Game Levels
Procedural Generation

Syllabus

- Tools : random numbers, noise
- Modular approach
- Grammars
- Markov Chains
- Evolutionary Algorithm
- Cellular Automata
- Wave Function Collapse

Introduction

Mission / Game Space

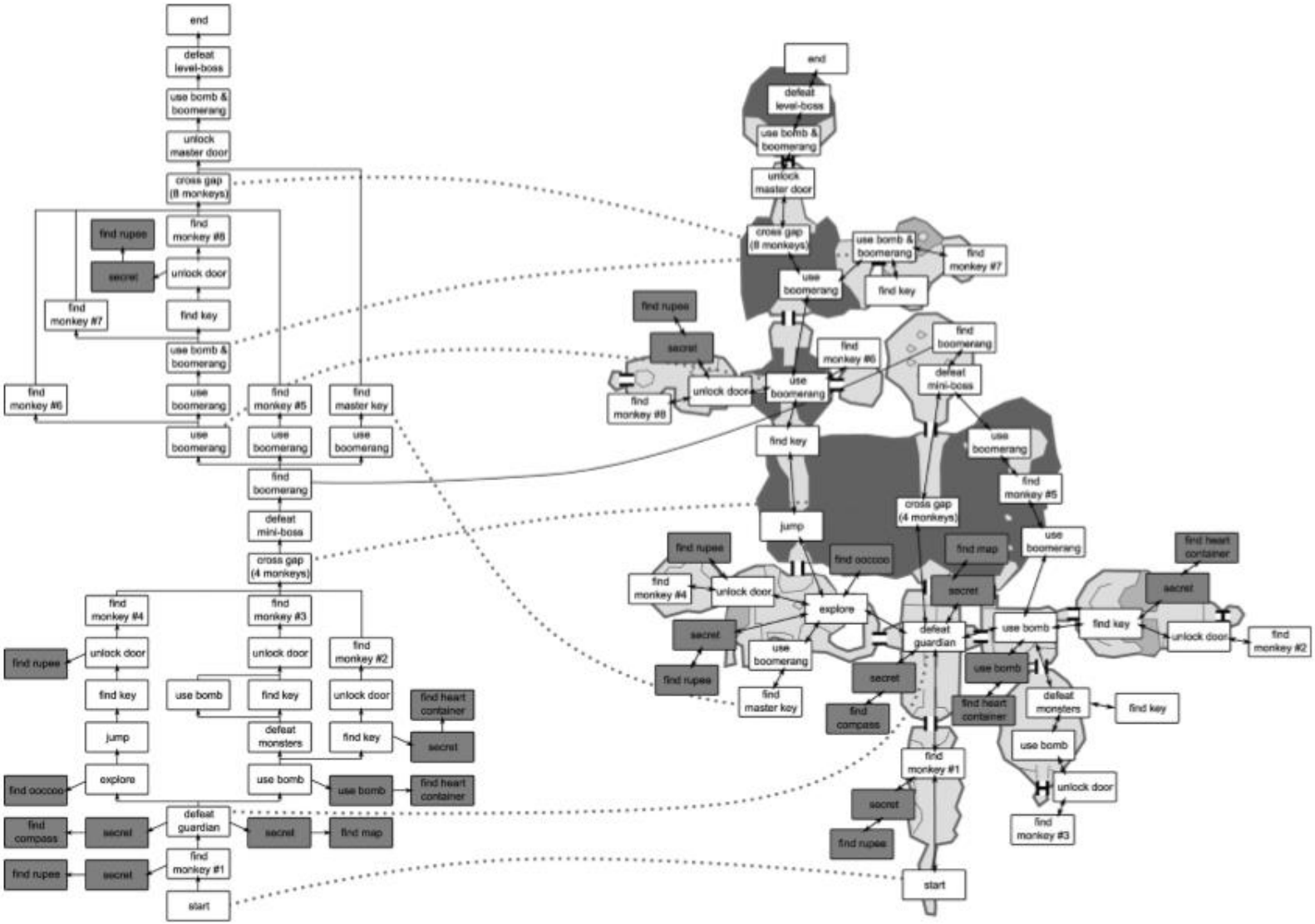


Fig. 1. Mission and space in the Forest Temple Level of *The Legend of Zelda: The Twilight Princess* [21].

Focus

- Main goal : focus generate game space
  - No planning algorithm

Positive aspects of generated game space

- Large possibility space
  - Curiosity, Exploration, Replayability
- Coherence
- Appropriation
- Cost ?

Drawbacks

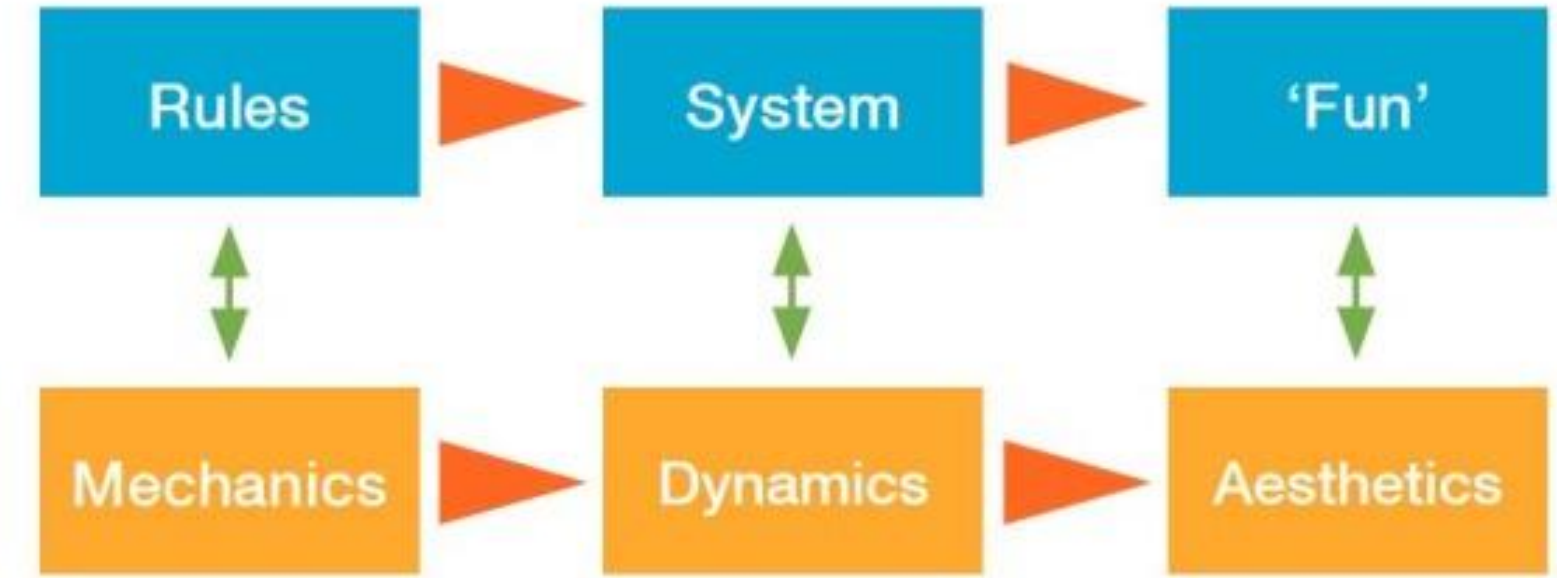
- Q.A.
  - Probabilist : can fail
- Hard to control / authorship
- Player get lost in uniformity
  - Just another sine wave mountain
- Lack of meaning
  - Why is there a cave here ?

Mixed Initiative Approach

Design Process

- To design a generator, you first need to talk with those who are doing it by hand:
  - Domain ontology (concepts and their relationships)
  - Evaluation function (differentiate between good and bad design)
  - Game space creatio process (what choices, which steps)

MDA

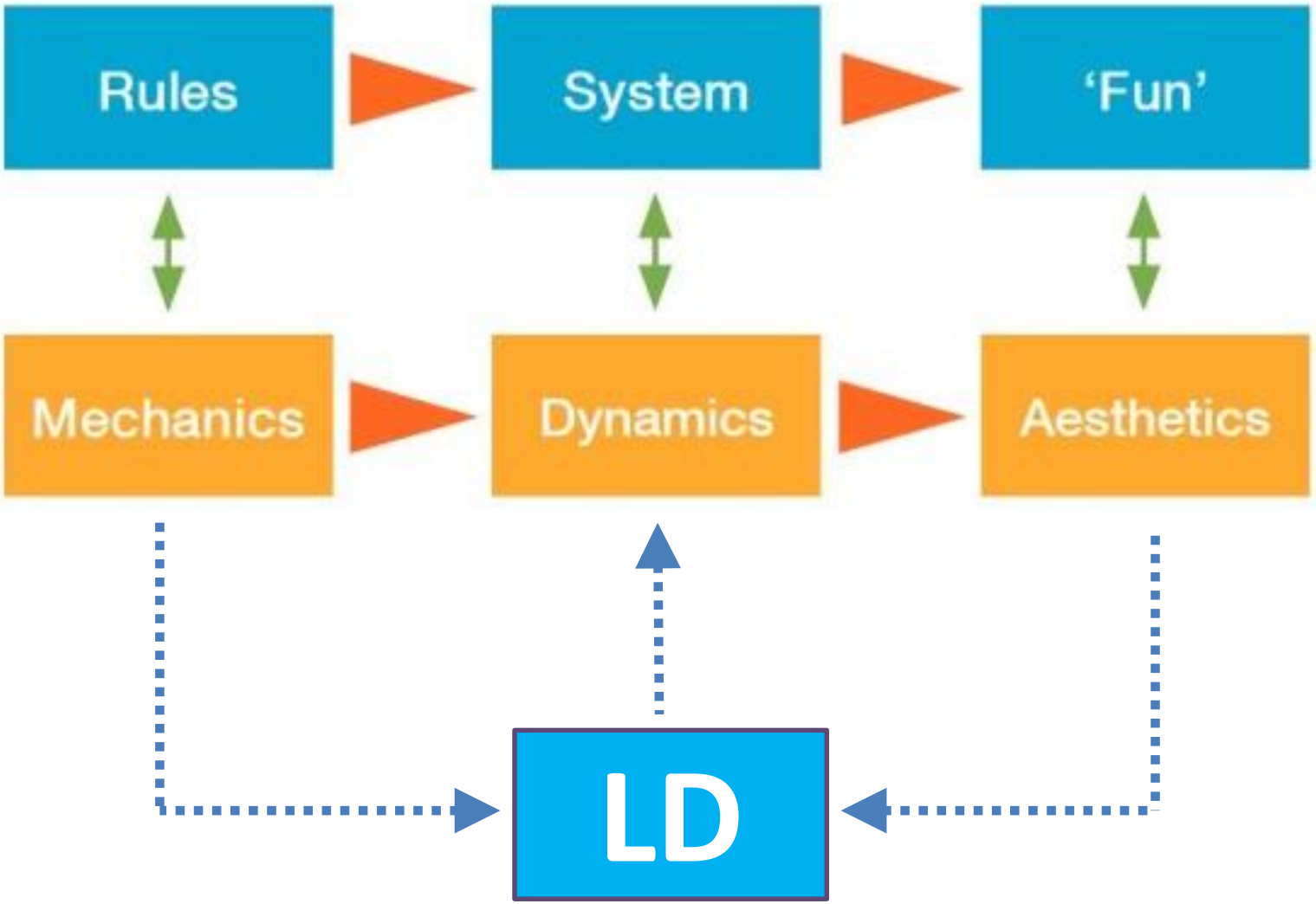


Mechanics : base components of the game - its rules, every basic action the player can take in the game, the algorithms and data structures in the game engine etc.

Dynamics : are the run-time behavior of the mechanics acting on player input and "cooperating" with other mechanics.

Aesthetics are the emotional responses evoked in the player - joy, frustration, fantasy, fellowship.

“For example, the mechanics of card games include shuffling, trick-taking and betting from which dynamics like bluffing can emerge” [Hunicke2004]

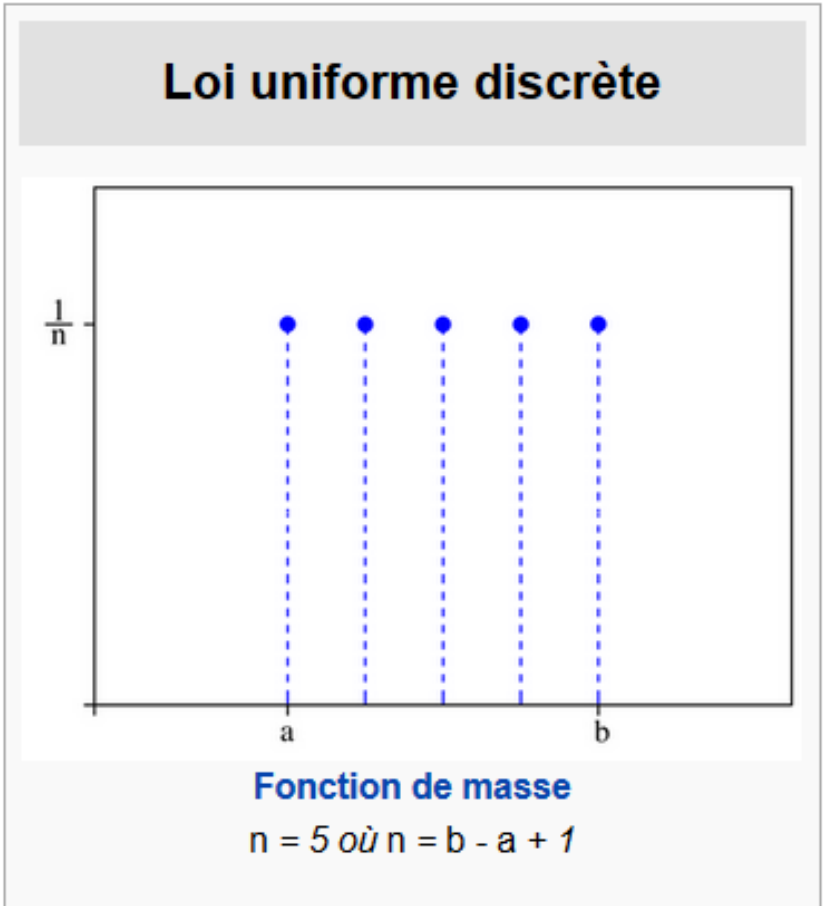

Rules
System
'Fun'
Mechanics
Dynamics
Aesthetics
LD

Basic tools

Randomness

Random numbers

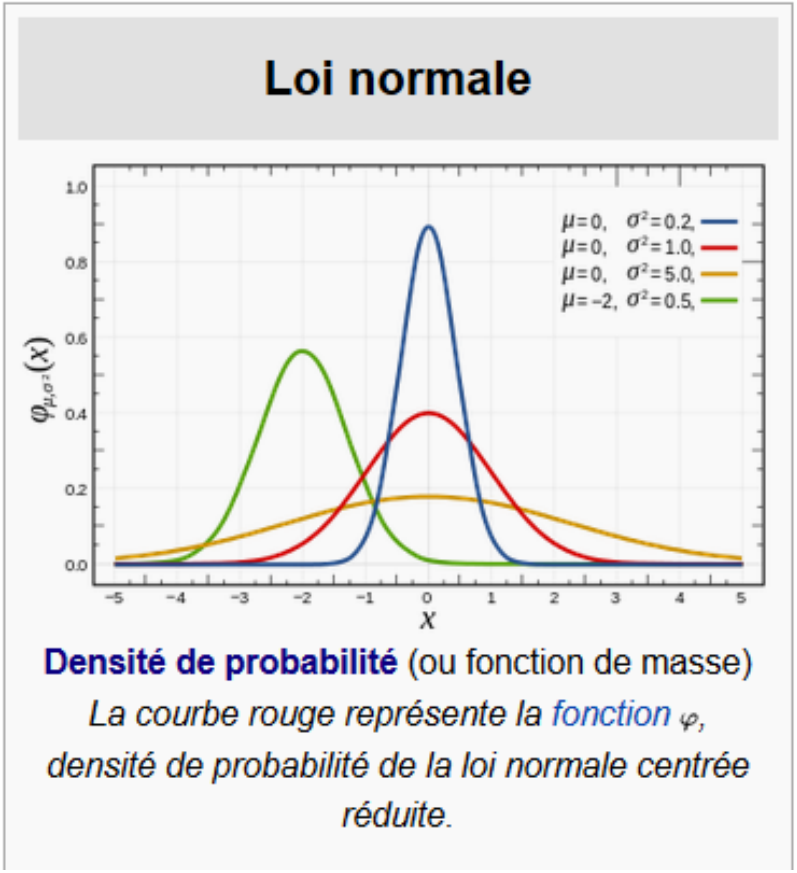
- rand() : pseudo random number generator, uniform distribution
  - Every number has the same probability to be observed
  - What if I need something else ?



Galton Board

Normal Distribution

- Two parameters : mean and standard deviation
- Natural: sum of independant random variables tends toward a normal distribution (central limit theorem)
  - Btw : be careful, summing rand() does not give another rand()



Rand() with normal distribution

$$\text{Si } \begin{cases} U \sim \mathcal{U}(0,1) \\ V \sim \mathcal{U}(0,1) \end{cases} \text{ alors } \begin{cases} \sqrt{-2\ln(U)}\cos(2\pi V) \sim \mathcal{N}(0,1) \\ \sqrt{-2\ln(U)}\sin(2\pi V) \sim \mathcal{N}(0,1) \end{cases}$$

- U(0,1) = loi uniforme dans]0,1]
- Puis on multiplie par écart type et on ajoute notre moyenne pour avoir un rand() normal avec nos paramètres.
- Il existe d'autres lois (poisson etc… à voir)

Tab Rand

- Give a specific probability to each choice.
  - Give each choice its probability (ex 0.2,0.3,0.5)
  - Normalize to have a sum of 1 (divide by sum of all numbers)
  - Draw a random number in [0,1]
  - If lower than first number : first event,
    - Elseif lower than second number : second event
    - Elseif… etc…

