Procedural Level Generation

IGGI AI based Game Design (Part II)

Vanessa Volz - v.volz@qmul.ac.uk
Office: CS.335

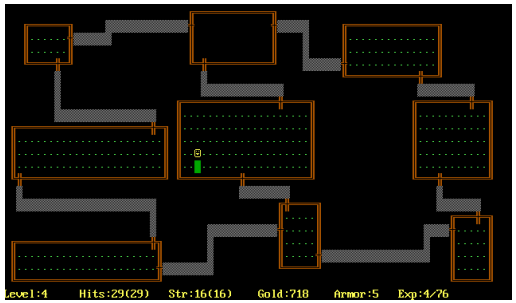
http://gameai.eecs.qmul.ac.uk

Queen Mary University of London

Outline

Examples

Rogue (1980)



Spelunky (2008)

Diablo (96/00/12)

Examples

Minecraft (2011)

No Man's Sky (2016)

Examples

Left 4 Dead (08/09)

Dirt 4 (2017)

Conclusion

Find the right tool for the job

Example Application: Mario Level Generation

Tile type	Symbol	Identity	Visualisation
Solid/Ground	X	0	
Breakable	S	1	
Empty (passable)	-	2	
Full question block	?	3	
Empty question block	Q	4	
Enemy	E	5	
Top-left pipe	<	6	
Top-right pipe	>	7	
Left pipe	[	8	
Right pipe	]	9	
Coin	o	10	

Levels: `https://github.com/TheVGLC/TheVGLC`

Taxonomy

1. online vs. offline
2. necessary vs. optional
3. degree and dimensions of control
4. generic vs. adaptive
5. stochastic vs. deterministic
6. constructive vs. generate-and-test
7. automatic generation vs mixed authorship

Taxonomy

1. online vs. offline
2. necessary vs. optional
3. degree and dimensions of control
4. generic vs. adaptive
5. stochastic vs. deterministic
6. constructive vs. generate-and-test
7. automatic generation vs mixed authorship

from: Procedural Content Generation in Games: A Textbook and an Overview of Current Research, Chapter 1 (`pcgbook.com`)

Desirable properties of a PCG solution

A Speed

B Reliability

C Controllability

D Expressivity and Diversity

E Creativity and Believability

from: Procedural Content Generation in Games: A Textbook and an Overview of Current Research, Chapter 1 (`pcgbook.com`)

- Speed: Requirements for speed vary wildly, from a maximum generation time of milliseconds to months, depending on (amongst other things) whether the content generation is done during gameplay or during development of the game.
- Reliability: Some generators shoot from the hip, whereas others are capable of guaranteeing that the content they generate satisfies some given quality criteria. This is more important for some types of content than others, for example a dungeon with no exit or entrance is a catastrophic failure, whereas a flower that looks a bit weird just looks a bit weird without this necessarily breaking the game.
- Controllability: There is frequently a need for content generators to be controllable in some sense, so that a human user or an algorithm (such as a player-adaptive mechanism) can specify some aspects of the content to be generated. There are many possible dimensions of control, e.g. one might ask for a smooth oblong rock, a car that can take sharp bends and has multiple colours, a level that induces a sense of mystery and rewards perfectionists, or a small ruleset where chance plays no part.

from: Procedural Content Generation in Games: A Textbook and an Overview of Current Research, Chapter 1 (`pcgbook.com`)

- Expressivity and diversity: There is often a need to generate a diverse set of content, to avoid the content looking like it's all minor variations on a tired theme. At an extreme of non-expressivity, consider a level "generator" that always outputs the same level but randomly changes the colour of a single stone in the middle of the level; at the other extreme, consider a "level" generator that assembles components completely randomly, yielding senseless and unplayable levels. Measuring expressivity is a non-trivial topic in its own right, and designing level generators that generate diverse content without compromising on quality is even less trivial.
- Creativity and believability: In most cases, we would like our content not to look like it has been designed by a procedural content generator. There is a number of ways in which generated content can look generated as opposed to human-created.

Outline

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

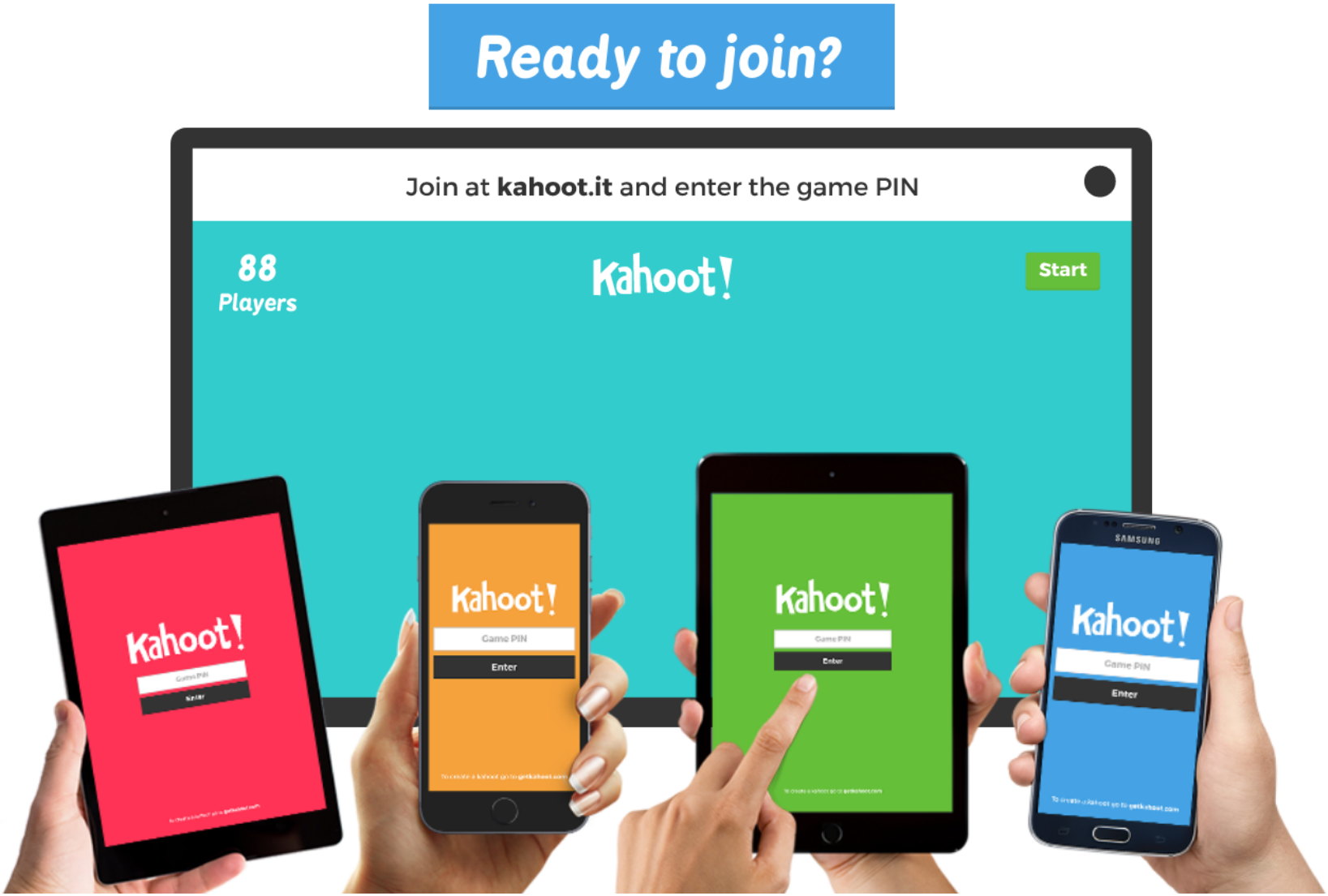
V. Volz, J. Schrum, J. Liu, S. Lucas, A. Smith, S. Risi. 2018. Evolving Mario Levels in the Latent Space of a Deep Convolutional Generative Adversarial Network.In Genetic and Evolutionary Computation Conference (GECCO 2018). ACM Press, New York, NY.

Generative Adversarial Networks learn to emulate Mario levels

0:38 / 4:58

https://youtu.be/NObqDuPuk7Q?t=35

Become the Adversary!



Generative Adversarial Networks (GANs)

