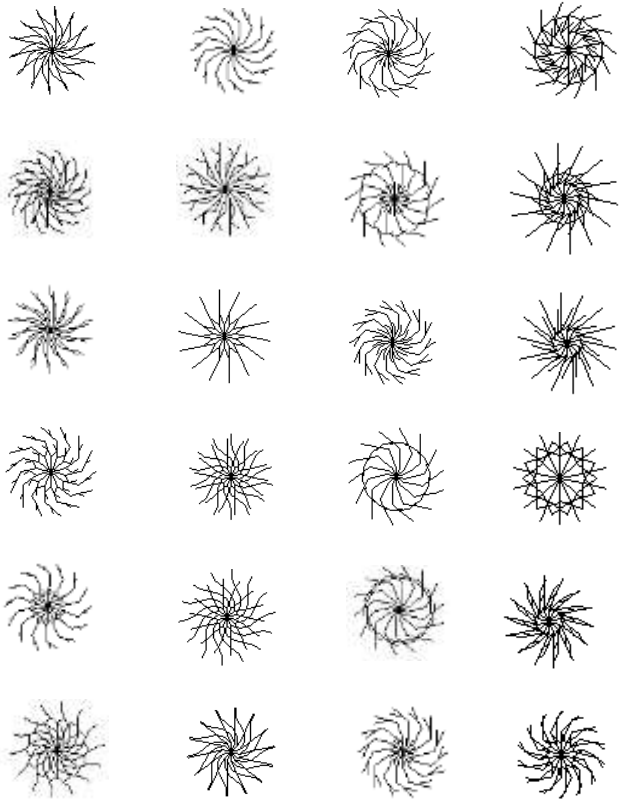
All 3

Phototropism

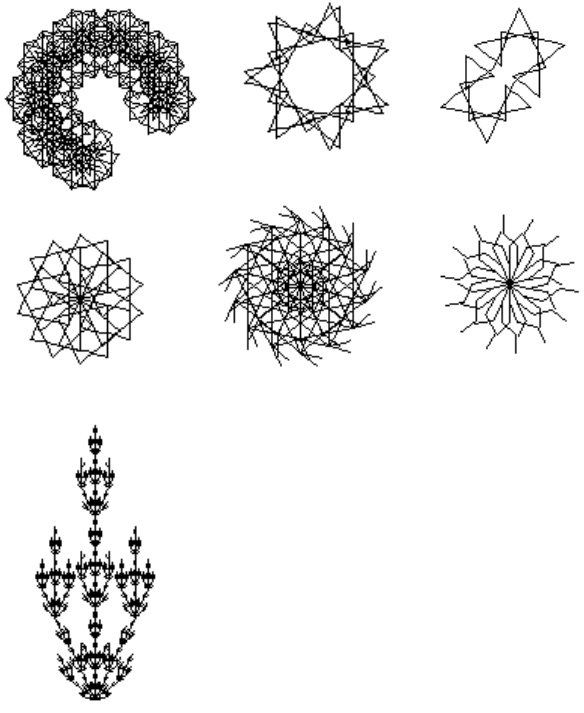


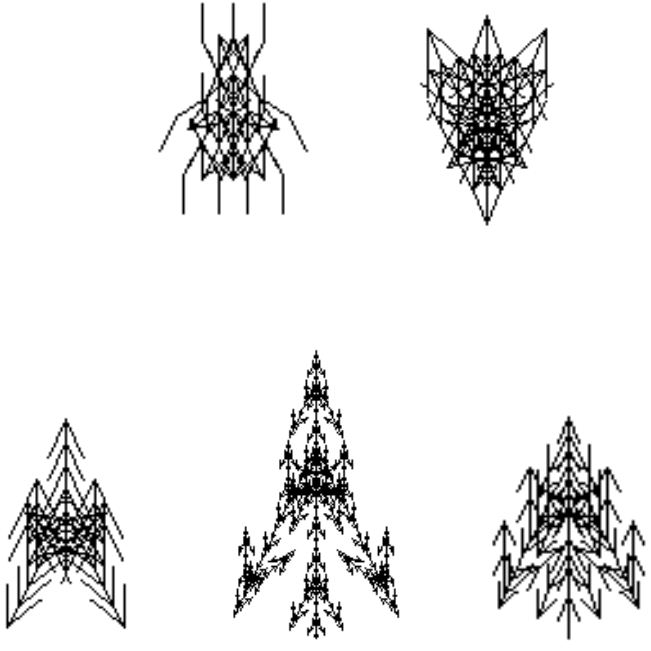
Phototropism + Symmetry

Evolved Systems



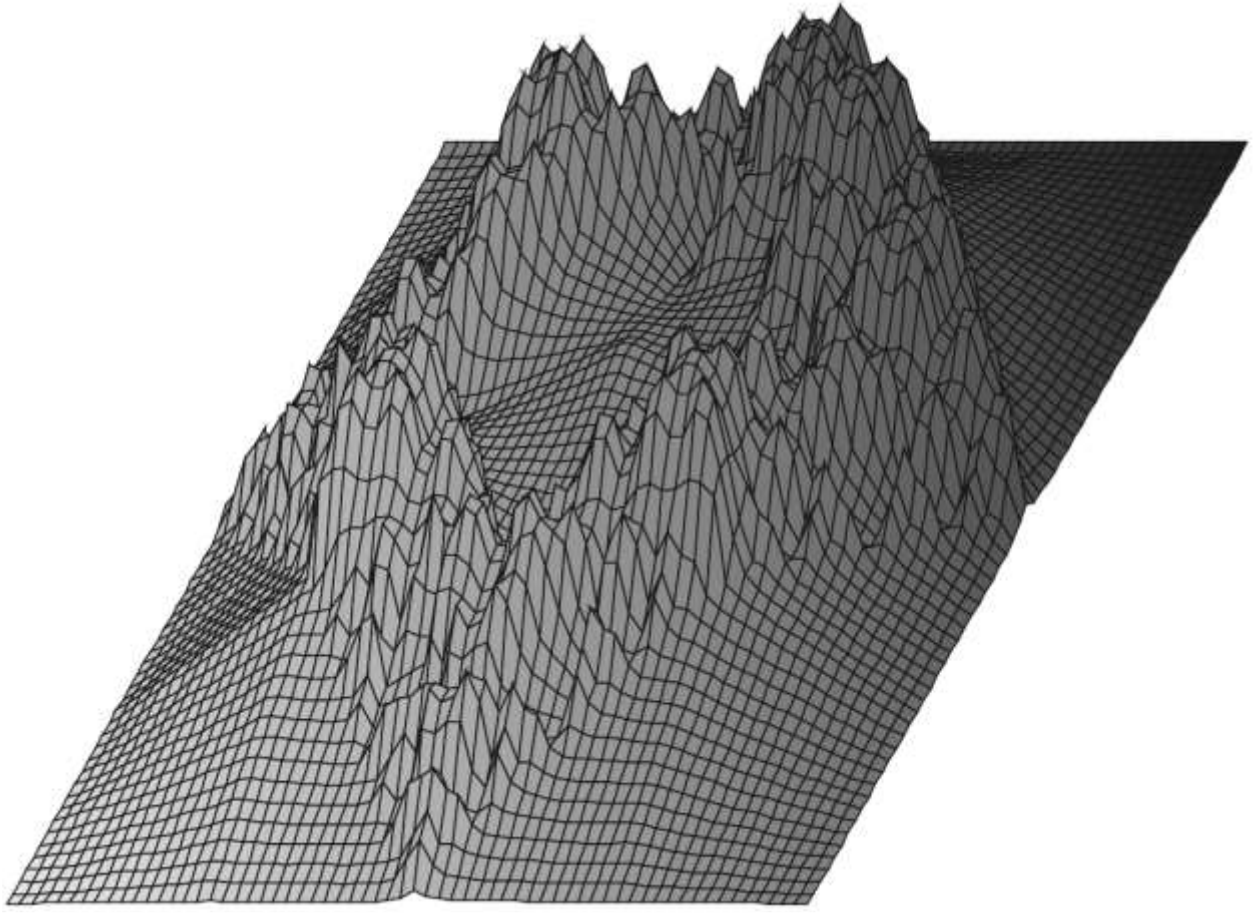
Evolved Systems



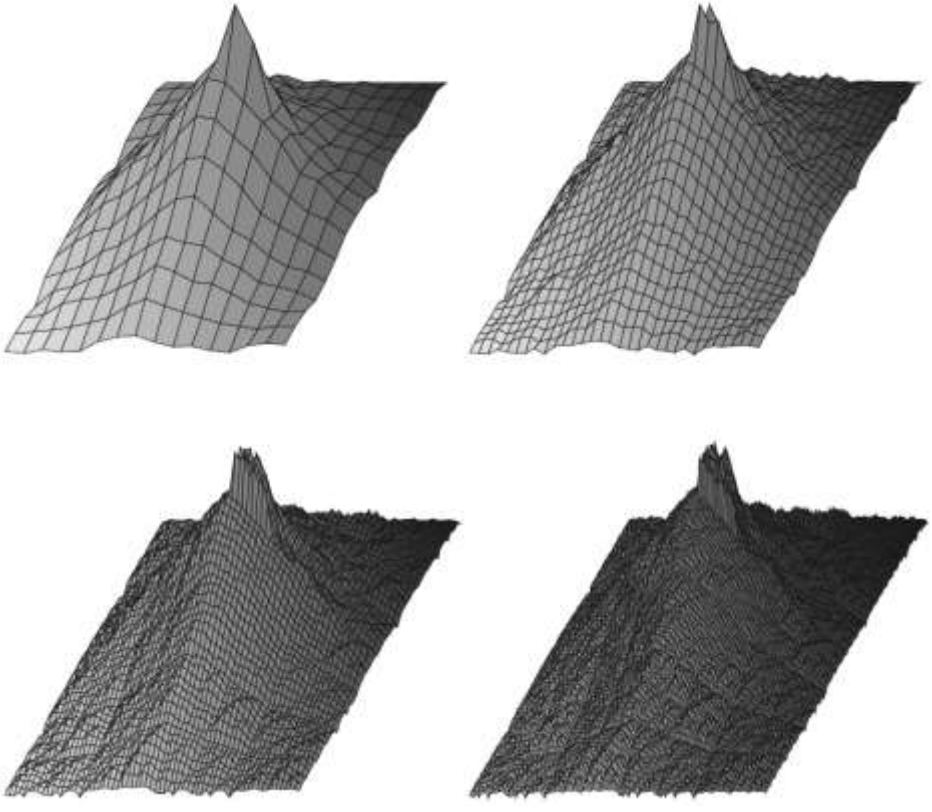


…and this was an extremely simple L-system!

Evolved 2D L-system Terrains



Evolved 2D L-system Terrains



Very short specification, yet infinite resolution!

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
- Reference: Summerville et al. (2018): Procedural Content Generation via Machine Learning (PCGML)

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

(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

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

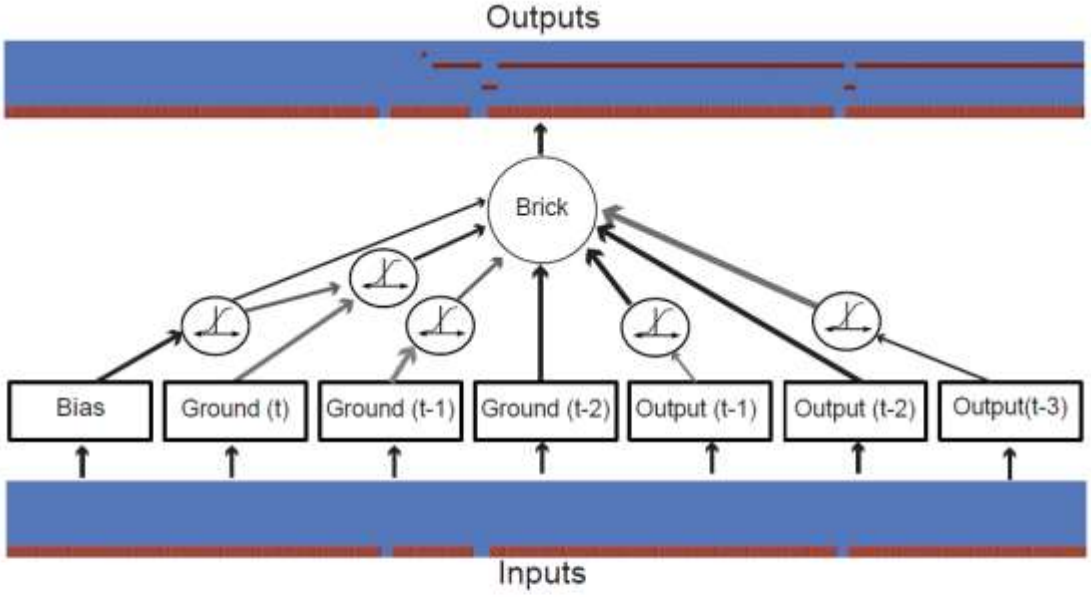


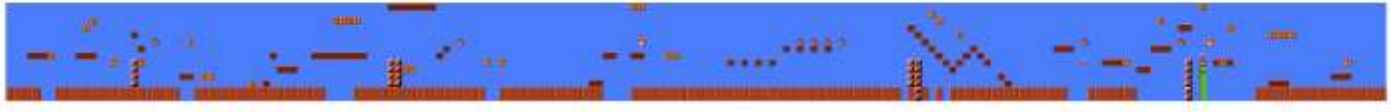
Figure 1: **The ANN Representation**. At each column in a given level, an ANN inputs visual assets (i.e., musical voices) from at least one tile type of *Super Mario Bros.* and outputs a different type of visual asset. This example figure shows ground tiles being input to the ANN while the brick tile 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



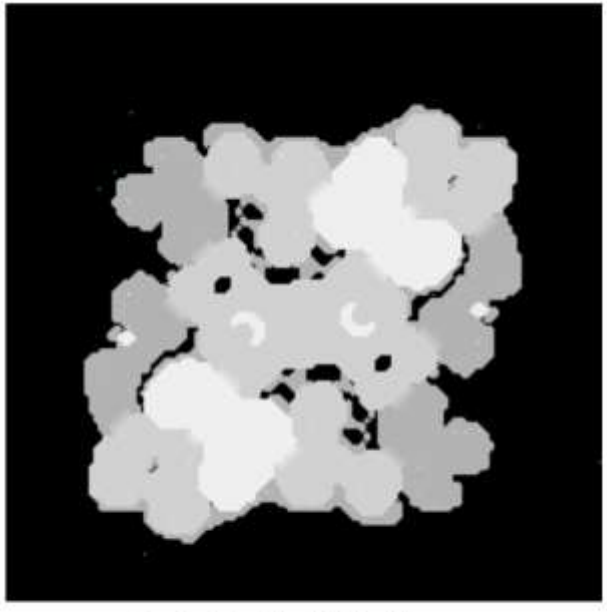
(e) Additional voices generated by trained networks for the author-created level.

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.

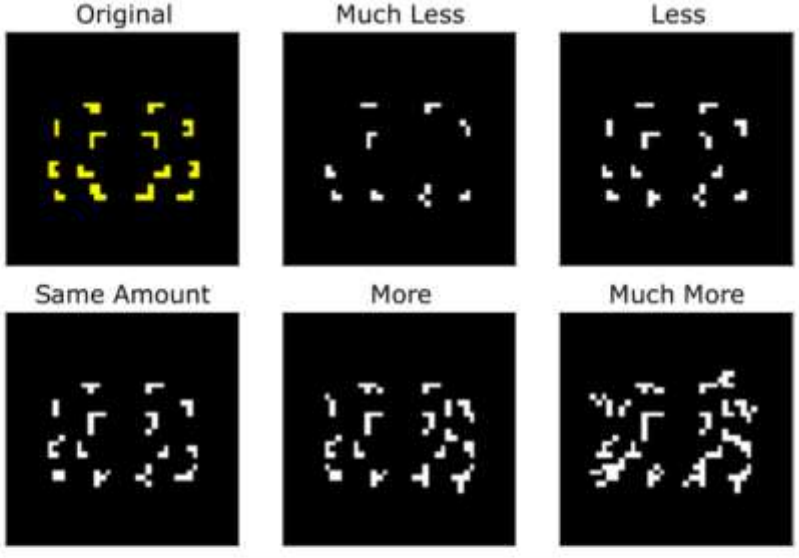
Convolutional Neural Nets

- 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

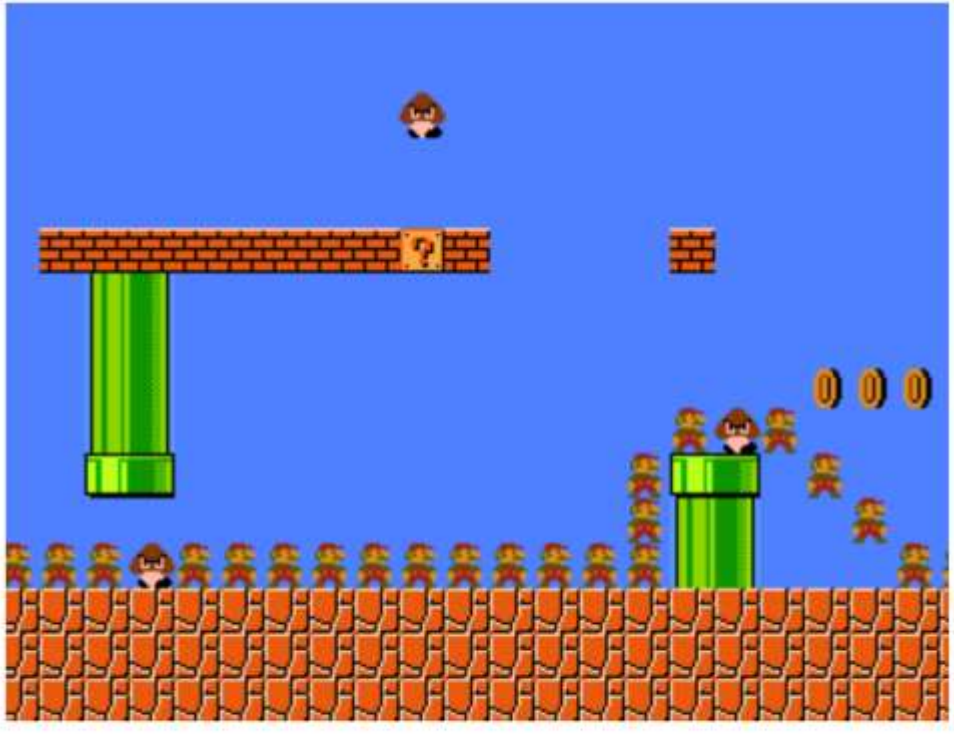


Figure 2: Example output of the LSTM approach, including generated exemplar player path.

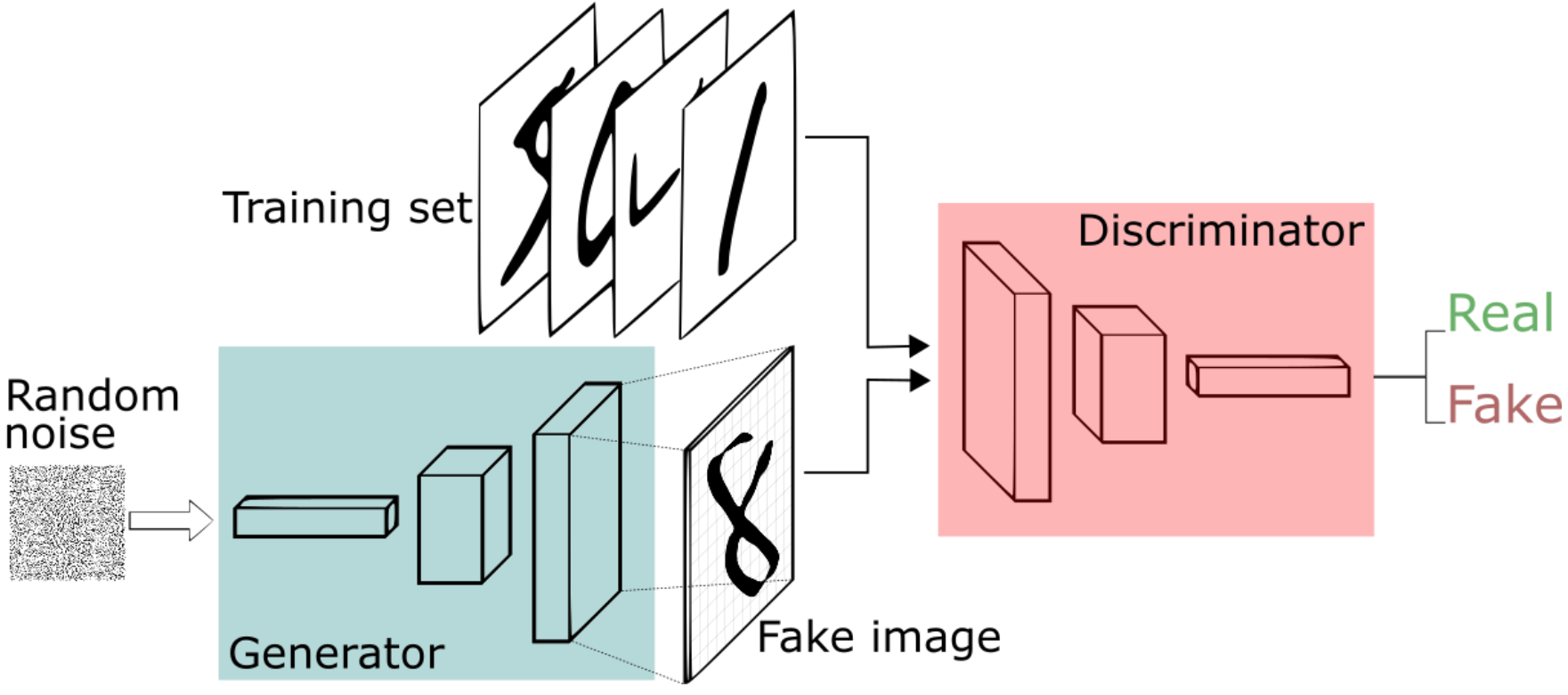
LSTM for Magic Cards

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

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



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

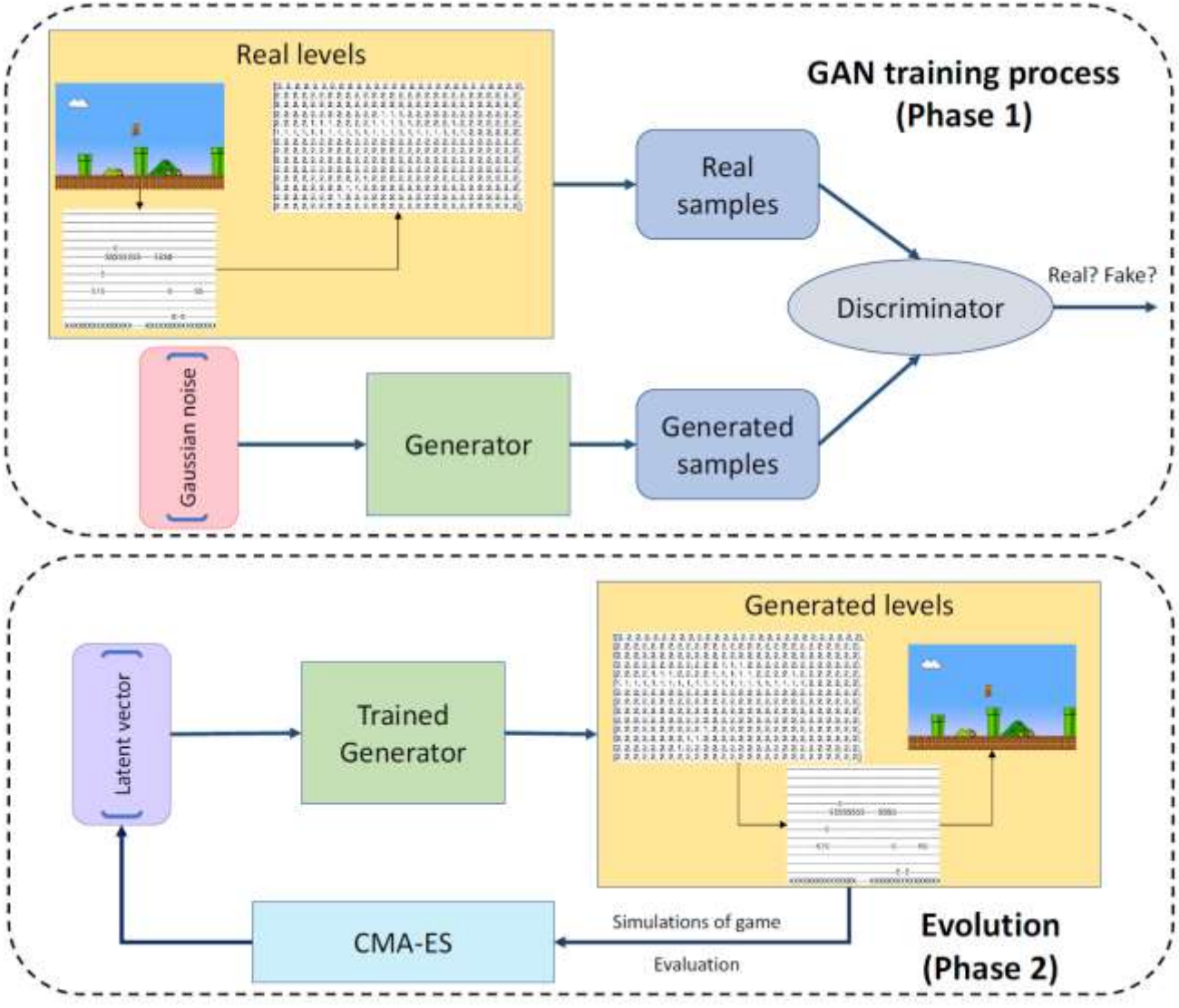
Jialin Liu
Queen Mary University of London
London, UK
jialin.liu@qmul.ac.uk