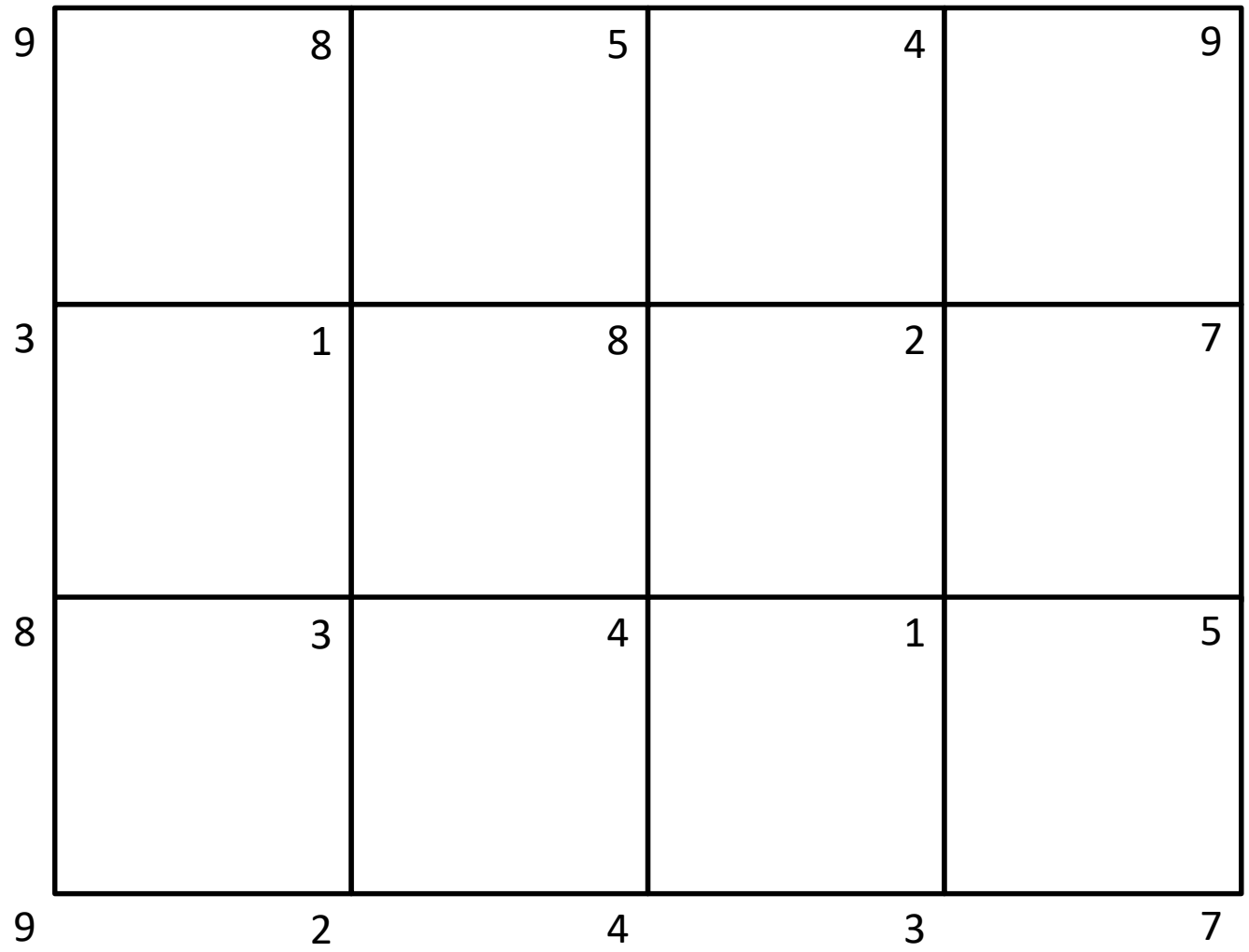
Tab Rand

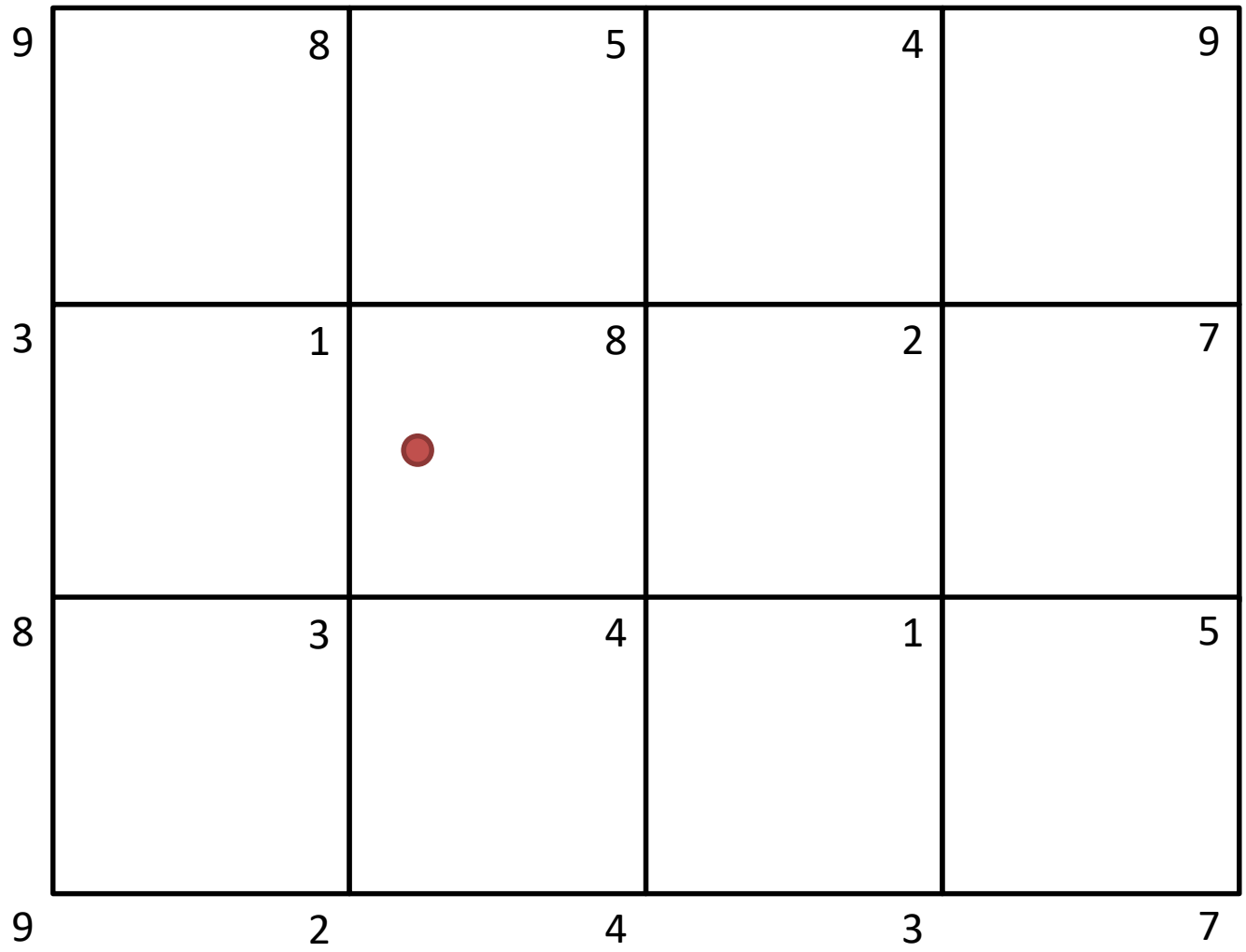
Basic tools

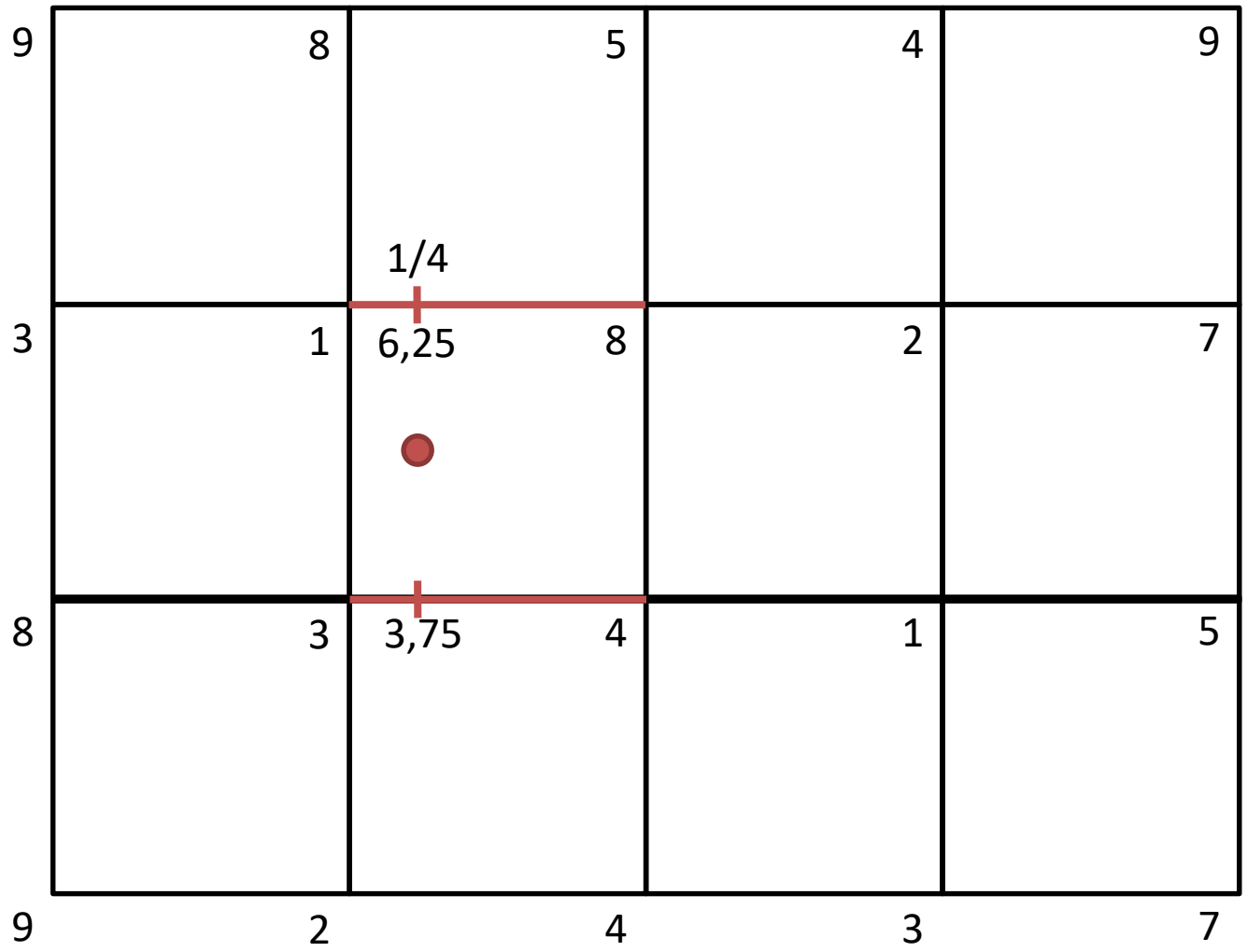
Noise Functions

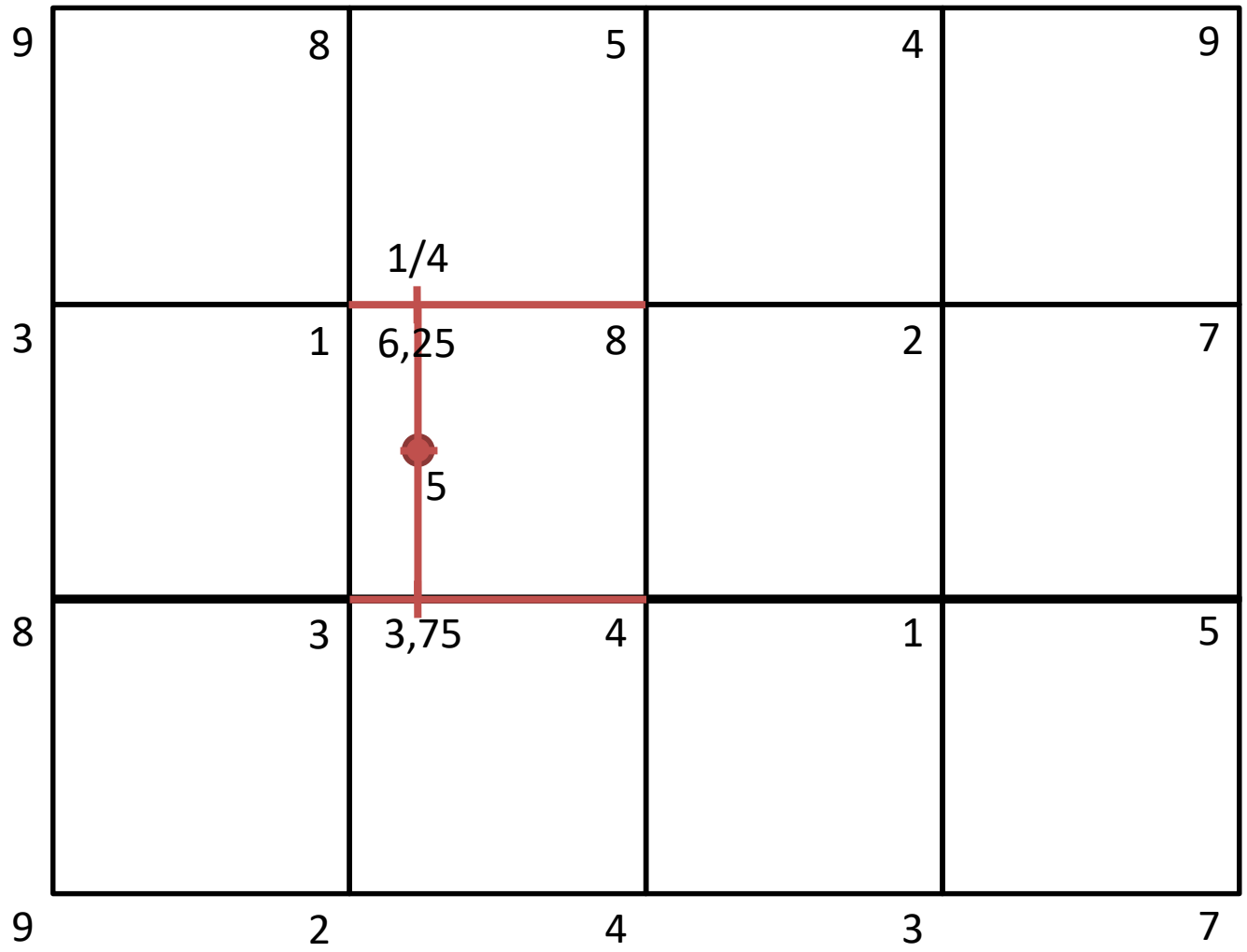
Value Noise

- Random numbers arranged on a grid
- Interpolation between cell corners (x then y)

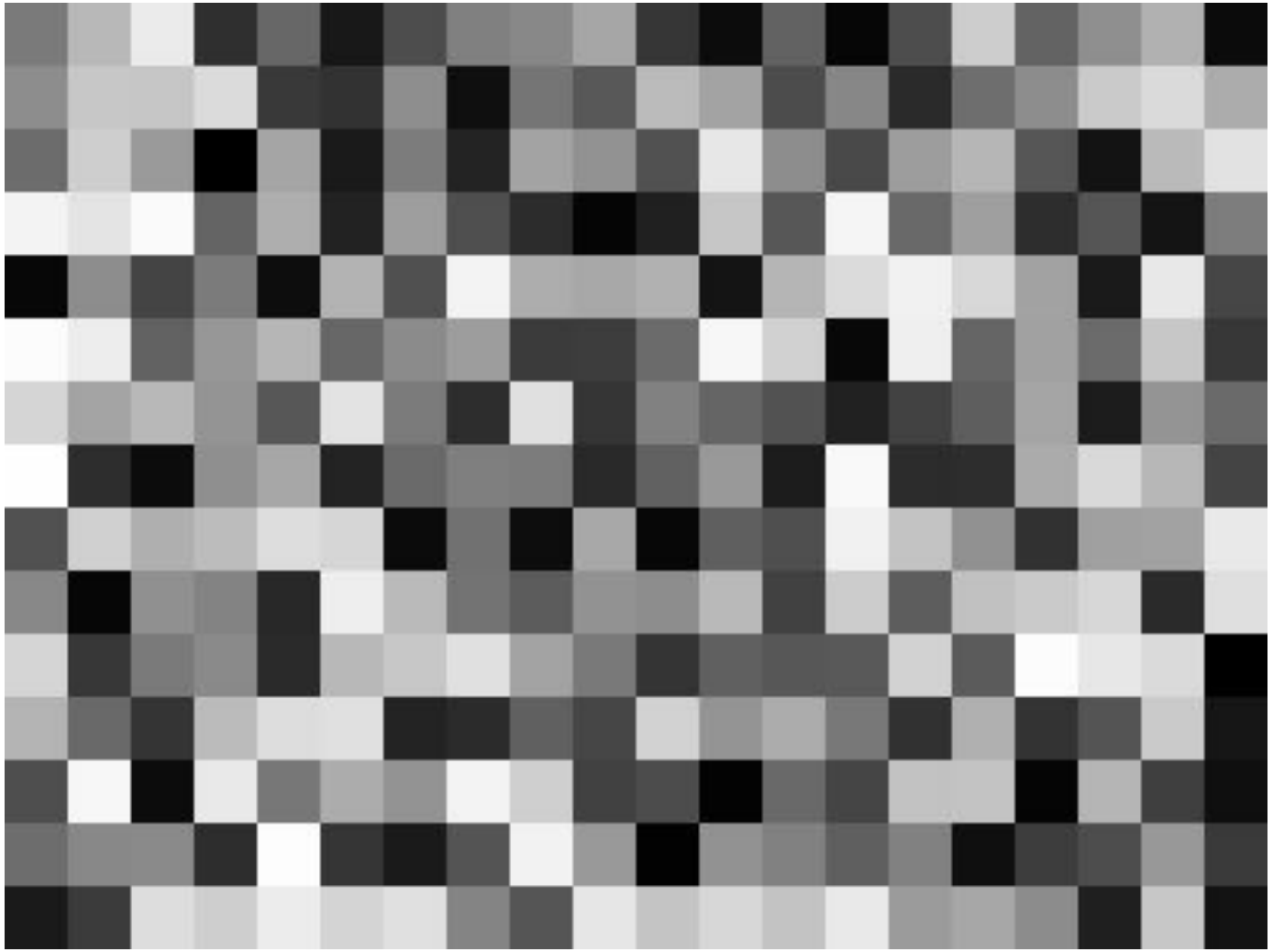

9 8 5 4 9
3 1 8 2 7
8 3 4 1 5
9 2 4 3 7


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


9
8
5
4
9
1/4
3
1
6,25
8
2
7
8
3
3,75
4
1
5
9
2
4
3
7

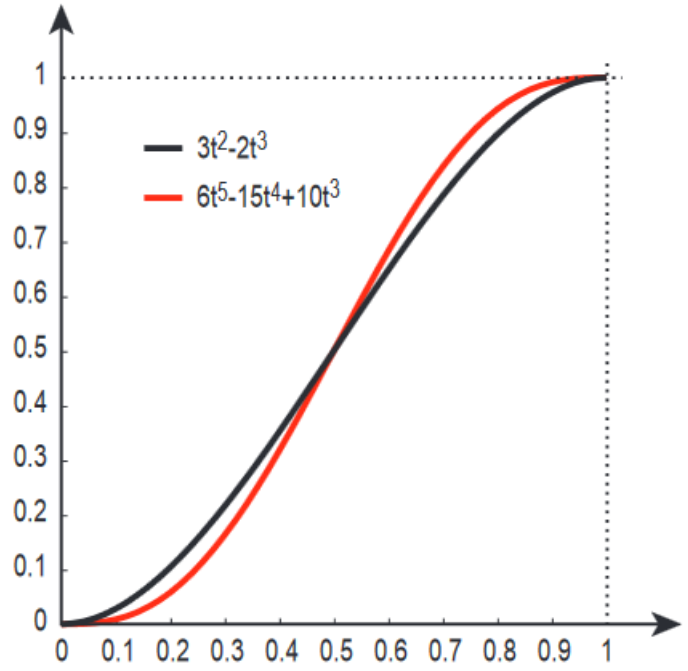

9 8 5 4 9
1/4
3 1 6,25 8 2 7
5
8 3 3,75 4 1 5
9 2 4 3 7

Value noise (cell top left corner)



Interpolation

- Linear :
  - (1-t)*v1 + t*v2, t dans [0,1]
- Hermite smooth step (black)
  - Continuous first order derivative
- X5 smooth step (red) :
  - dérivée seconde également continue



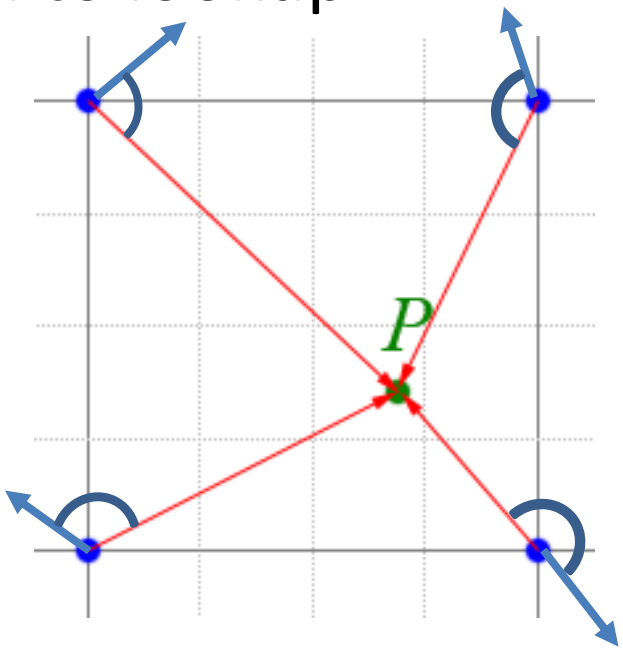
Linear interpolation

Hermite

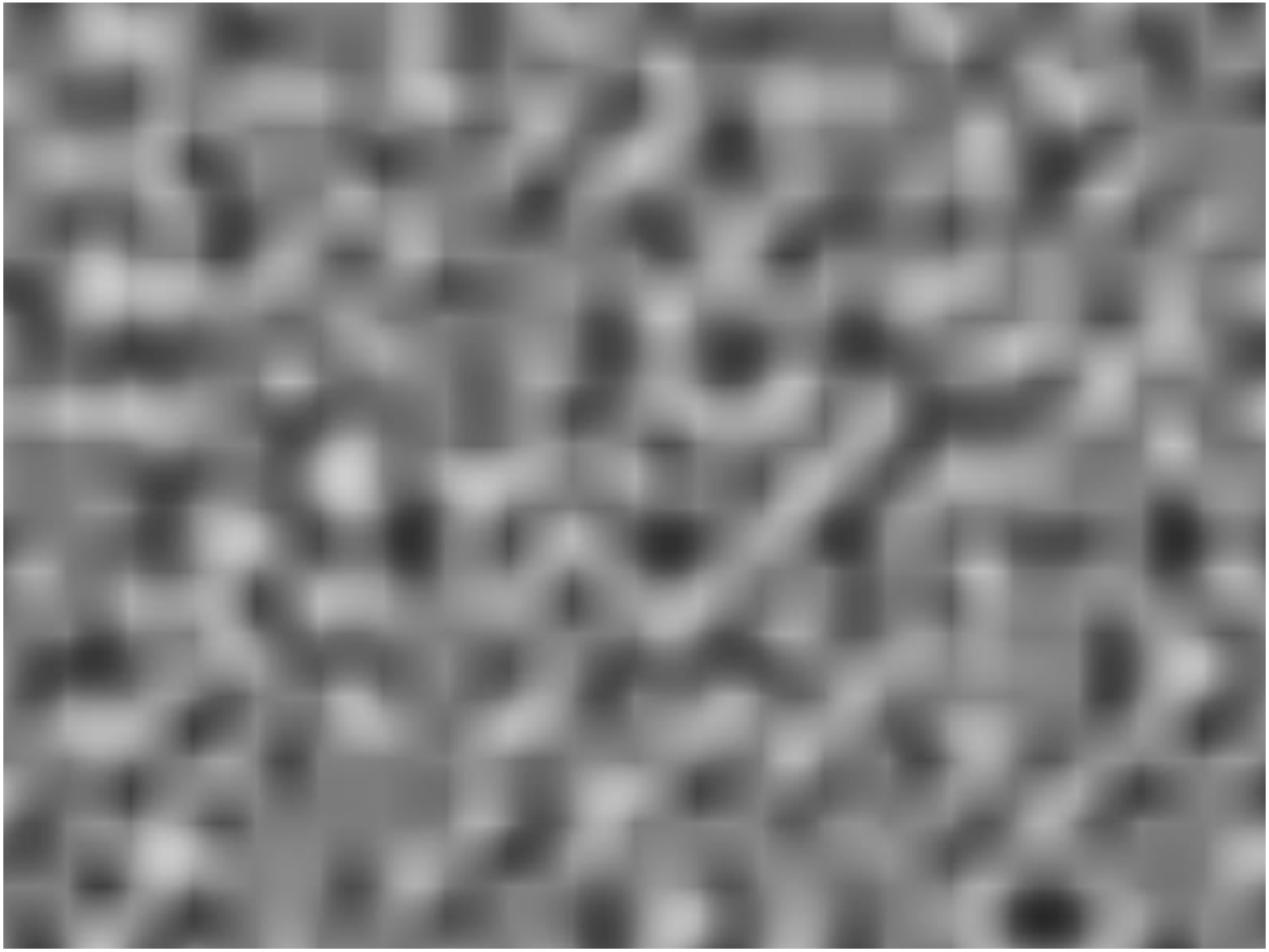
x5

Gradient Noise (Perlin)