NVIDIA 2017

Radford et al. 2015

The Adversaries

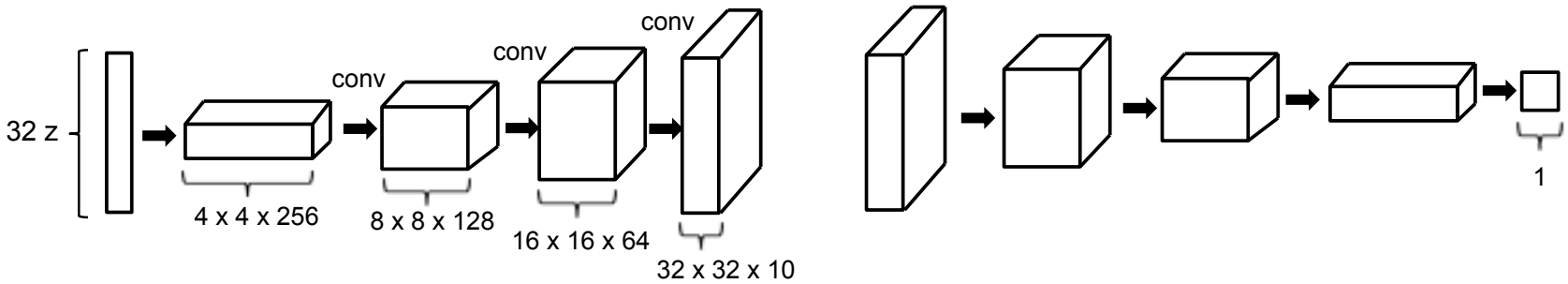
Generator

$G(\mathbf{z};\theta_g)$

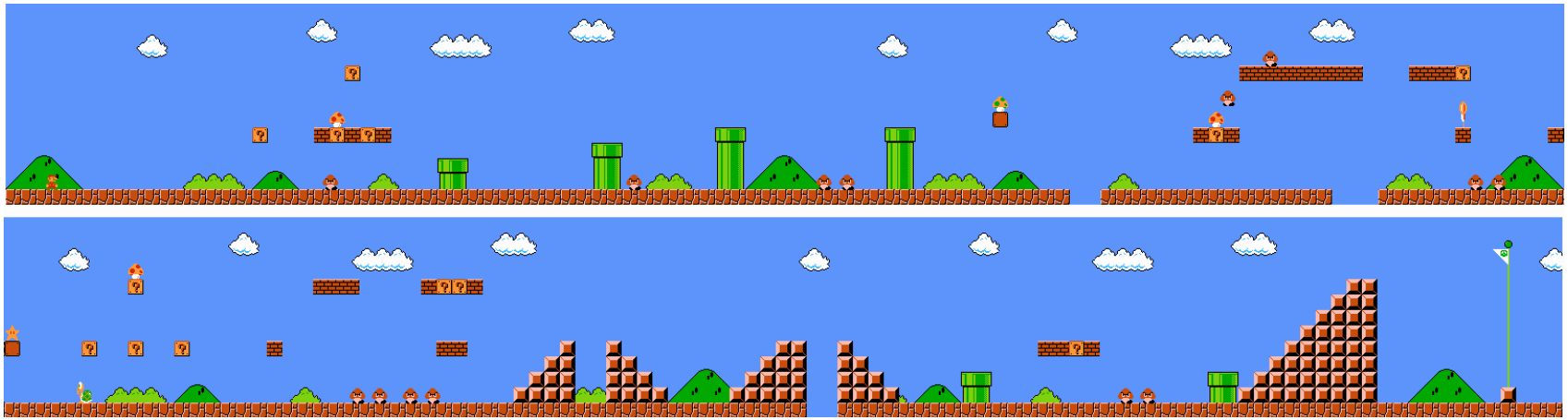
Discriminator

$D(\mathbf{x};\theta_d)$

$p_{\mathbf{z}}(\mathbf{z})$



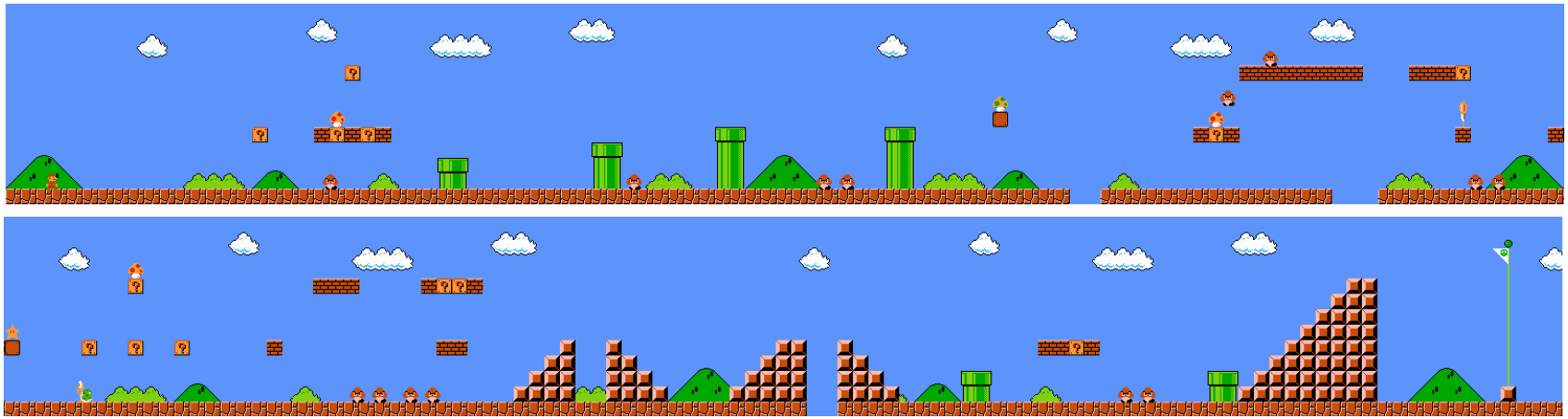
Training



p_{data}:

-	0.8667	X	0.1004
S	0.0107	E	0.0053
[	0.0039	]	0.0039
Q	0.0035	<	0.0021
>	0.0021	?	0.0011

Formal Description



Let $D(\mathbf{x})$ be the probability that $\mathbf{x}$ came from data distribution p_{data} rather than generator distribution p_g.

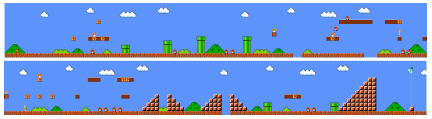
Train:

- D: max probability of correct label
- G: min $\log(1 - D(G(z)))$

Minimax game:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

Formal Description



Let $D(\mathbf{x})$ be the probability that $\mathbf{x}$ came from data distribution p_{data} rather than generator distribution p_g.

Train:

- D: max probability of correct label
- G: min $\log(1 - D(G(z)))$

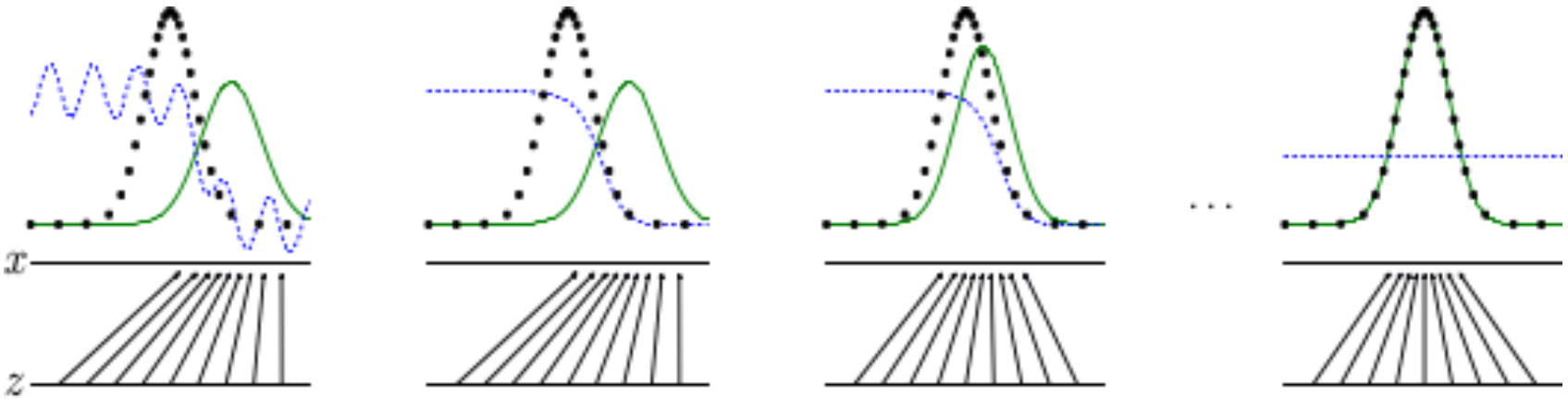
Minimax game:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

from: I Goodfellow, J Pouget-Abadie, M Mirza, B Xu, D Warde-Farley, S Ozair, A Courville, Y. Bengio. 2014. Generative Adversarial Networks. In "Advances in neural information processing systems", 2672-2680.

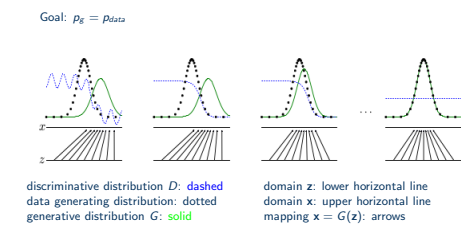
Training Visualisation

Goal: $p_g = p_{data}$



discriminative distribution D: dashed
data generating distribution: dotted
generative distribution G: solid

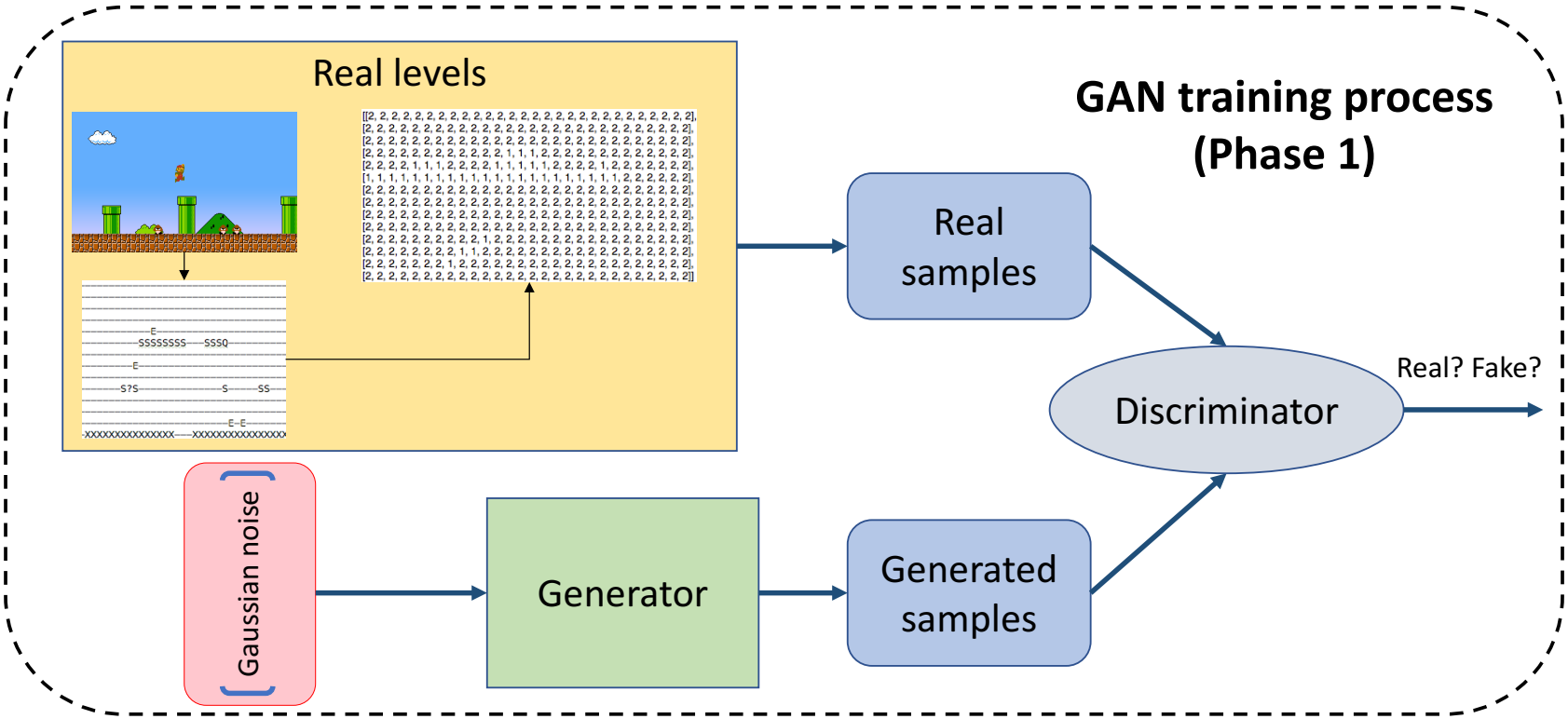
domain $\mathbf{z}$: lower horizontal line
domain $\mathbf{x}$: upper horizontal line
mapping $\mathbf{x} = G(\mathbf{z})$: arrows