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)

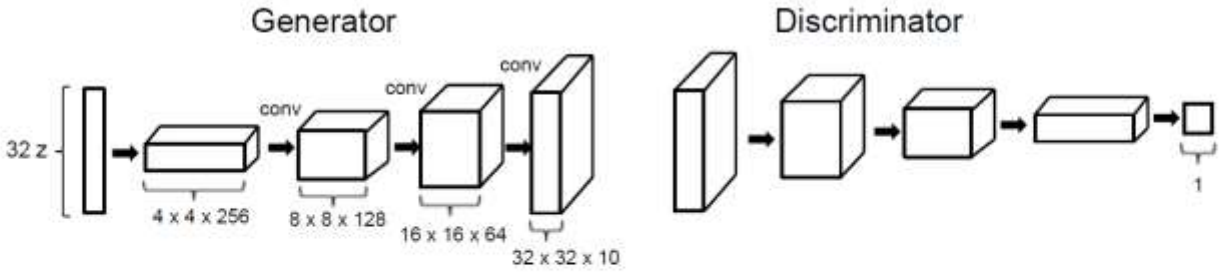


Figure 3: The Mario DCGAN architecture.

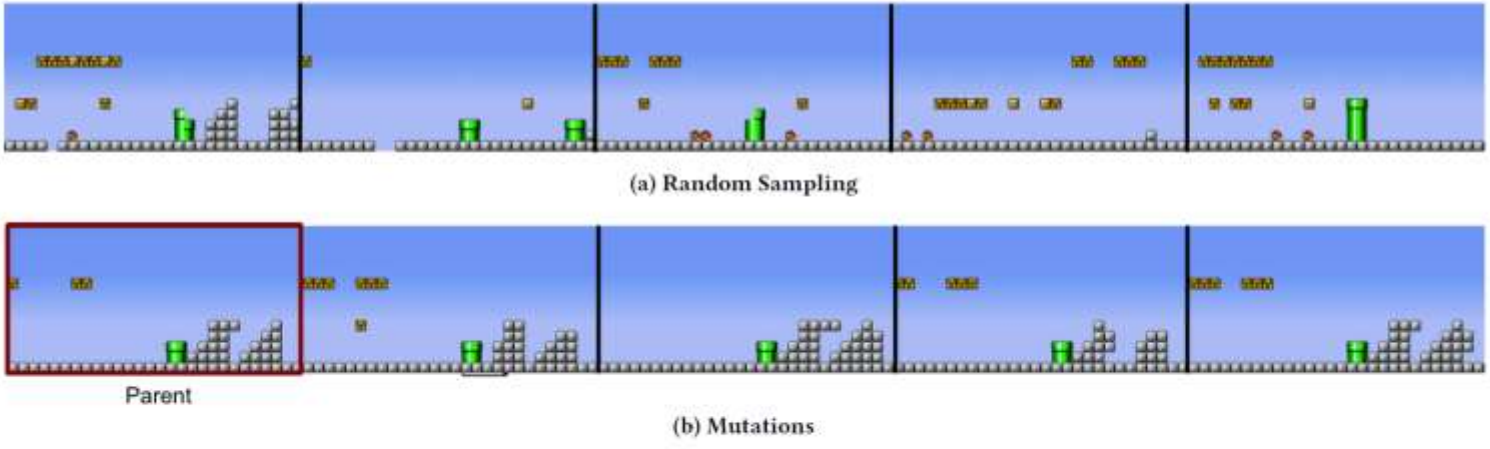
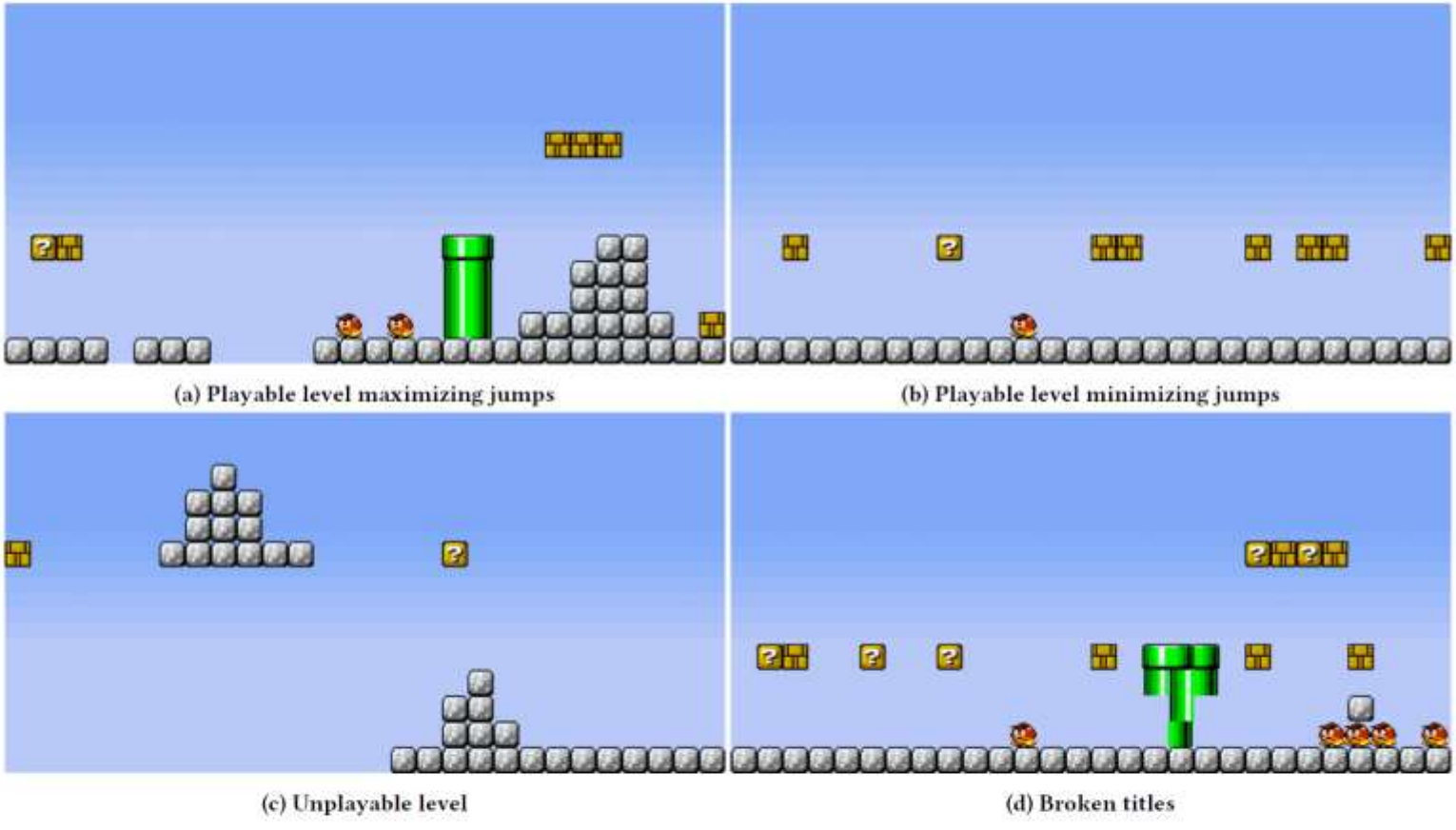


Figure 4: Generated Examples. Shown are samples produced by the GAN by (a) sampling random latent vectors, and (b) randomly mutating a specific latent vector. The main result is that the generator is able to produce a wide variety of different level layouts, but varied offspring still resemble their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



(a) Playable level maximizing jumps

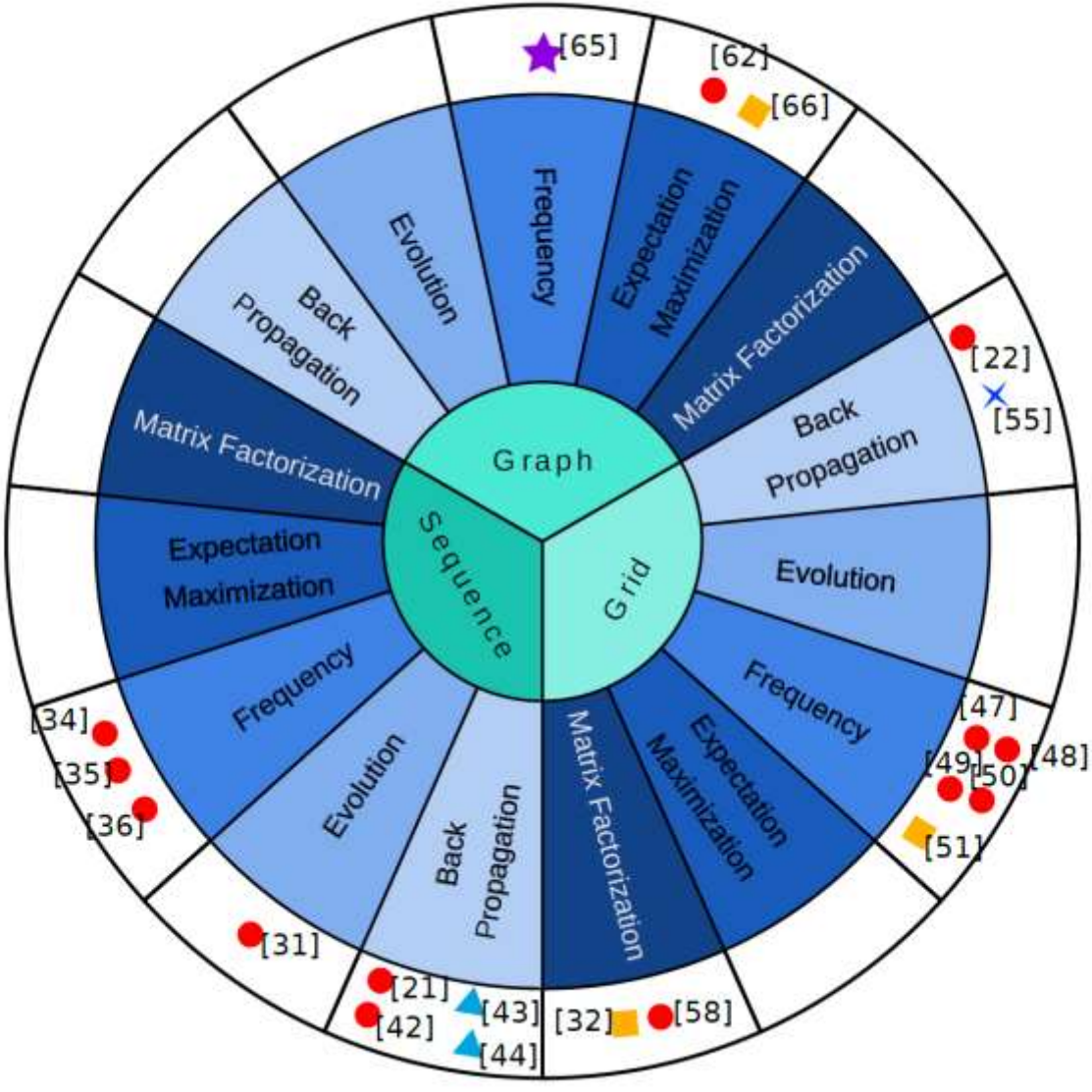
(b) Playable level minimizing jumps

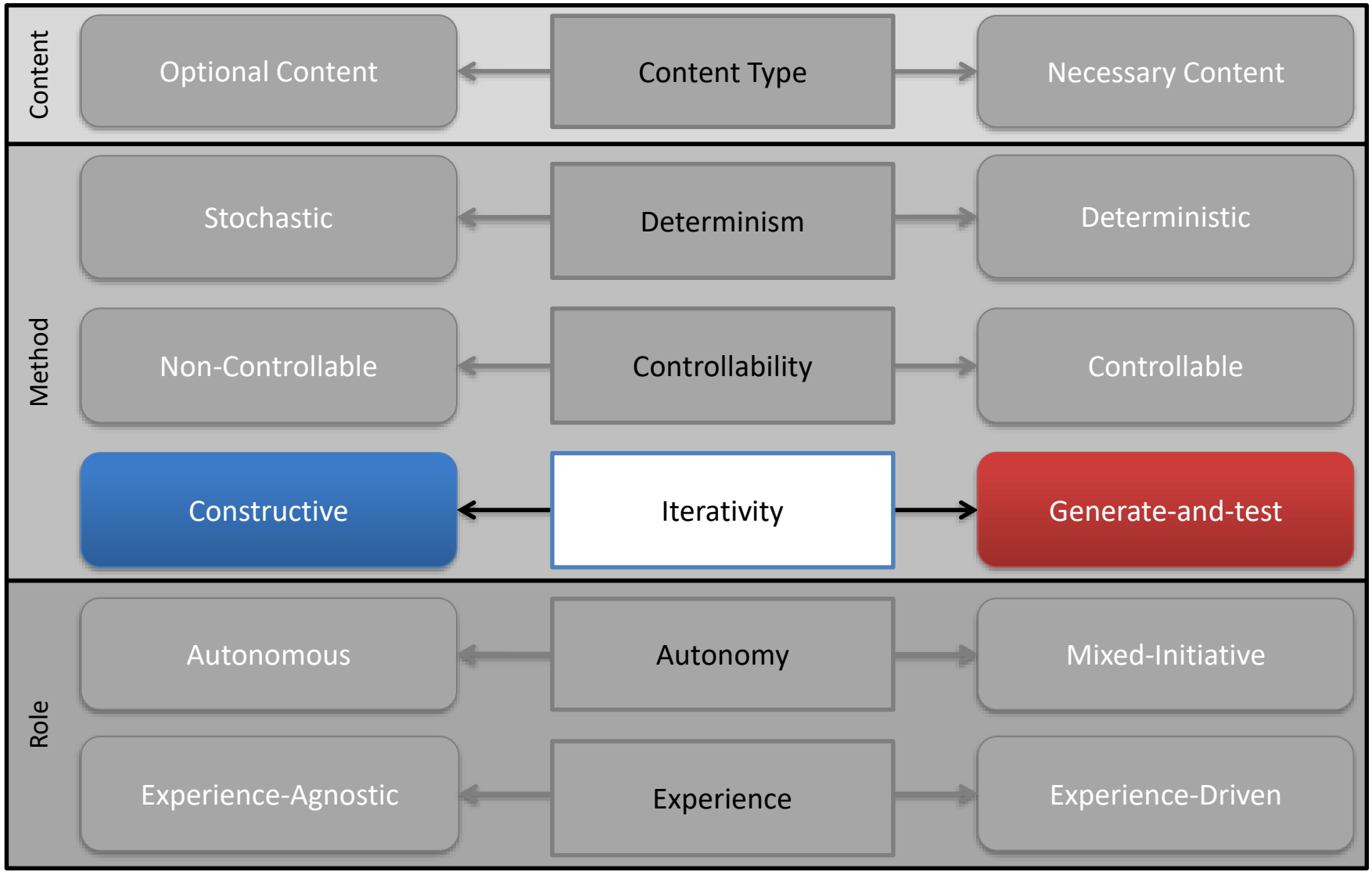
(c) Unplayable level

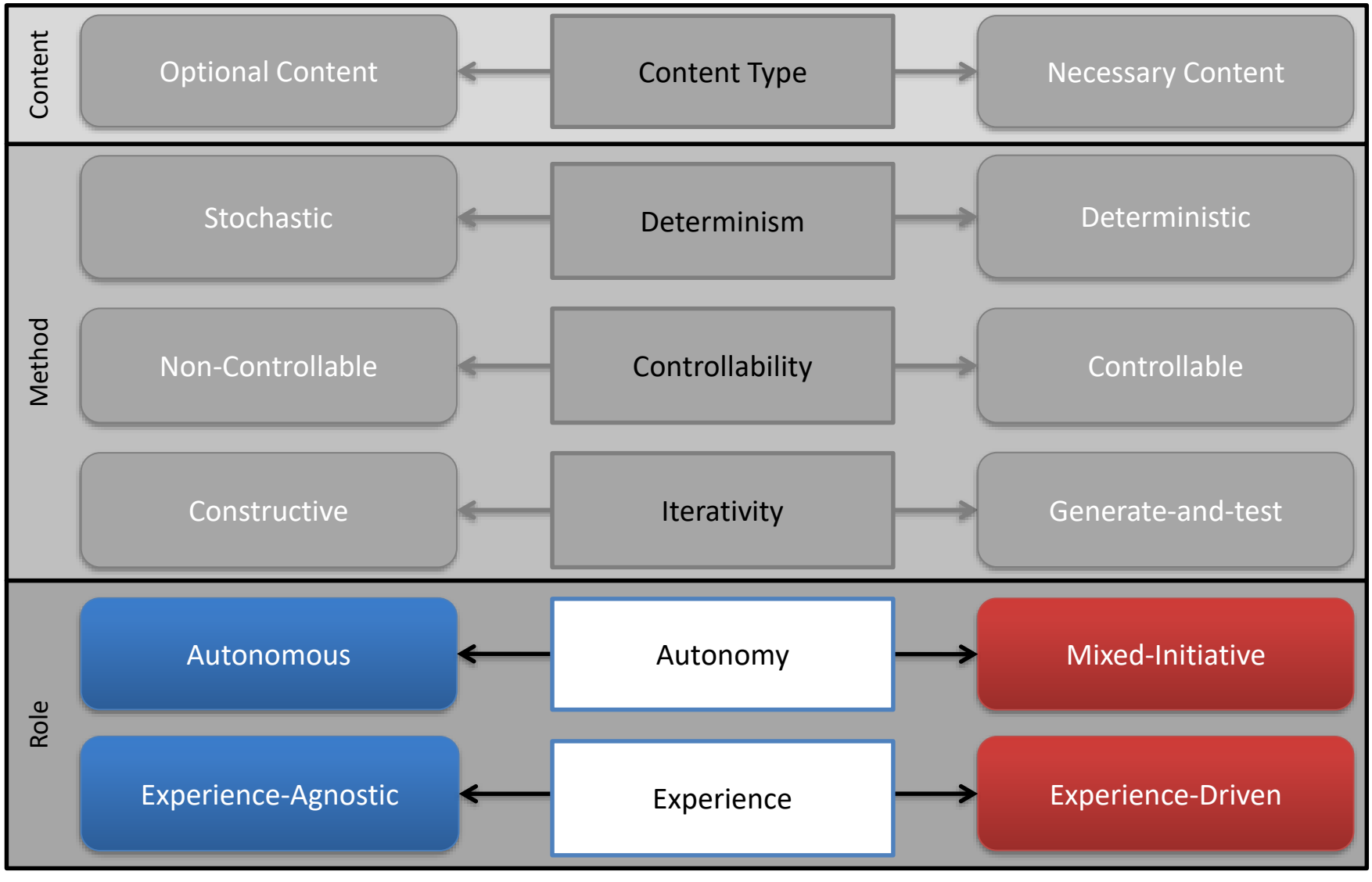
(d) Broken titles

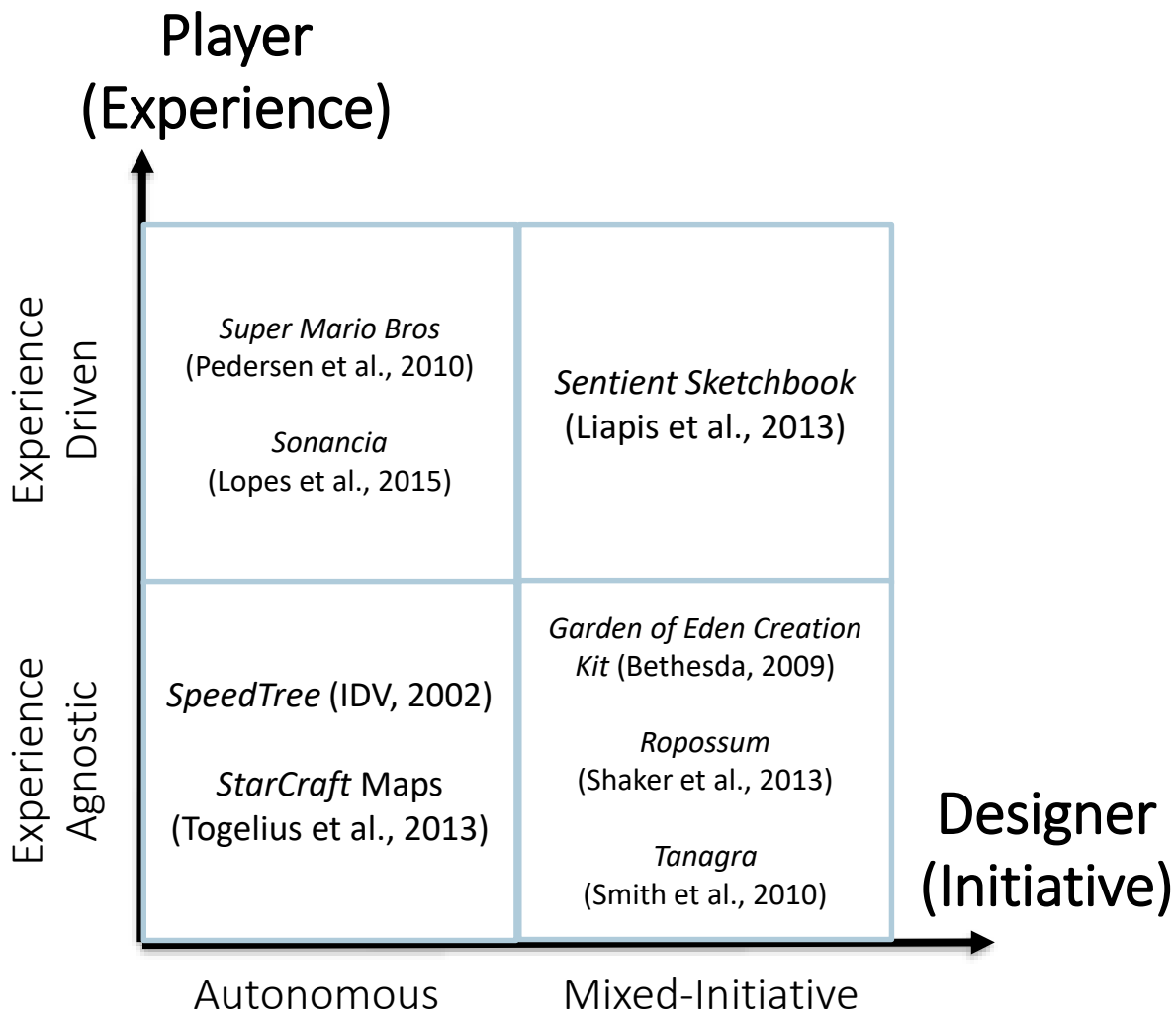
Figure 7: Agent-based optimization examples. (a) and (b) show good examples of levels in which the number of jumps is maximized (F_1), and minimized (F_2), respectively. (c) shows an example of one of the worst individuals found (not playable, F_1). An example of an individual that reaches high fitness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