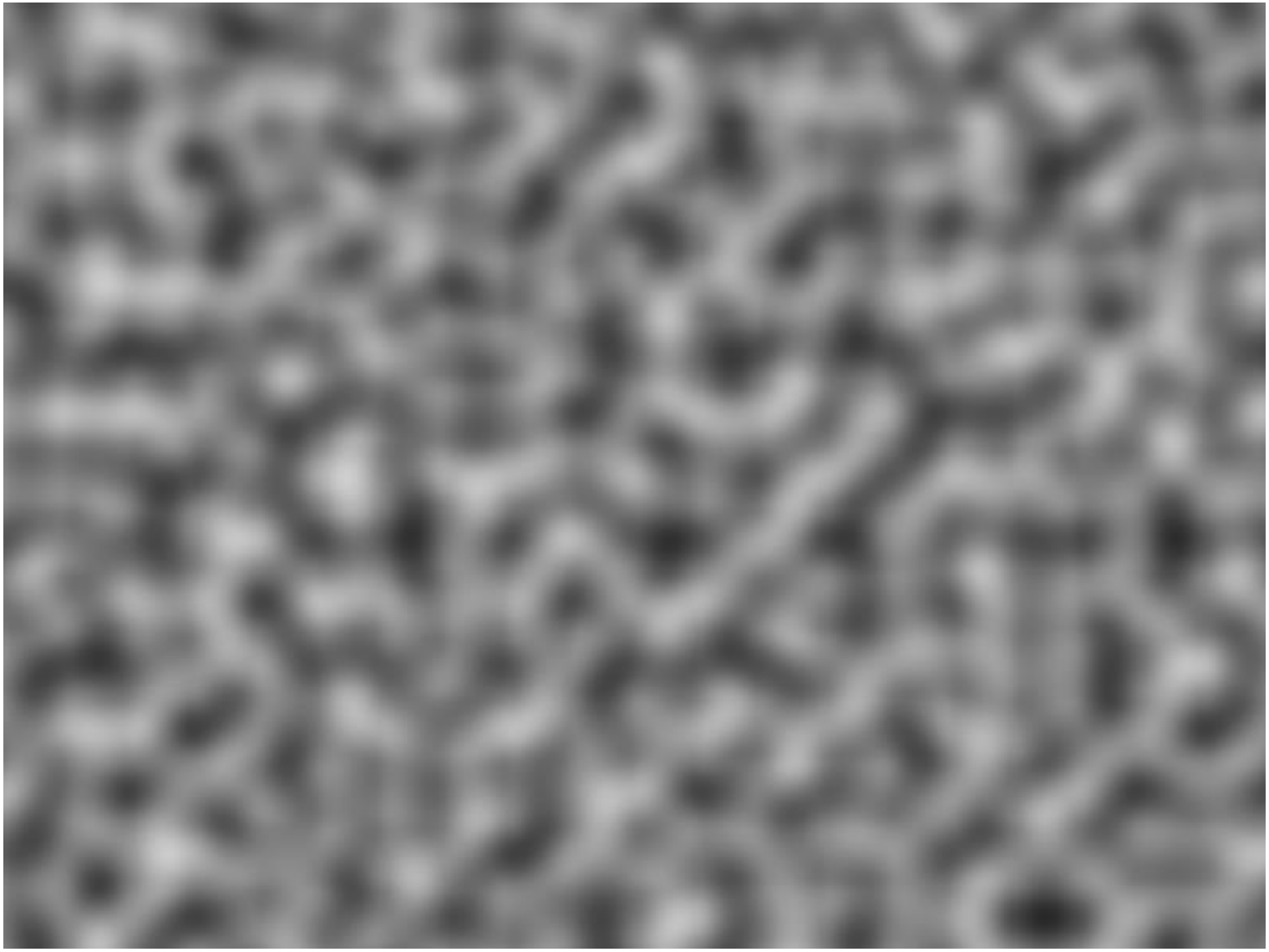
- Each point of the grid is a normalized 2D vector (can be extended to 3D.. etc)
- Compute dot products betwen random direction at each grid point and direction to lookup position
  - Changes with distance (0 at grid point)
  - Changes with angle (0 when perpendicular)
- Then interpolate values like we did with value noise.



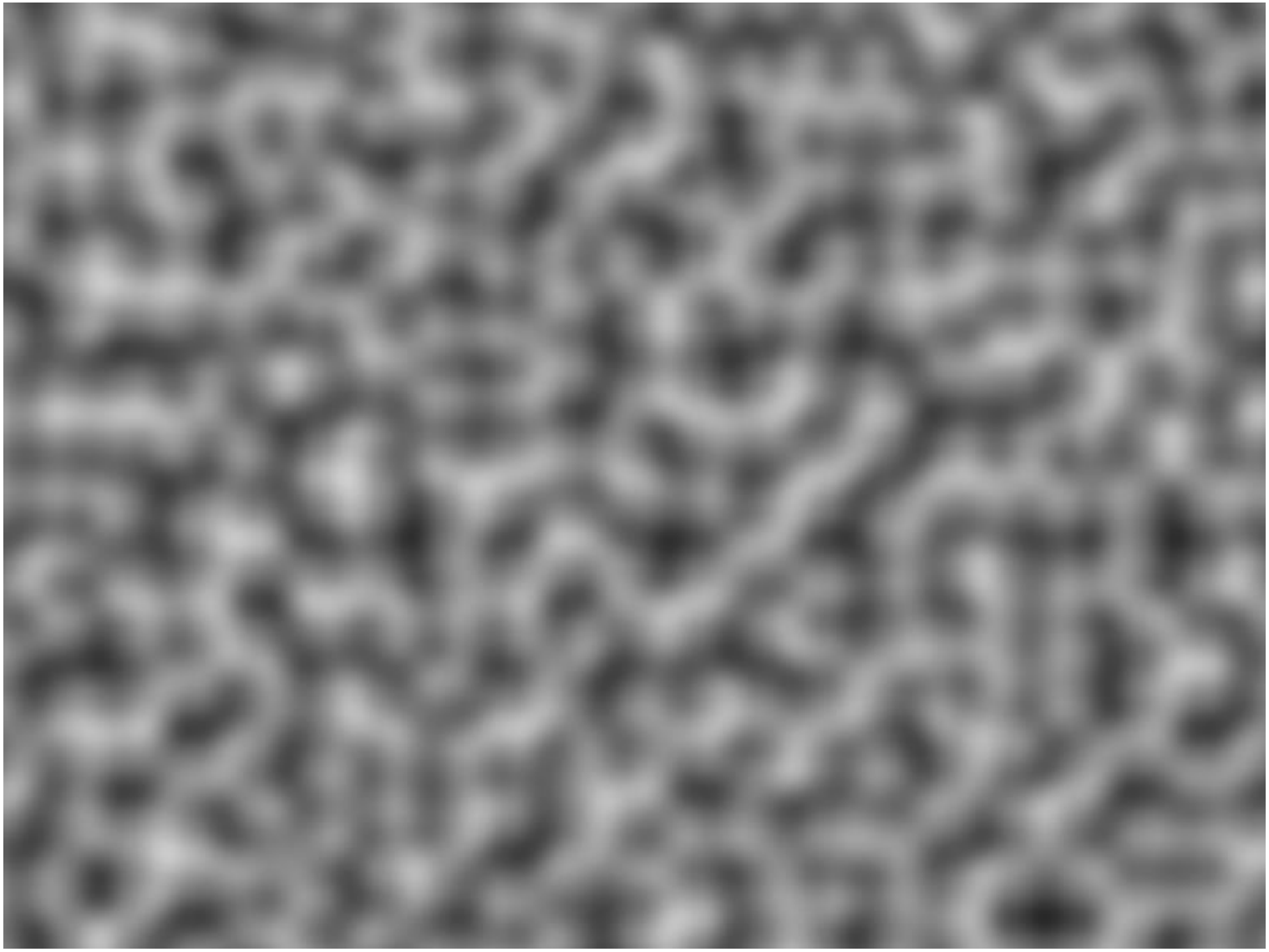
Linear Interpolation



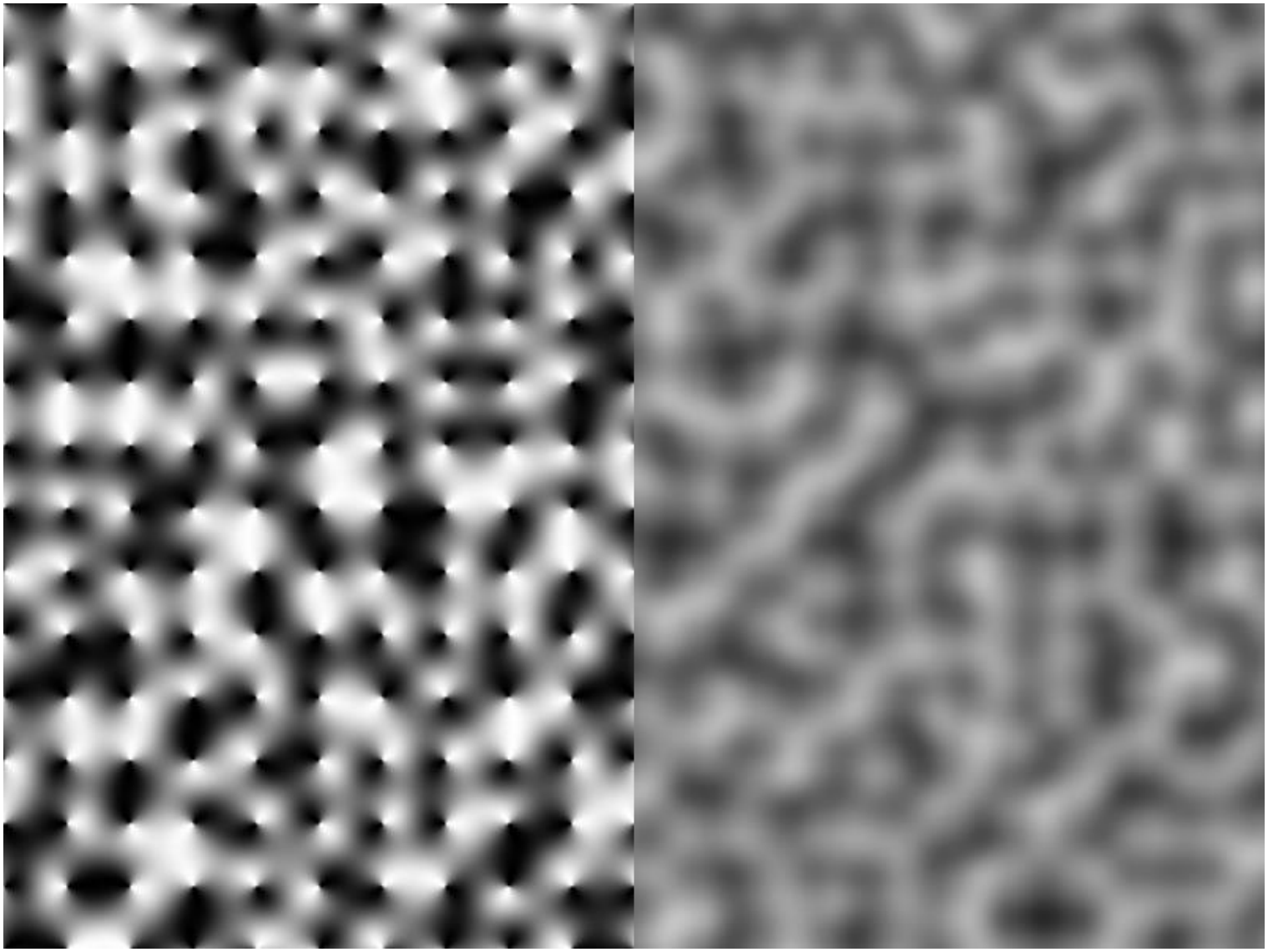
Hermite



x5



Normalized direction

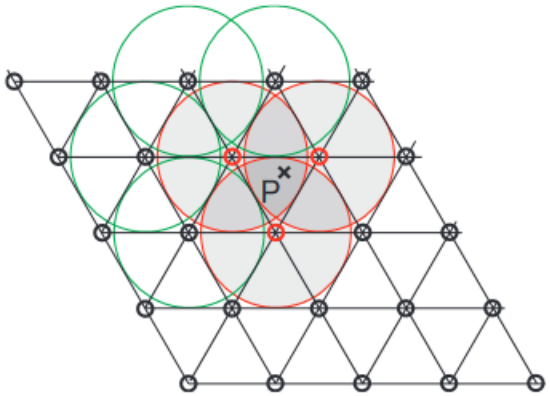


Gradient Noise (Perlin)

- Toujours 0 aux points de la grille
- Quand on part d'un point de la grille, il y a forcément des directions où la valeur varie et une autre ou la valeur est constante.
  - La taille de la grille définit la fréquence du bruit, les gradients l'orientation des variations.

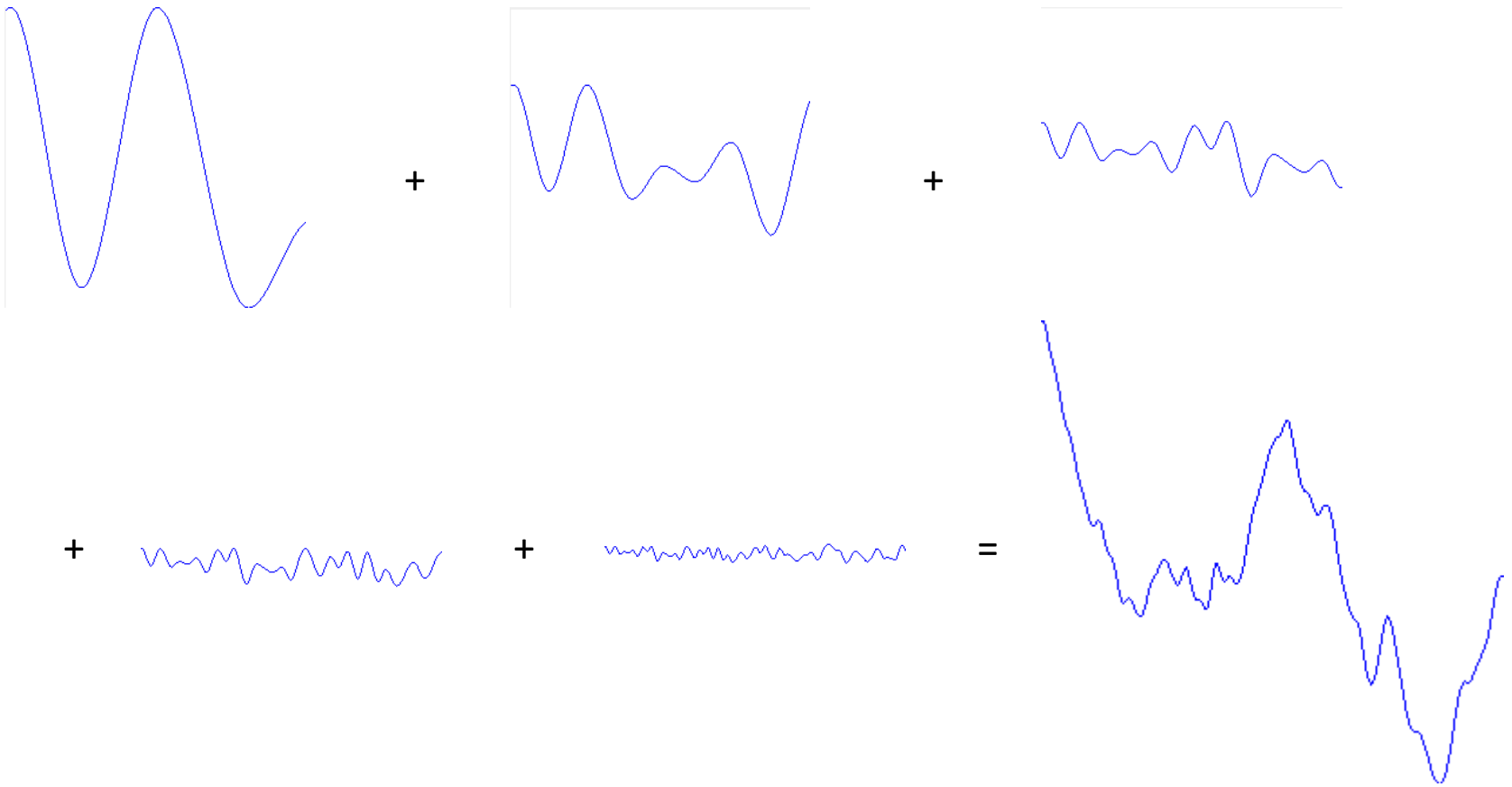
Simplex Noise

- Much harder to explain
- Simplex : simplest nD geometrical object (triangle in 3D)
- Better version of perlin nois. Faster to compute, grandient can be easily obtained (useful for shading for example)



A point P inside a simplex gets contributions to its value only from the three kernels centered on the surrounding corners (shaded, red circles). Kernels at corners farther away (green circles) decay to zero before they cross the boundary to the simplex containing P. Thus, the noise value at each point can always be calculated as a sum of three terms.