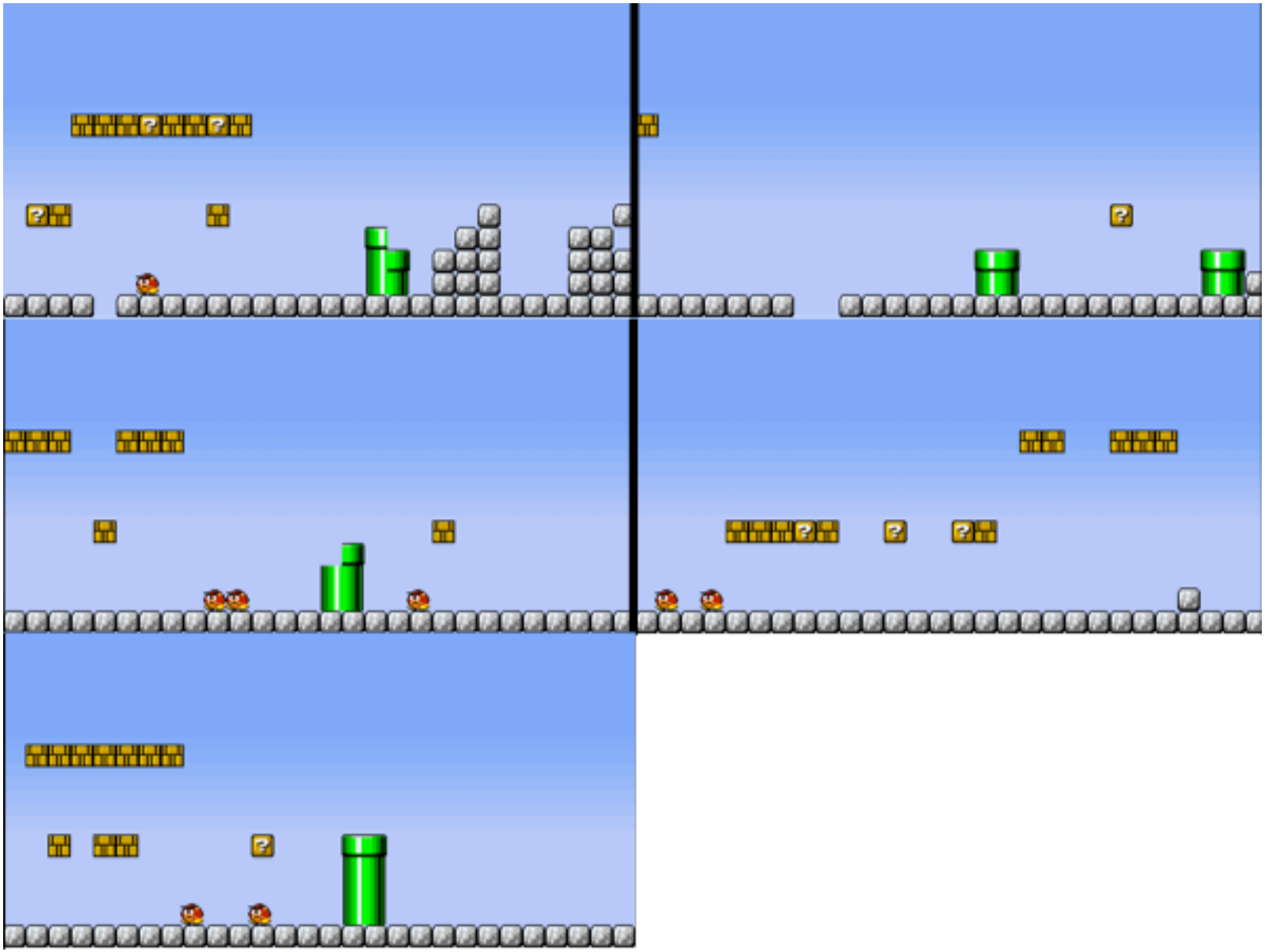
from: I Goodfellow, J Pouget-Abadie, M Mirza, B Xu, D Warde-Farley, S Ozair, A Courville, Y. Bengio. 2014. Generative Adversarial Networks. In "Advances in neural information processing systems", 2672-2680.

Formal proof in original paper `http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf`

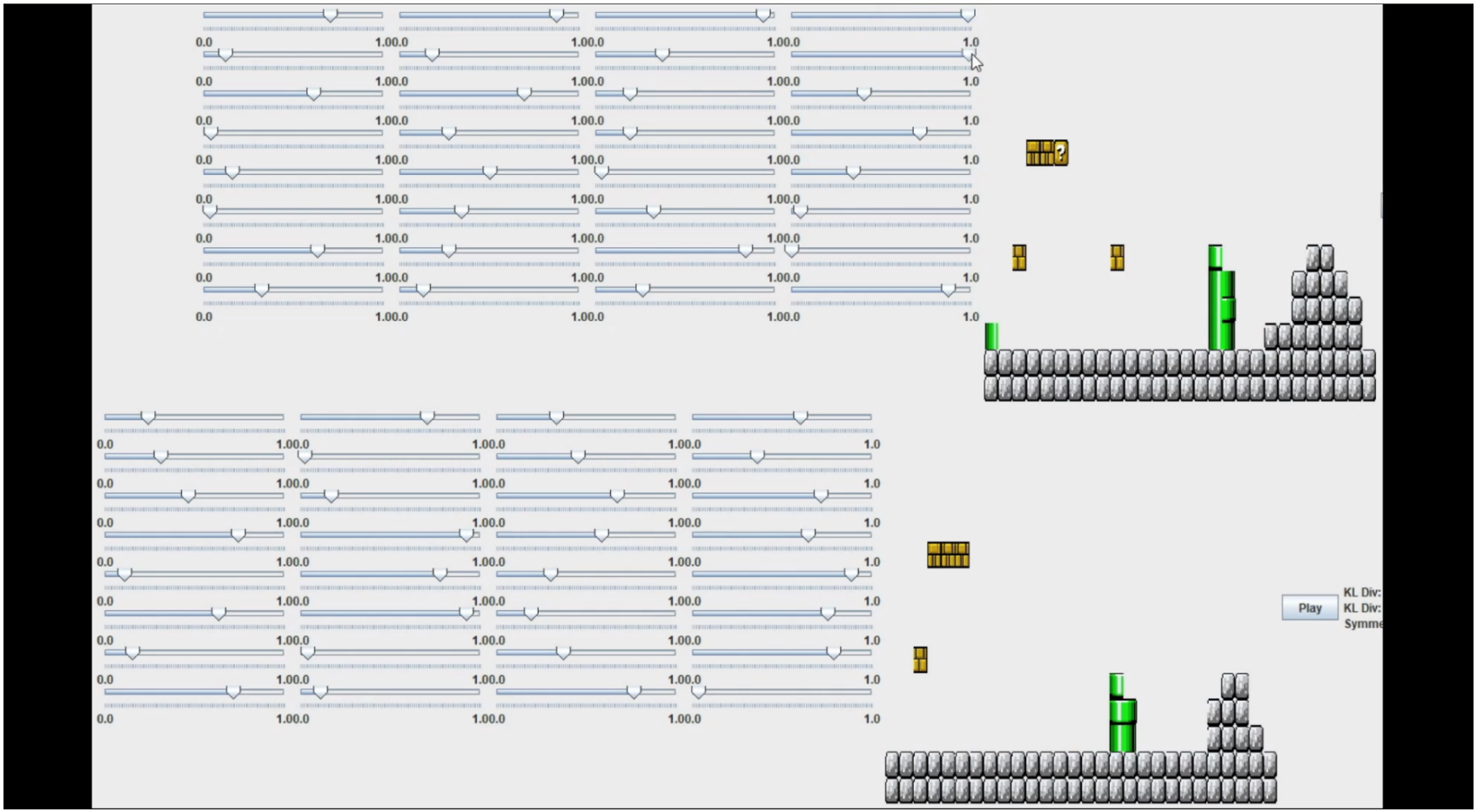
MarioGAN Training (Phase 1)



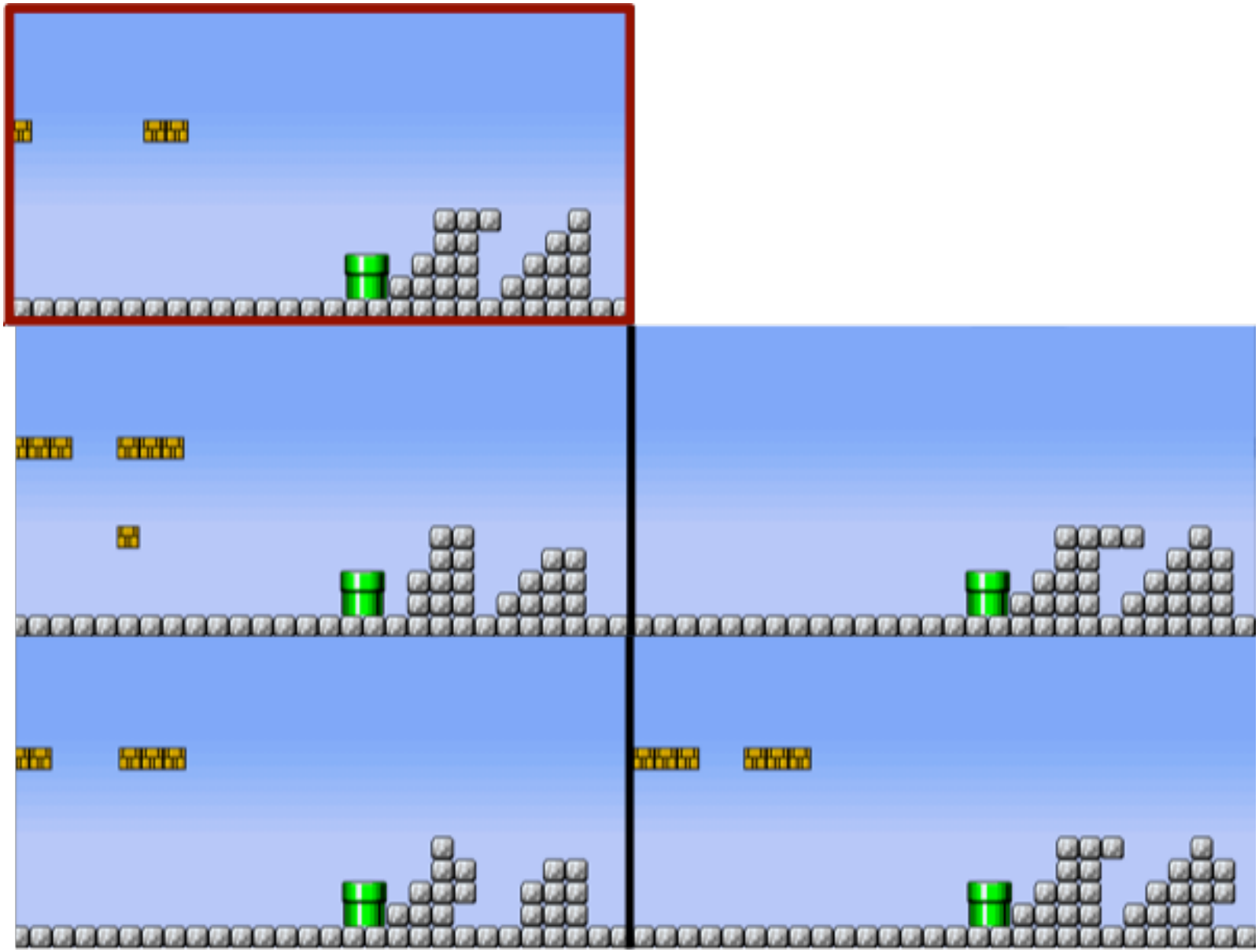
Results (Random Sample)