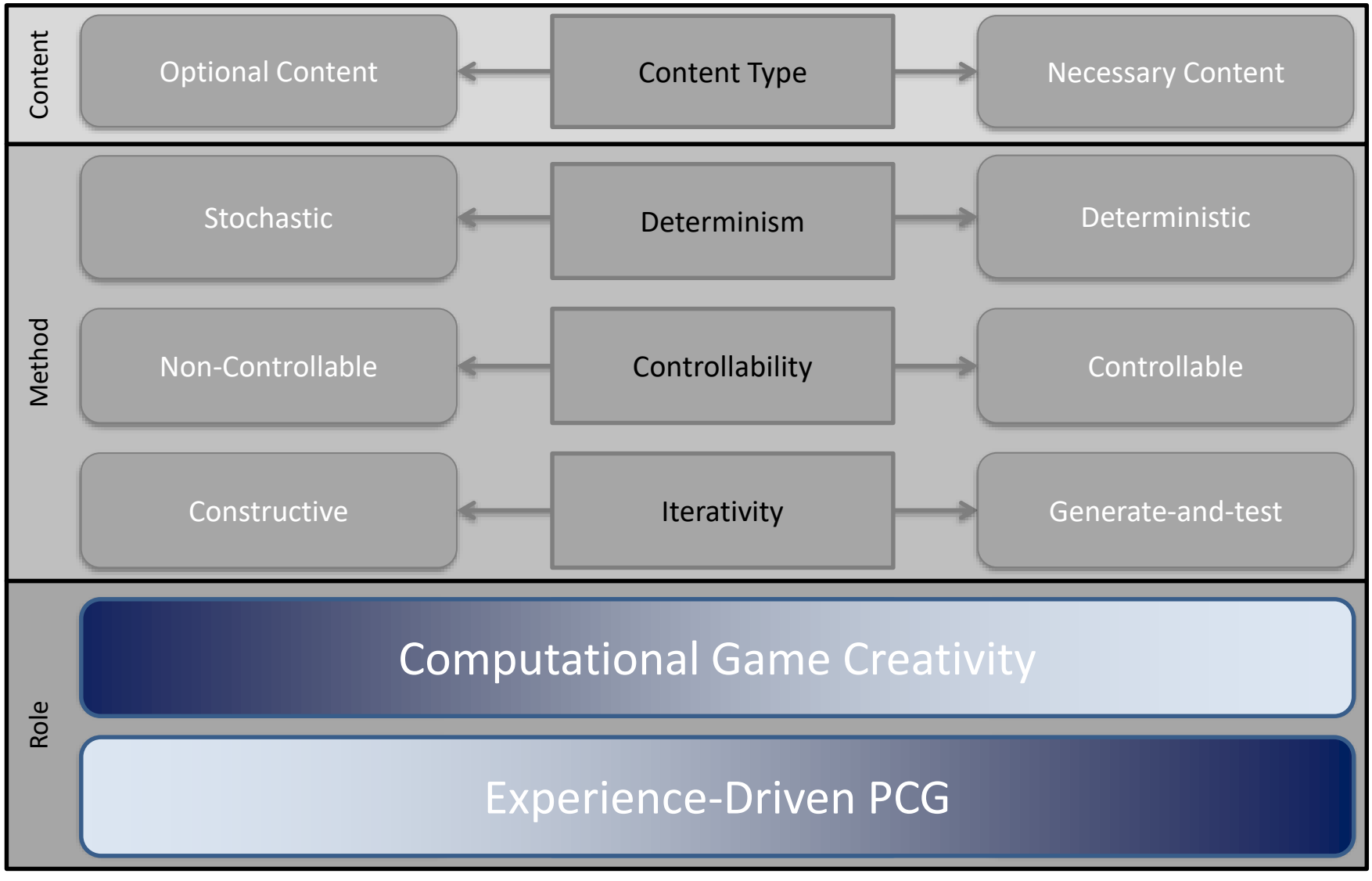
Figure 6: Level with increasing difficulty. Our LVE approach can create levels composed of multiple parts that gradually increase in difficulty (less ground tiles, more enemies). In the future this approach could be used to create a level in real-time that is tailored to the particular skill of the player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


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


Content
Optional Content
Content Type
Necessary Content
Method
Stochastic
Determinism
Deterministic
Non-Controllable
Controllability
Controllable
Constructive
Iterativity
Generate-and-test
Role
Autonomous
Autonomy
Mixed-Initiative
Experience-Agnostic
Experience
Experience-Driven


Content
Optional Content
Content Type
Necessary Content
Method
Stochastic
Determinism
Deterministic
Non-Controllable
Controllability
Controllable
Constructive
Iterativity
Generate-and-test
Role
Autonomous
Autonomy
Mixed-Initiative
Experience-Agnostic
Experience
Experience-Driven


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


Content
Optional Content
Content Type
Necessary Content
Method
Stochastic
Determinism
Deterministic
Non-Controllable
Controllability
Controllable
Constructive
Iterativity
Generate-and-test
Role
Computational Game Creativity
Experience-Driven PCG

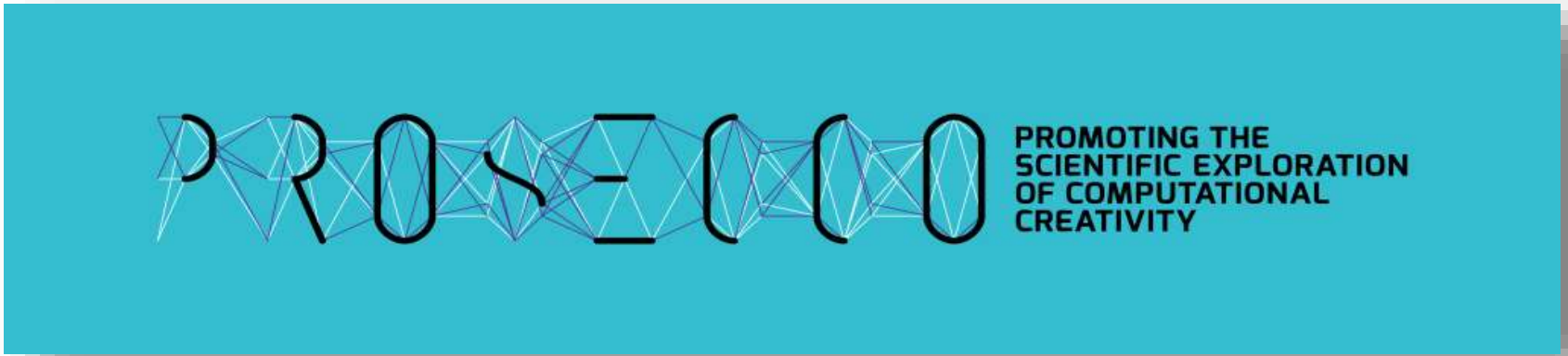
Computational Creativity **in** and **for** Games!

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



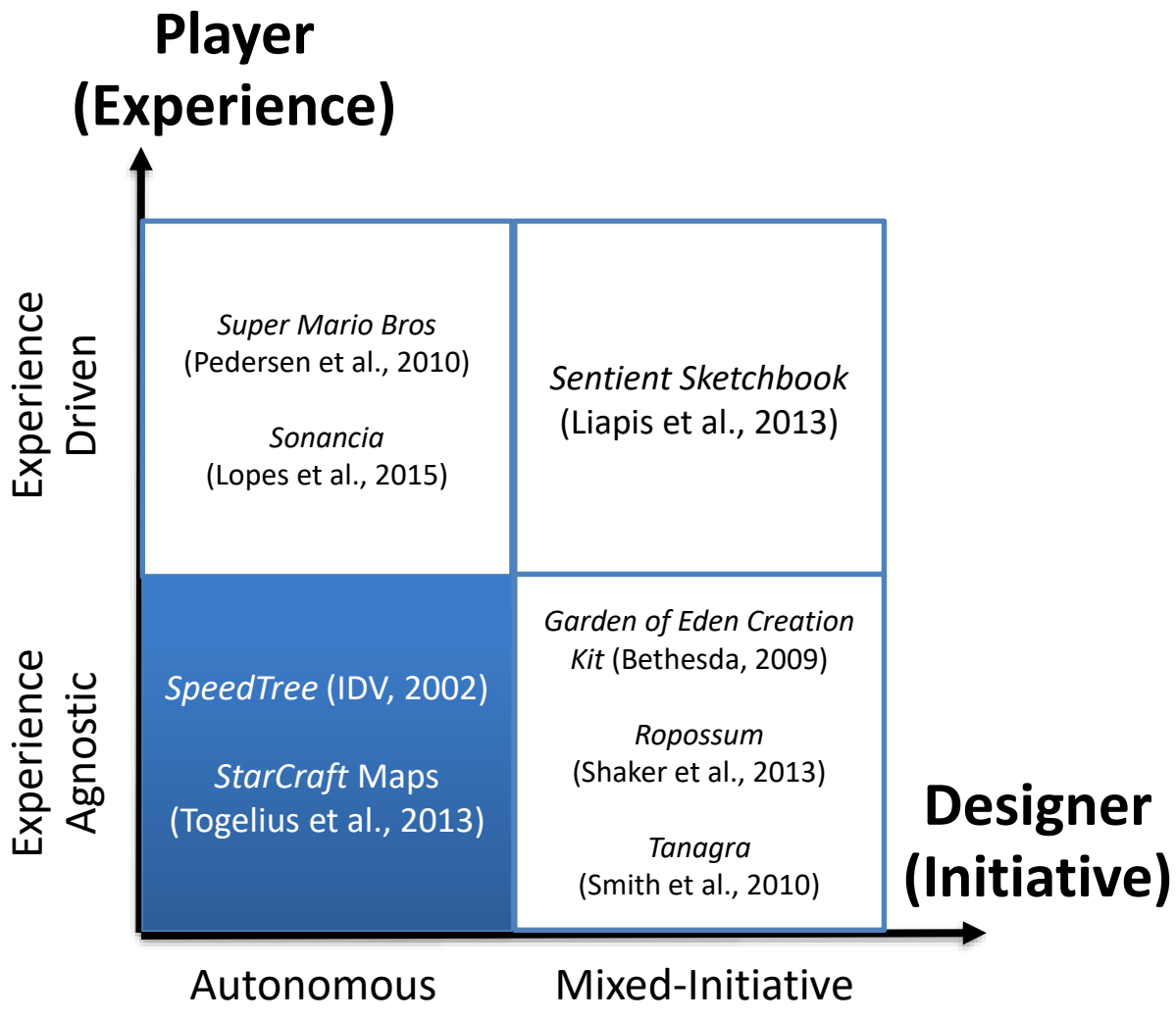
Computational **Game** Creativity

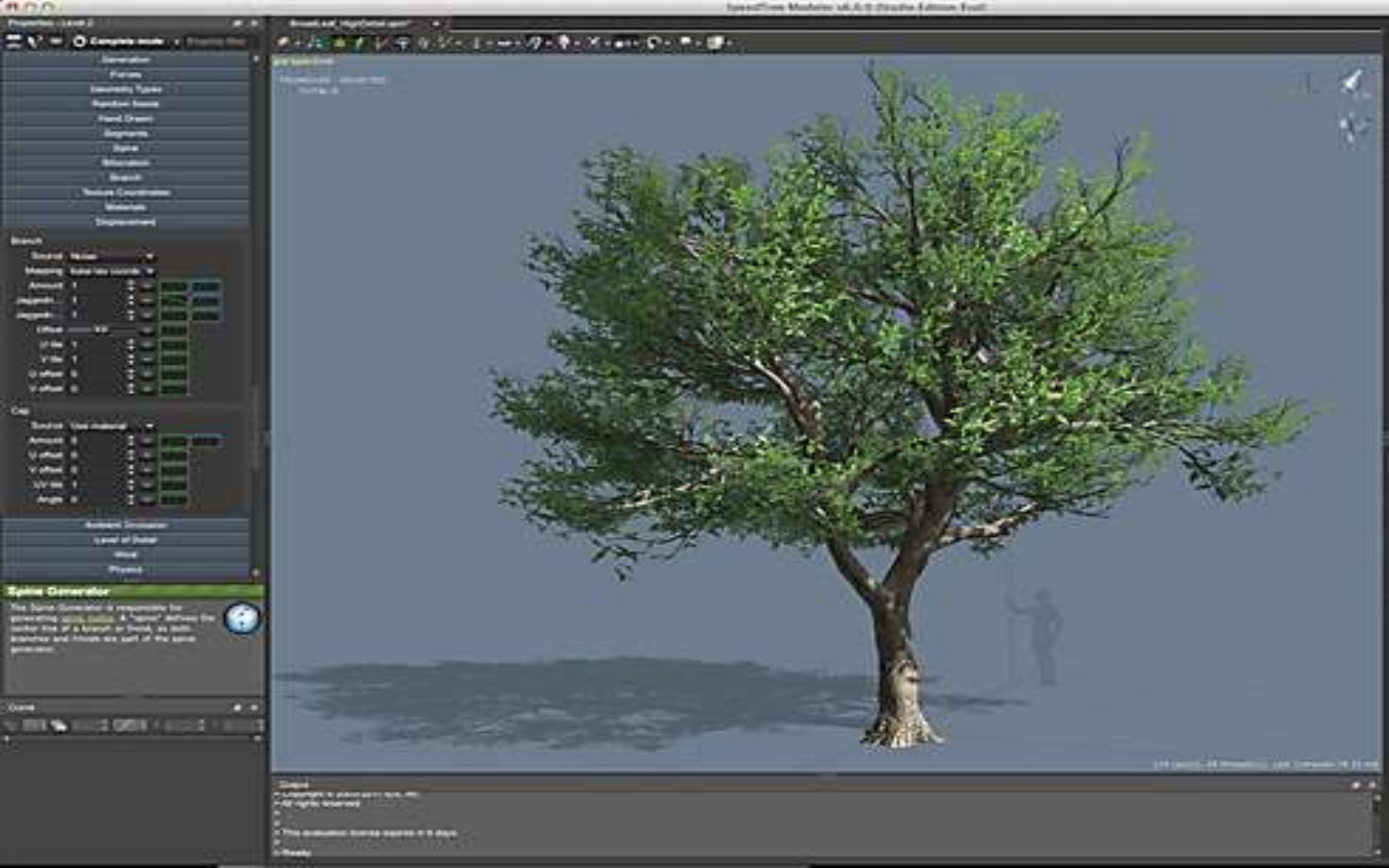
"The study of Computational Creativity **within** and **for** digital games"

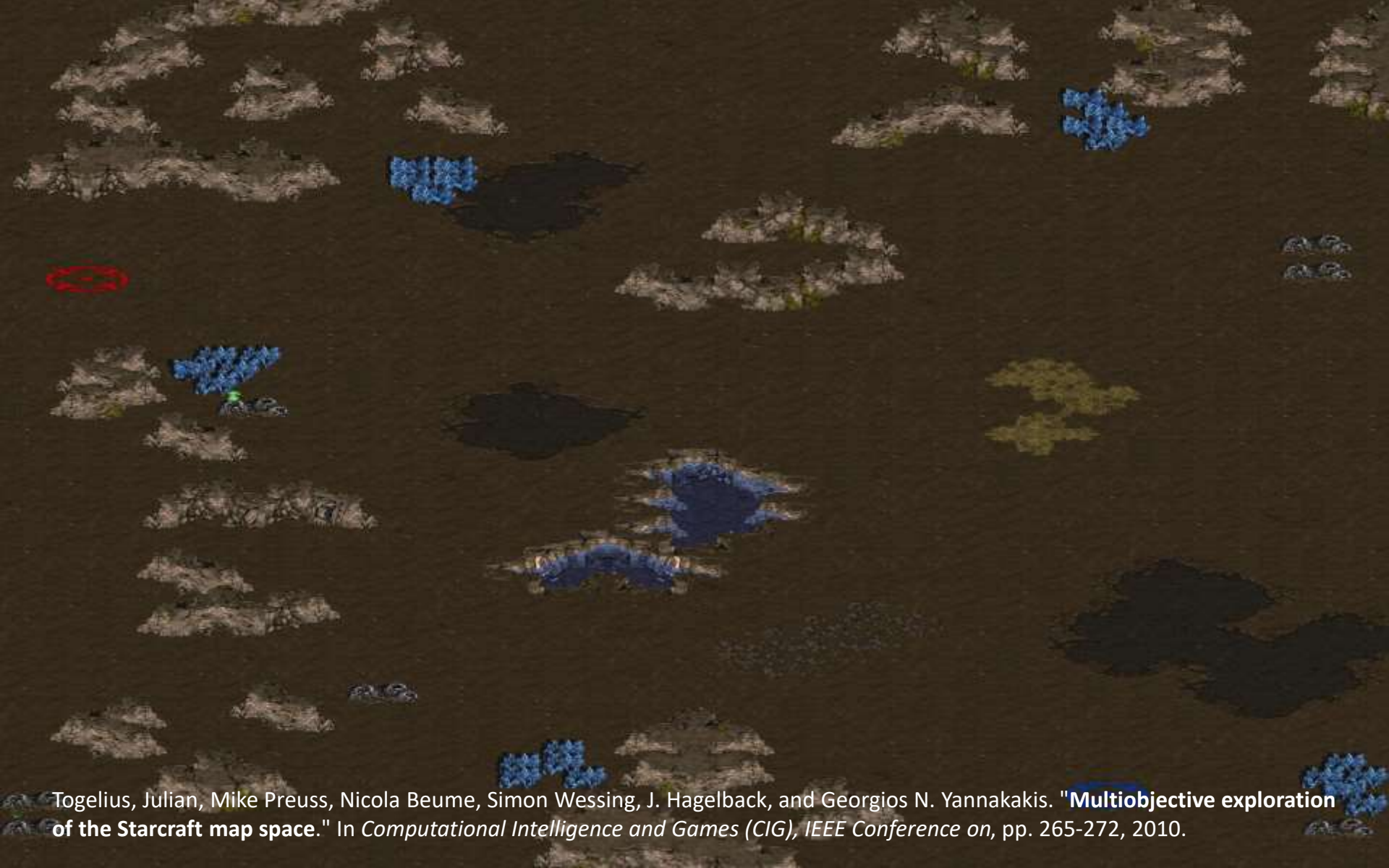
Within: Games as an ideal canvas for studying CC

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

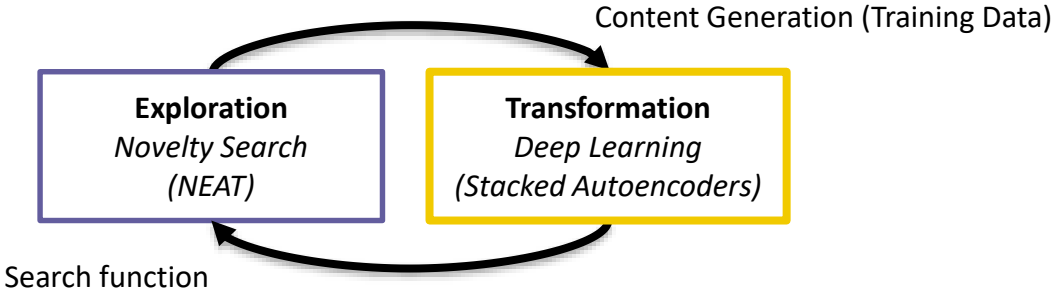

Generation
Fronds
Geometry Types
Random Seeds
Hard Edges
Segments
Spine
Structure
Branch
Texture Coordinates
Materials
Optimizations
Branch
Cap
Ambient Occlusion
Level of Detail
Wind
Physics
Spine Generator



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.