Adding multiple frequencies

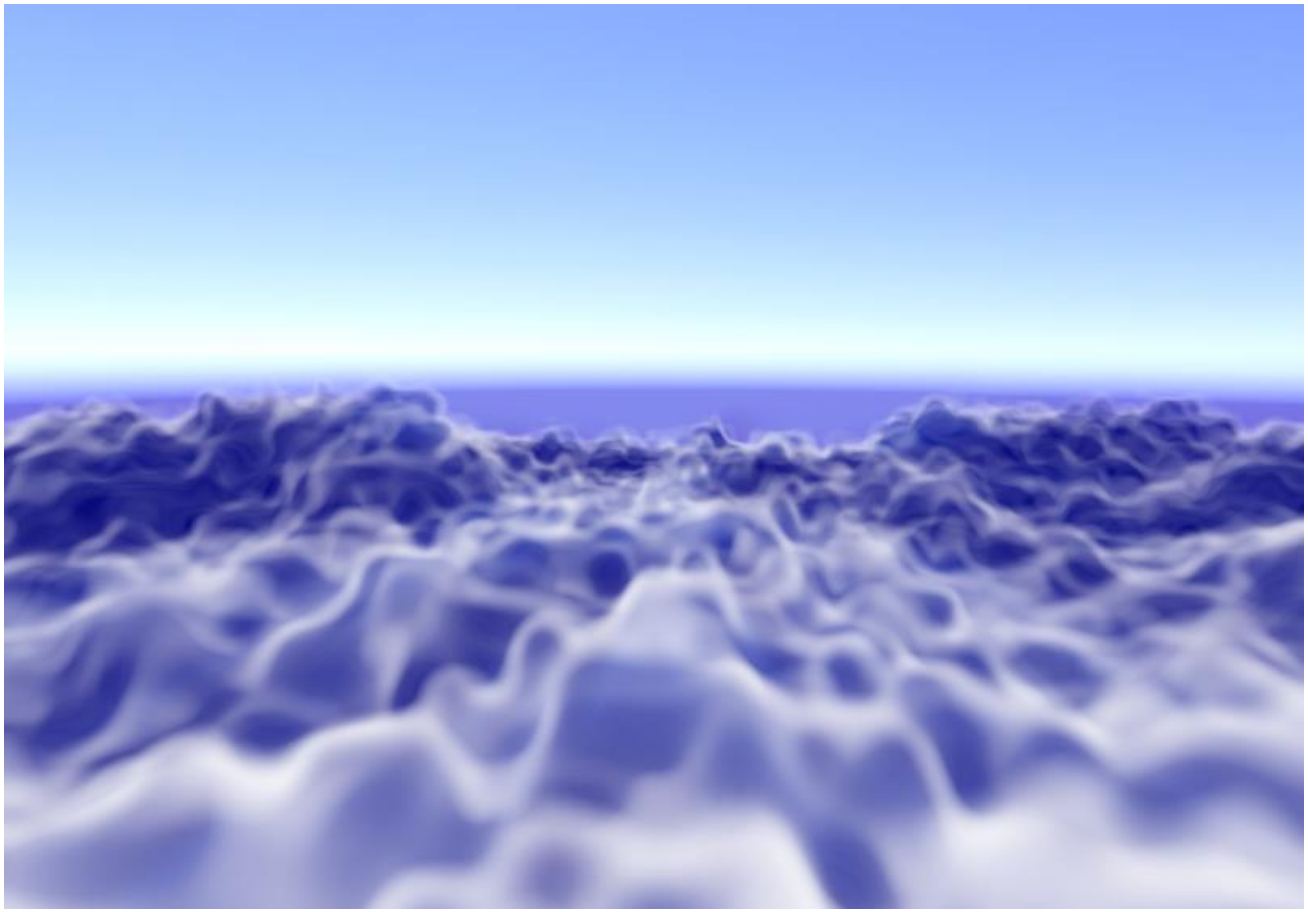


Compute other frequencies

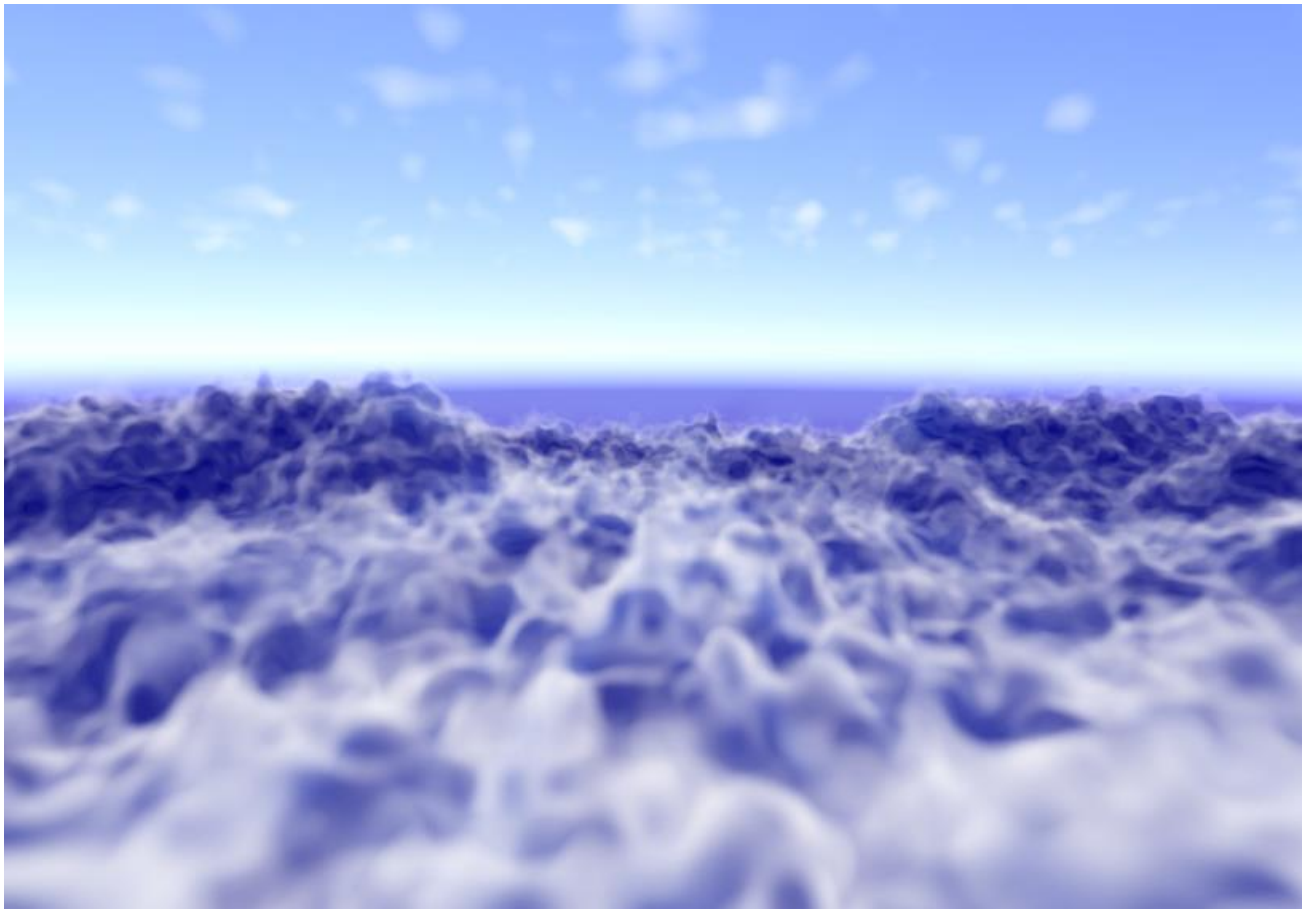
- Multiply input coordinates to change frequency

Raymarched Perlin Noise

Ajout de hautes fréquences

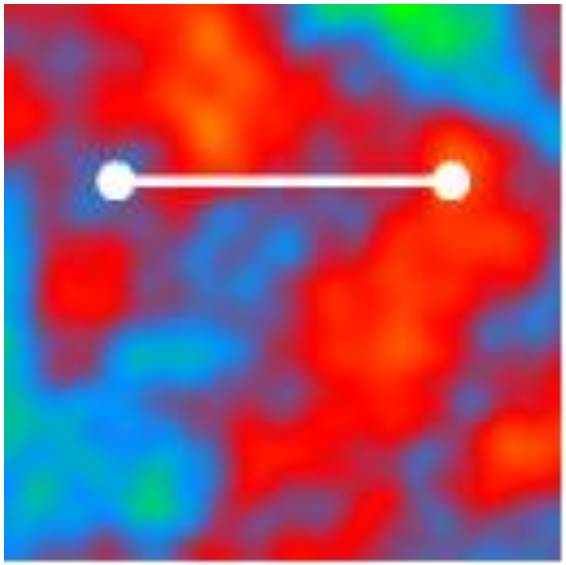


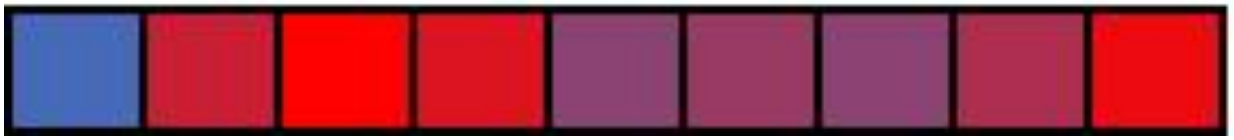
Ajout de hautes fréquences



Ajout de hautes fréquences

Perlin Worm





Modular Approach

High level of control

Templates with local random changes

Modular approach

- Use manually crafted chunks of level
- Chunks can be connected to each other
- Each chunk may have local random features:
  - They should maintain the chunck fundamental gameplay properties (e.g. traversability)
  - They provide some variability

Diablo 3

- Exterior zones :
  - static borders, roads, and town placements.
  - most of the area in-between is static too,
  - chunks of the map in various shapes that are essentially cut out. We then create pieces to fit those shapes and sizes, a bunch of them for each, and the game randomly picks which ones to use.
- Exteriors static for the most part :
  - Having to search to find where the town/questgivers are just wasn't a fun use of randomization.
  - You also couldn't easily or quickly meet a friend anywhere outside of town because nothing was ever in the same place.
  - And lastly it helps us create a sense of a world that actually exists, the towns and cities and at least the important points of interest exist in stationary locations.
- The dungeons, in addition to being random in design and flow, also use the adventure system to bring in random events and quests.

Diablo 3


DIABLOWIKI.NET+


DIABLOWIKI.NET+
DIABLOWIKI.NET+
DIABLOWIKI.NET+
DIABLOWIKI.NET+

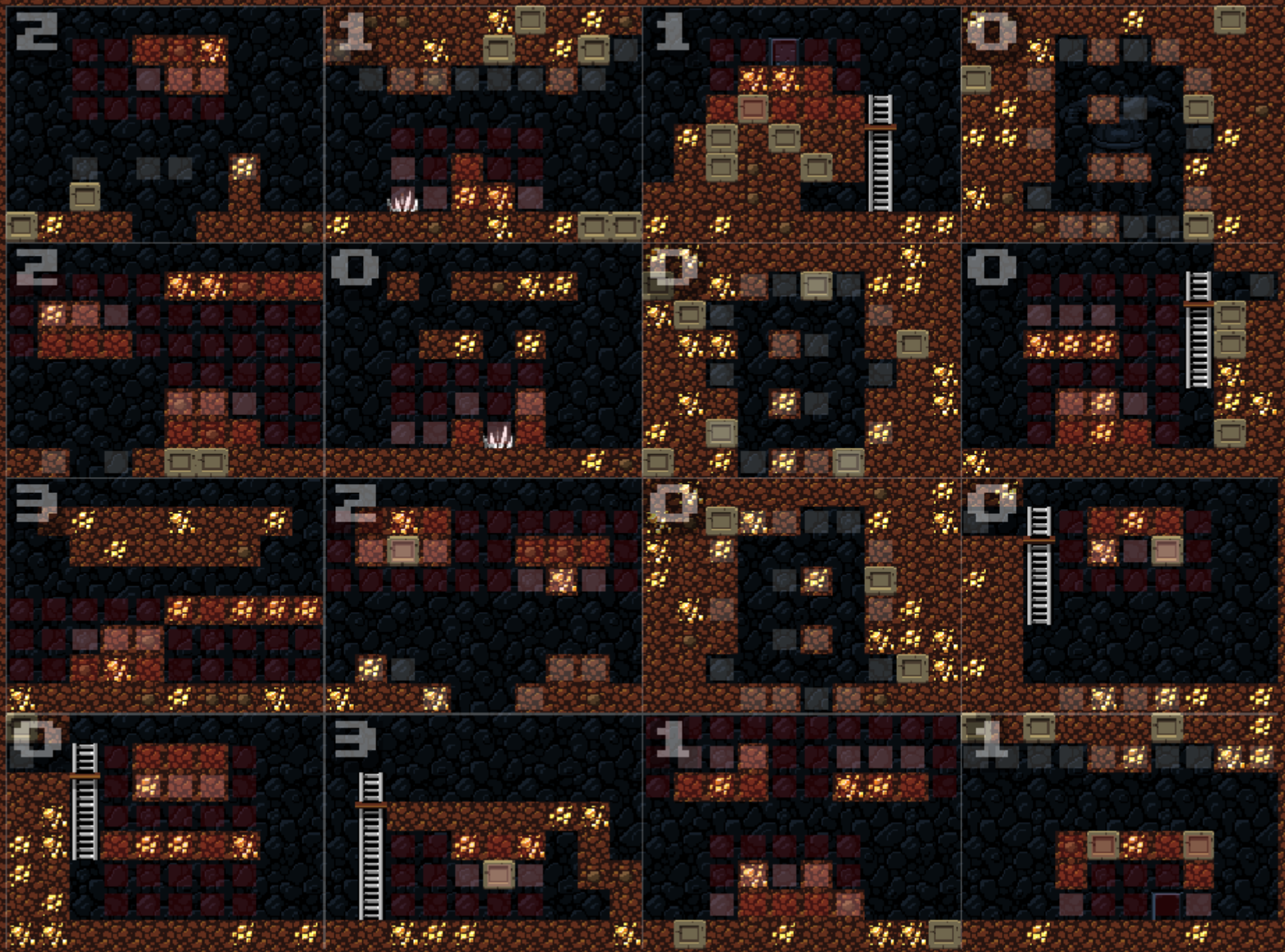
Spelunky

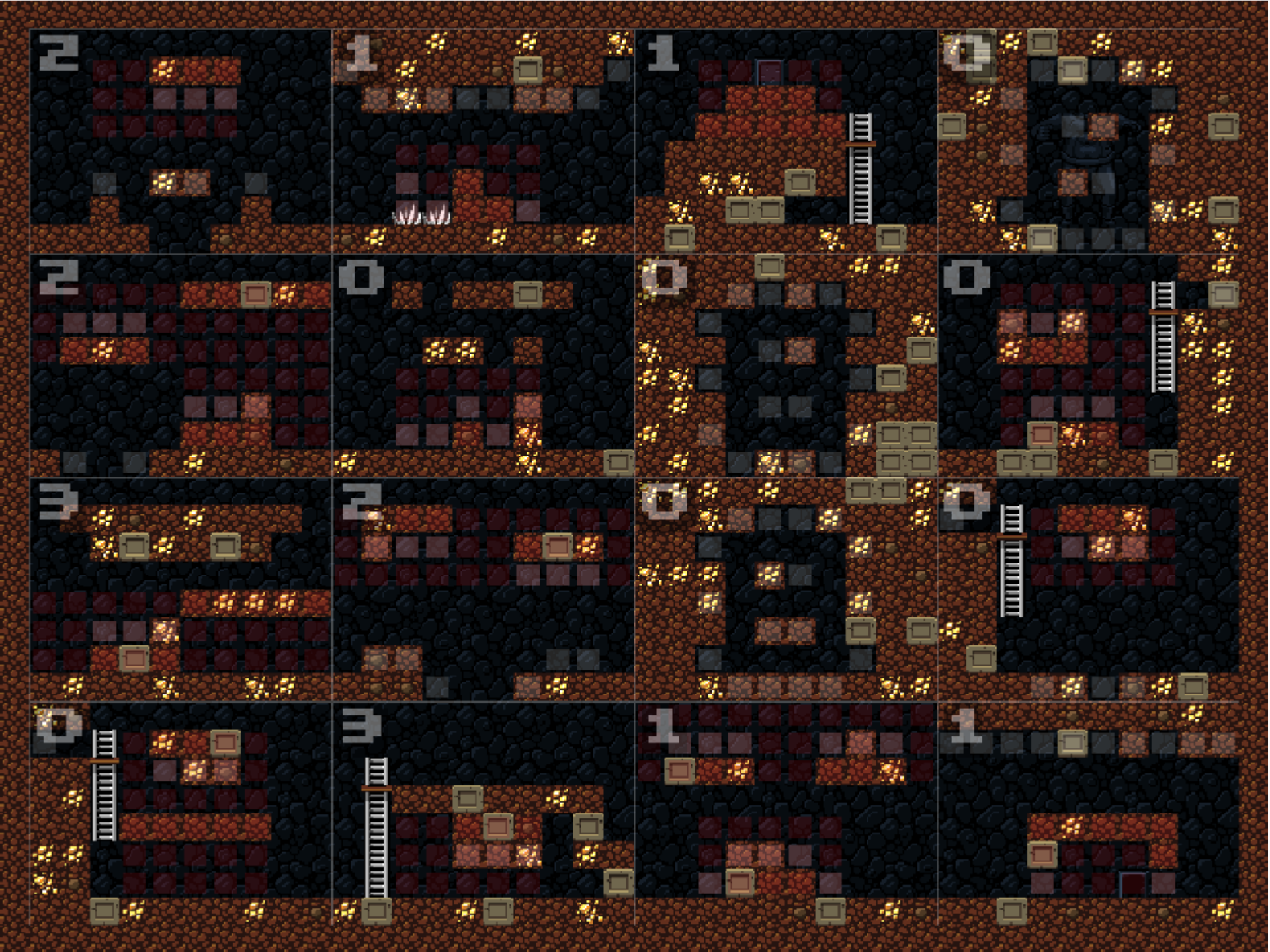
Spelunky

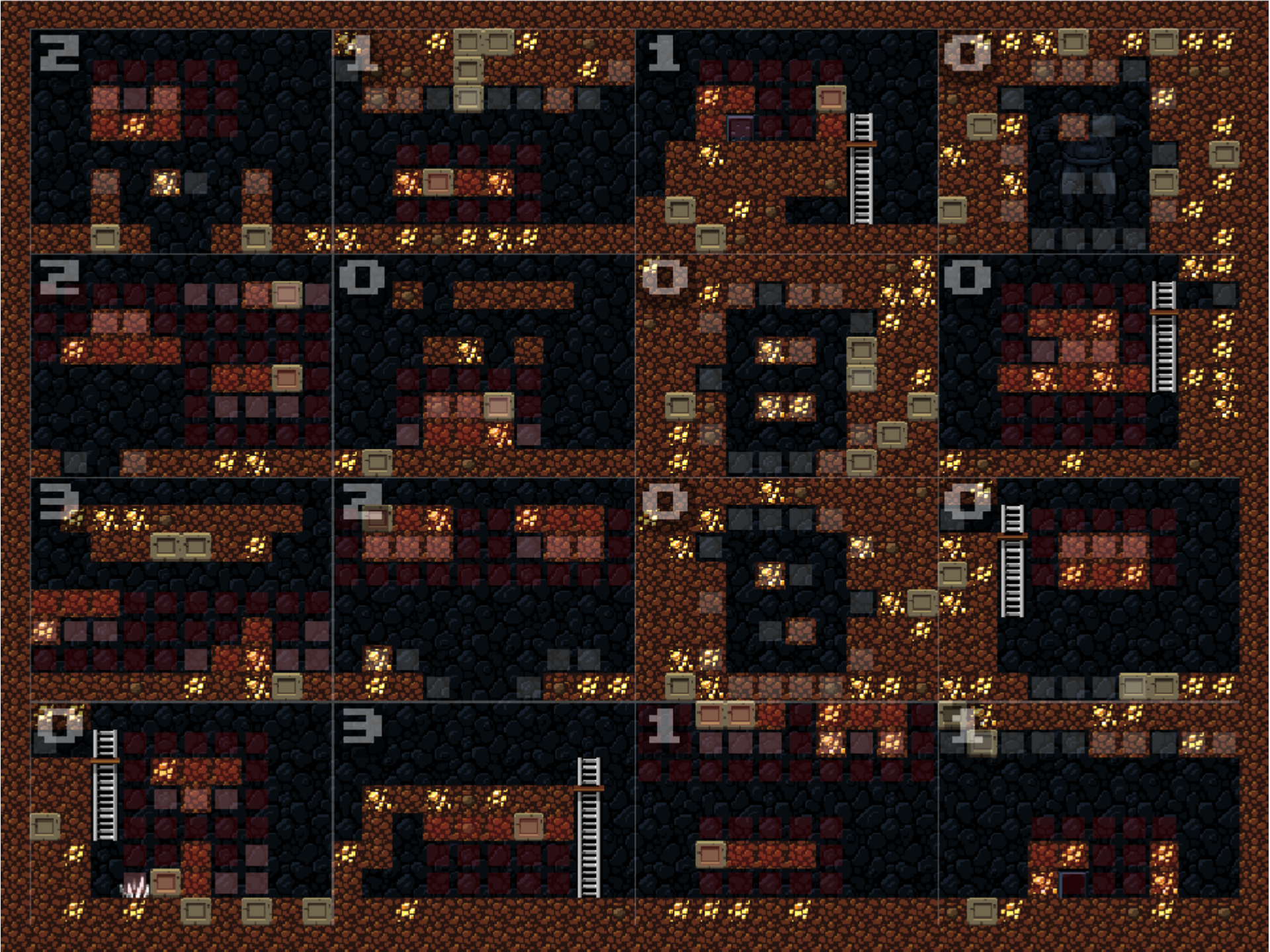
- 16 rooms (4*4)
- Room types:
  - 0: various, not on solution path
  - 1: left, right exit
  - 2: left, right, bottom exit
  - 3: left, right, top exit

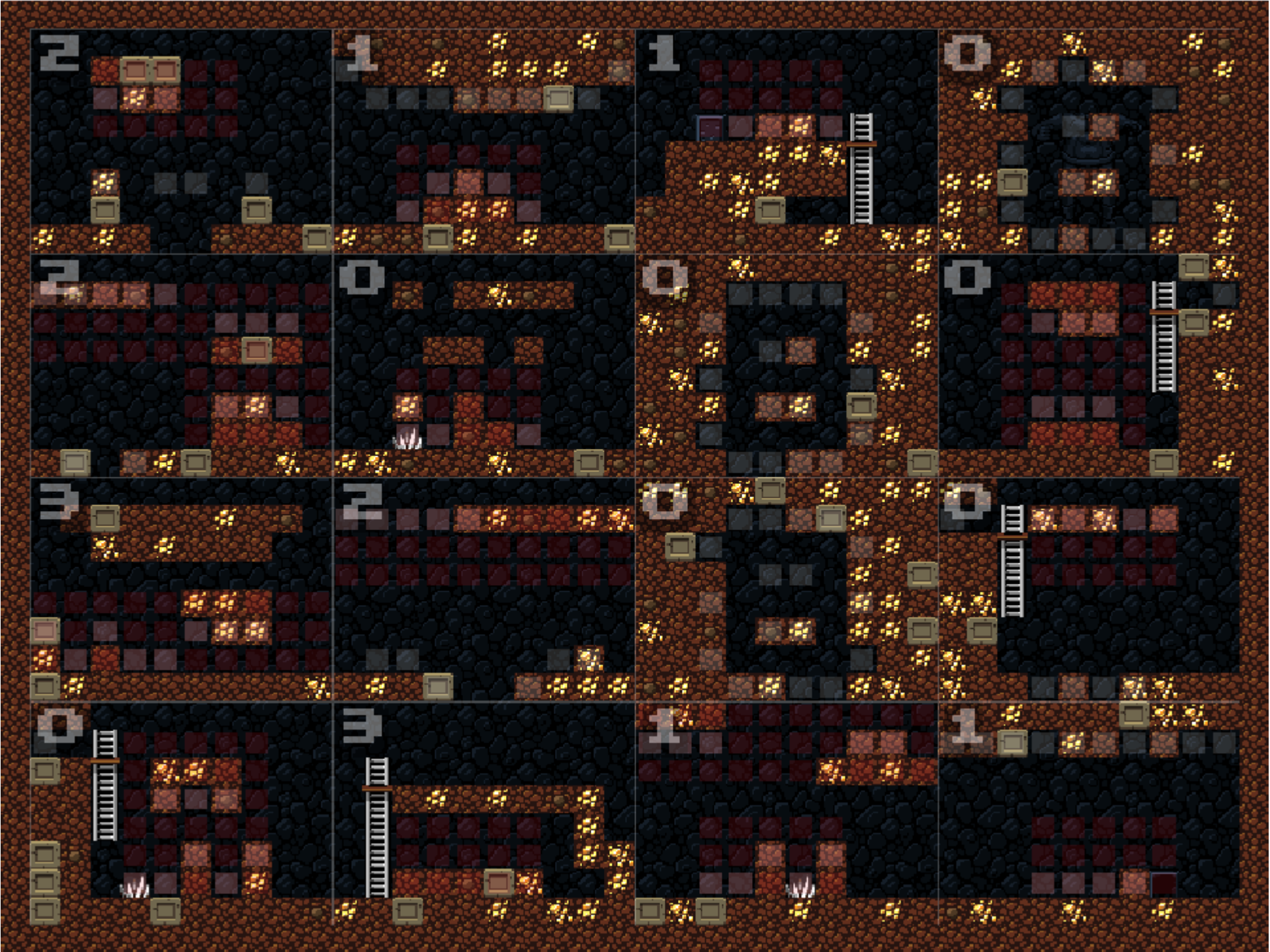
Spelunky algorithm (en gros)