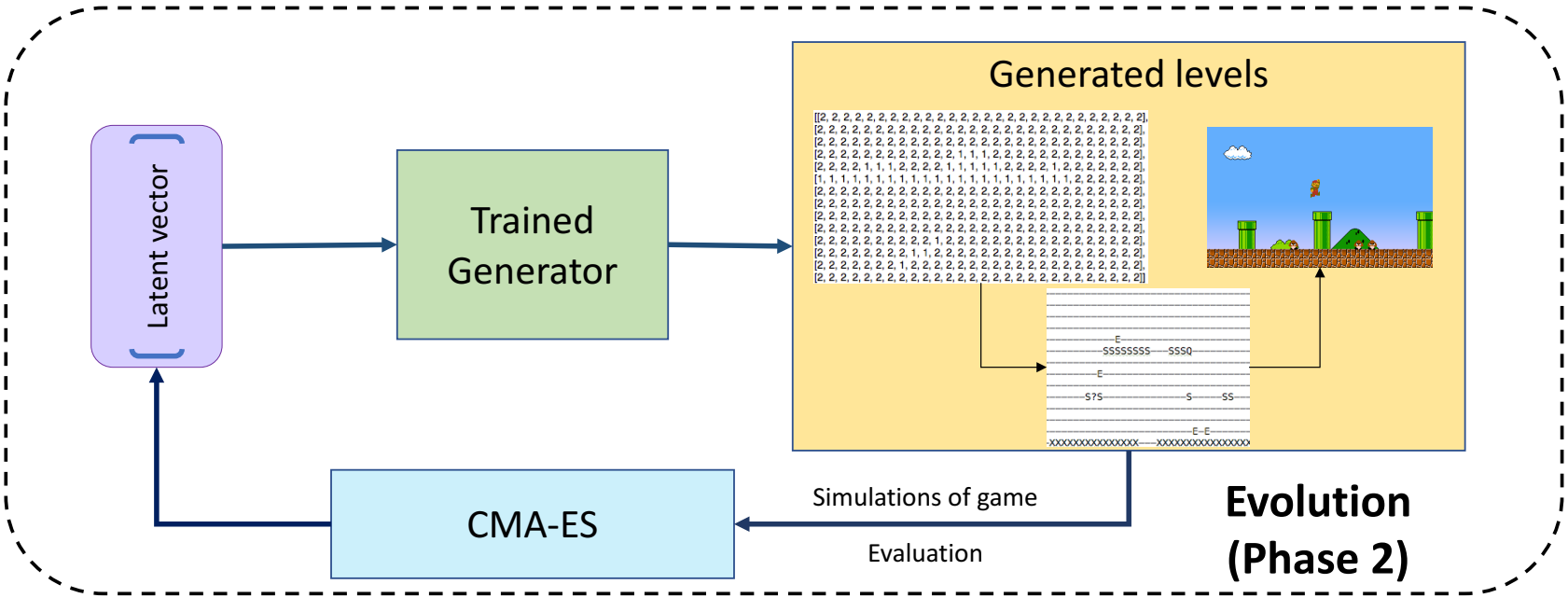
Exploring Latent Space (Interactive)



Exploring Latent Space (Mutations)



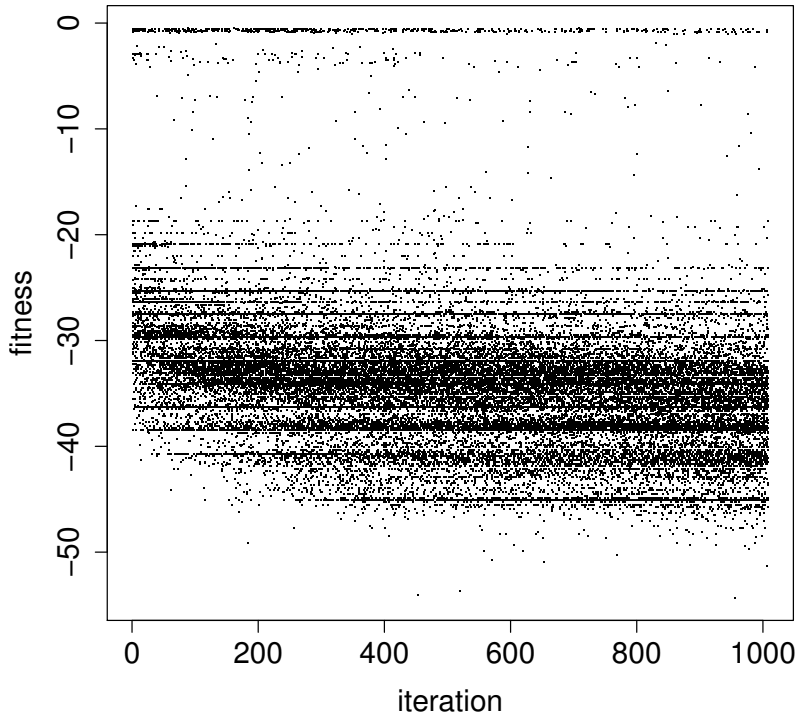
MarioGAN Evolution (Phase 2)



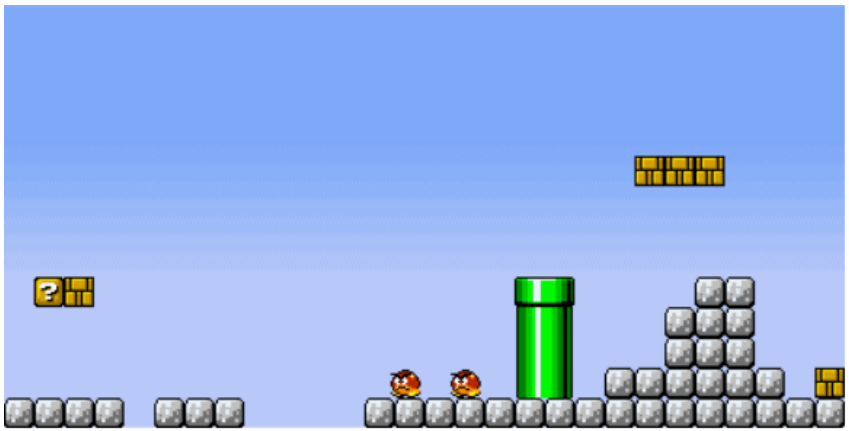
Fitness Function

$$F = \begin{cases} -p & \text{for } p < 1 \\ -p - \#jumps & \text{for } p = 1, \end{cases}$$

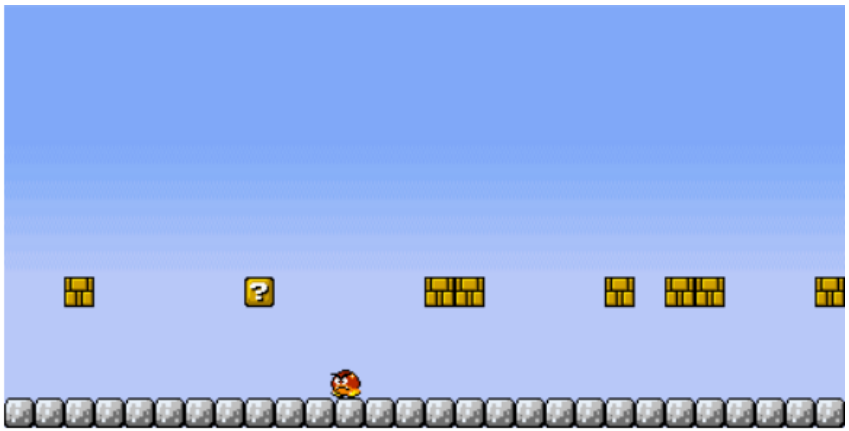
where p is the fraction of the level that was completed in terms of progress on the x-axis.



Example Results



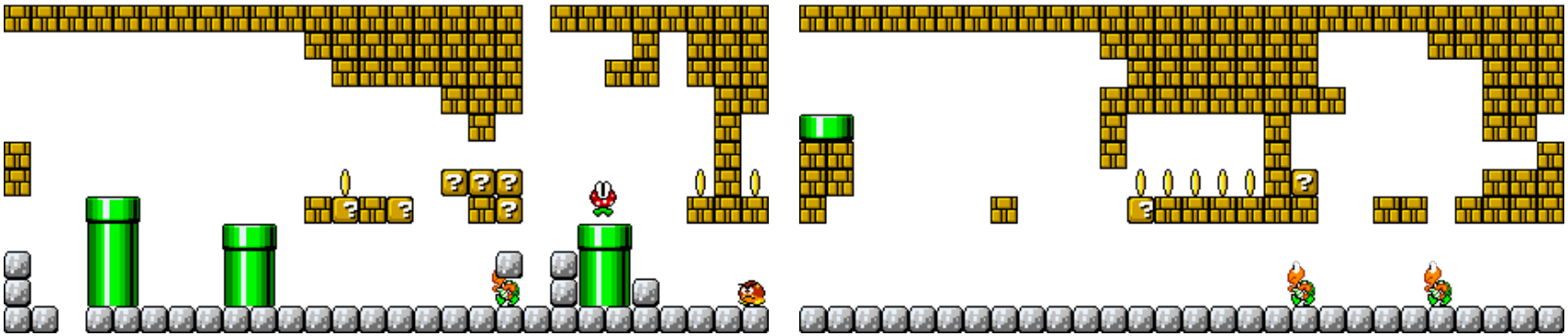
(a) Playable level maximizing jumps



(b) Playable level minimizing jumps

Updates

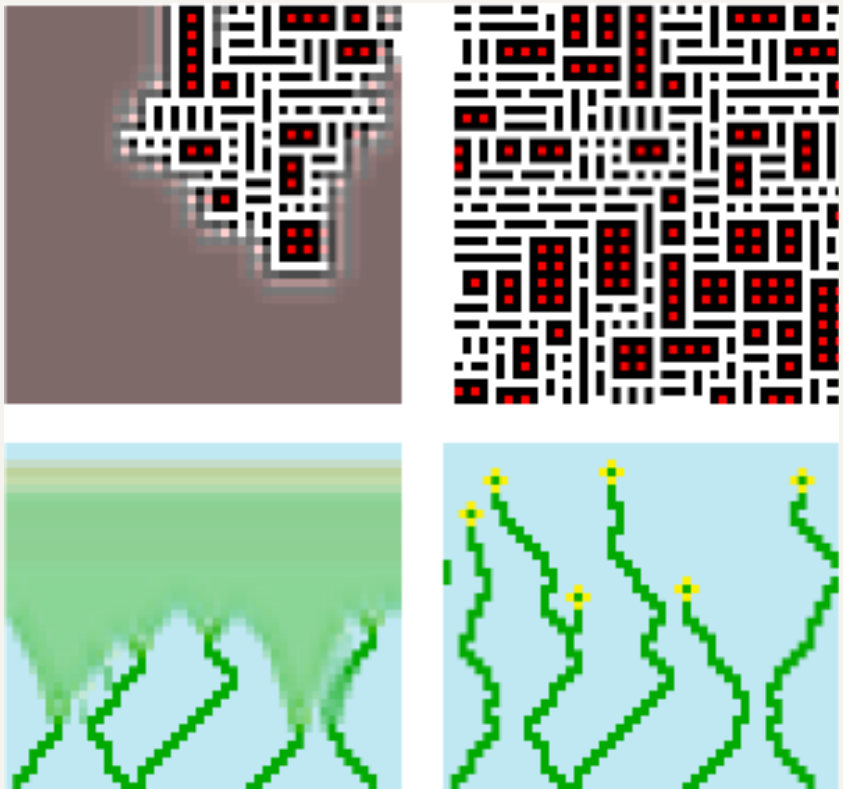
- Encoding
- Extended training sets
- Fitness functions
- Benchmark