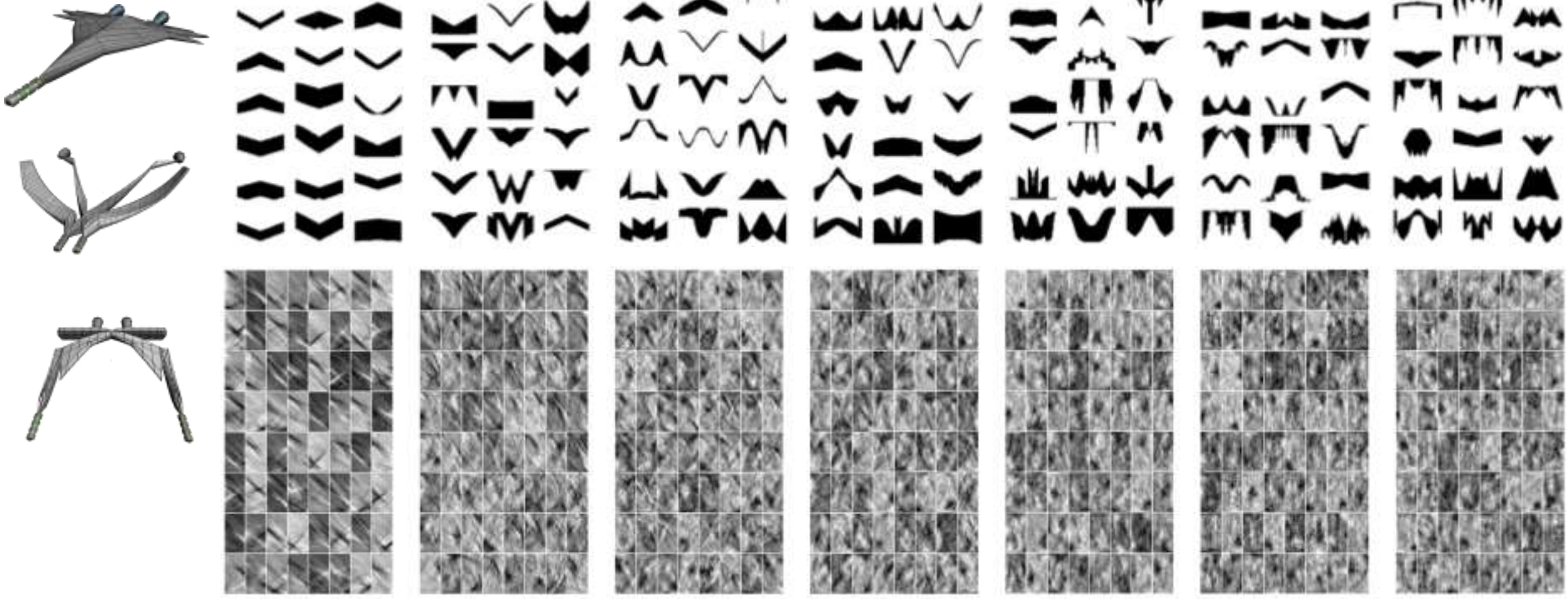
Deep Learning Meets Novelty Search

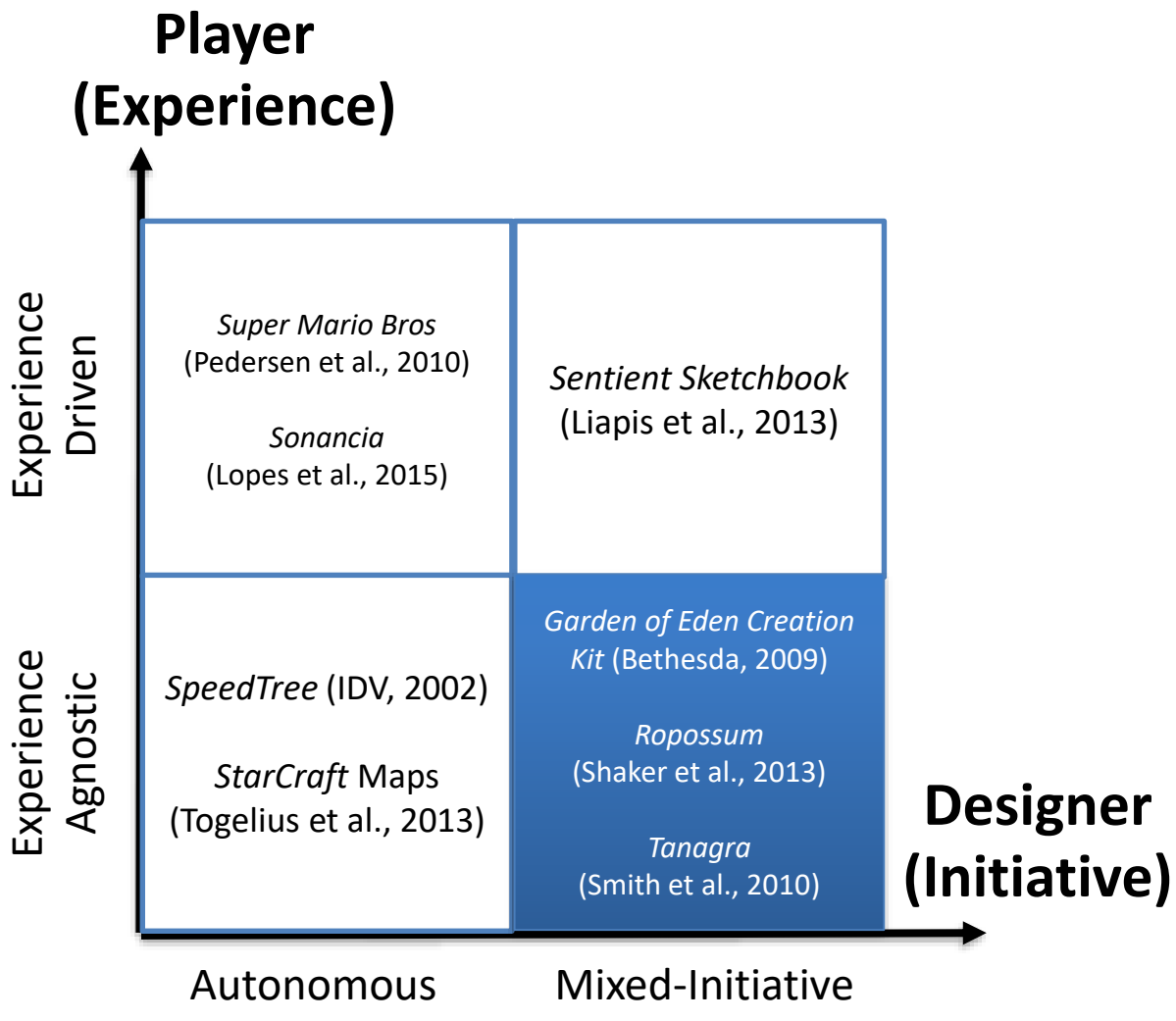
Liapis, Martínez, Togelius, and Yannakakis: **"Transforming Exploratory Creativity with DeLeNoX,"** in *Proceedings of the Fourth International Conference on Computational Creativity*, 2013.

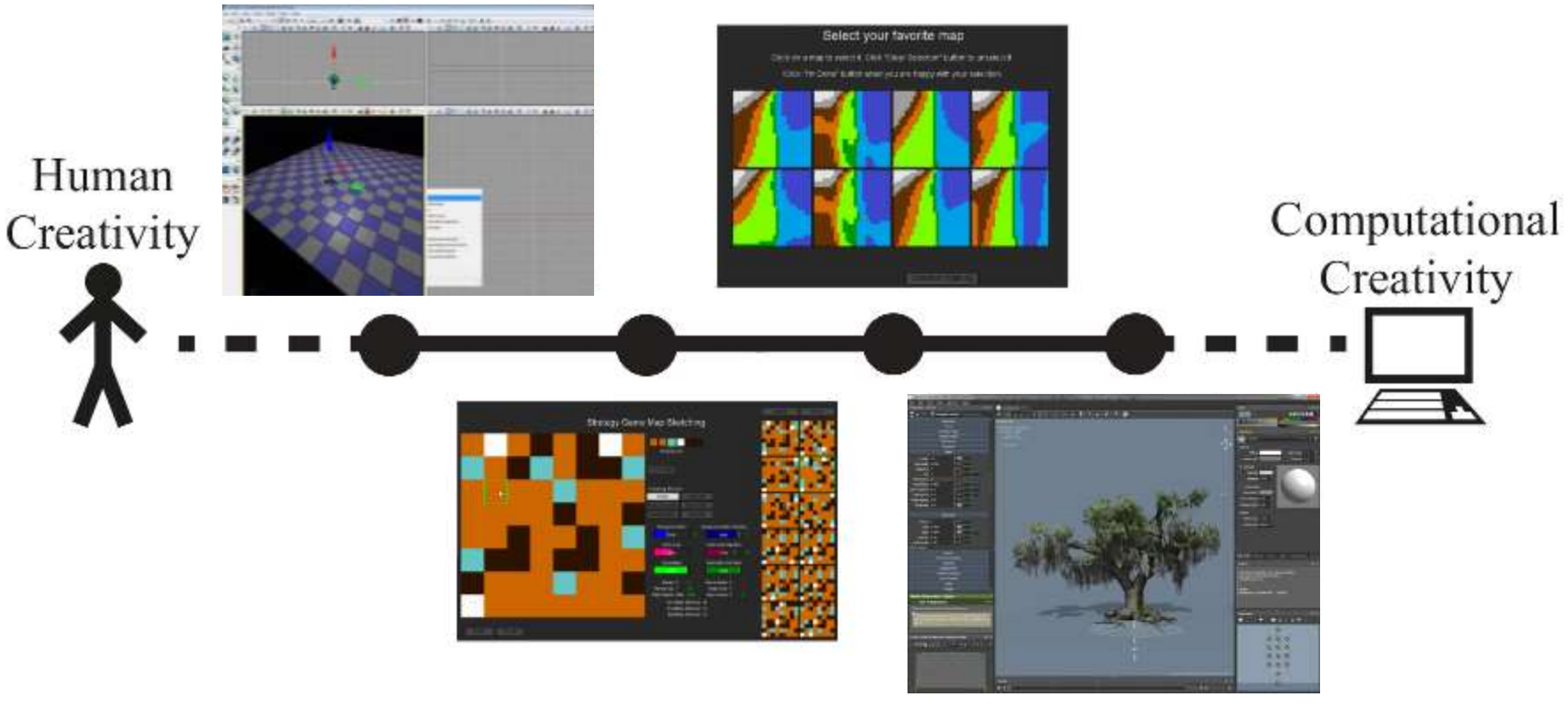


Deep Learning Meets Novelty Search

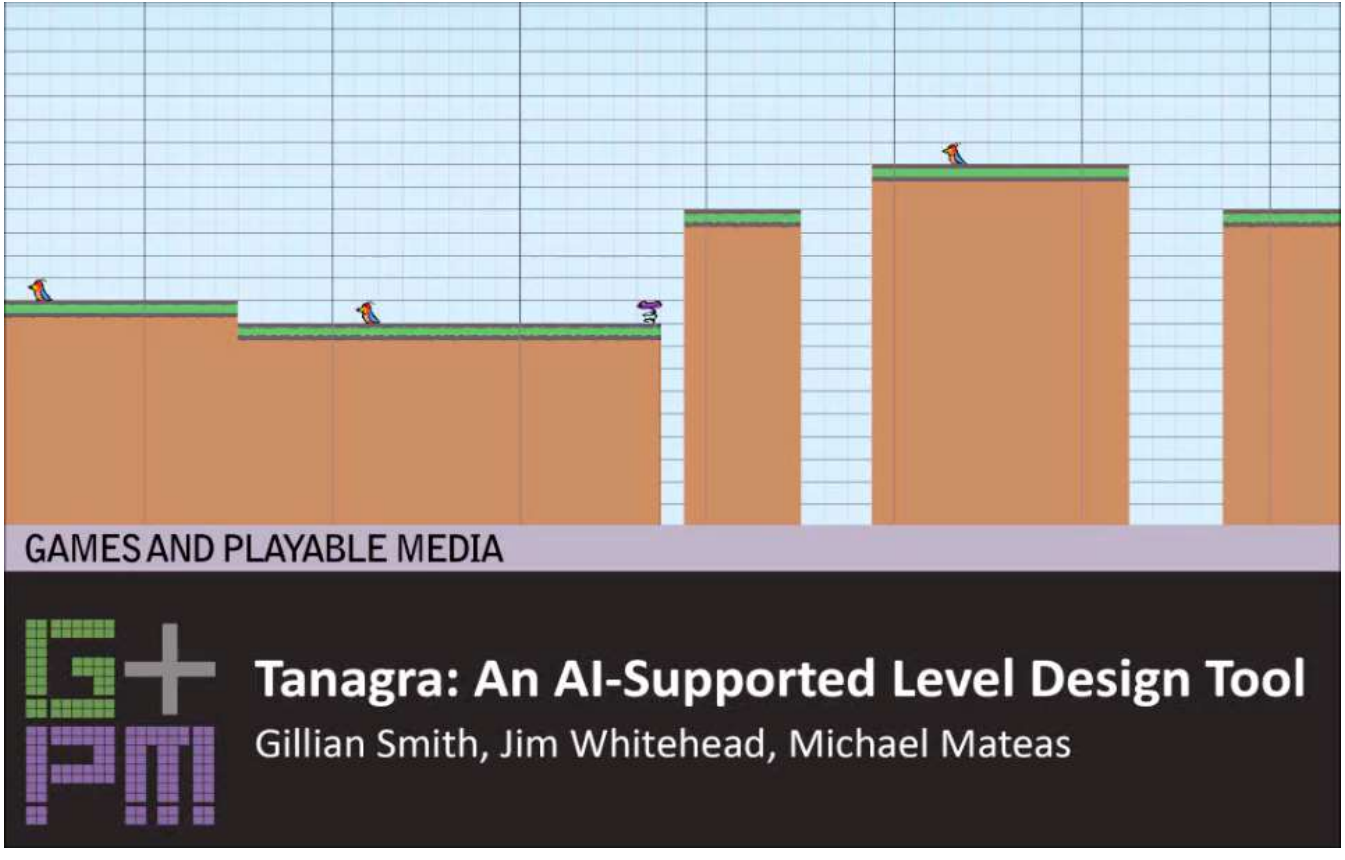
Liapis, Martínez, Togelius, and Yannakakis: **"Transforming Exploratory Creativity with DeLeNoX,"** in *Proceedings of the Fourth International Conference on Computational Creativity*, 2013.




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


Human
Creativity
Select your favorite map
Computational
Creativity

Tanagra: Constraint Solver for MI-PCG



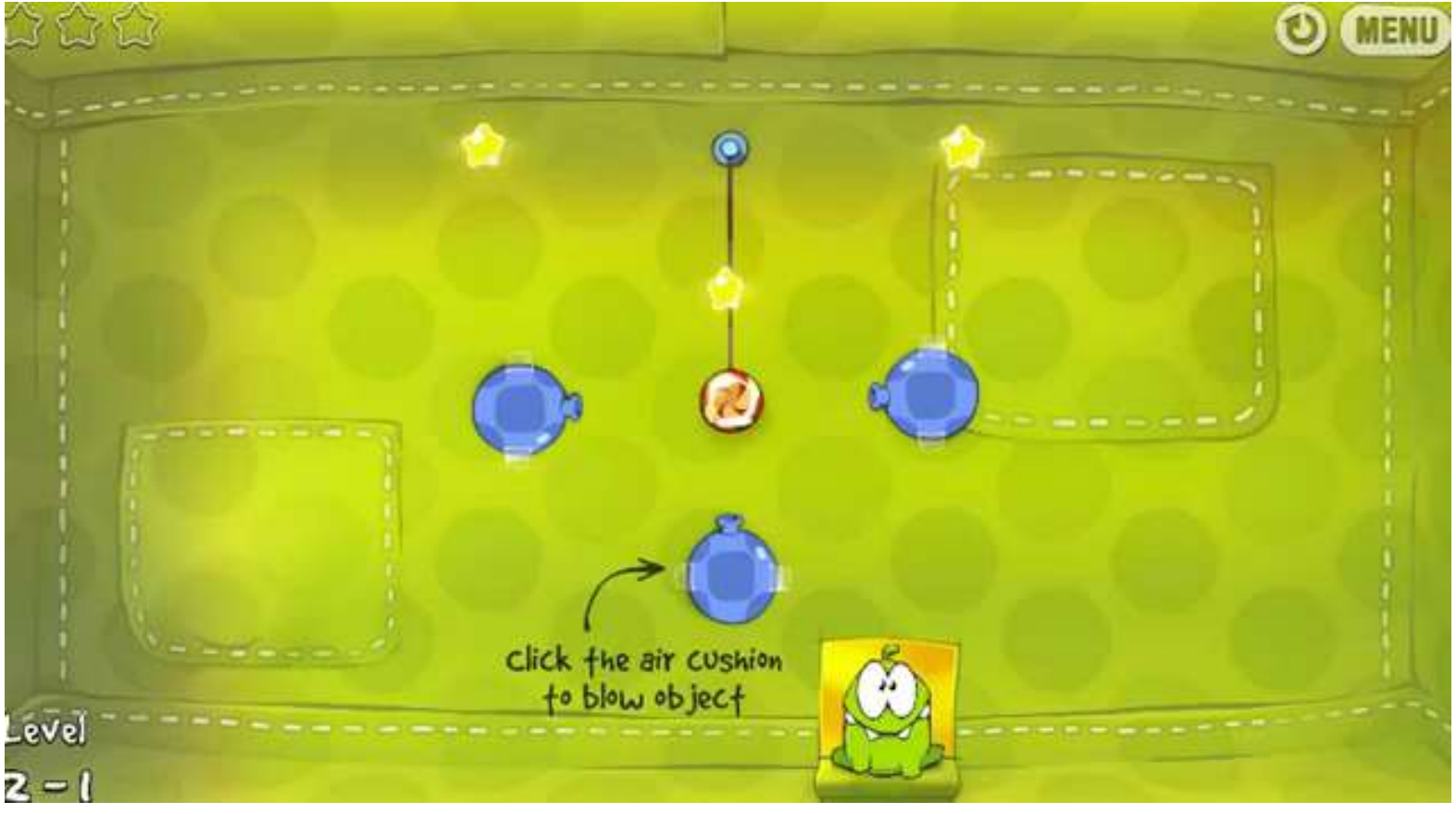
Ropossum

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

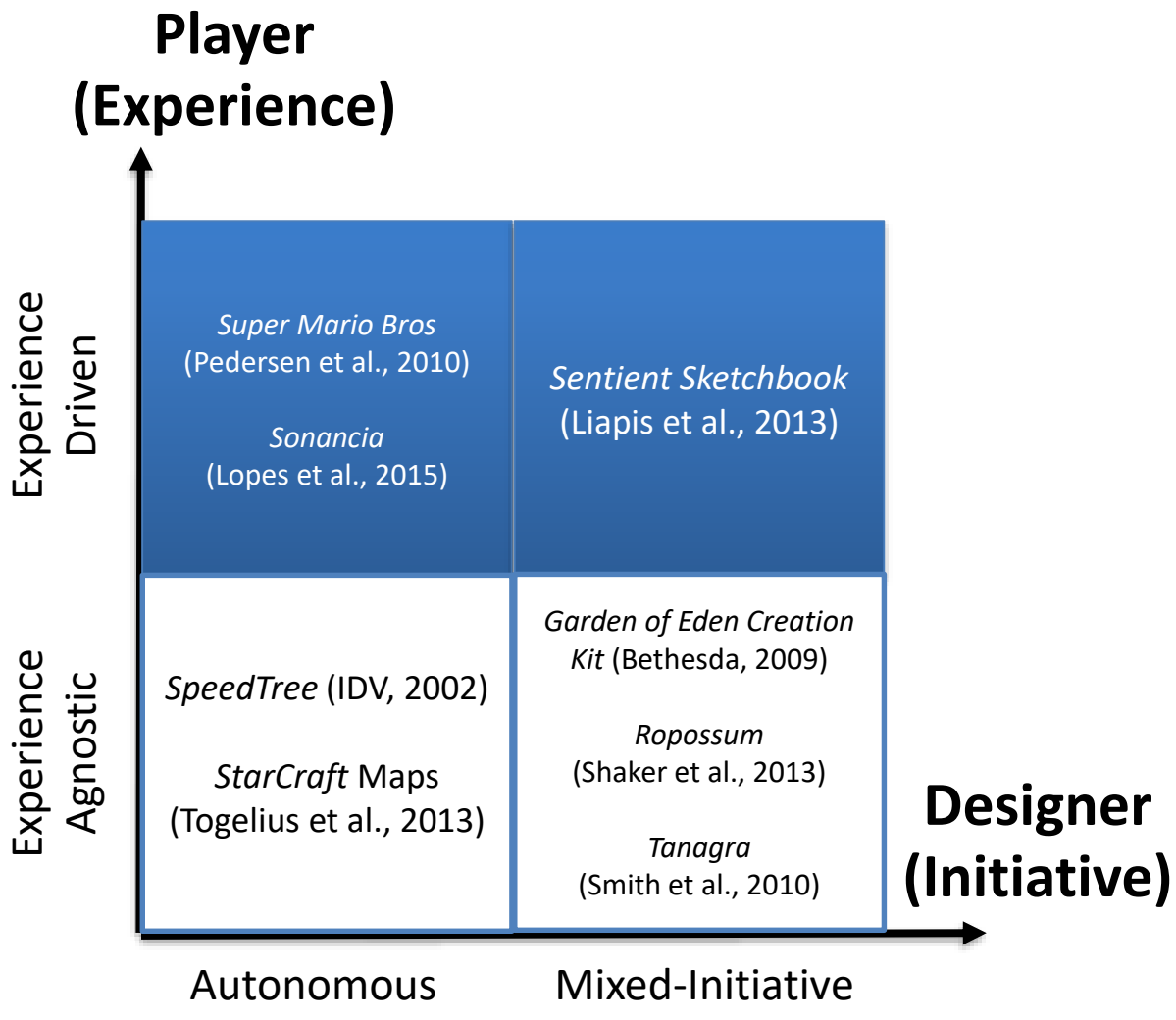
M. Shaker, M. H. Sarhan, O. A. Naameh, N. Shaker, and J. Togelius. Automatic generation and analysis of physics-based puzzle games. In Computational Intelligence in Games (CIG), 2013 IEEE Conference on.