- Start room at random position, top row.
  - Select direction randomly
  - Select room type to stay connected to previous room
  - If bottom row, end and set blank rooms to 0 type.
- Each room has a template :
  - Fixed elements to guarantee solution
  - Random elements for variability

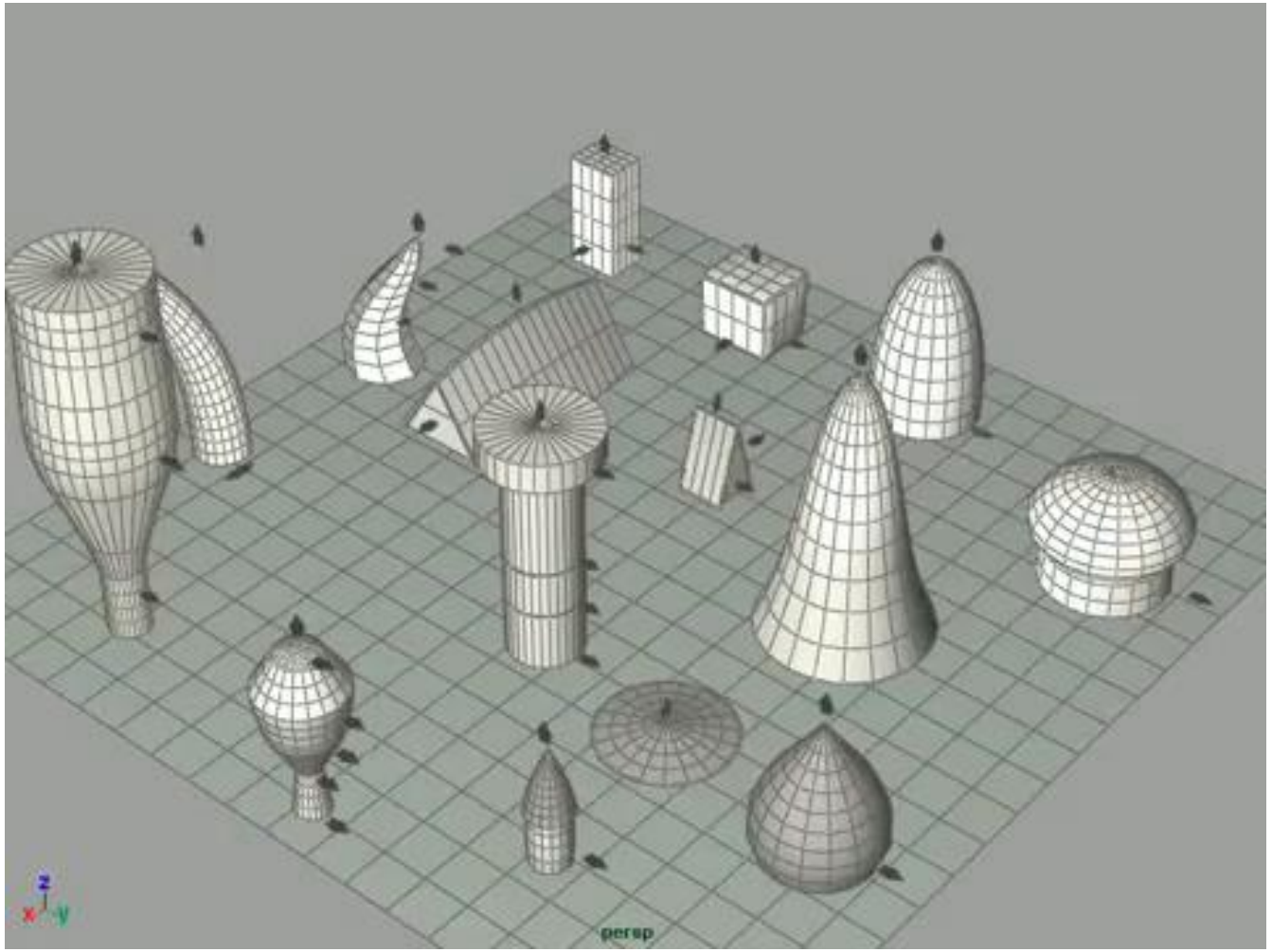

2
1
1
0
2
0
0
0
3
2
0
0
0
3
1
1


2
1
1
0
2
0
0
0
3
2
0
0
0
3
1
1

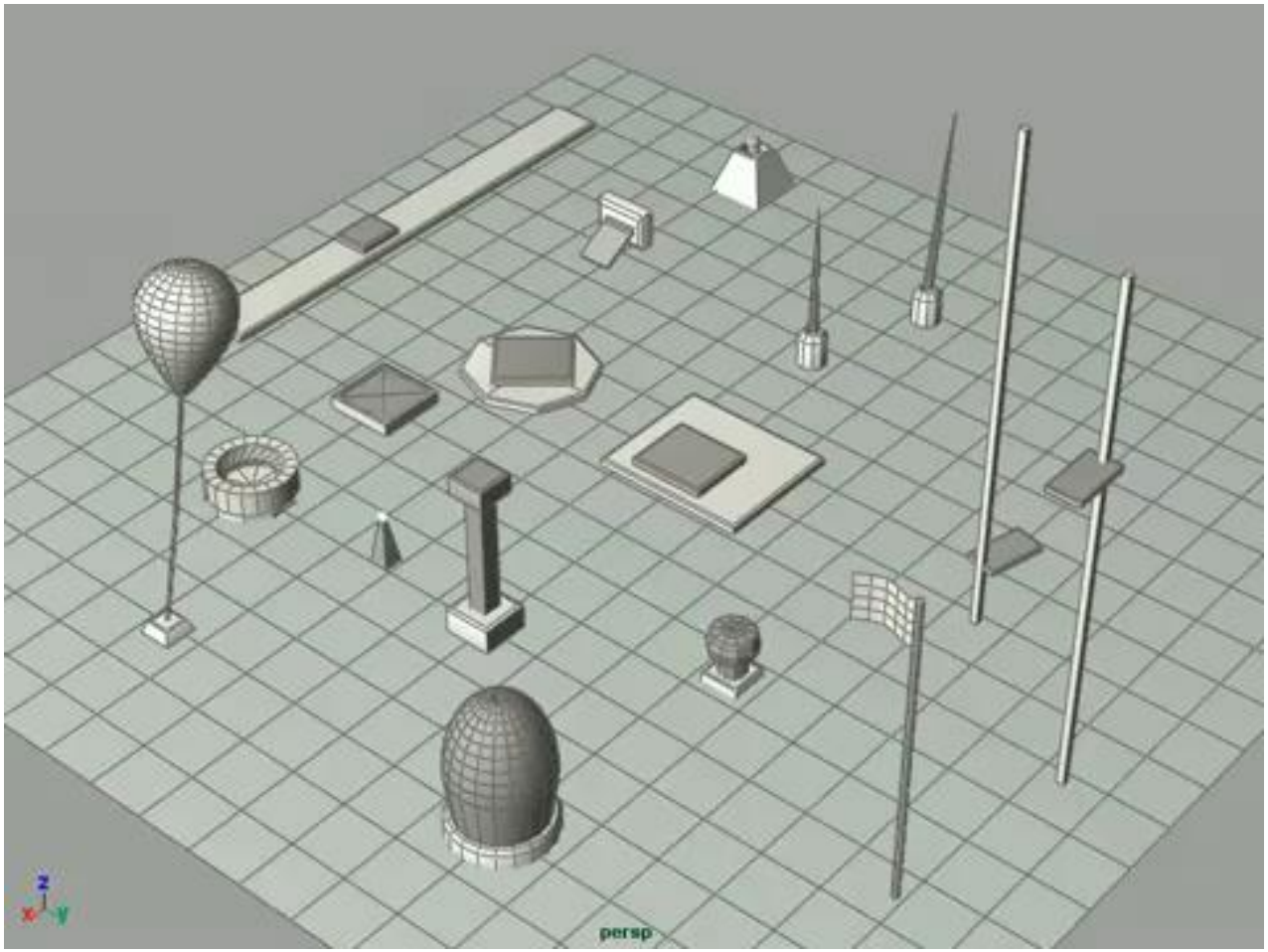



2
1
1
0
2
0
0
0
3
2
0
0
0
3
1
1

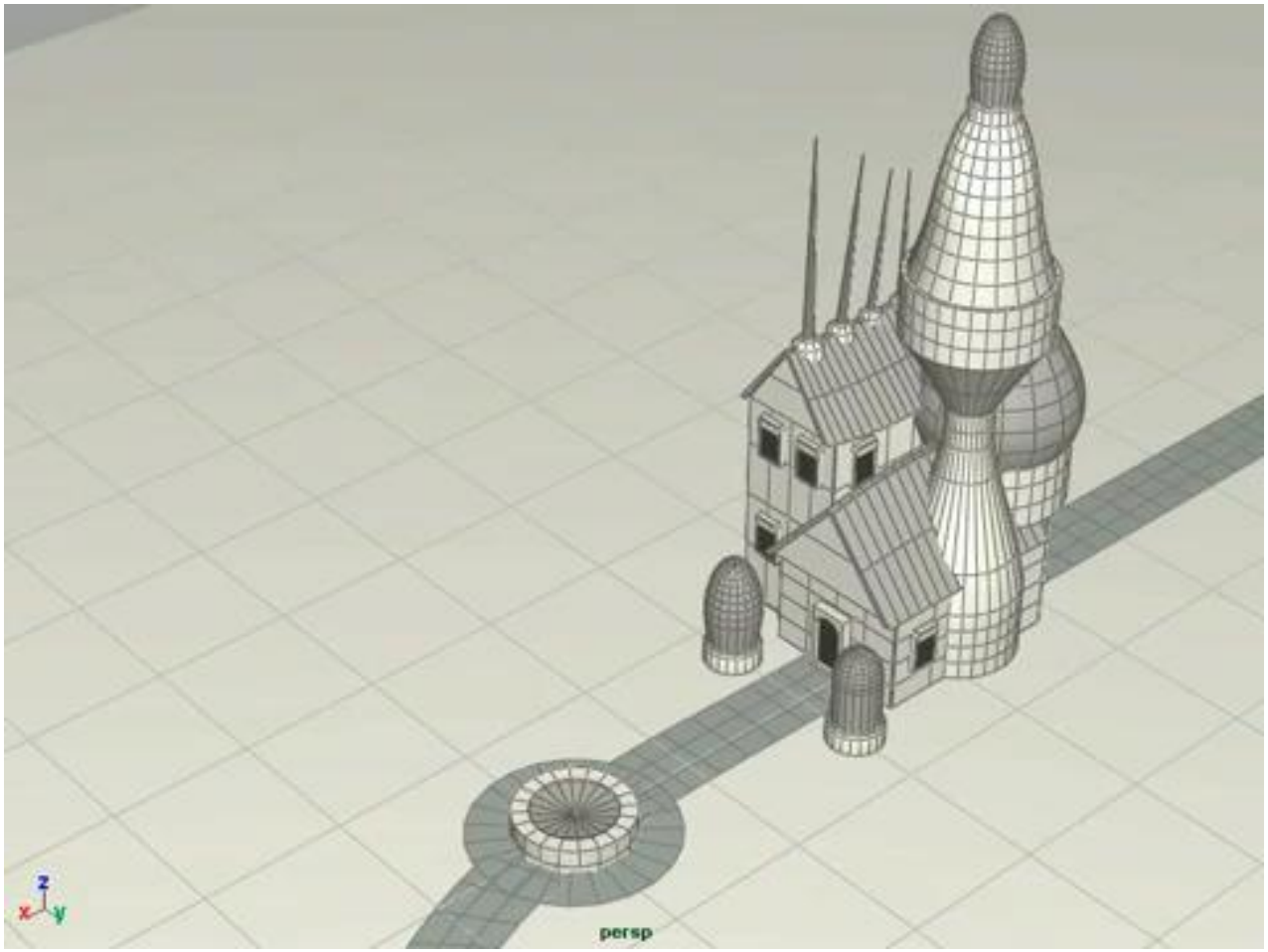
Spore



Spore



Spore



Grammars

Describe generation rules

Maintain control over the structure

Generative Grammar

- Symbols
- Rewriting rules

1) dungeon → obstacle + treasure;
2) obstacle → key + obstacle + lock + obstacle;
3) obstacle → monster + obstacle;
4) obstacle → room.

- -> Always respects the rules structure (always « valid », helpful to encode design pricniples)

[dormans2011]

Mission et Game Space

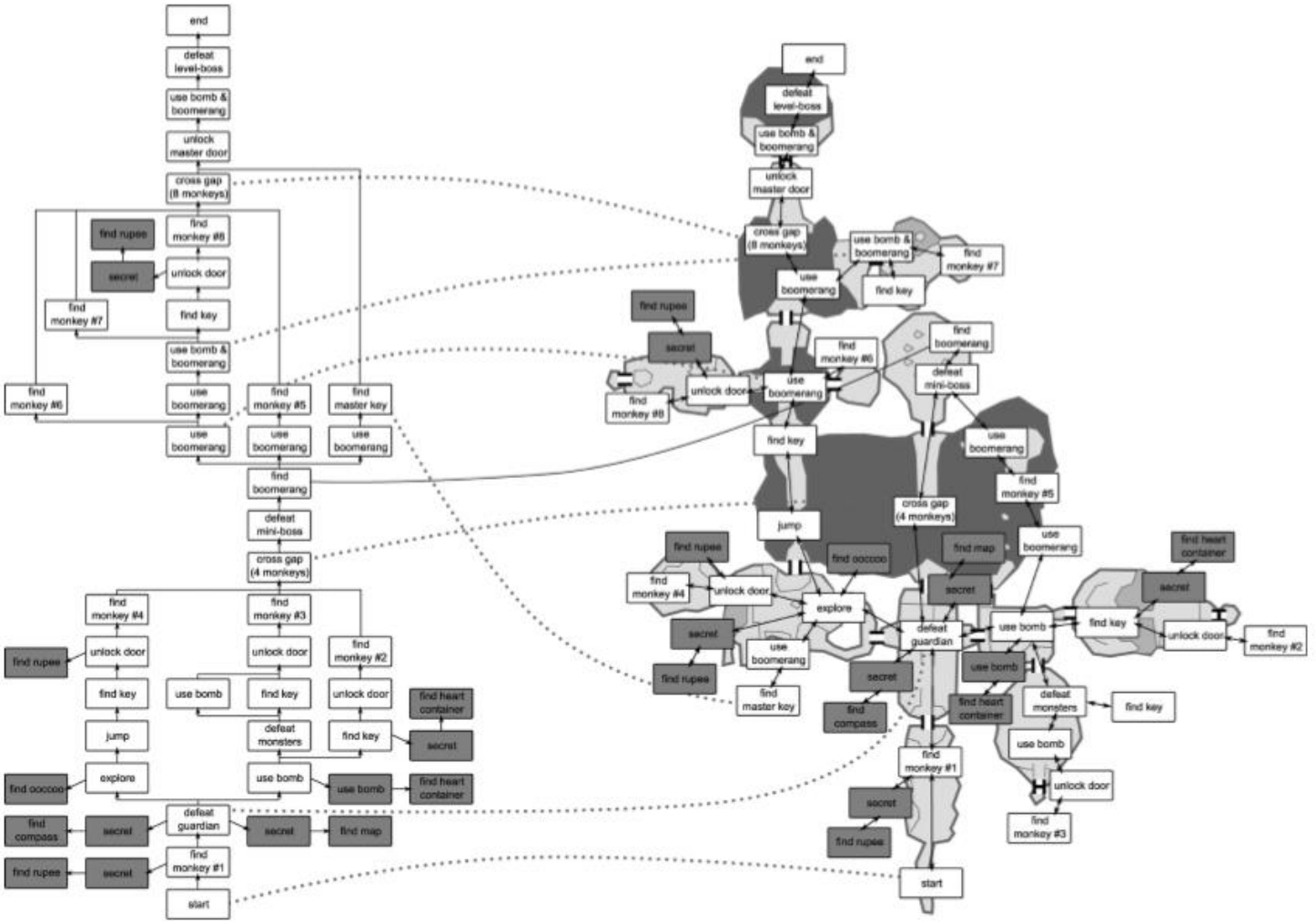


Fig. 1. Mission and space in the Forest Temple Level of *The Legend of Zelda: The Twilight Princess* [21].

Graph Grammar

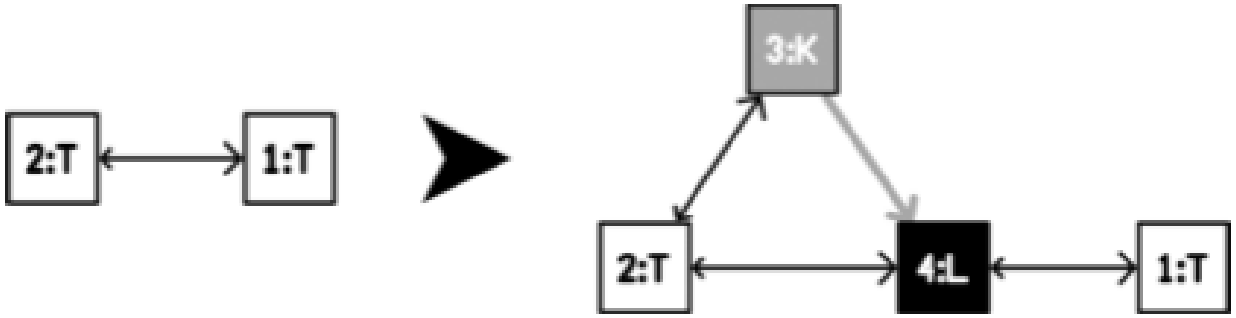


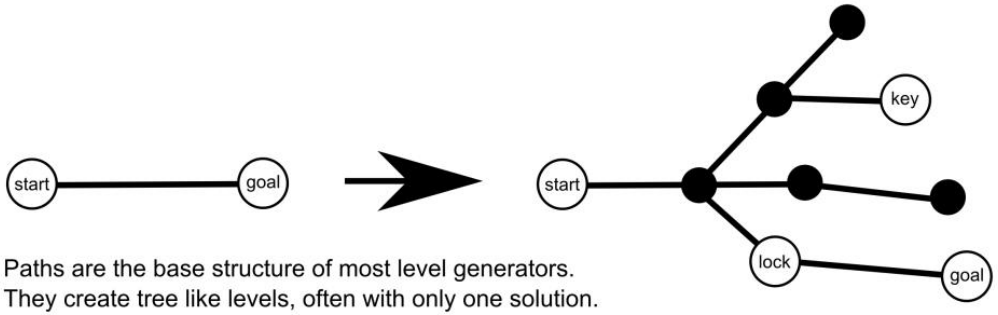
Fig. 7. Rule to add a key and lock. The colors of the nodes are only used to help identify the node types, the gray lines indicate which key opens what lock.

[dormans2011]

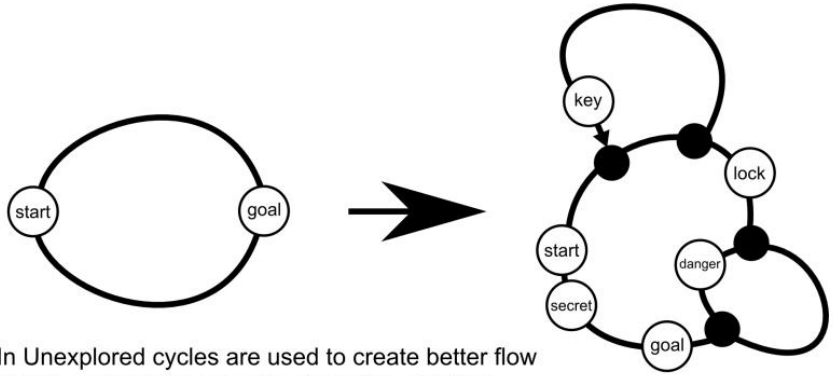
Unexplored

[dormans2016]

Unexplored



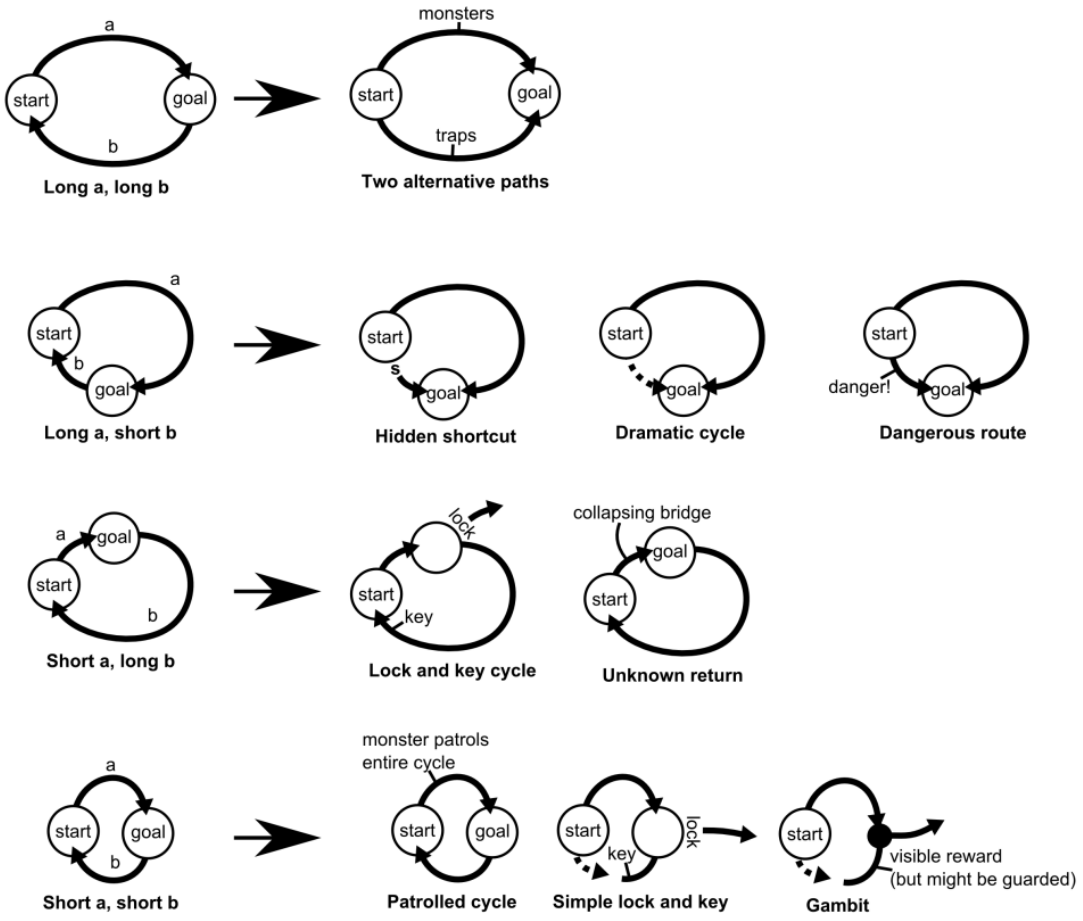
Paths are the base structure of most level generators.
They create tree like levels, often with only one solution.