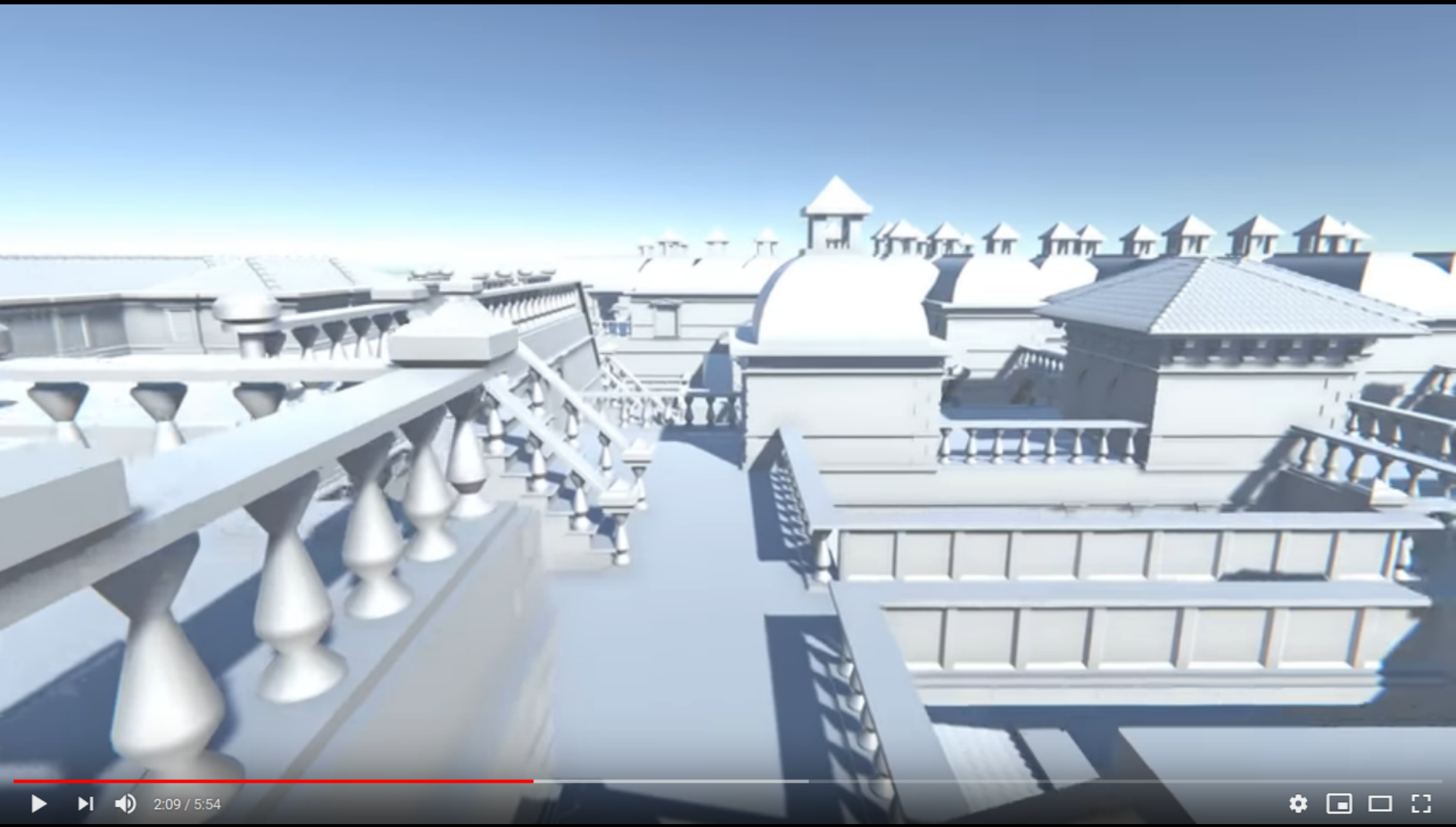


Outline

Paper: Isaac Karth, Adam M. Smith. 2017. WaveFunctionCollapse is Constraint Solving in the Wild. FDG 2017. `https://adamsmith.as/papers/wfc_is_constraint_solving_in_the_wild.pdf`

Talk: EPC2018 - Oskar Stalberg - Wave Function Collapse in Bad North
`https://www.youtube.com/watch?v=0bcZb-SsnrA`





https://www.youtube.com/watch?v=-W7zt8181Zo

Pseudo Code

```
1: procedure RUN
2:     PatternsFromSample()
3:     BuildPropagator()
4:     while not finished do
5:         Observe()
6:         Propagate()
7:     end while
8:     return level
9: end procedure
```

PatternsFromSample()

pattern

Particular, unique configuration of input tiles

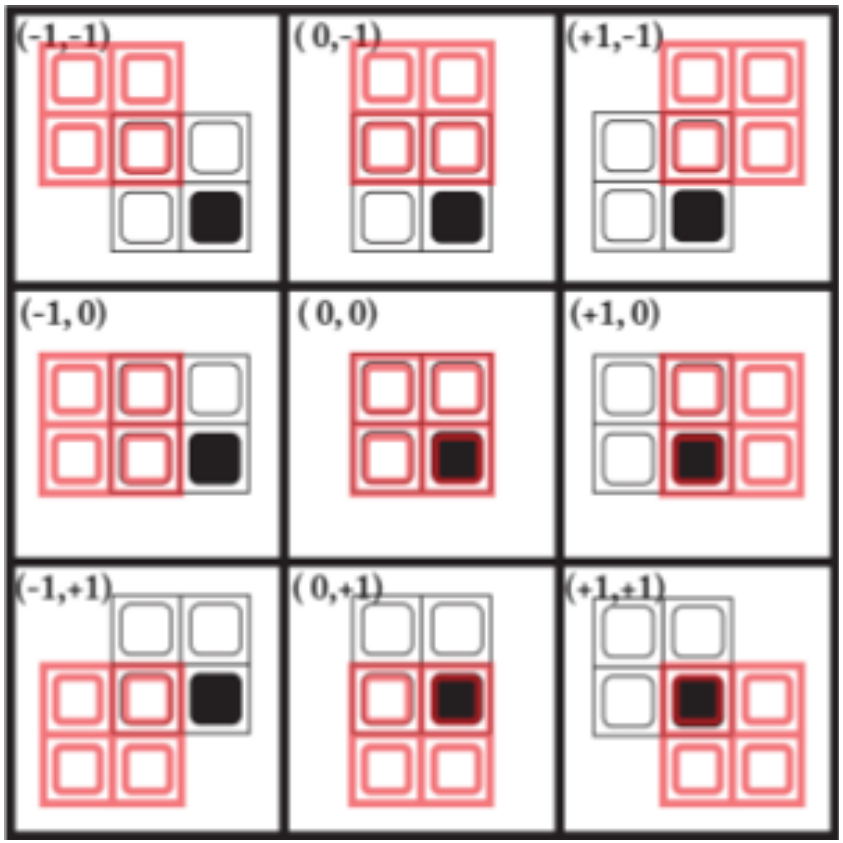
1×1 patterns (10)

-	0.8667	X	0.1004
S	0.0107	E	0.0053
[	0.0039	]	0.0039
Q	0.0035	<	0.0021
>	0.0021	?	0.0011

2×2 patterns (57)

-	-	0.8029	[	]	0.0023
-	-		X	X	

BuildPropagator()



Manual definition or automatic from training data

Observe(coefficient_matrix)

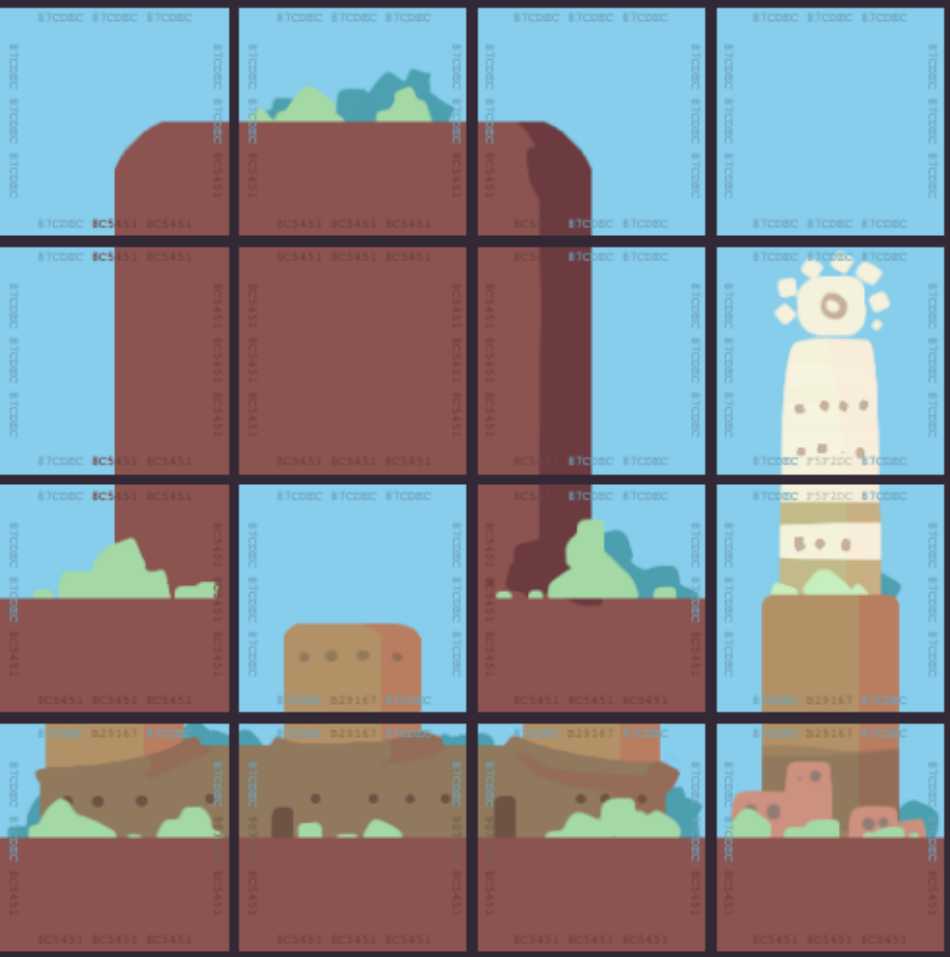
```
1:  procedure OBSERVE(coefficient_matrix)
2:      cell=FindLowestEntropy()
3:      if contradiction then
4:          throw error and quit
5:      end if
6:      if entropy=0 ∀ cells then
7:          return level                          ▷ Processing complete
8:      end if
9:      Choose pattern by random sample           ▷ ∼ pattern frequency
10:     All except chosen pattern invalid for cell
11:     Mark cell for update
12: end procedure
```

Propagate(coefficient_matrix)

```
1:  procedure PROPAGATE(coefficient_matrix)
2:      while ∃ cell marked for update do
3:          for all neighbour cells do
4:              for all valid patterns do
5:                  if pattern now invalid then
6:                      Set corresponding indicator to false
7:                      Mark cell for update
8:                  end if
9:              end for
10:         end for
11:     end while
12: end procedure
```