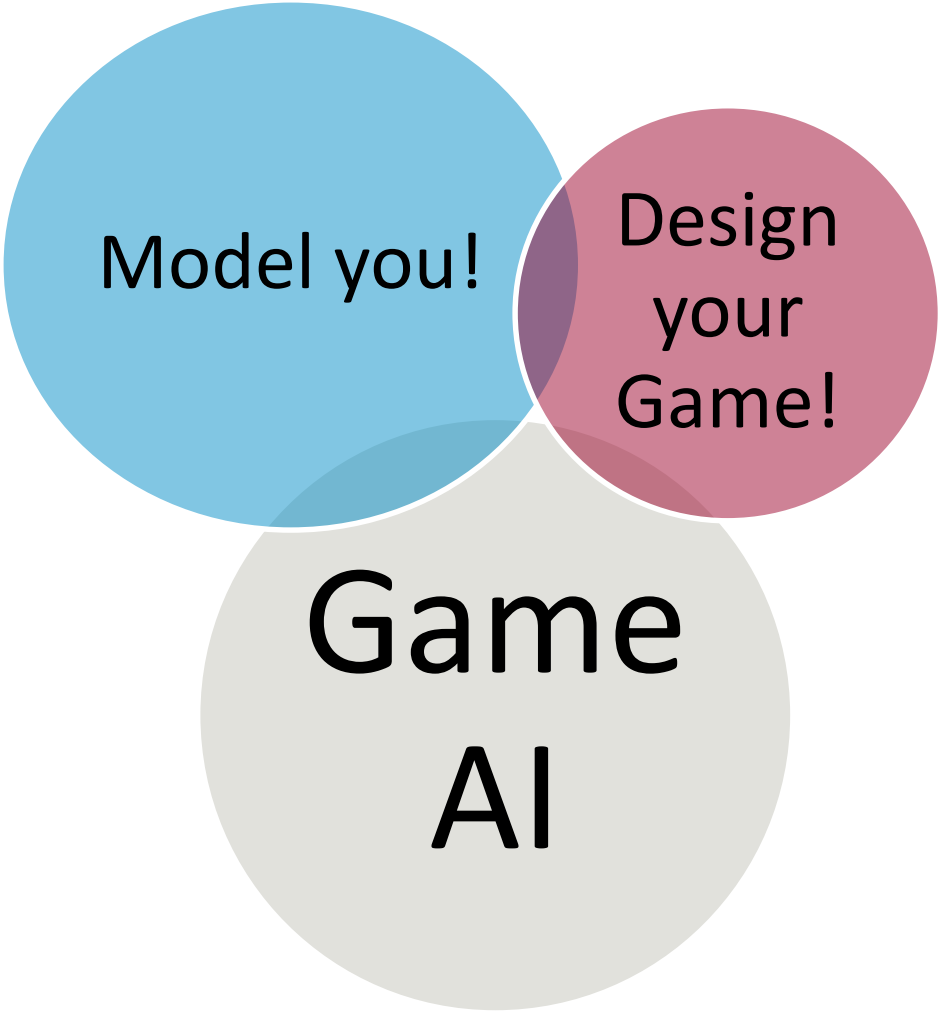
Ropossum



Ropossum

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

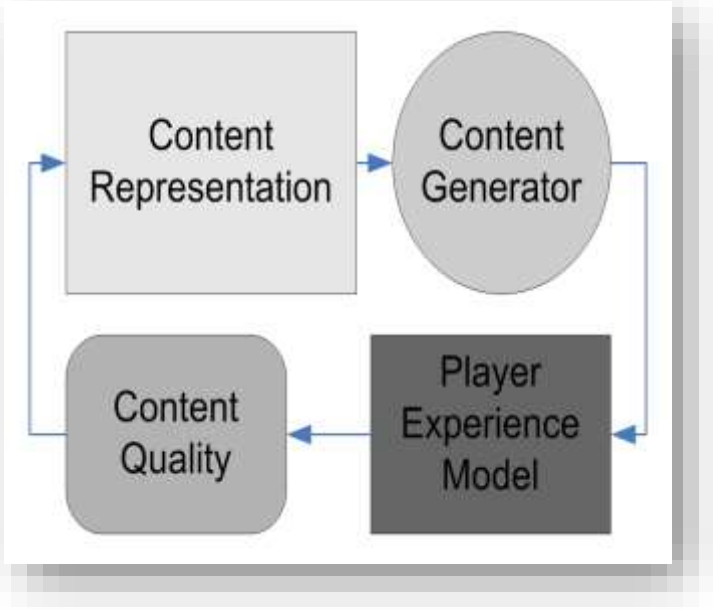

Player
(Experience)
Experience
Driven
Experience
Agnostic
Super Mario Bros
(Pedersen et al., 2010)
Sonancia
(Lopes et al., 2015)
Sentient Sketchbook
(Liapis et al., 2013)
SpeedTree (IDV, 2002)
StarCraft Maps
(Togelius et al., 2013)
Garden of Eden Creation Kit (Bethesda, 2009)
Ropossum
(Shaker et al.,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



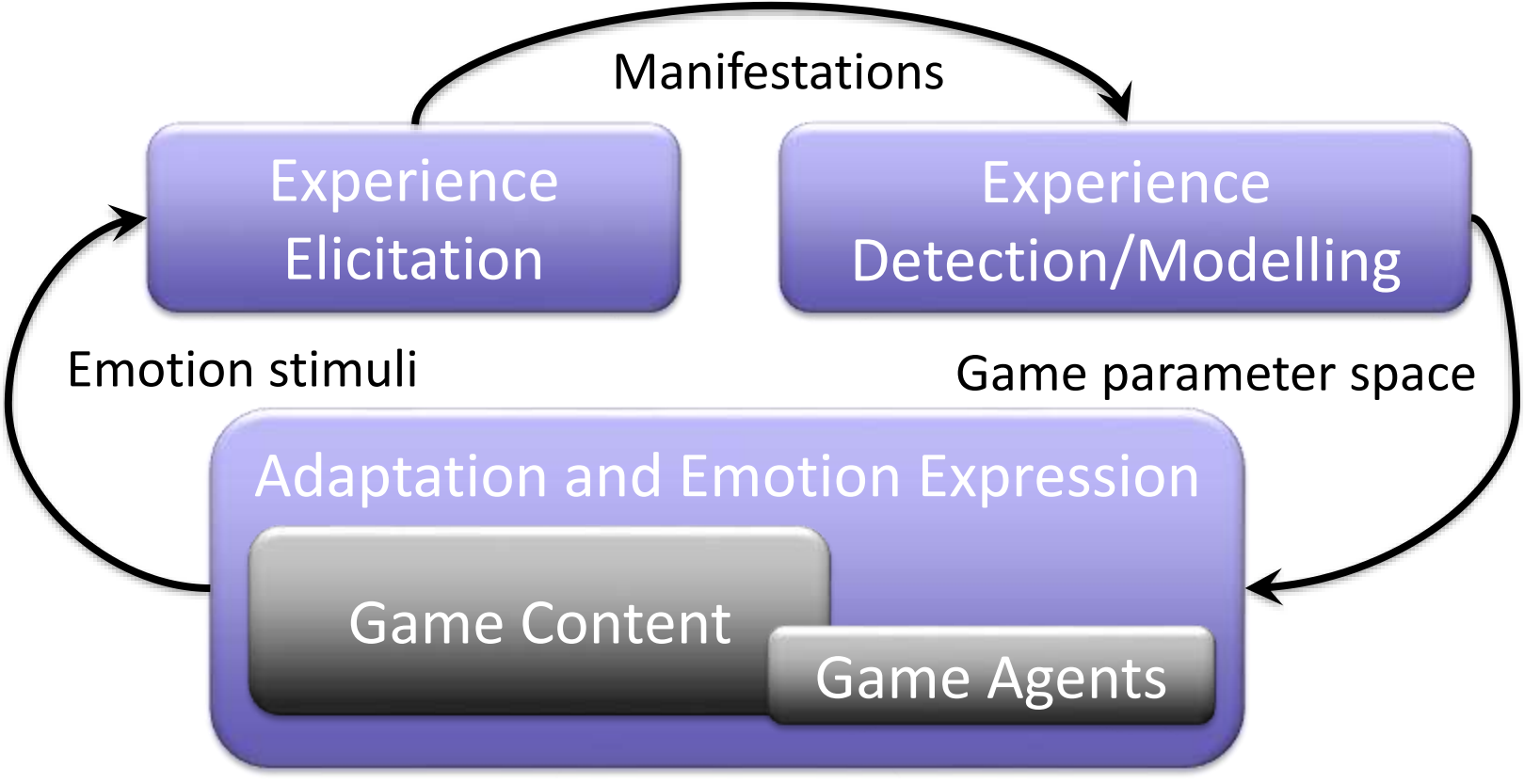
EDPCG: What is it?

"**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

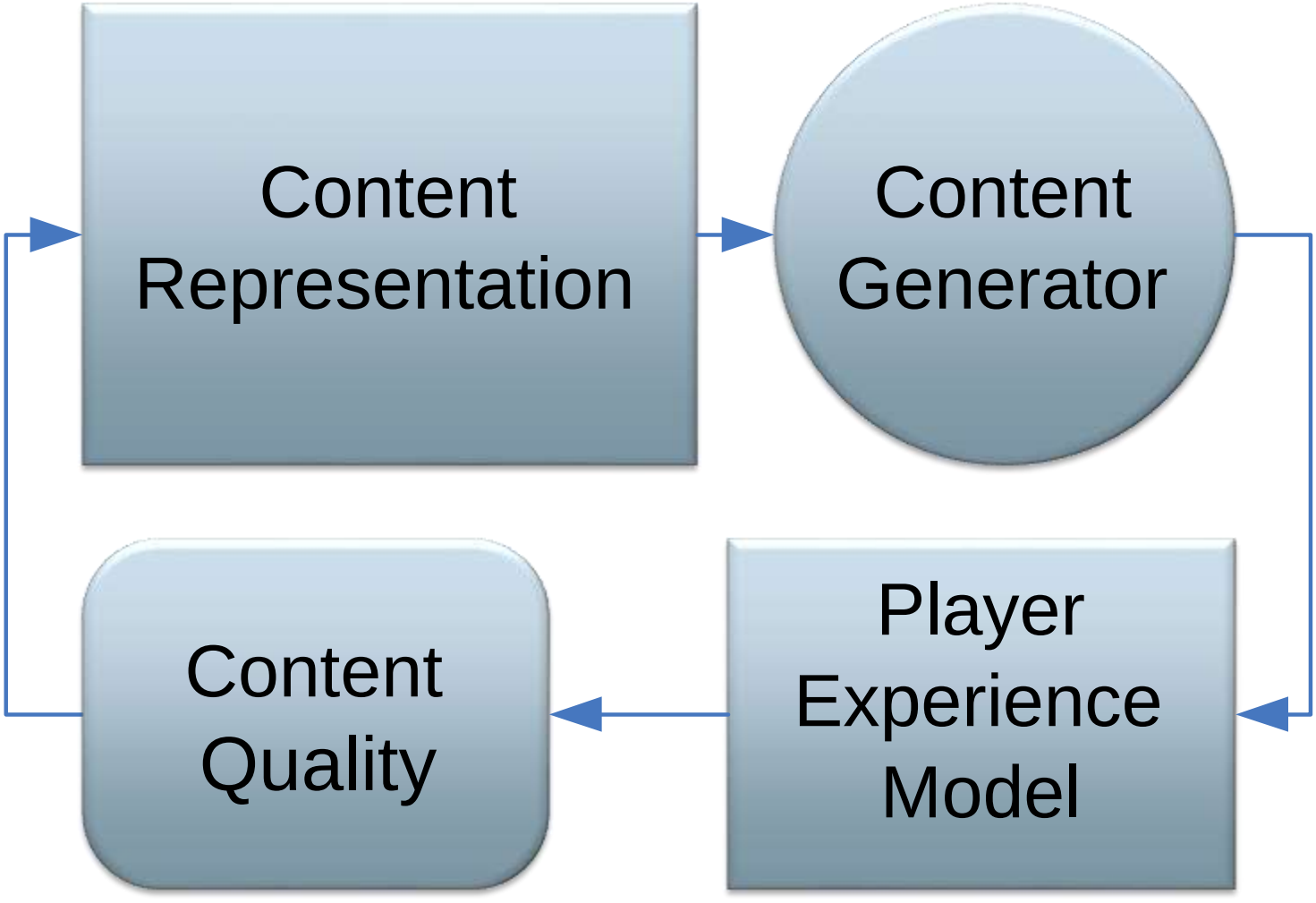
Yannakakis and Paiva, **Emotion in Games**, in *Handbook of Affective Computing*, 2013

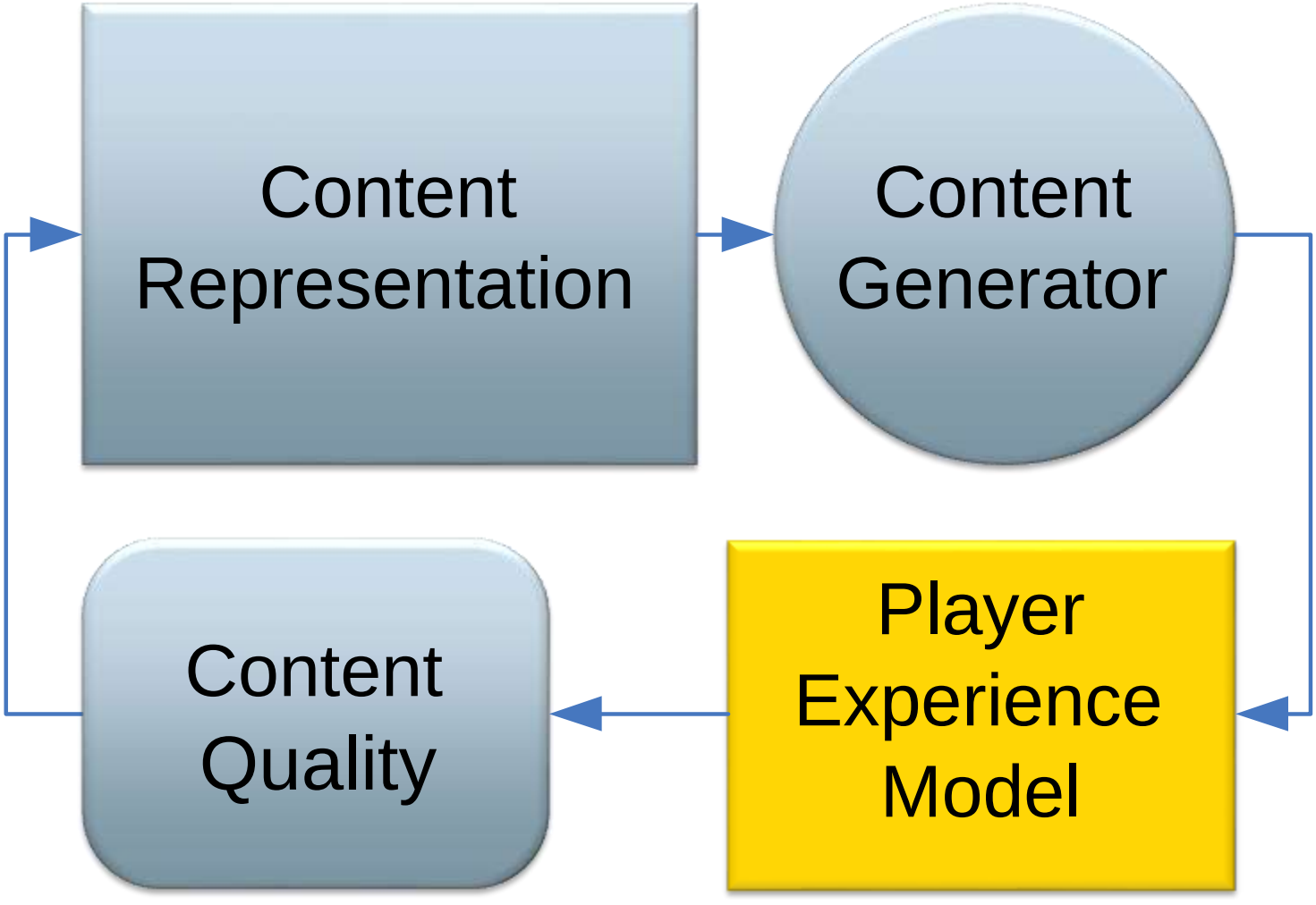


Experience-Driven Procedural Content Generation

"… 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

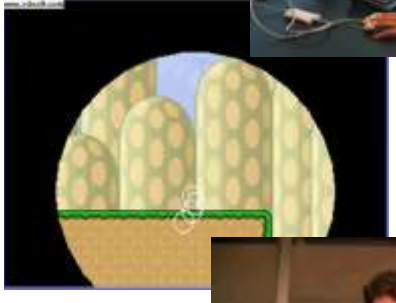
X is frustrating

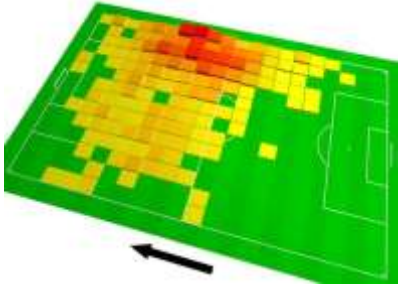
Disagree — Agree

1 2 3 4 5

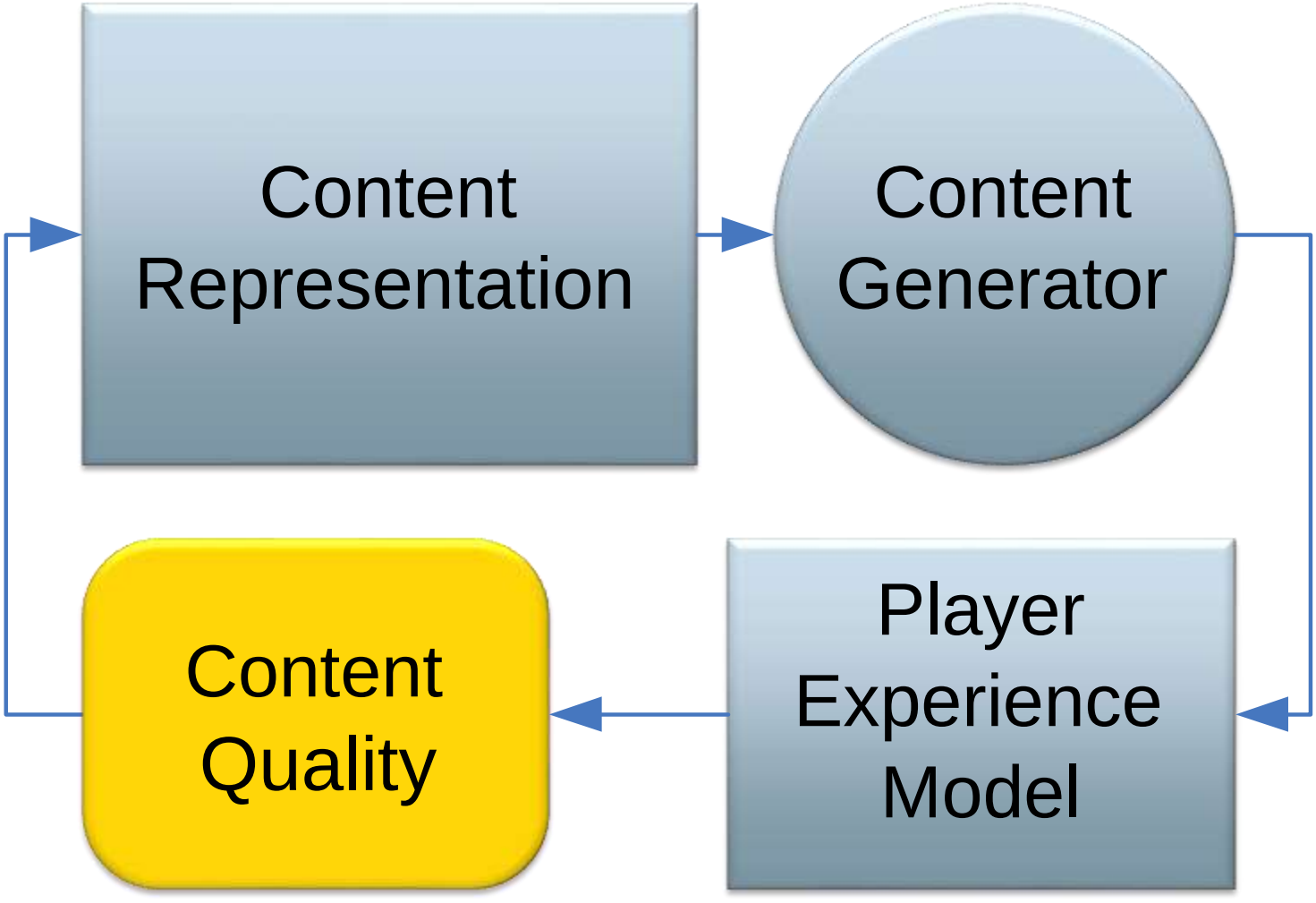
Is X or Y **more** frustrating?