In Unexplored cycles are used to create better flow
and to present players with alternative solutions.

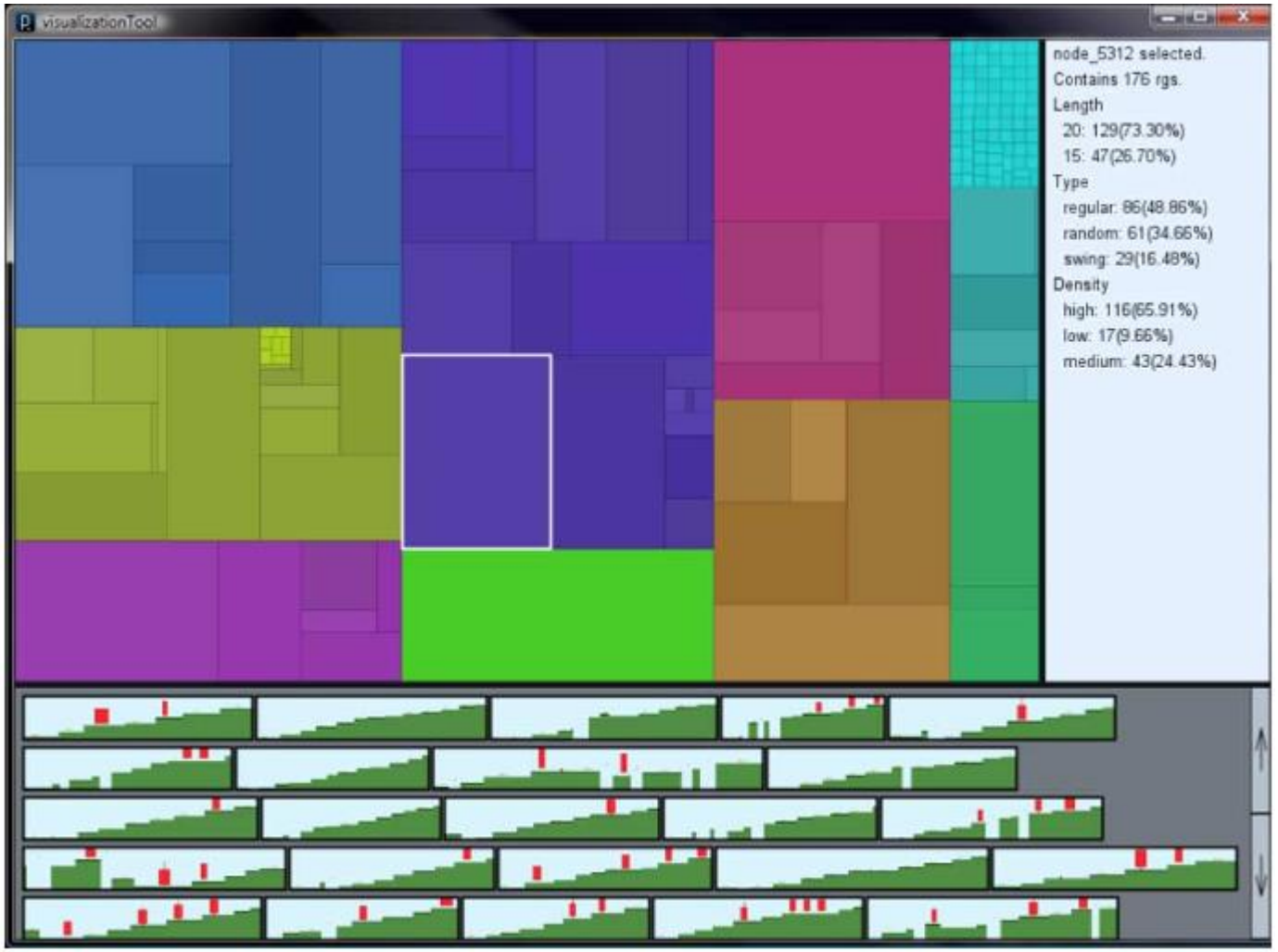
[dormans2011]

Unexplored



[dormans2011]

Launchpad



[Smith2009 – 2013]

Launchpad

- Generate the level's rythm from a chosen structure and density
  - Regular (regular beat)
  - Swing (short then long)
  - Random
- Rythm is then translated to a playable level using a grammar
- Use a bunch of scoring functions to select the best level

[Smith2009 – 2013]

Rythm examples

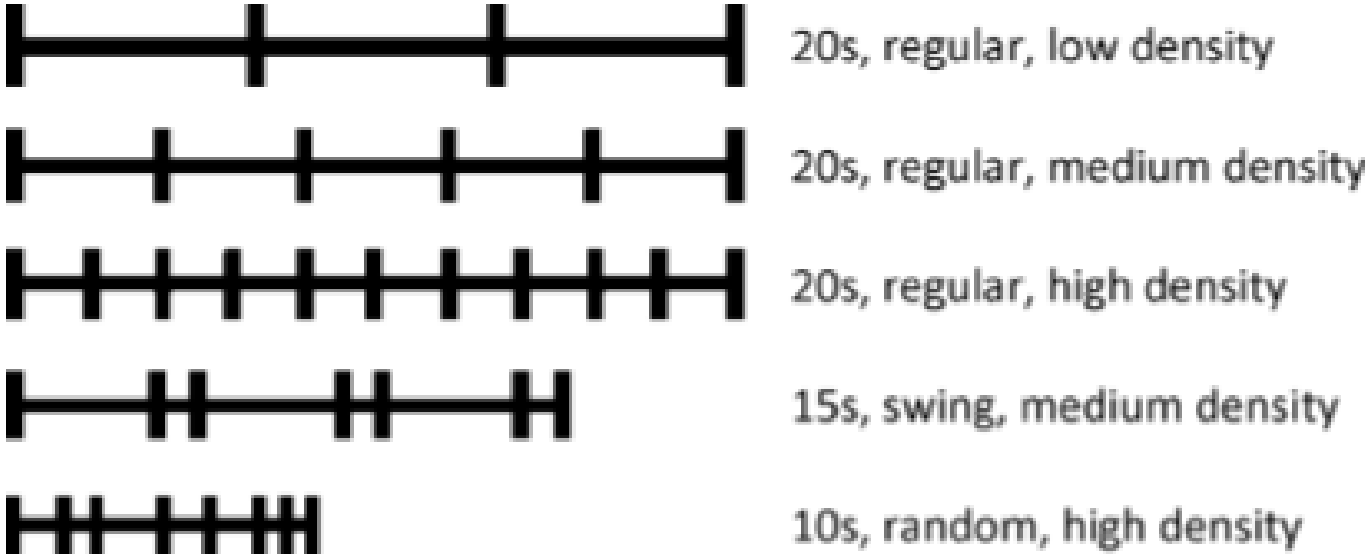


Fig. 7. Example rhythms. These timelines show the effects of varying the length, type, and density of a rhythm. Lines indicate the length of the rhythm, and hatch marks indicate the times at which an action will begin.

[Smith2009 – 2013]

Example

```
move 0 4
jump 2 2.25
move 6 10
jump 6 6.25
jump 8 8.75
```

[Smith2009 – 2013]

From rythm to geometry

Moving → Sloped | flat_platform

Sloped → Steep | Gradual

Steep → steep_slope_up | steep_slope_down

Gradual → gradual_slope_up | gradual_slope_down

Jumping → flat_gap
| (gap | no_gap) (jump_up | Down | spring | fall)
| enemy_kill
| enemy_avoid

Down → jump_down_short | jump_down_medium | jump_down_long

Waiting-Moving → stomper

Waiting-Moving-Waiting -> moving_platform_vert
| moving_platform_horiz

[Smith2009 – 2013]

move	0.00	8.00
jump	2.00	2.25
jump	4.00	4.24
jump	6.00	6.25
move	10.00	12.00
move	14.00	20.00
jump	16.00	16.25
jump	18.00	18.25

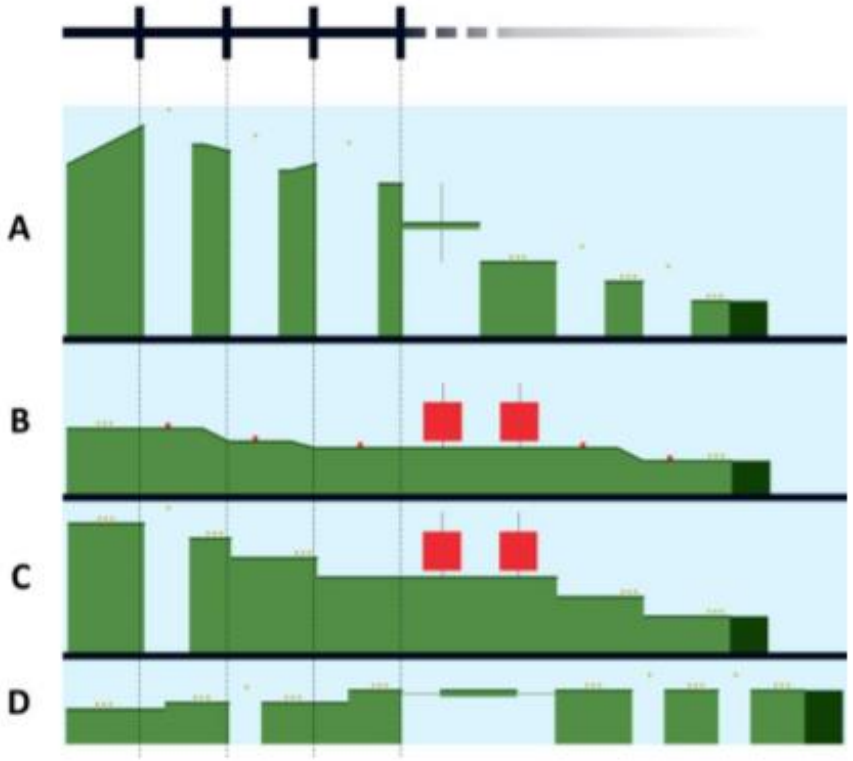
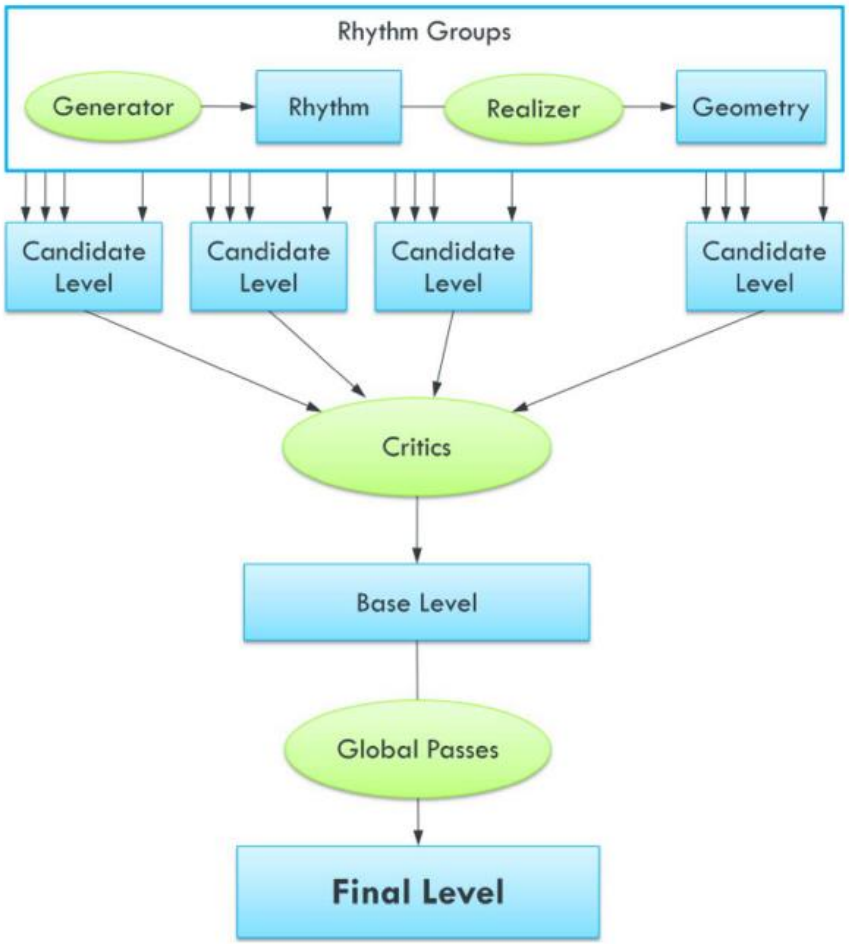


Fig. 11. Geometry interpretations of a rhythm. This figure shows four different geometric interpretations of the provided rhythm. Small red boxes denote enemies to kill, large red boxes are stompers that follow the associated line, and platforms on green lines are moving platforms that follow that path. The large, dark green platform at the end of the rhythm group is the joiner from this rhythm group to the next.

[Smith2009 – 2013]

Whole generator



[Smith2009 – 2013]

Metrics

- Line critic :

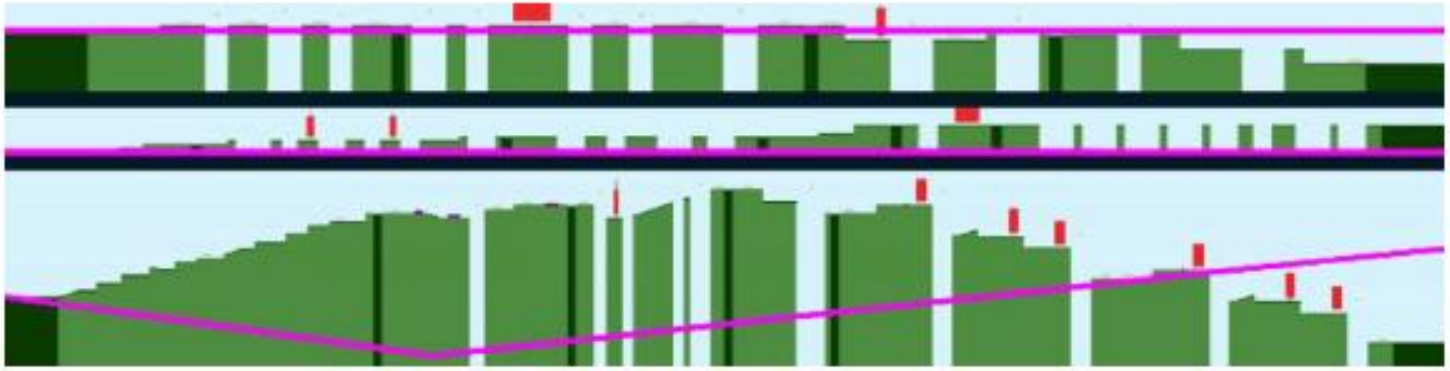


Fig. 12. Line critic scores. Examples of levels that are different distances from a specified control line (shown in pink). The top level has a distance measure of 1.75, the middle a distance measure of 5.08, and the bottom a distance measure of 23.06.

[Smith2009 – 2013]

Metrics

- Component Frequency Critic:
  - Jump and wait frequencies were defined at the beginning.
  - The generated level may not respect these frequencies,
  - Evaluate the distance between actual and wanted frequencies.

[Smith2009 – 2013]

Metrics

- Linéarity :
  - Different from line critic. Evaluates if the level can be fitted to a line

[Smith2009 – 2013]

Metrics

- Leniency (difficulty) :

-1.0	gaps enemies falls
-0.5	springs, stompers
$+0.5$	moving platforms
$+1.0$	jumps with no gap associated.

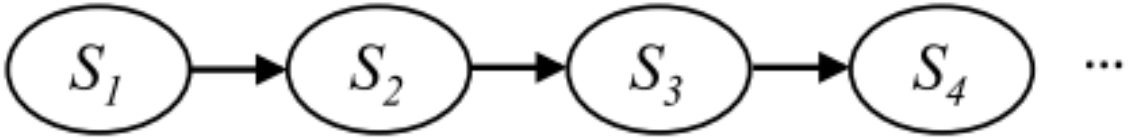
[Smith2009 – 2013]

Markov chains

Extract a probabilist structure

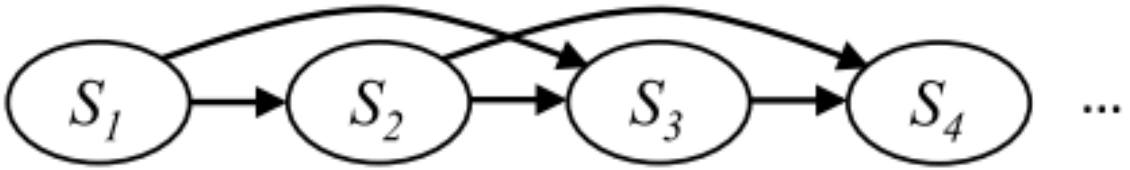
Markov Chain

a) Order 1 Markov Chain

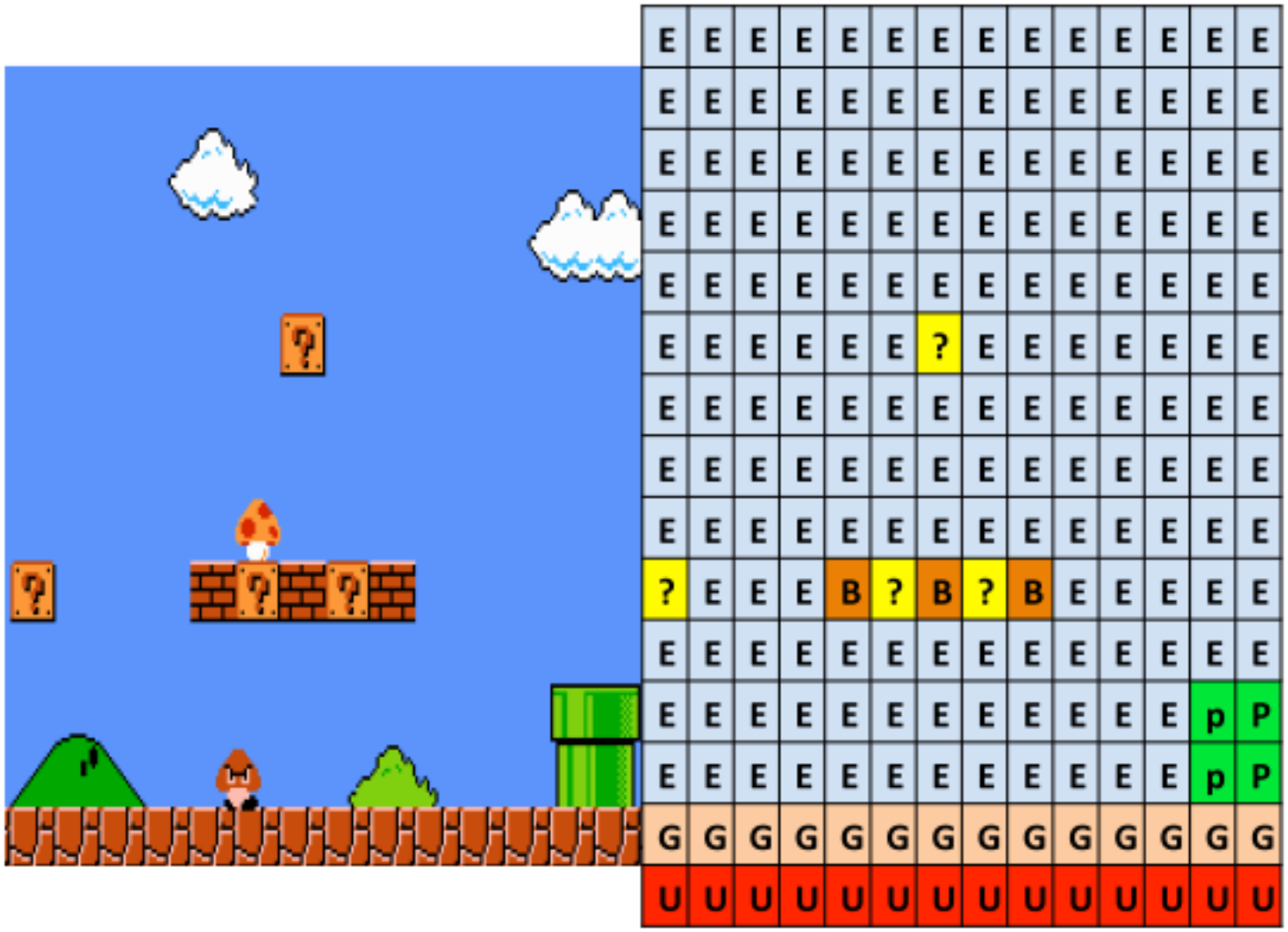


$P(S_t \mid S_{t-1})$

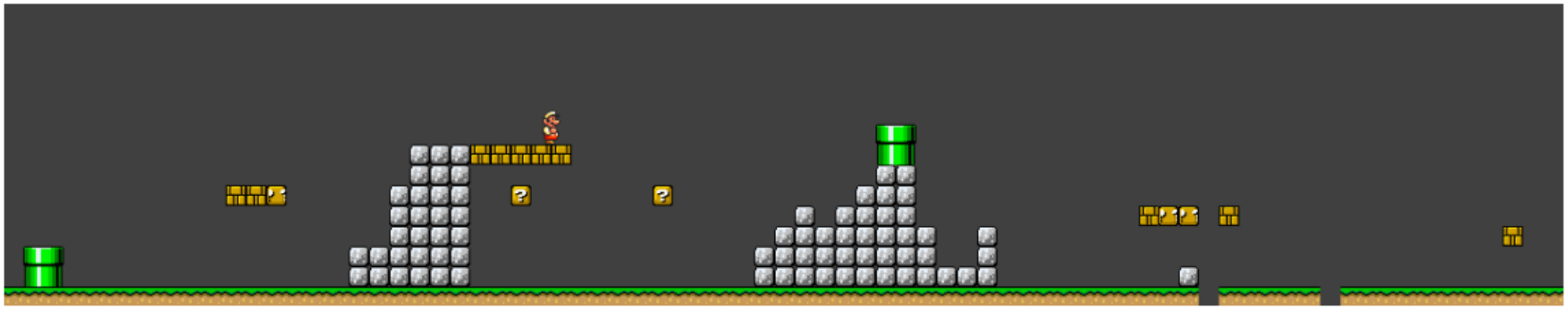
b) Order 2 Markov Chain



$P(S_t \mid S_{t-1}, S_{t-2})$



[Snodgrass2014]