MODULES

SLOTS

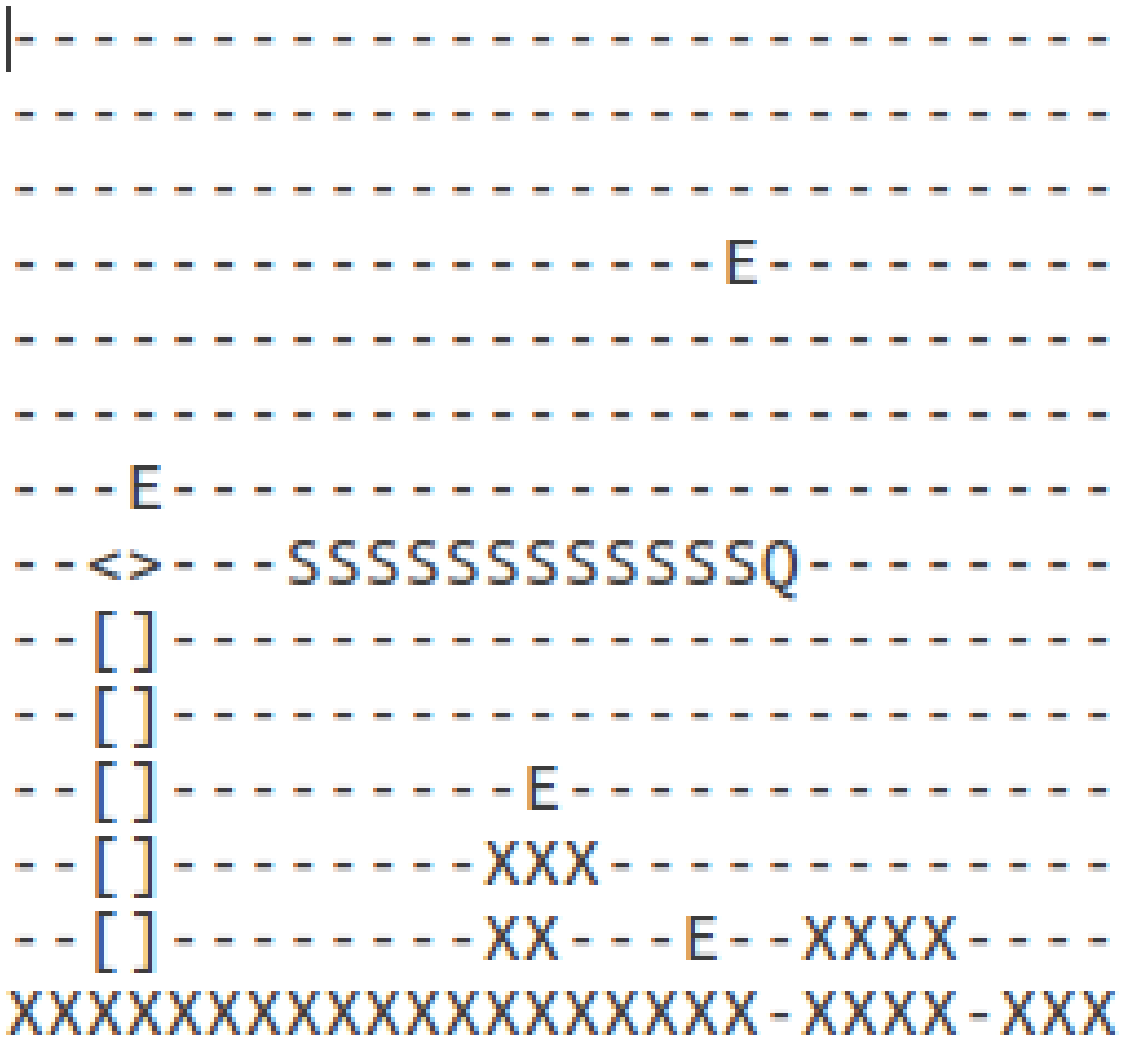


ALL NONE RESET SOLVE

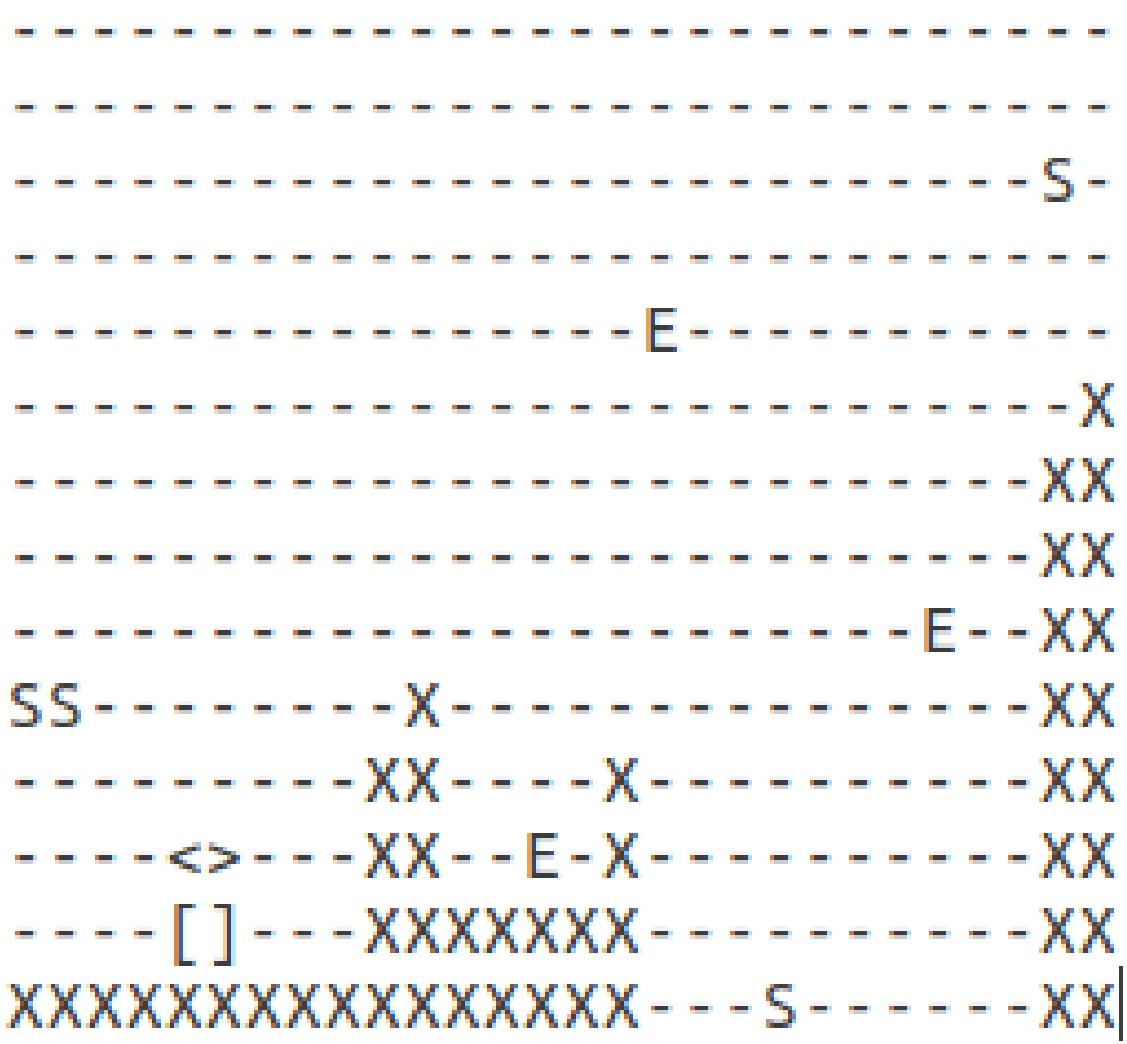
http://oskarstalberg.com/game/wave/wave.html

Wave Function Collapse: Mario Demo

WFC Mario: Filters



WFC Mario: Borders



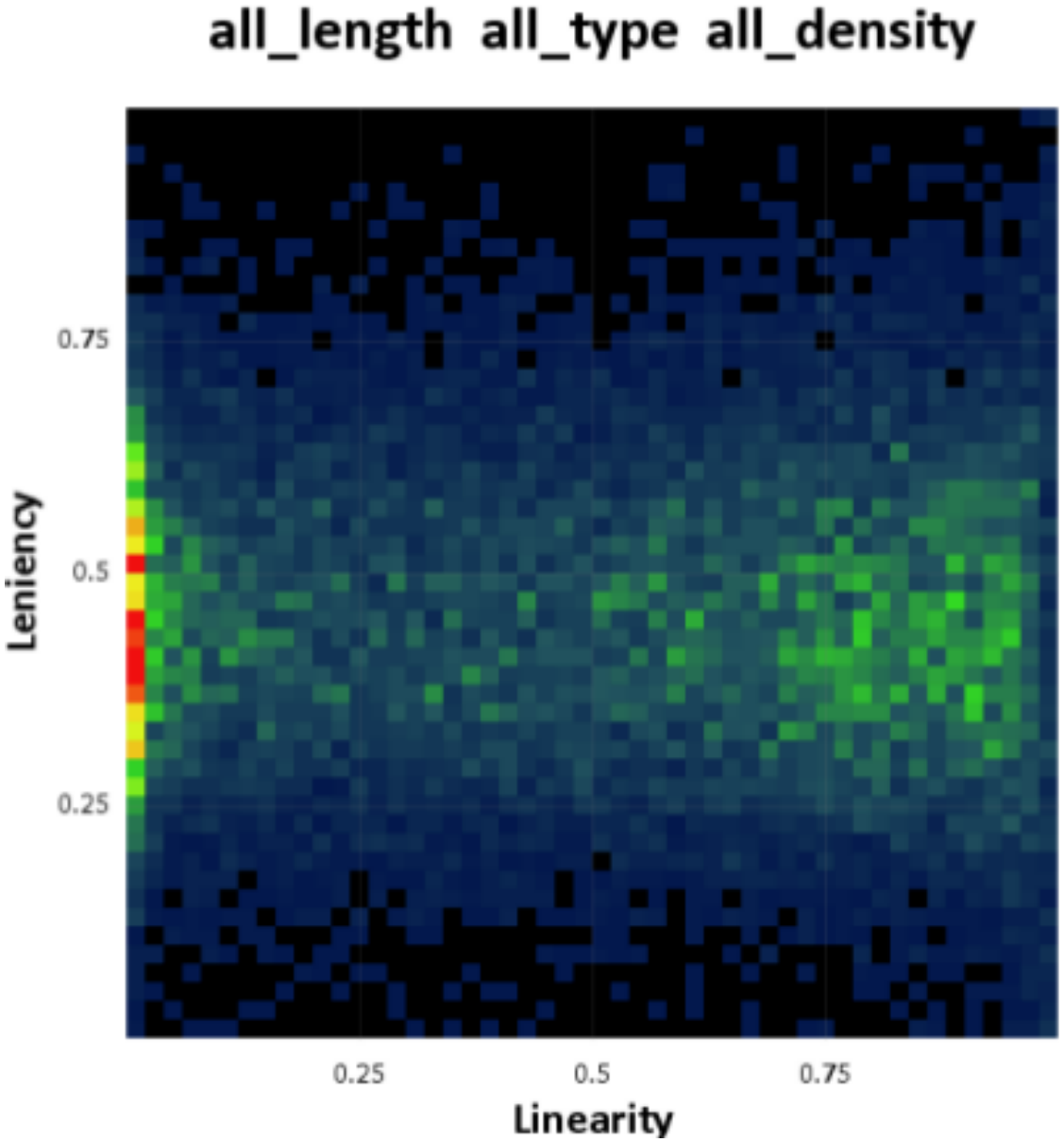
Outline

PCGbook: Noor Shaker, Julian Togelius, and Mark J. Nelson (2016). Procedural Content Generation in Games: A Textbook and an Overview of Current Research. Springer. ISBN 978-3-319-42714-0.