- ☐ X ☐ Y
- ☐ Both are **equally** frustrating
- ☐ **Neither** is frustrating

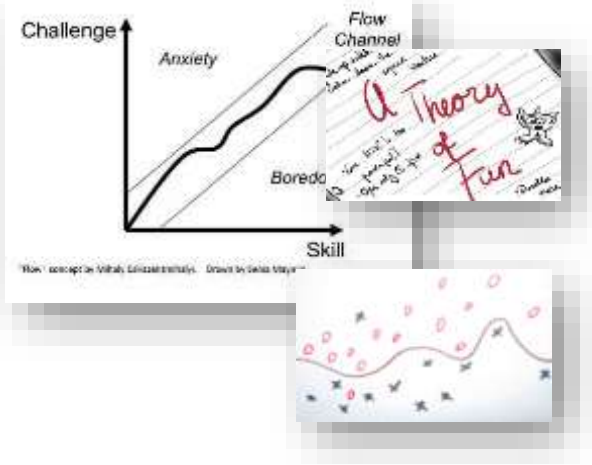




Player Experience Model		
Subjective	**Objective**	**GamePlay-Based**
Free-Response	**Model-based** (e.g. arousal-valence dimensions of emotion)	**Model-based** (e.g. OCC, BDI)
Forced Rating vs. Preference	**Model-free** (e.g. facial expression annotation)	**Model-free** (e.g. player modeling)

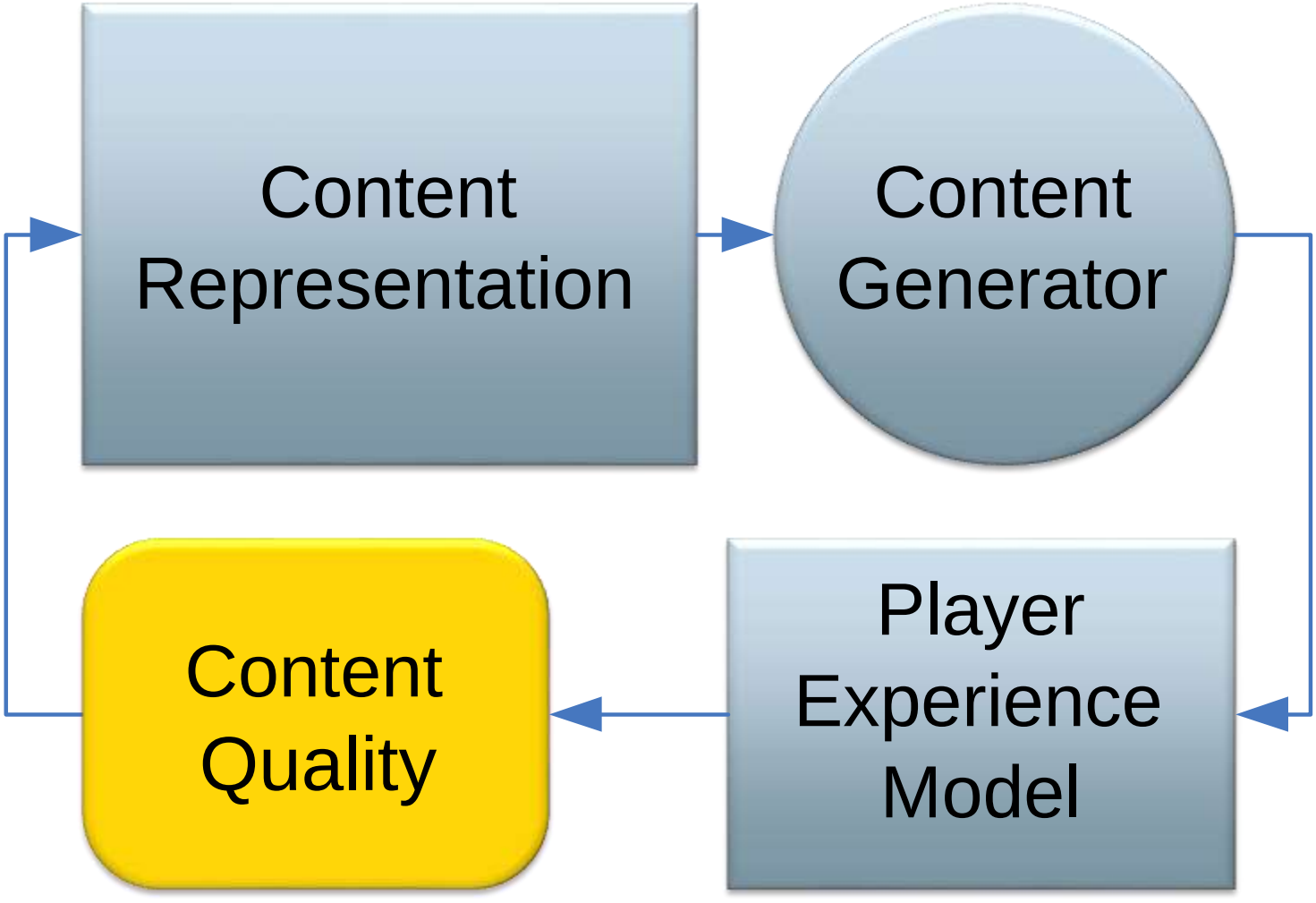

Content Representation
Content Generator
Player Experience Model
Content Quality

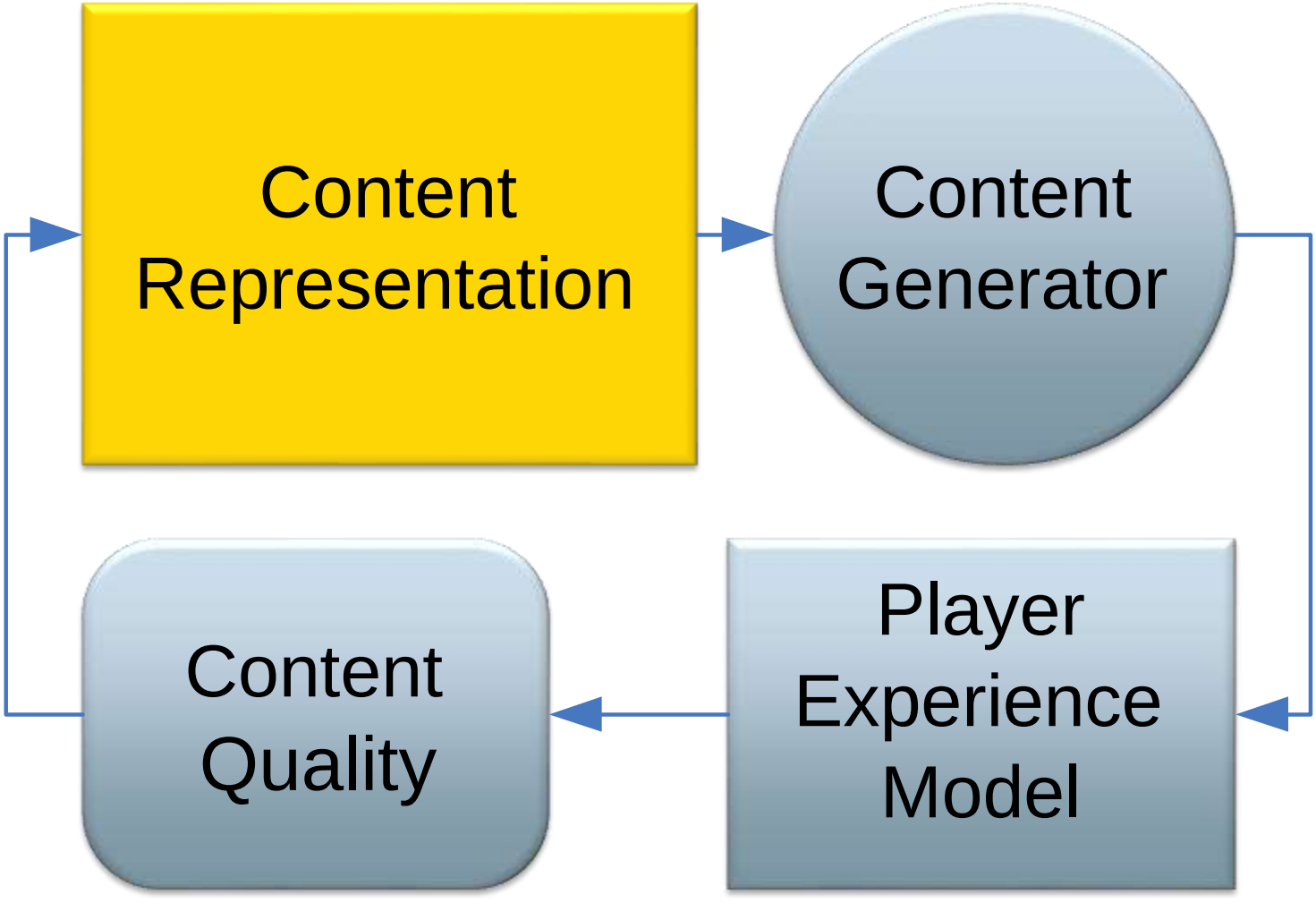
Content Quality

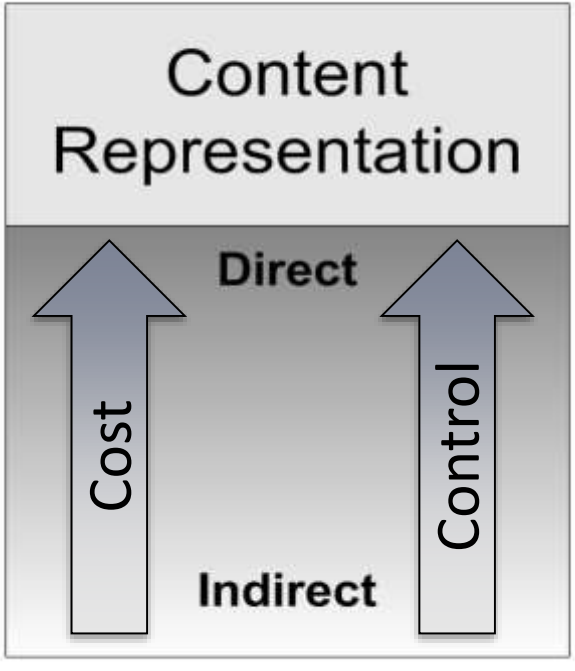
Direct	Simulation-based	Interactive
		

Content Quality

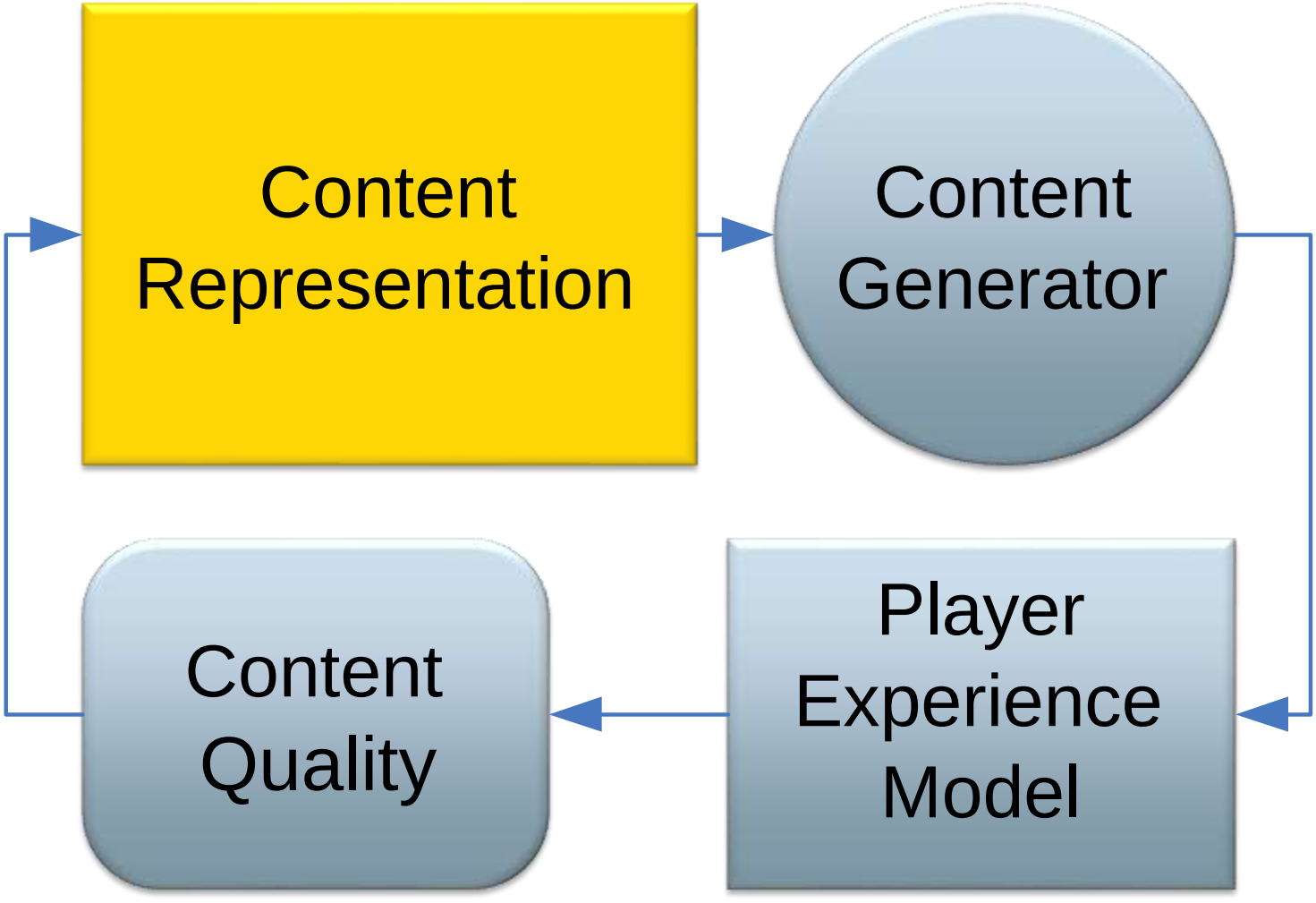
Direct	Simulation-based	Interactive
Theory-driven vs. Data-driven	Static vs. Dynamic	Implicit vs. Explicit

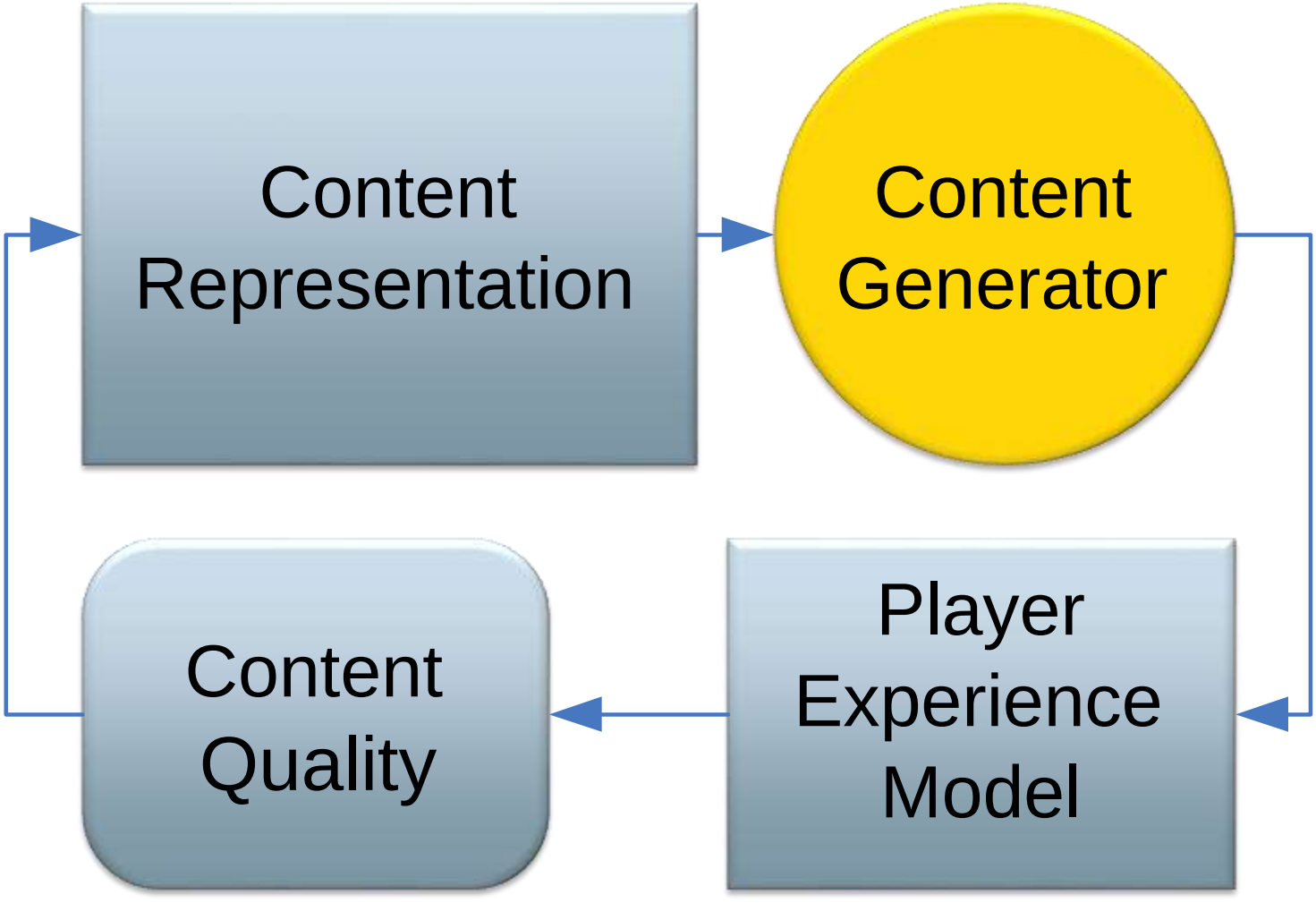

Content Representation
Content Generator
Player Experience Model
Content Quality

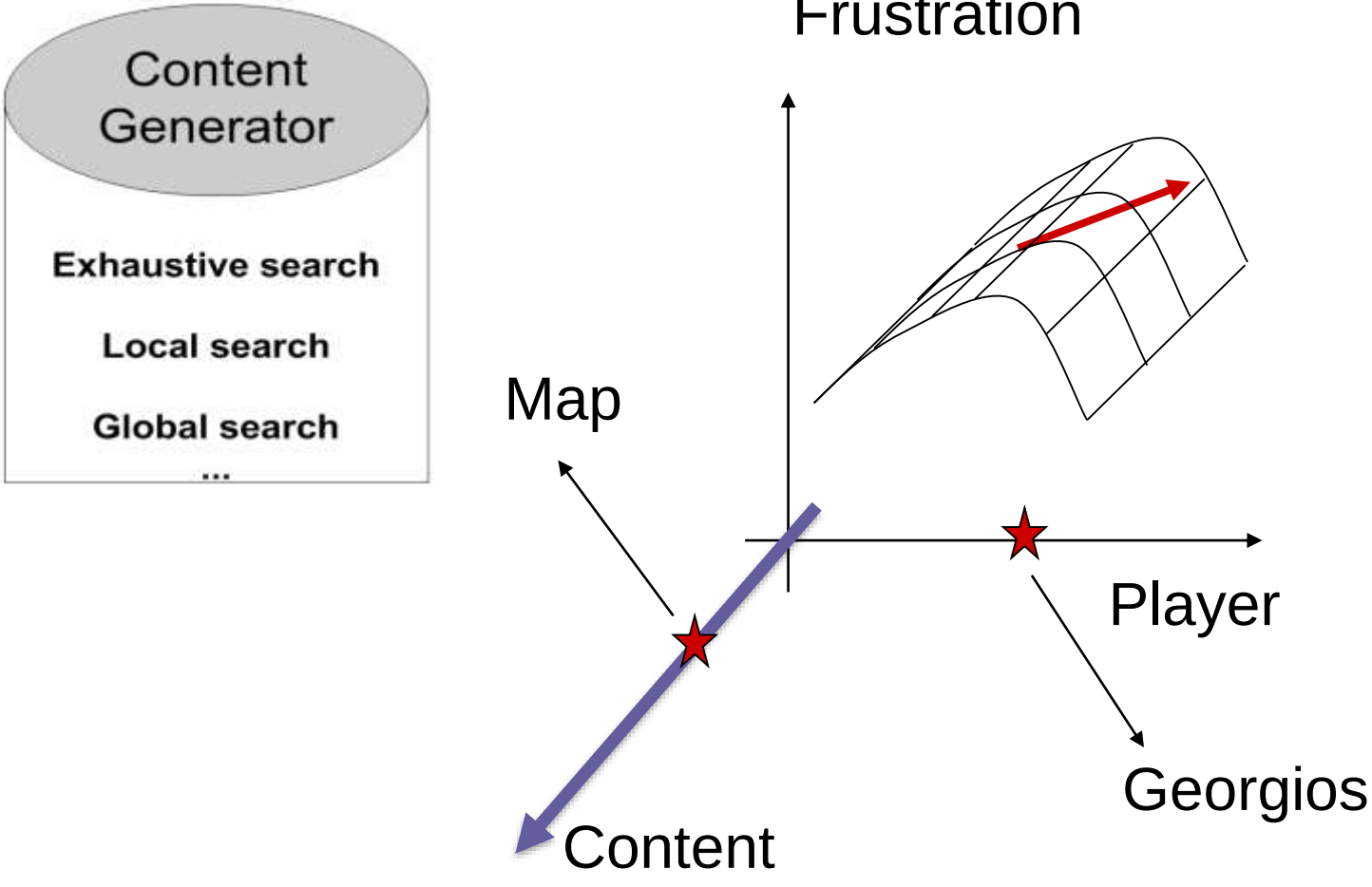

Content Representation
Content Generator
Player Experience Model
Content Quality



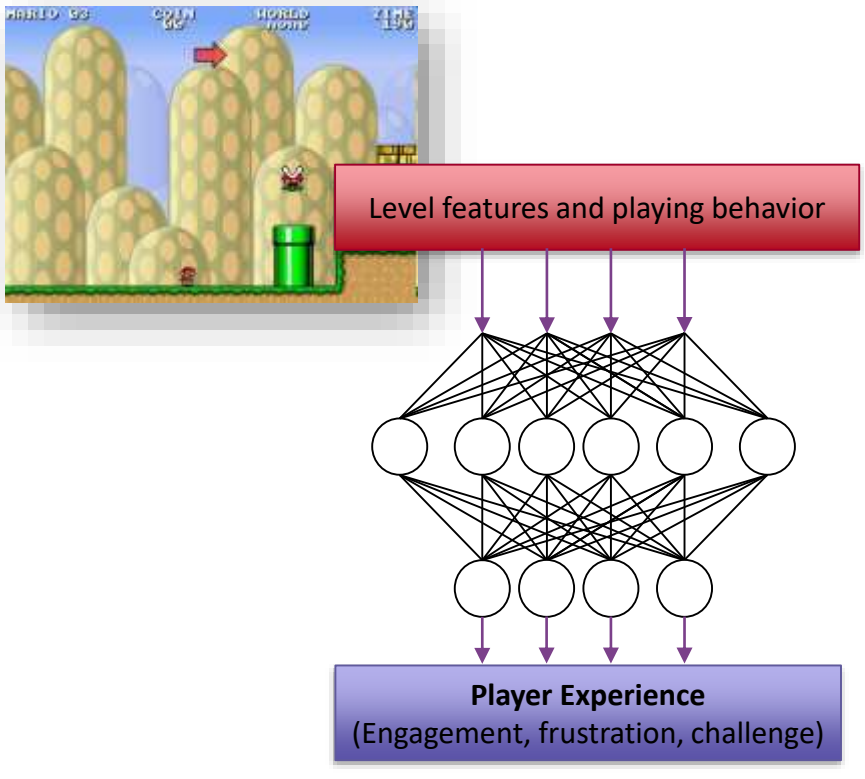
→ Grid
→ Position and orientation of walls
→ Patterns of walls and floor
→ Number of rooms and doors
→ Random seed


Content Representation
Content Generator
Player Experience Model
Content Quality


Content Representation
Content Generator
Player Experience Model
Content Quality

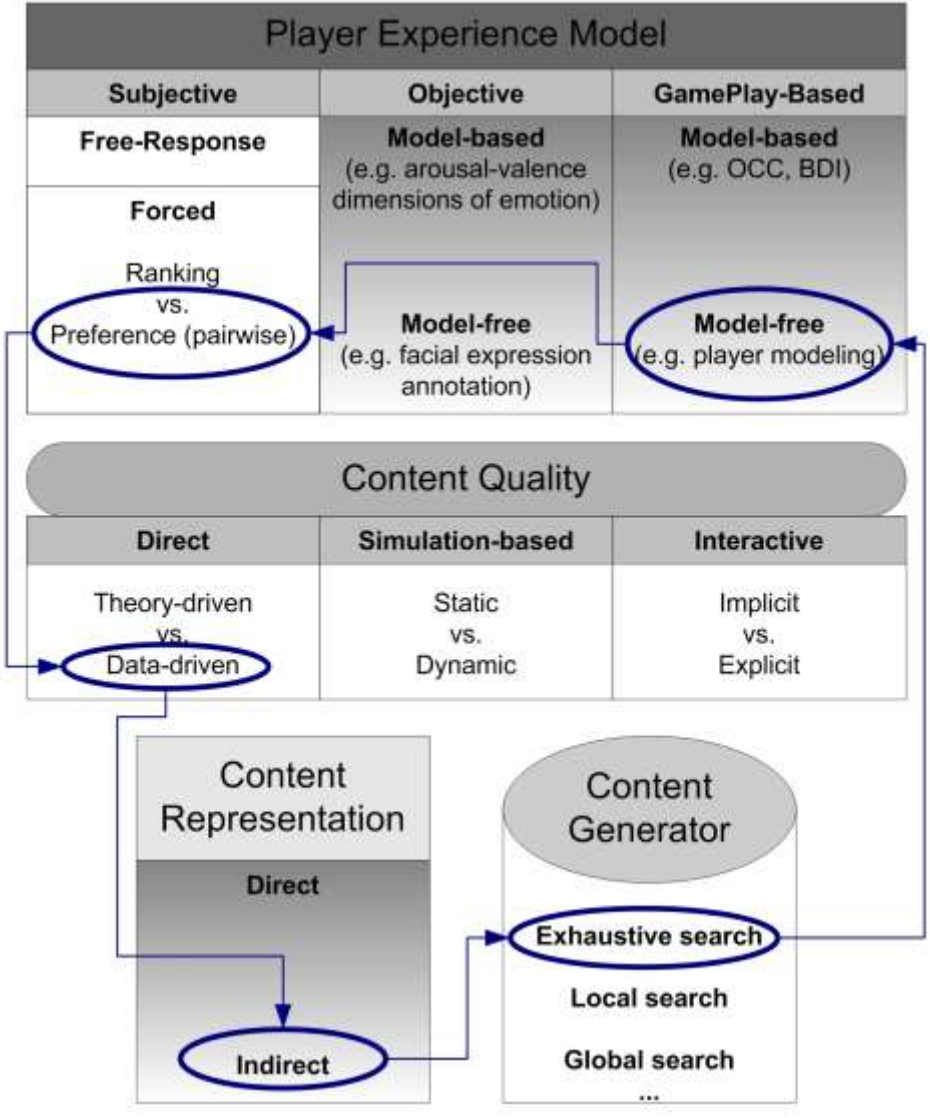

Content Generator
Exhaustive search
Local search
Global search
...
Frustration
Map
Player
Georgios
Content

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**!