$$D_5 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 2 \end{pmatrix}$$

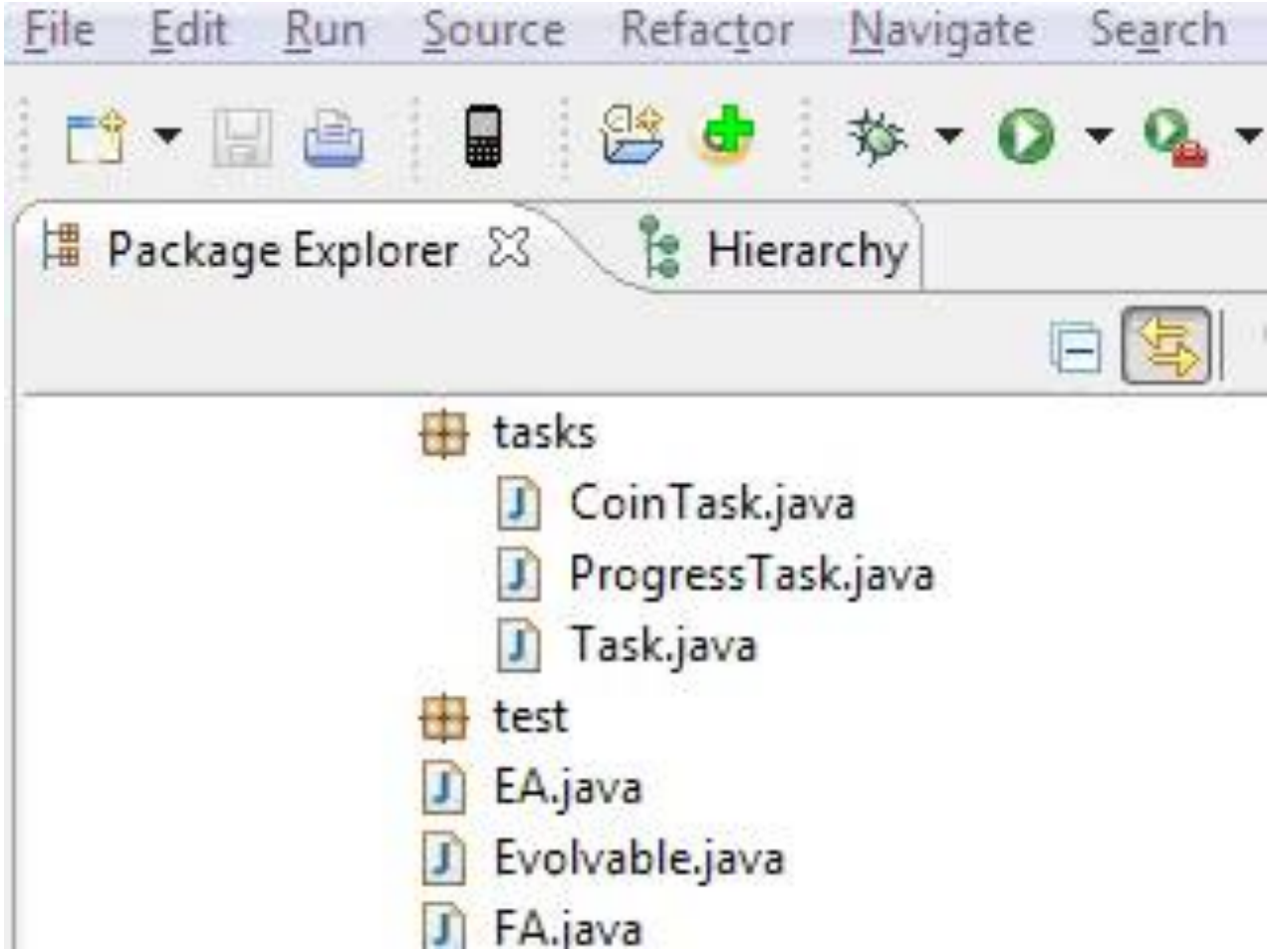
[Snodgrass2014]

Search based generation

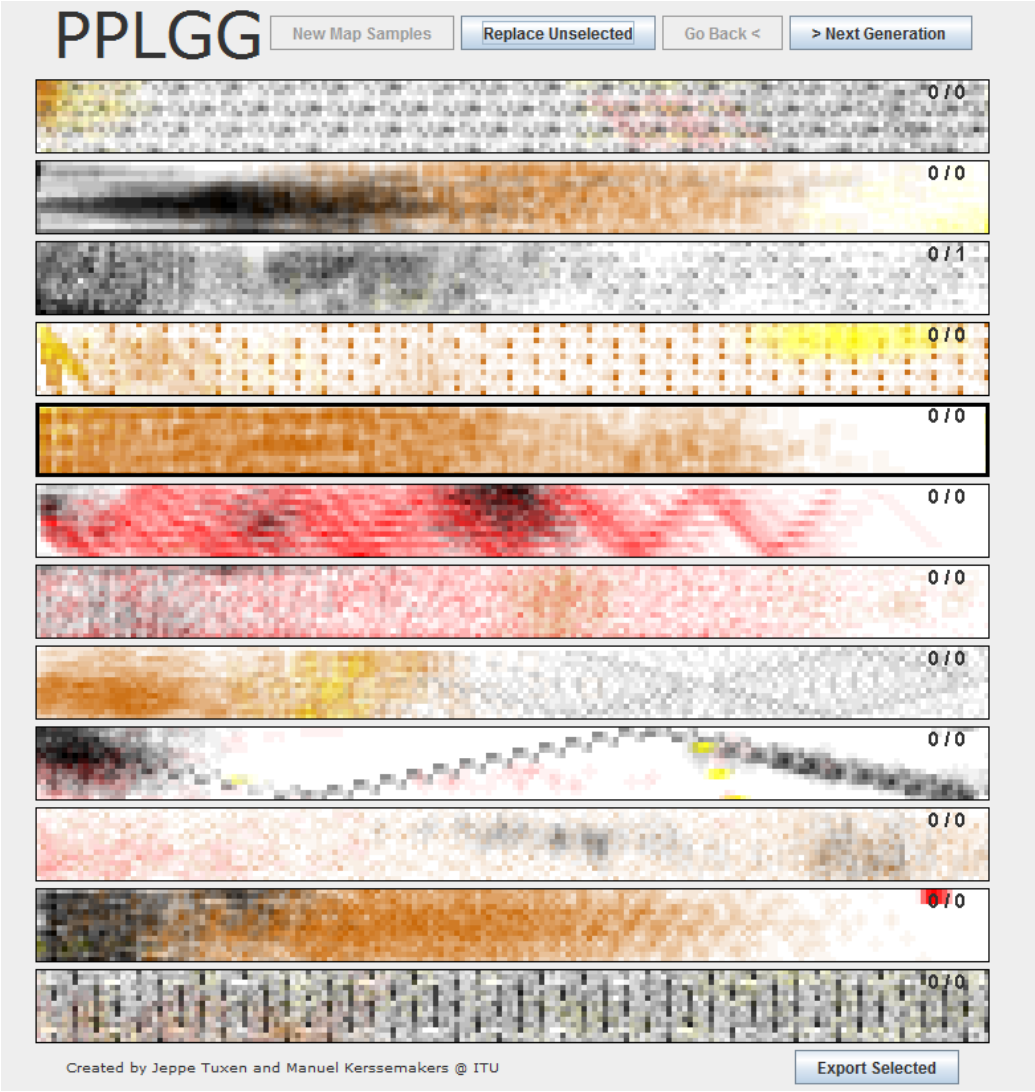
Explore the levels space

It’s all about the score function

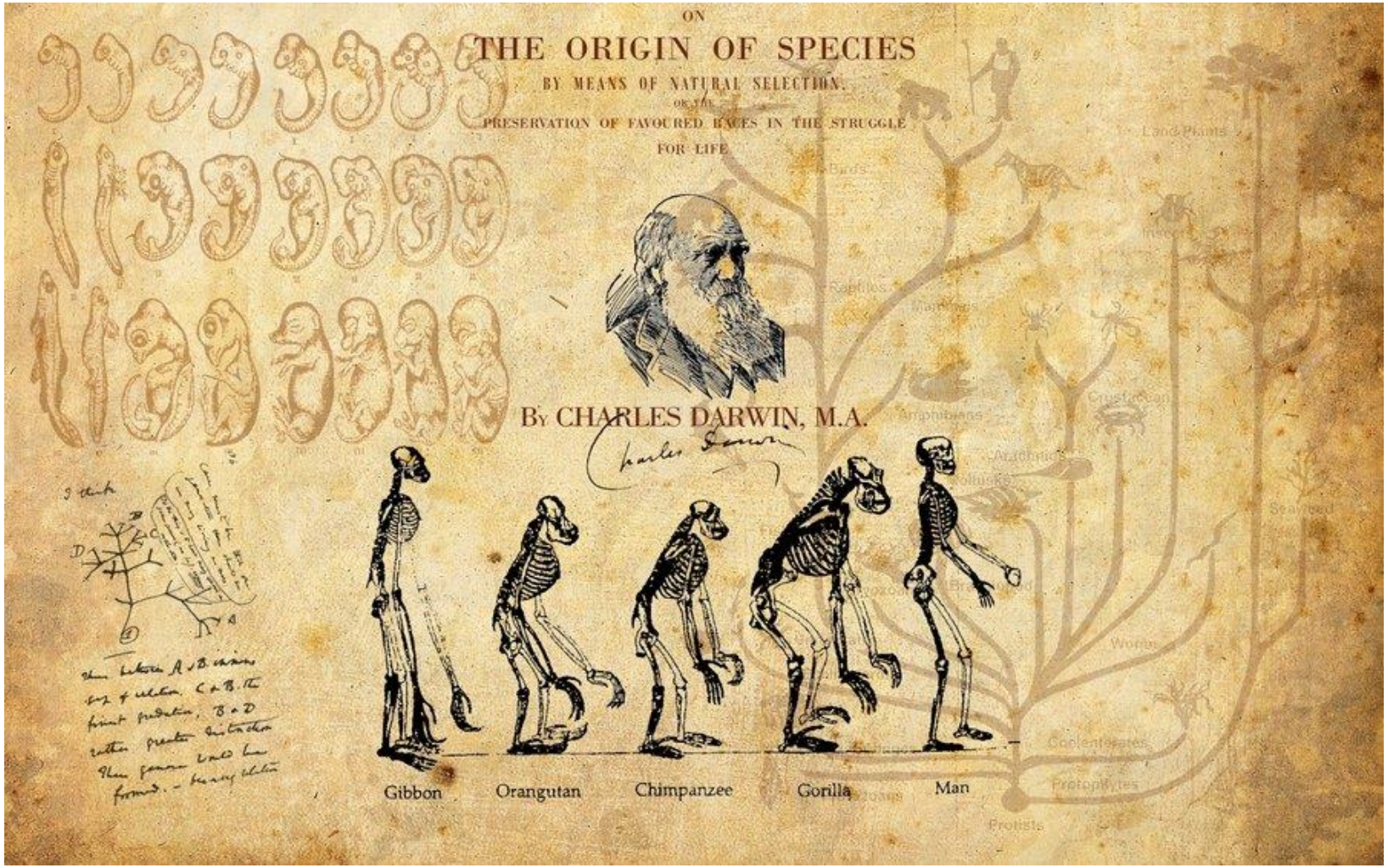
Agent based scoring



Mixed initiative, agent based



Evolutionary Algorithms



Genetic Algorithms

```
BEGIN
  INITIALISE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
    1 SELECT parents;
    2 RECOMBINE pairs of parents;
    3 MUTATE the resulting offspring;
    4 EVALUATE new candidates;
    5 SELECT individuals for the next generation;
  OD
END
```

Genetic Algorithms

- Sorry it’s a bit long
- But it’s bread and butter of PCG research ☺

Solution encoding

- **Phénotype** : a game level. What we actually want to be gnereated at the end.
- **Genotype** (or **chromosome**) : abstraction of a phenotype, using n variables. Thus a point in a n dimensions space. We search this space for a good solutions
- **Gène** (or locus): One variable of a chromosome
- **Allele**: possible value for a gene.
- **Encoding :** phenotype to genotype
- **Decoding :** genotype to phenotype. Maximum of 1 phenotype per genotype.

Fitness Function

- Evluates the quality of a chromosome
- Most often :
  - Decode chromosome
  - Evaluate phenotype (ai plays the level, etc…)
- We want to maximize this fitness function (find the best solution)

Population

- Set of solutions currently found (individuals)
- Constant size from loop to loop
  - We ar going to add, modify and delete individuals.
- **Population's diversity :** are individuals different from each other ?
  - Number of different genotypes
  - Number of different fitness values
  - Entropy

Entropy

- How much is a system predictable
- X random variable, values {xi,…,xn}
- Entropy H(X) = $-\sum_{i=1}^{n} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i),$

- Propriétés :
  - log(x) : [0,1] -> [-inf,0]
  - x*log(x)
    - if x = 0 : entropy is 0 (event never happens, we know it will not)
    - if x = 1 : entropy is 0 (event happens all the time, we know it will)
  - **Entropy is maximal when P(X) uniform**, i.e. P(xi) = 1/n
    - All event have the same probability, we cannot make any prediction, we know nothing.

Parents Selection

- Some individuals are selected for next generation.
- Depends on individual fitness and selection probability.
- « Bad » inbdividuals also have a chance to be selected :
  - Maintains the population diversity
  - Allows to find better solution that were not promising at first sight (Avoid local maximum)

Operators

- Create new individuals from old ones
- **Mutate** : unary operator. Random, **unbiased** modification of parent genotype to make a new one.
  - **Unbiased** : if you mutation operator tries to fix what you may consider a solution weakness, it's not a mutation operator (narrows the solution space instead of widening it)
- **Crossover** : binary operator : make to childrens by mixing two parents.
  - Allows to combine good properties of different solution in a single one

Ex: Binary Mutation

Fig. 4.1. Bitwise mutation for binary encodings

Ex: Binary Crossover

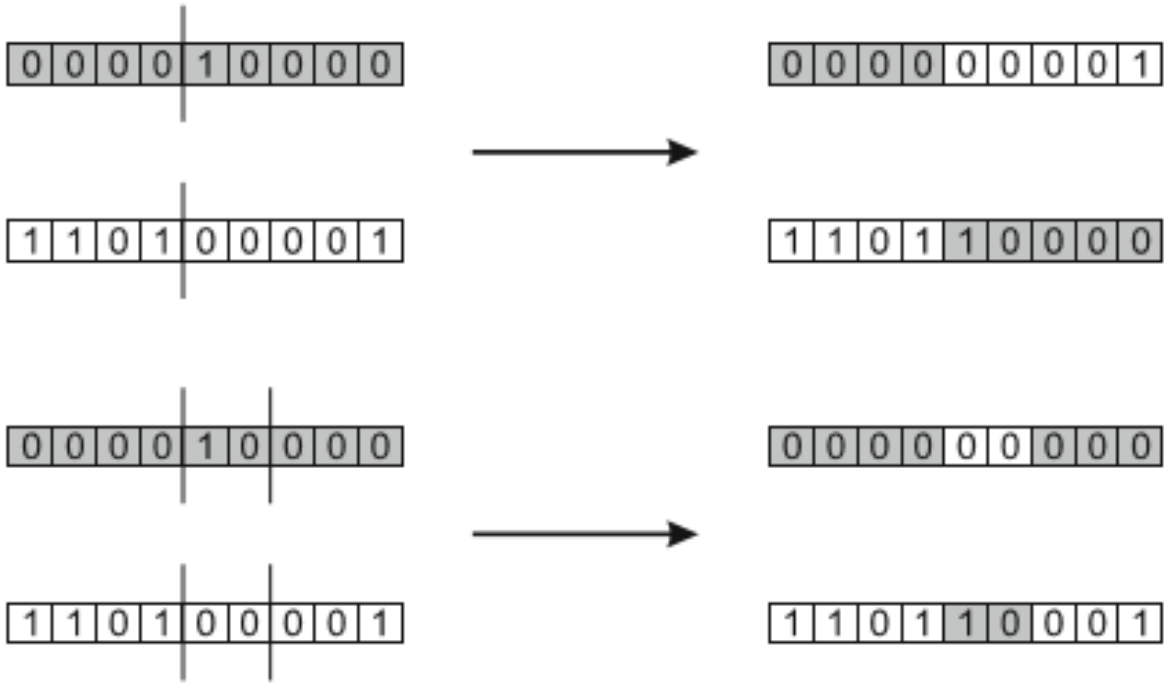


Fig. 4.2. One-point crossover (top) and n-point crossover with $n = 2$ (bottom)

Ex: Binary Crossover

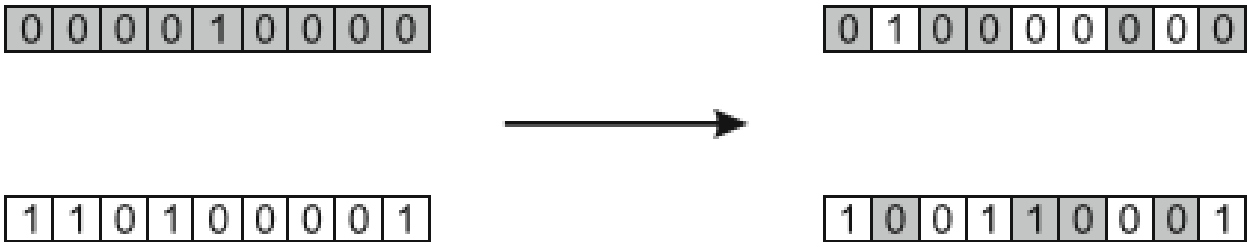


Fig. 4.3. Uniform crossover. The array [0.3, 0.6, 0.1, 0.4, 0.8, 0.7, 0.3, 0.5, 0.3] of random numbers and $p = 0.5$ were used to decide inheritance for this example.

Time to kill individuals !

- Population needs to keep a constant size
- Use quality and age of individuals :
  - Ex1: sort by fitness, keep n best individuals
  - Ex2: make the same but only with childrens (kill all the parents)

Stop condition

- The maximum allowed CPU time elapses.
- The total number of fitness evaluations reaches a given limit.
- The fitness improvement remains under a threshold value for a given period of time (i.e., for a number of generations or fitness evaluations).
- The population diversity drops under a given threshold.

Multi objective

- Multiple fitness functions:
  - Many ways to evaluate a solution
  - No apriori weight to combine functions (as if they were mixed in a single one)
- **Dominance**: individual A dominates B if A has a score at least as good as B for all fitness functions, and at least one of them, A is better than B.
- **Pareto front** : set of solution that are dominated by no one

Example : Togelius2010

Genotype

- Bases : polar coordinates 2D theta,phi (angle & distance)
- Mineral : x,y
- Gas : x,y
- Blocking zones:
  - Start X,y
  - P(left turn)
  - P(right turn)
  - P(gap)

- 3 bases, 4 minerals, 4 gas, 5 blocking zones
  - 3*2 + 4*2 + 4*2 + 5*5 = 47 floats

Phenotype

- Map of 64*64 tiles
- Bases
  - Forces maximal spread : for instance, if 3 bases, 120° between each other
  - Distances between 0.5 and 1.0 size of map
  - If base is outside map, orthogonal projection on side of map
    - Generates more maps with bases on the side of the map, which is a feature.

Phenotype

- Blocking zones
  - Starts in x,y
  - Adds new blocking tiles while not in a blocking zone, following probabilities
  - If 5 times straight, forces a turn
  - **To have only one phenotype per genotype** : always uses the same seed

Fitness function

- Playability :
  - Needs to be able to build your base (minimal space) and attack enemies
- Balancing :
  - If players have the same level, they must have the same chances to win
- Skill difference :
  - Better tactic need to win more often
  - A map should let players use different tactics
- Map's "beauty"
  - Maps should not look like other maps, be too symmetrical or too empty.

Fitness function

- fb0: Base space
  - For playability, some space for other buildings is required next to the base. Out of the 5*5 cells surrounding a base, the base space is defined as the fraction of these cells that are passable and reachable within 5 steps (using A*) from the base. This fitness value is the mean of the base space of all bases.