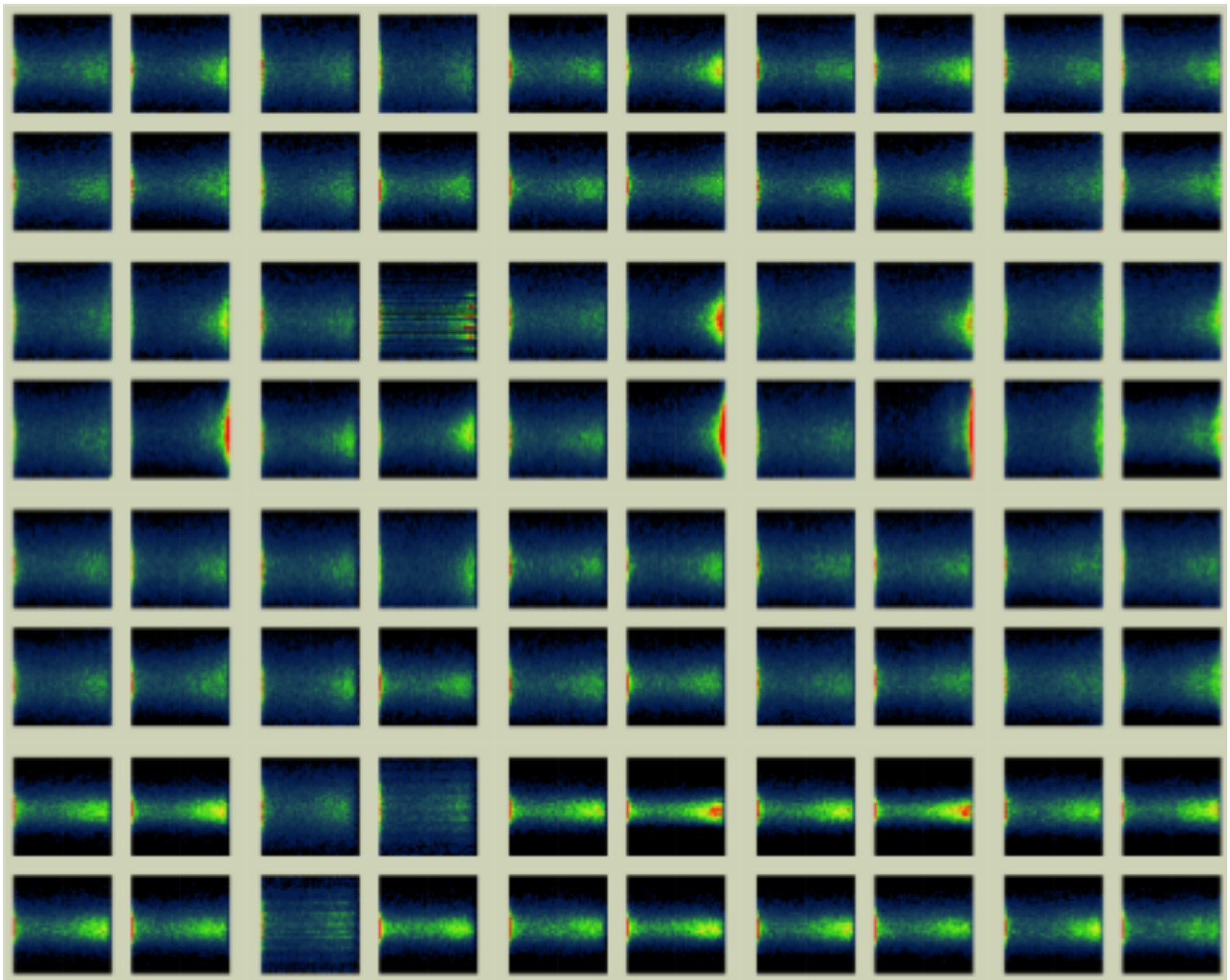
Specifically chapter 12: **Evaluating content generators**