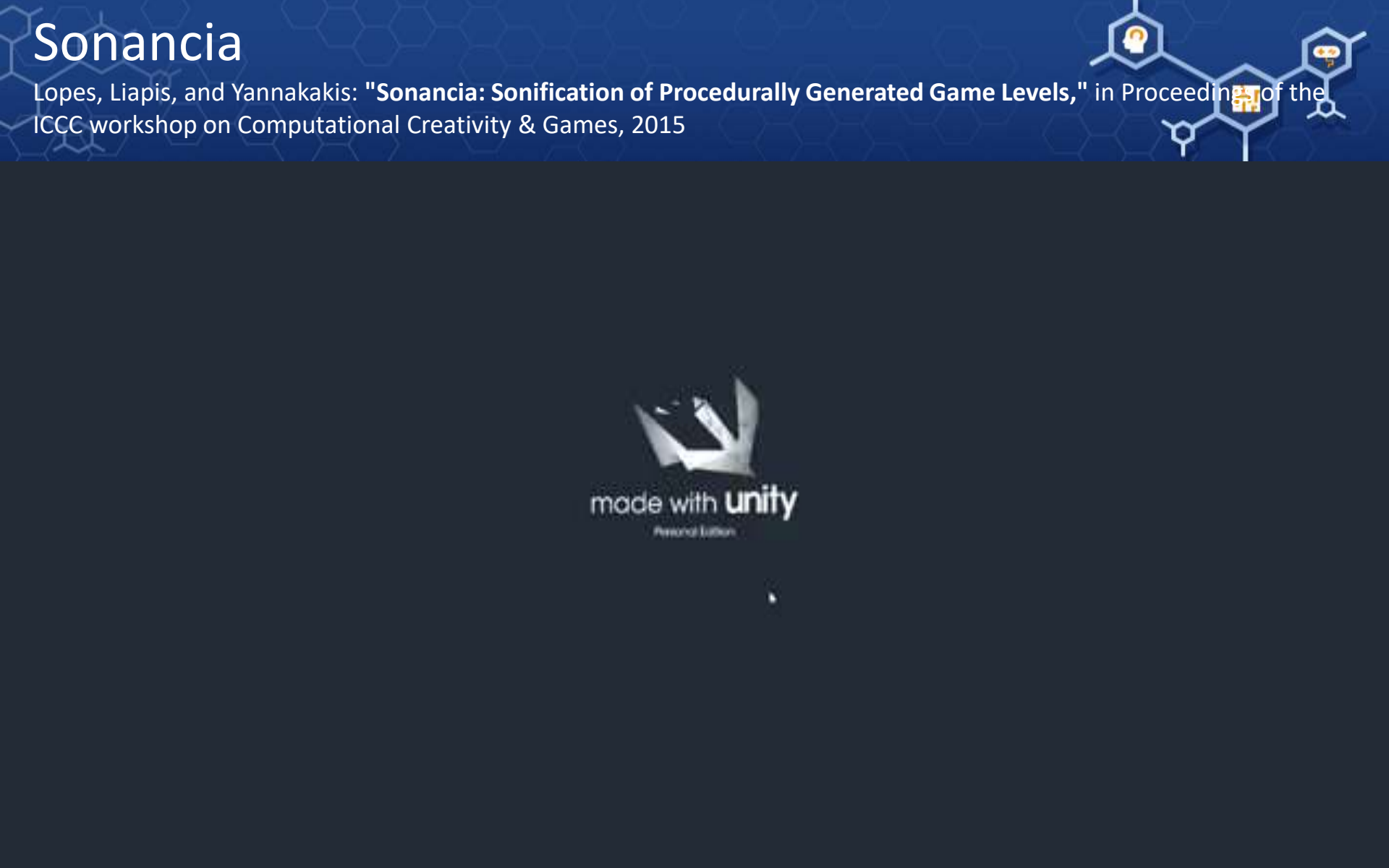
Shaker, N., Yannakakis, G. N., & Togelius, J. (2010, October). Towards Automatic Personalized Content Generation for Platform Games. In *AIIDE*.


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

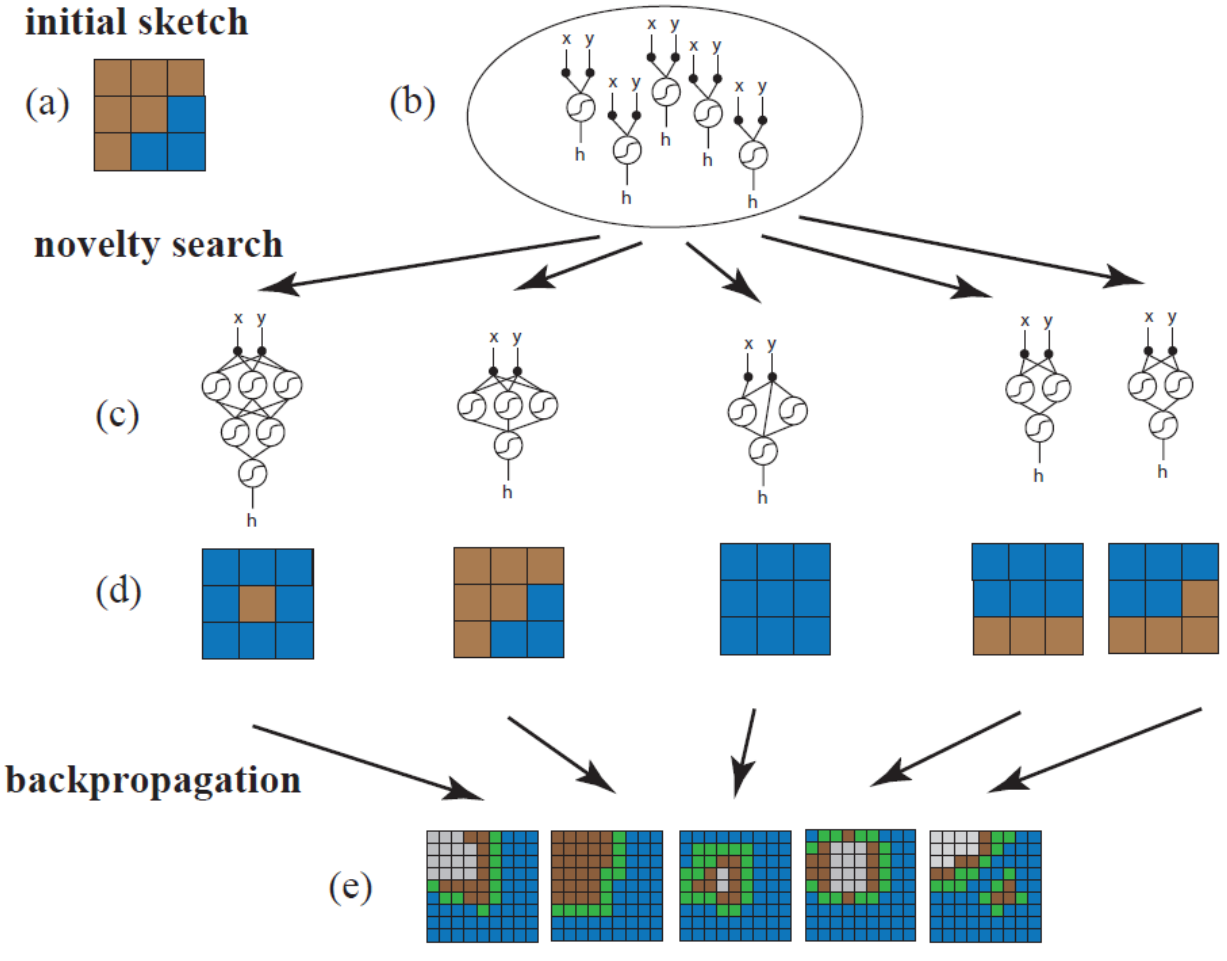


EDPCG for Serious Games: *Village Voices*

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

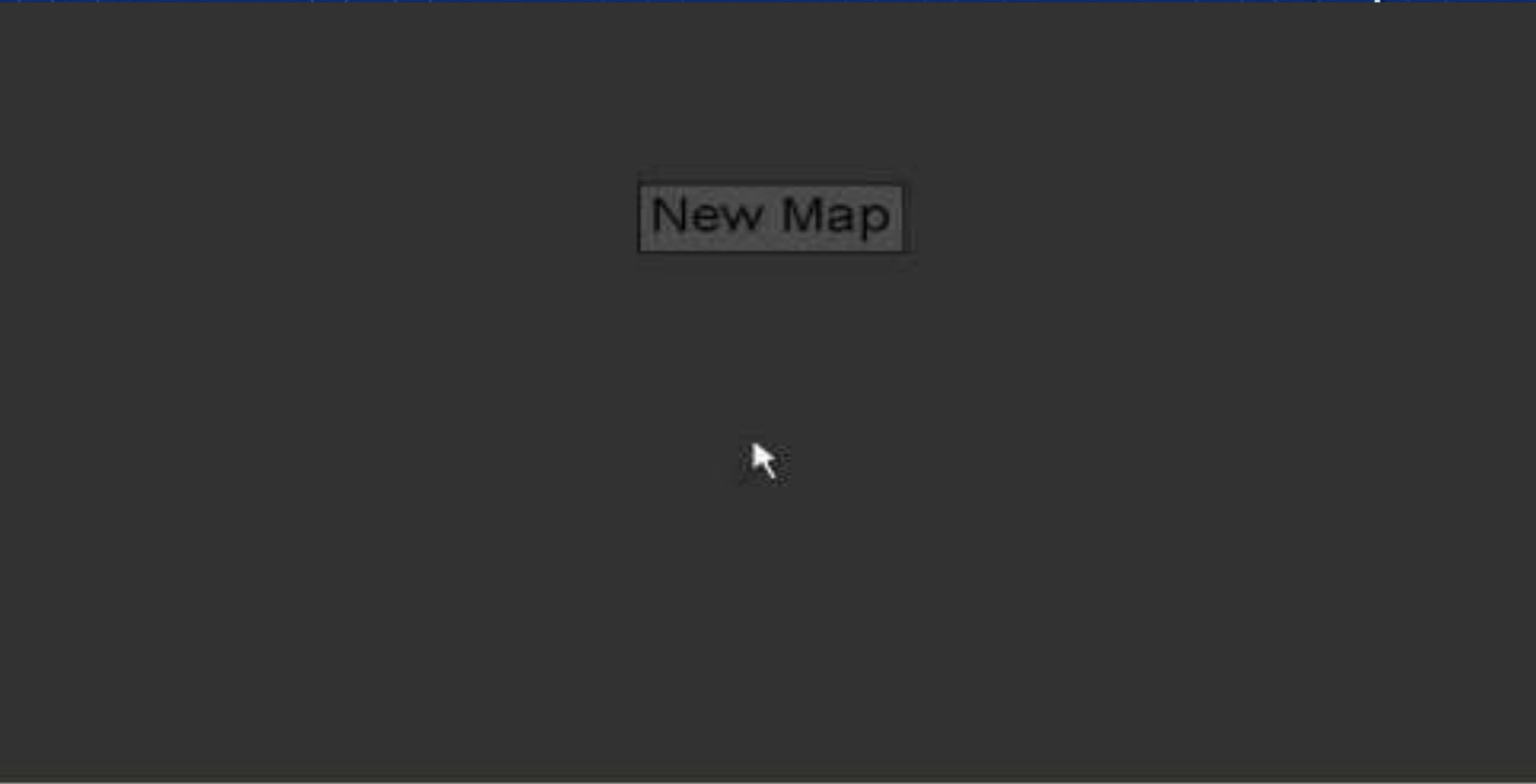
Sentient World:
Hybridizing Evolution and Gradient Search

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



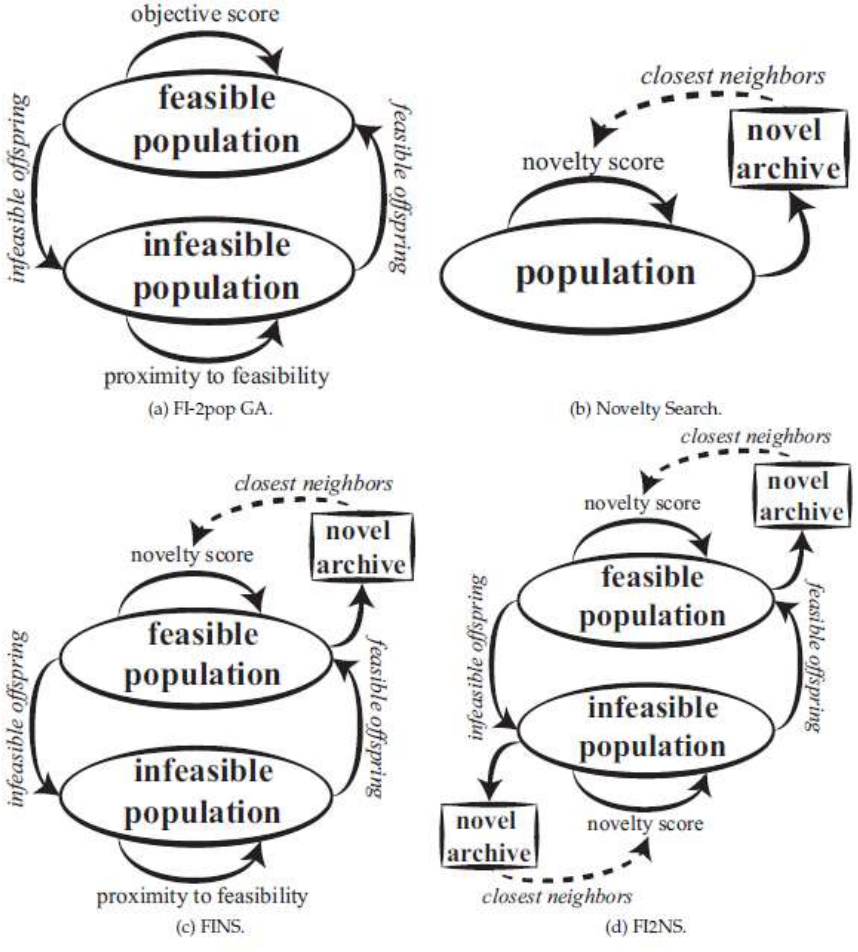
Sentient World:
Hybridizing Evolution and Gradient Search

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



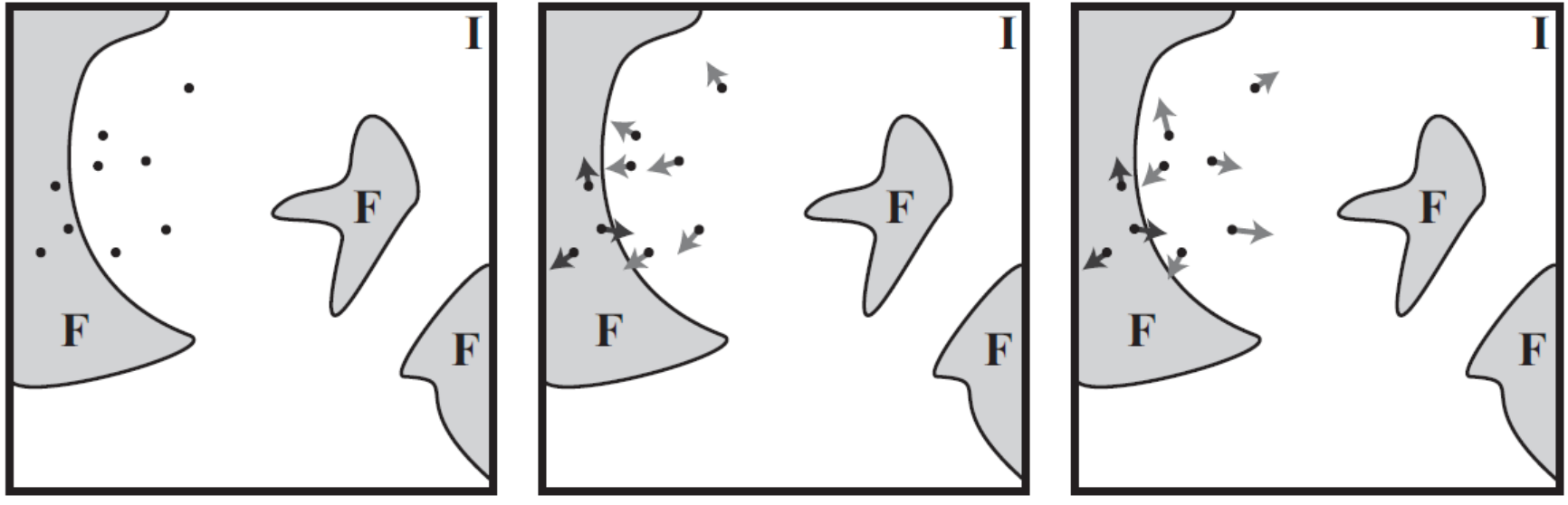
Constrained Novelty Search for PCG

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



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

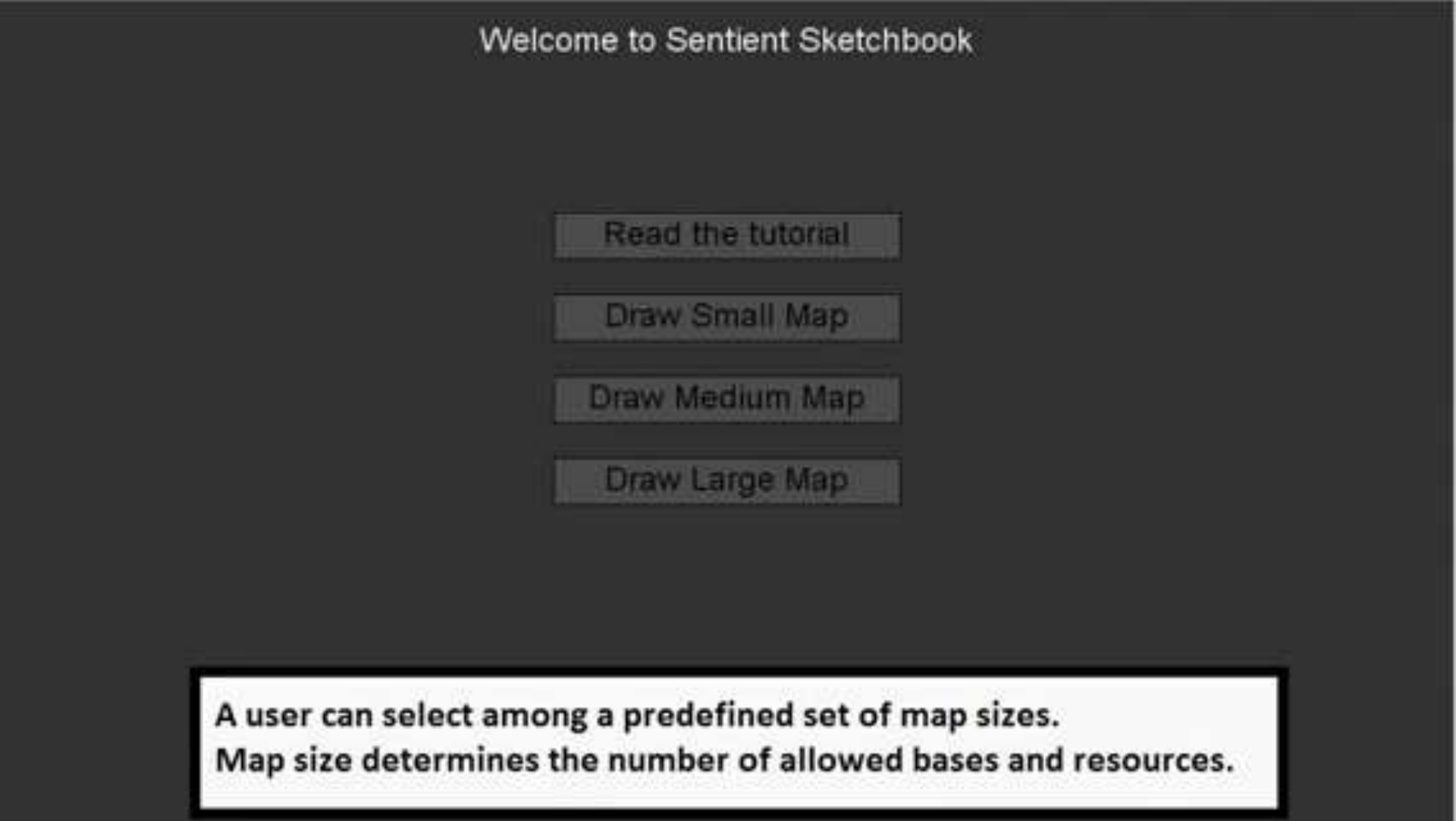


(a) Feasible and Infeasible Spaces. (b) FINS search process. (c) FI2NS search process.