`http://pcgbook.com/wp-content/uploads/chapter12.pdf`

Expressive Range



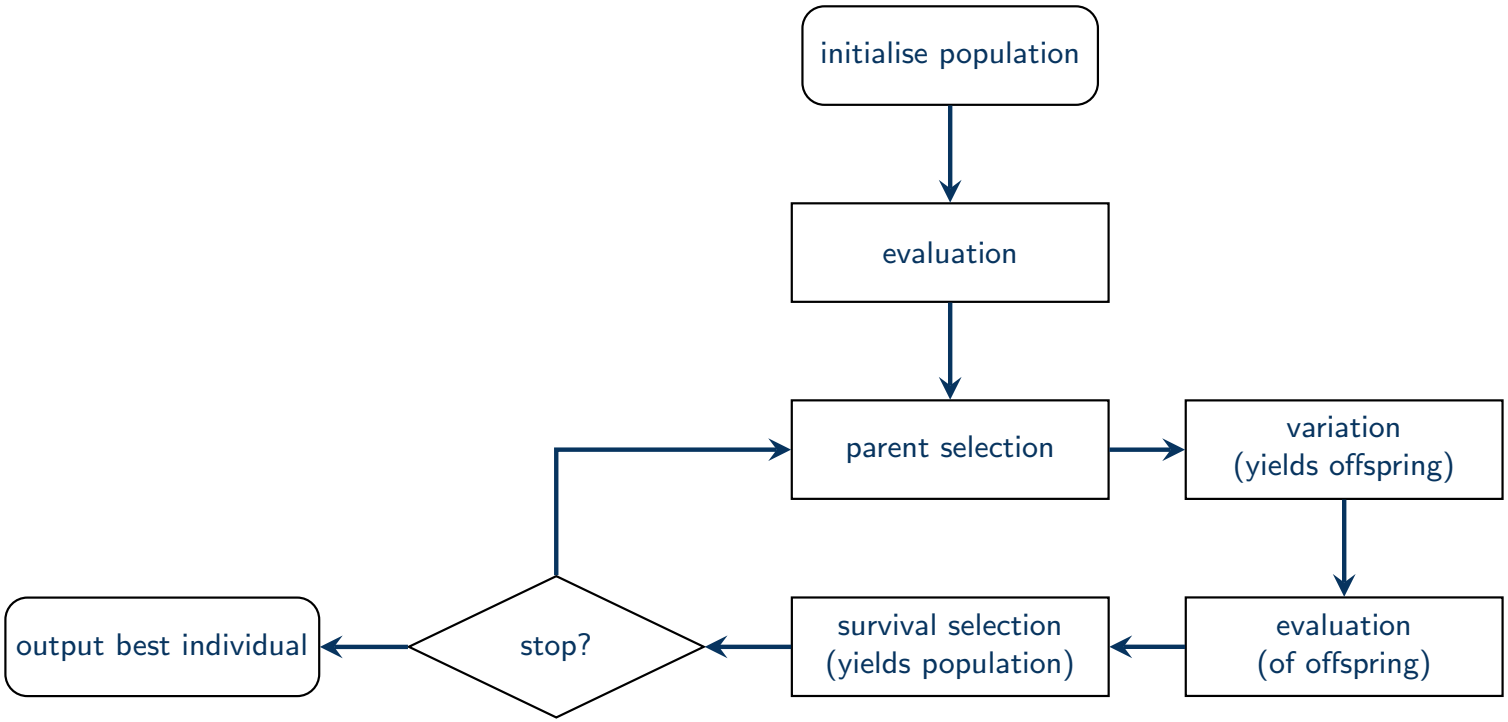
Controllability and Expressive Range



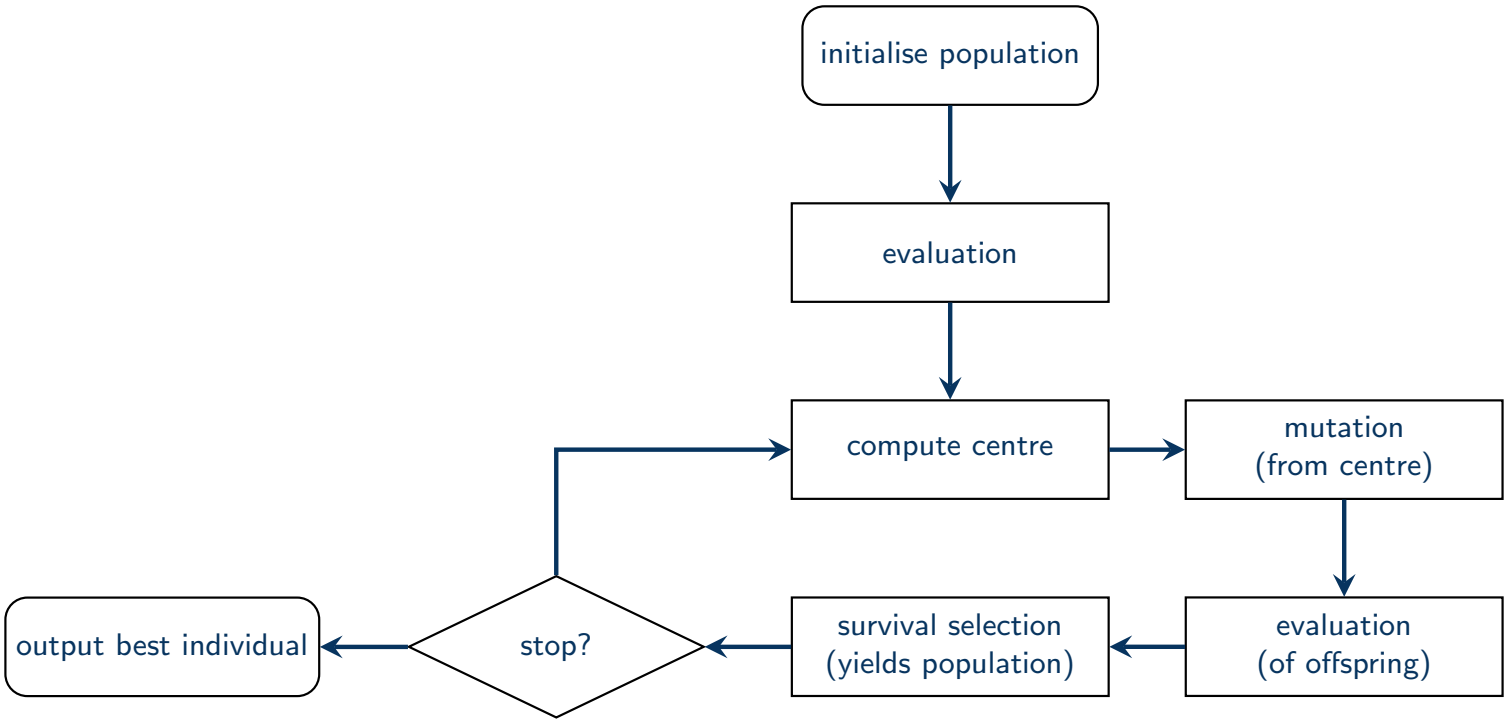
Playtests

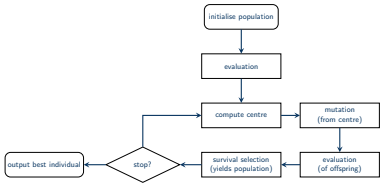
Appendix

Evolutionary Algorithms



CMA-ES Outline





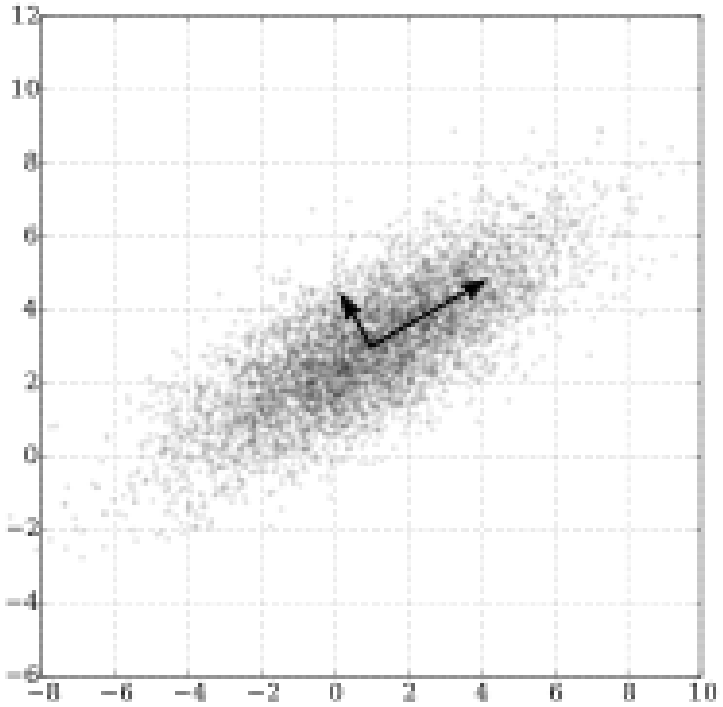
Resources used in this section

- Course Computational Intelligence (Günter Rudolph, TU Dortmund University)
- CMA-ES Tutorial PPSN 2006 (Nikolaus Hansen, INRIA)
- Nikolaus Hansen homepage: `http://www.cmap.polytechnique.fr/~nikolaus.hansen/`

Covariance Matrix

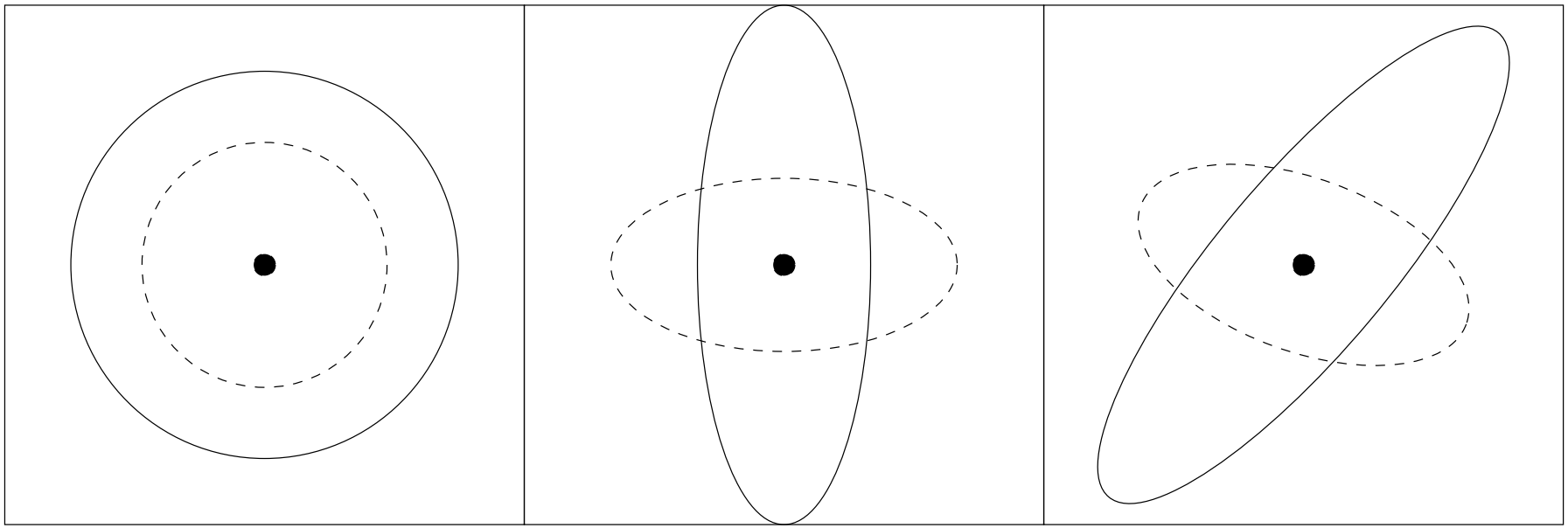
Covariance Matrix

Each element i, j describes the covariance of i-th and j-th element of a random vector. The covariance measures the joint variability of two random variables.



The **covariance matrix** $\mathbf{C}$ determines the shape. It has a nice **geometrical interpretation**: any covariance matrix can be uniquely identified with the iso-density ellipsoid $\{\boldsymbol{x} \in \mathbb{R}^n \,|\, \boldsymbol{x}^{\mathrm{T}}\mathbf{C}^{-1}\boldsymbol{x} = 1\}$

Lines of Equal Density



$\mathcal{N}\left(\boldsymbol{m}, \sigma^2\mathbf{I}\right) \sim \boldsymbol{m} + \sigma\mathcal{N}(\mathbf{0}, \mathbf{I})$
one free parameter σ
components of $\mathcal{N}(\mathbf{0}, \mathbf{I})$ are independent standard normally distributed

$\mathcal{N}\left(\boldsymbol{m}, \mathbf{D}^2\right) \sim \boldsymbol{m} + \mathbf{D}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$
n free parameters
components are independent, scaled

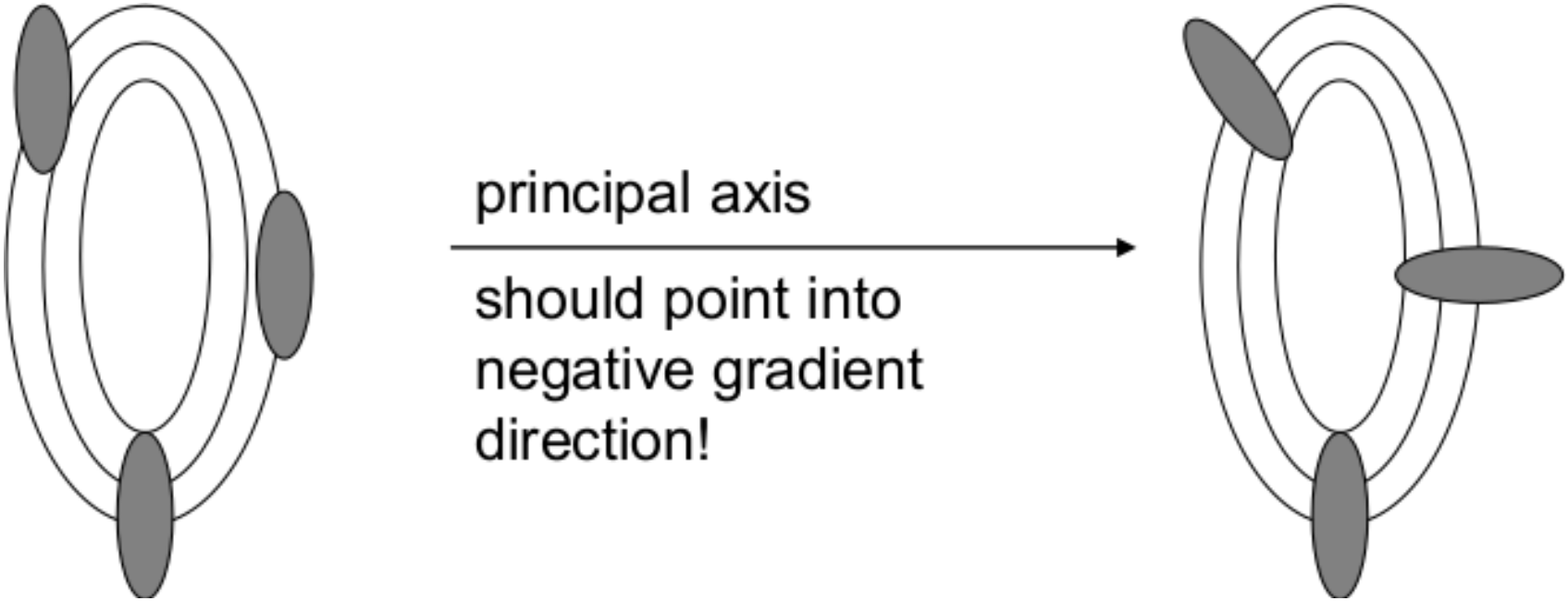
$\mathcal{N}(\boldsymbol{m}, \mathbf{C}) \sim \boldsymbol{m} + \mathbf{C}^{\frac{1}{2}}\mathcal{N}(\mathbf{0}, \mathbf{I})$
$(n^2 + n)/2$ free parameters
components are correlated

where $\mathbf{I}$ is the identity matrix (isotropic case) and $\mathbf{D}$ is a diagonal matrix (reasonable for separable problems) and $\mathbf{A} \times \mathcal{N}(\mathbf{0}, \mathbf{I}) \sim \mathcal{N}\left(\mathbf{0}, \mathbf{A}\mathbf{A}^{\mathrm{T}}\right)$ holds for all $\mathbf{A}$.

Mutation in CMA-ES

$Y = X + Z$, with $Z \sim N(0, C)$

- maximum entropy distribution for support $\mathbb{R}^n$,
- given expectation vector and covariance matrix



Covariance Matrix Adaptation with Rank-One Update

Iterative Approximation!

Iteration 0: initial covariance matrix $C^{(0)} = \mathbb{I}_n$

- λ offspring iteration t
- sort offspring: $f(x_{1:\lambda}) \leq f(x_{2:\lambda}) \leq \cdots \leq f(x_{\lambda:\lambda})$
- compute centre from selected offspring: $m = \frac{1}{\mu} \sum_{i=1}^{\mu} x_{i:\lambda}$
- update: $C^{(t+1)} = (1-\eta)C^{(t)} + \eta \sum_{i=1}^{\mu} w_i (x_{i:\lambda} - m)(x_{i:\lambda} - m)^T$,
  η: "learning rate" $\in (0, 1)$, w_i: weights, for example $\frac{1}{\mu}$