Sentient Sketchbook

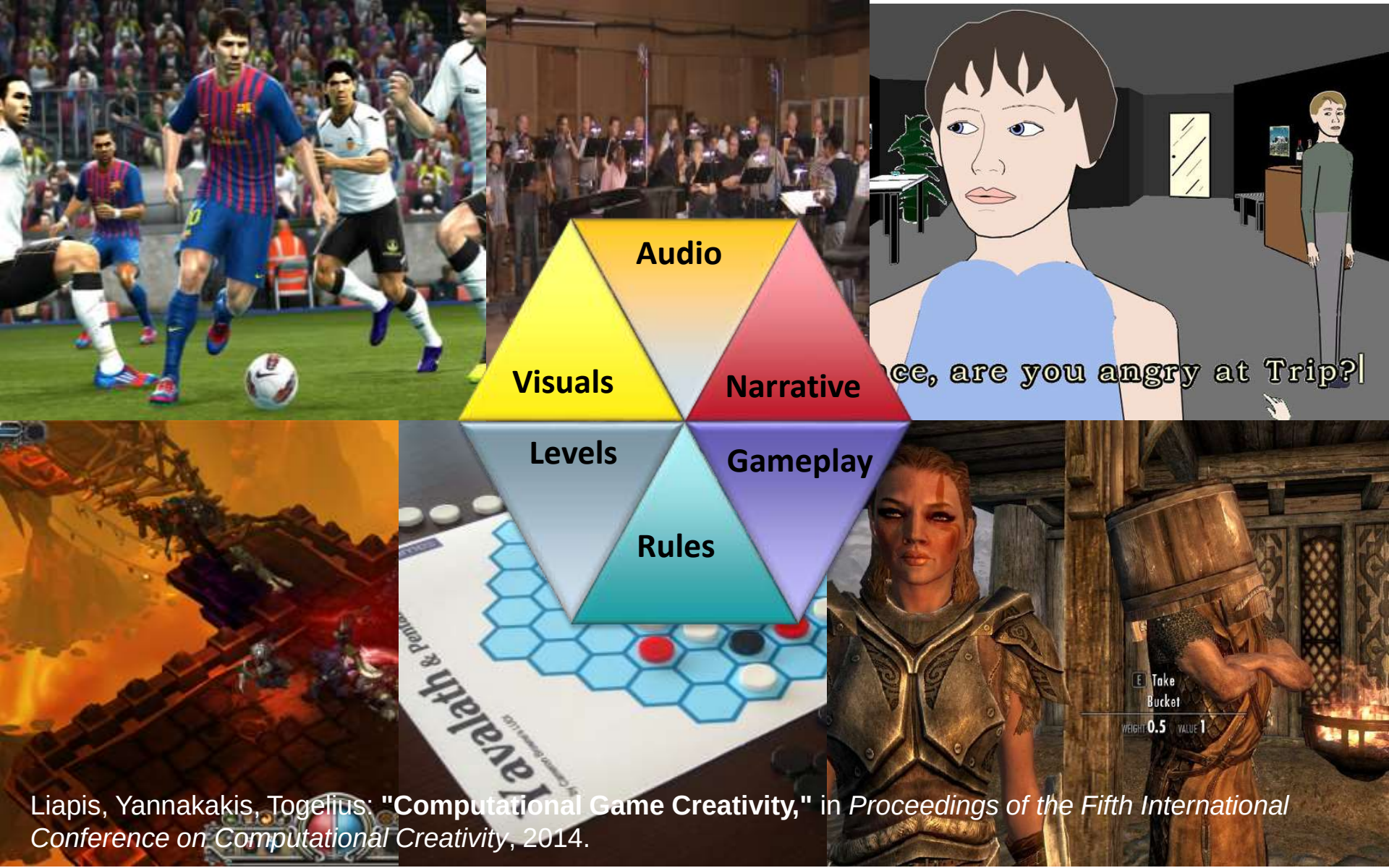
Georgios N. Yannakakis, Antonios Liapis and Constantine Alexopoulos:
"Mixed-Initiative Co-Creativity," in Proc. of the ACM Conference on Foundations of Digital Games, 2014.

- 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

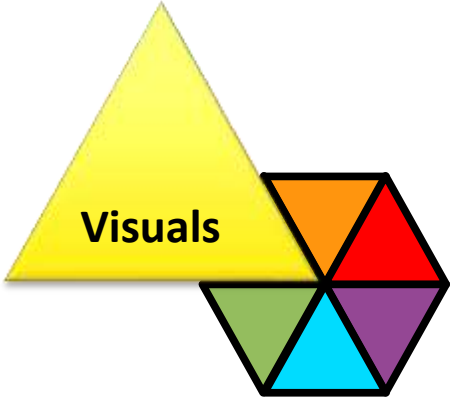


Georgios N. Yannakakis, Antonios Liapis and Constantine Alexopoulos: **"Mixed-Initiative Co-Creativity,"** in Proc. of the ACM Conference on Foundations of Digital Games, 2014.

What Could be Generated?



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


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.

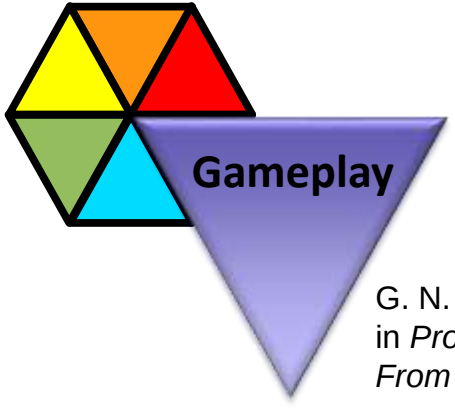

Narrative
ce, are you angry at Trip?

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.


E Take
Bucket
WEIGHT 0.5 VALUE 1
Gameplay


MARRY
ME?
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.



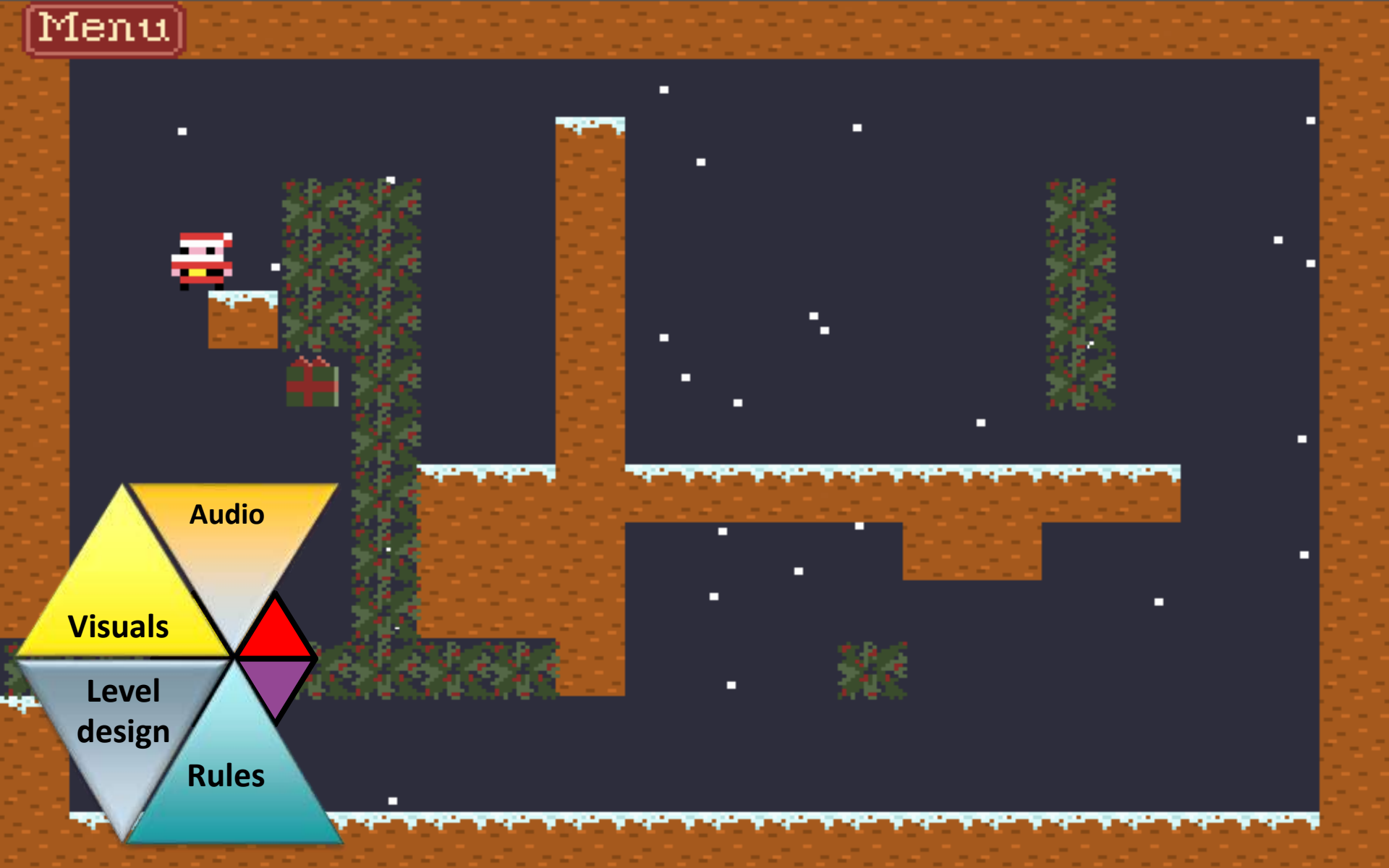
G. N. Yannakakis, and J. Hallam, "**Evolving Opponents for Interesting Interactive Computer Games**," in *Proceedings of the 8th International Conference on the Simulation of Adaptive Behavior (SAB'04); From Animals to Animats 8*, pp. 499-508, Los Angeles, CA, USA, July 13-17, 2004. The MIT Press.

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


Level
design

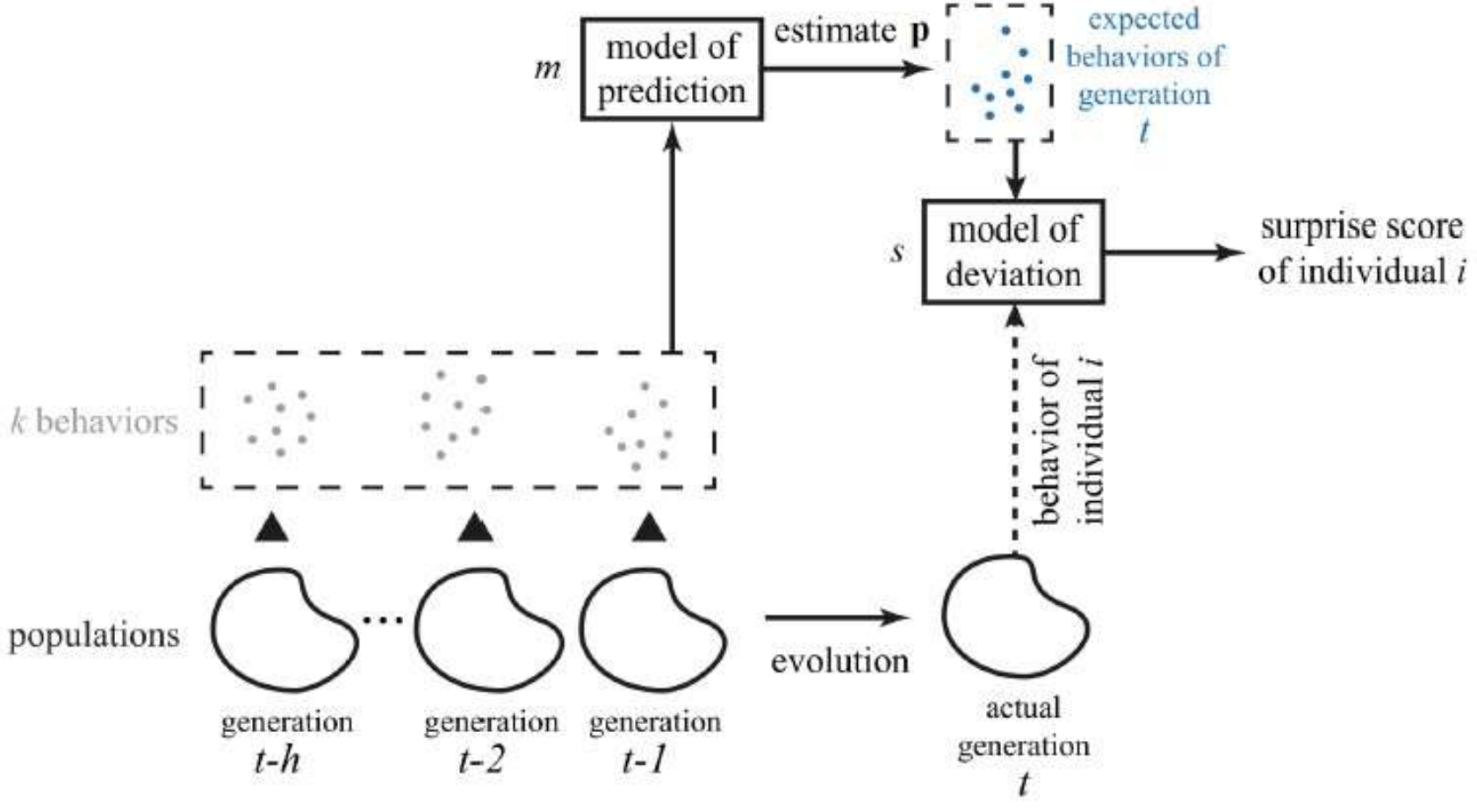
Complete Game Generation – Orchestration

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


Menu
Audio
Visuals
Level design
Rules

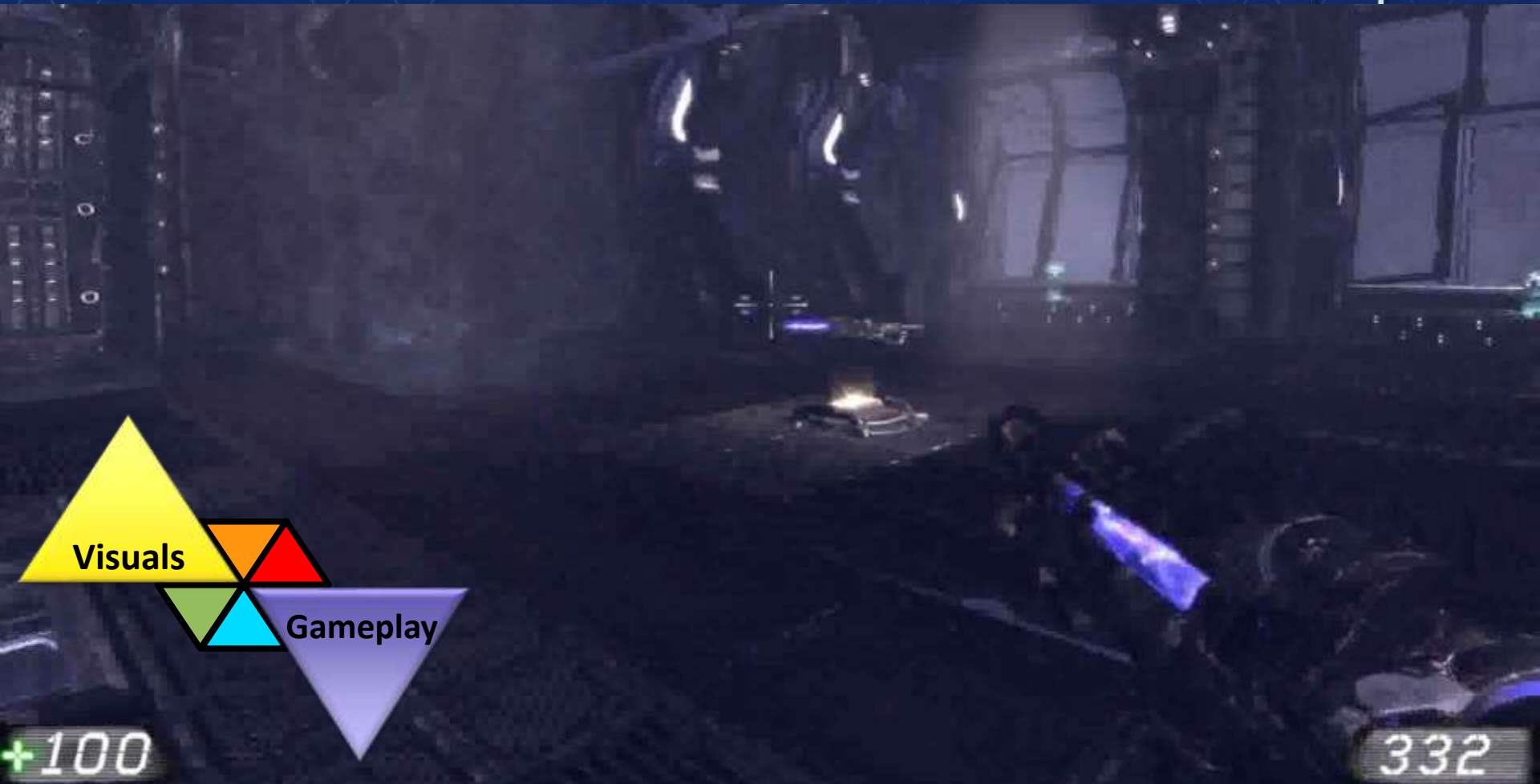
From Novelty Search to **Surprise Search**

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.

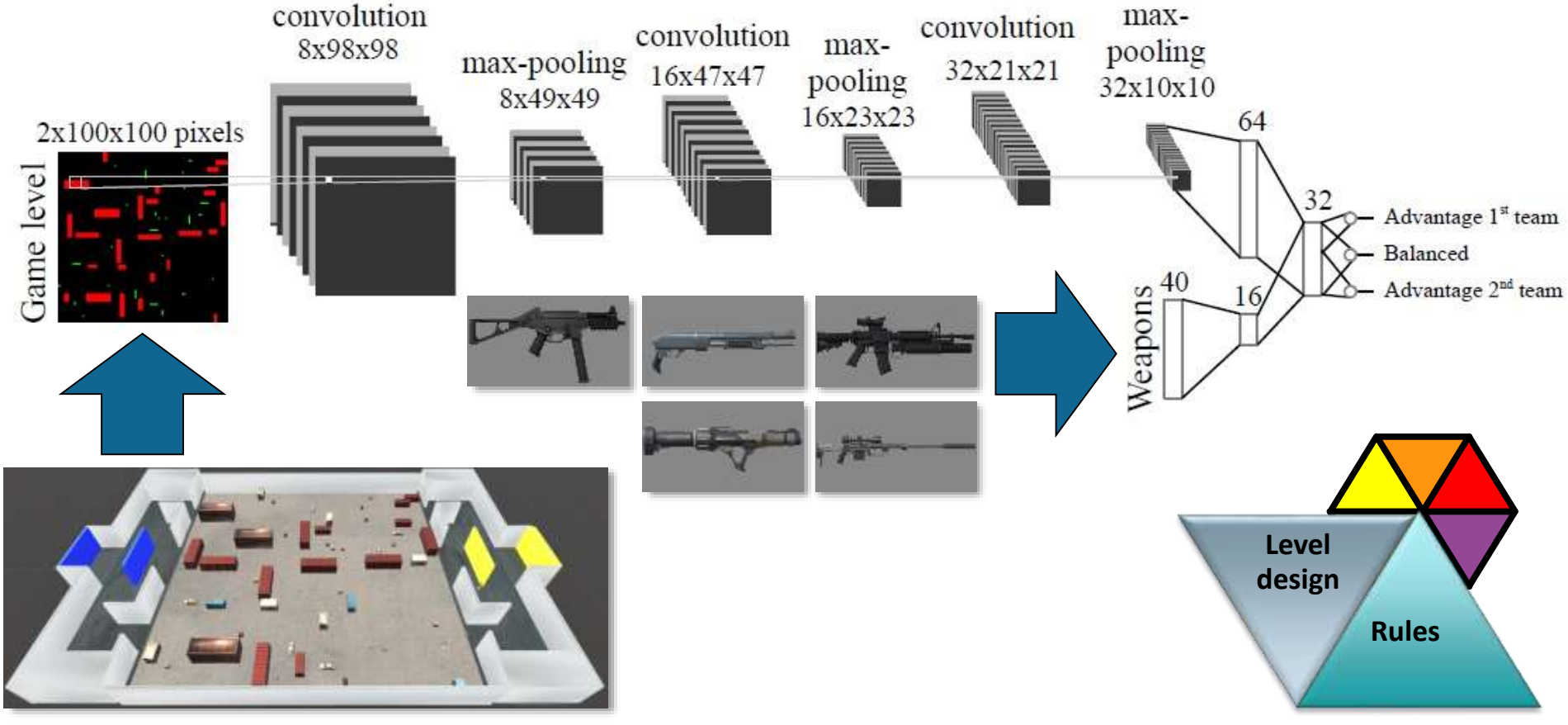


AudioInSpace: From Music to Weapons!

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.

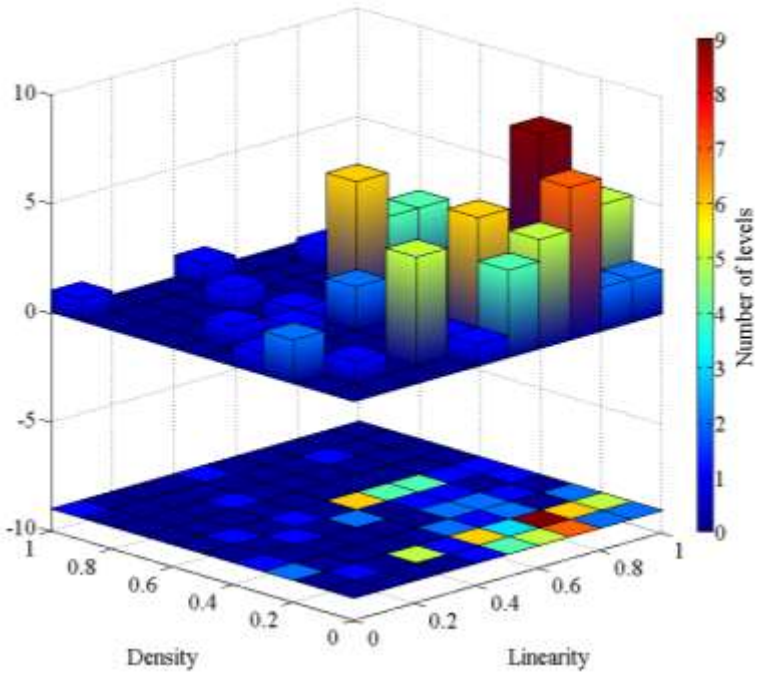


Evaluating Content Generators

How Can we Evaluate a Content Generator?

Generally speaking there are three ways:

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



Thank you!

gameaibook.org