Fitness function

- fb1: Base distance
  - The measure makes sure that the bases are not too easy to reach from each other so that the players have the opportunity to develop their base before clashing with the others. It contributes to playability and skill differentiation as the game is more difficult for all players when starting close to each other.
  - Fb1 is the minimum distance between any two bases, divided by the sum of the map’s width and height

Fitness function

- fp1: Choke points.
  - We consider the average narrowest gap on all paths between bases. The narrowest gap along a path is calculated by first calculating a **shortest path** and then traversing along the path and counting the **width** of the path at each cell.
  - **Path width** is calculated through determining whether the path is currently moving **horizontally** or **vertically** through comparison with the previous cell in the path, and searching **orthogonally** to the path direction until either an impassable cell or the border of the map is encountered.
  - Choke points contribute to **skill differentiation** in that a good player will be able to exploit such points through using a smaller defending force to stop a larger attacking force, which cannot use the strength of its numbers as they have to pass sequentially through the narrow

Algorithm

- Multi Objective GA (SMS-EMOA : Multiobjective selection based on dominated hypervolume)
- 50000 epochs
- 20 individuals
- crossover/mutation : floating point operators that simulate binary ones :
  - SBX (Simulated Binary crossover)
  - PM (Power Mutation)

Result

Generating game space

Specific algorithms
focusing on game space

Cellular Automata

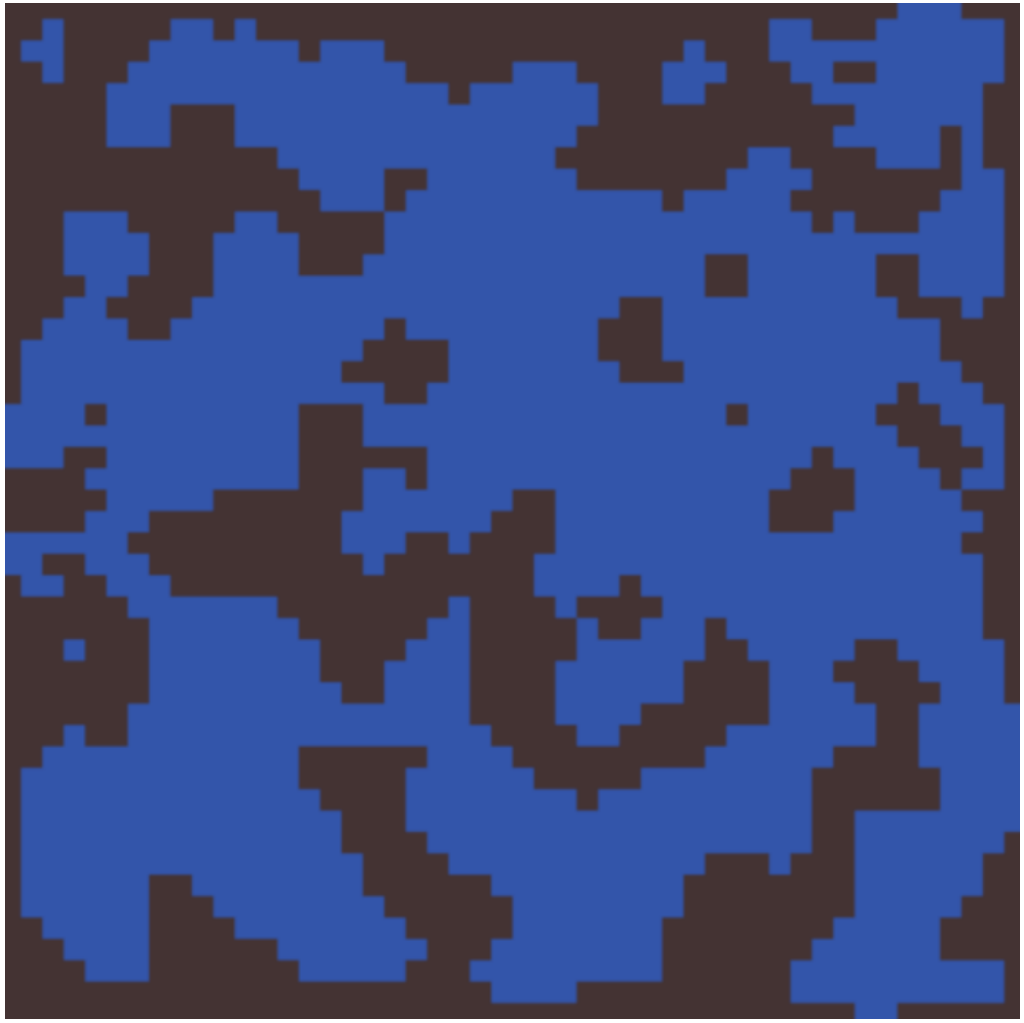
Emergent Space

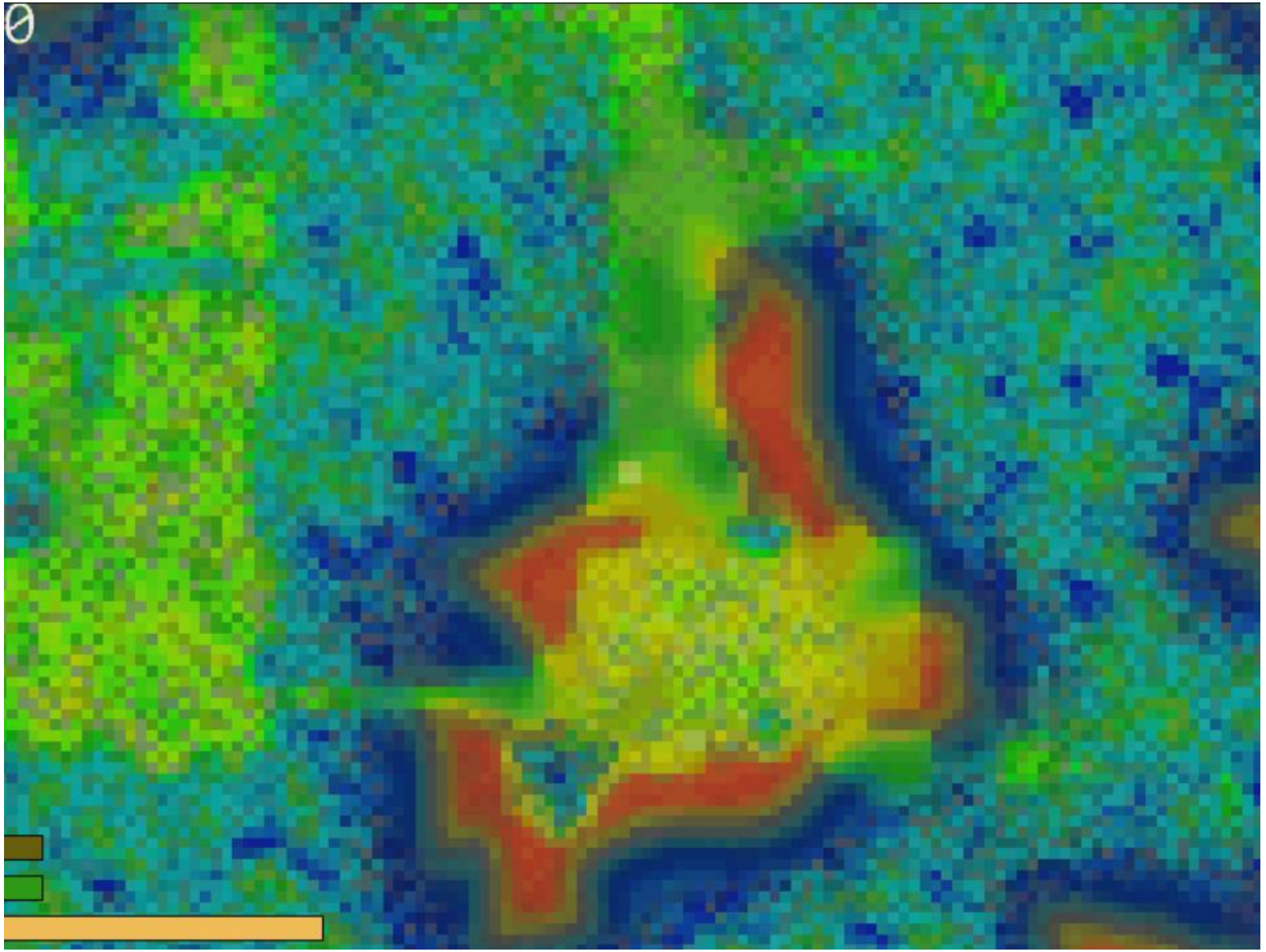
Cellular Automata

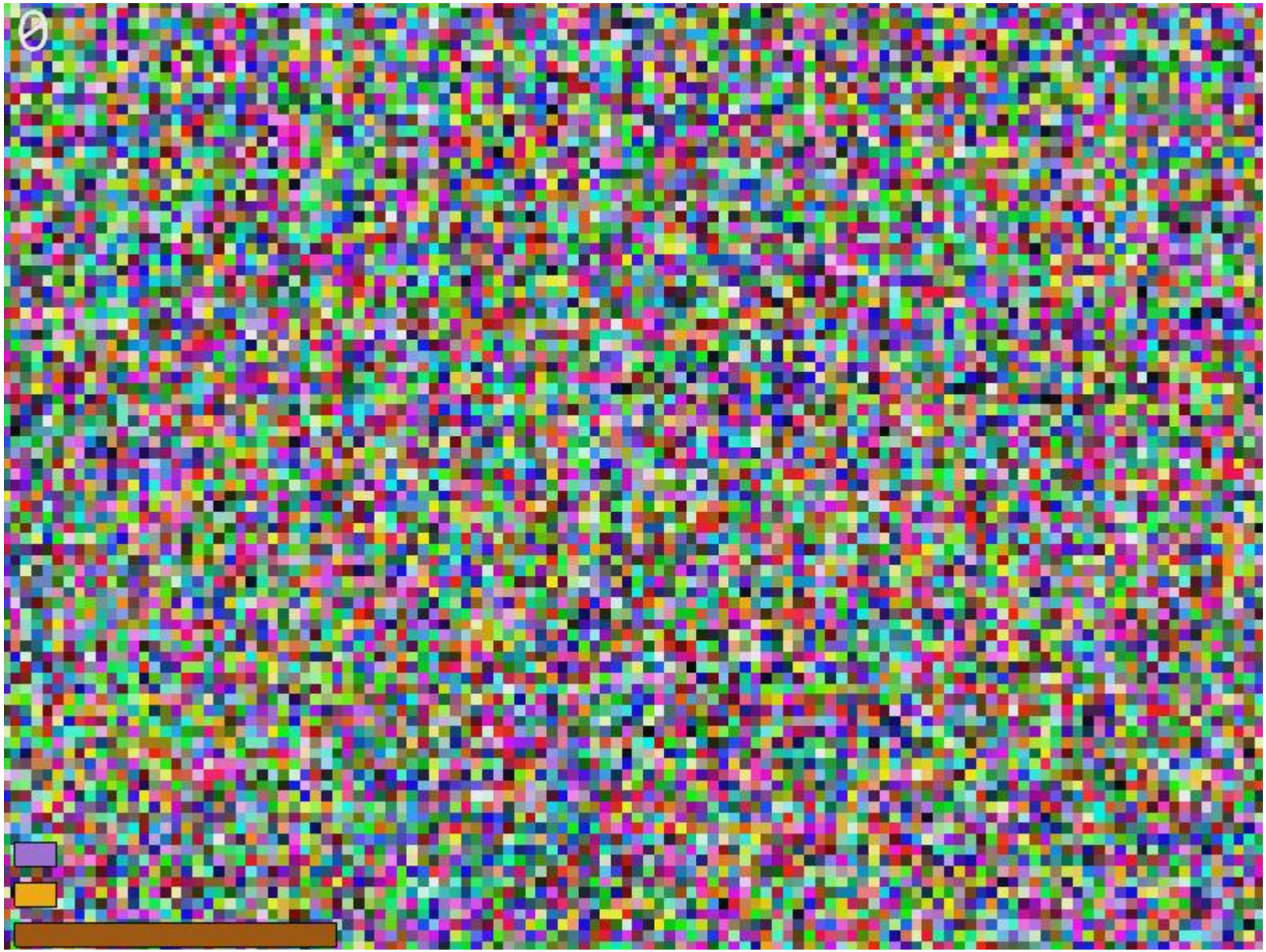
- Grid of cells
- State of each cell at t+1 calculated from
  - It's state at t
  - It's neighbours states at t

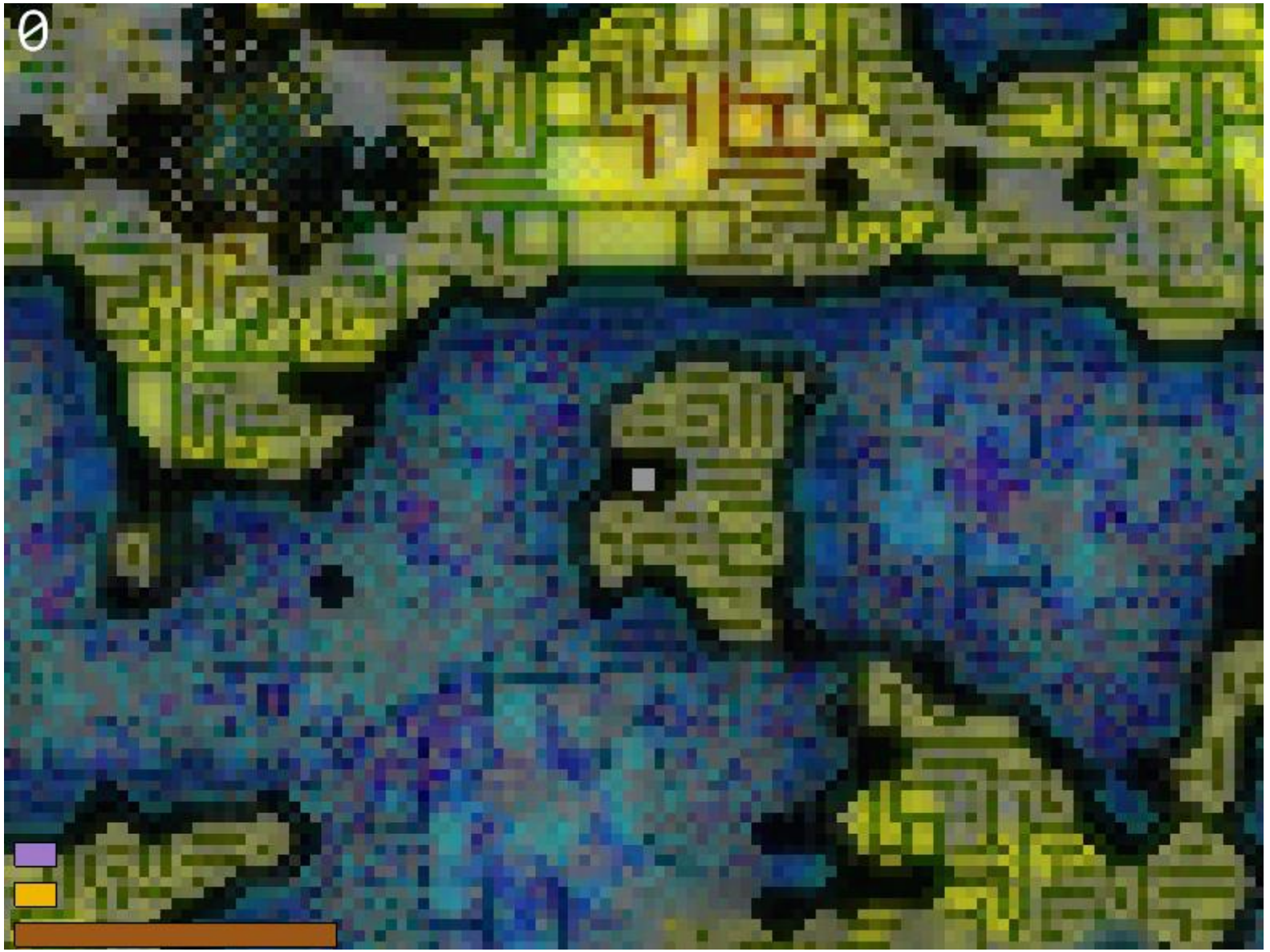
Game of Life

Generate Caves

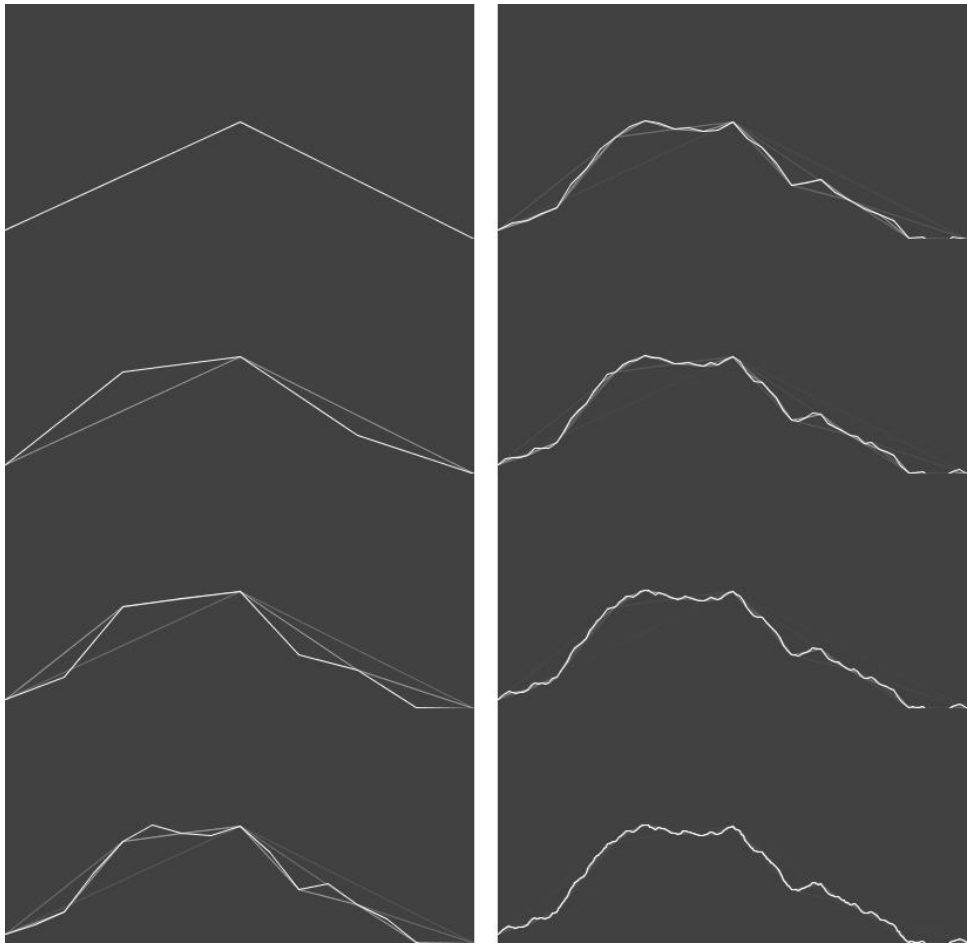



0


0

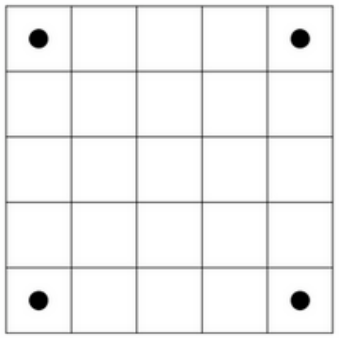

0

Fractal Brownian Noise

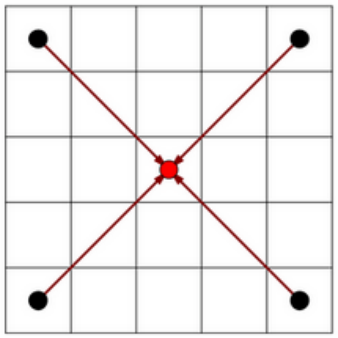


Maybe your noise will not be brownian but just fractal.
Not a big deal if your level looks great

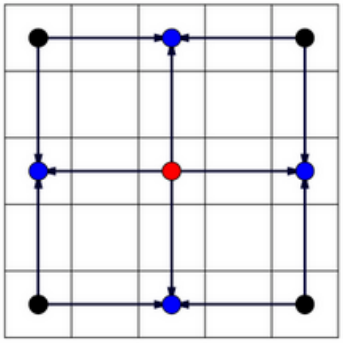
Diamond Square Algorithm



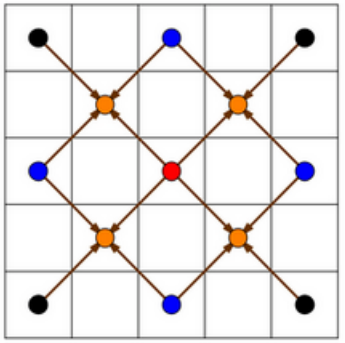
Initialisation des quatre coins



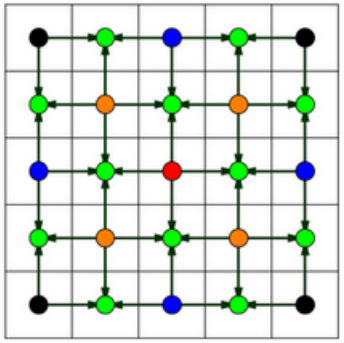
Phase du diamant avec un pas de 4



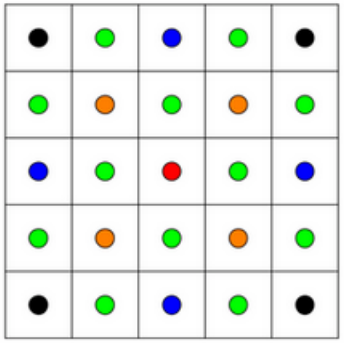
Phase du carré avec un pas de 4



Phase du diamant avec un pas de 2

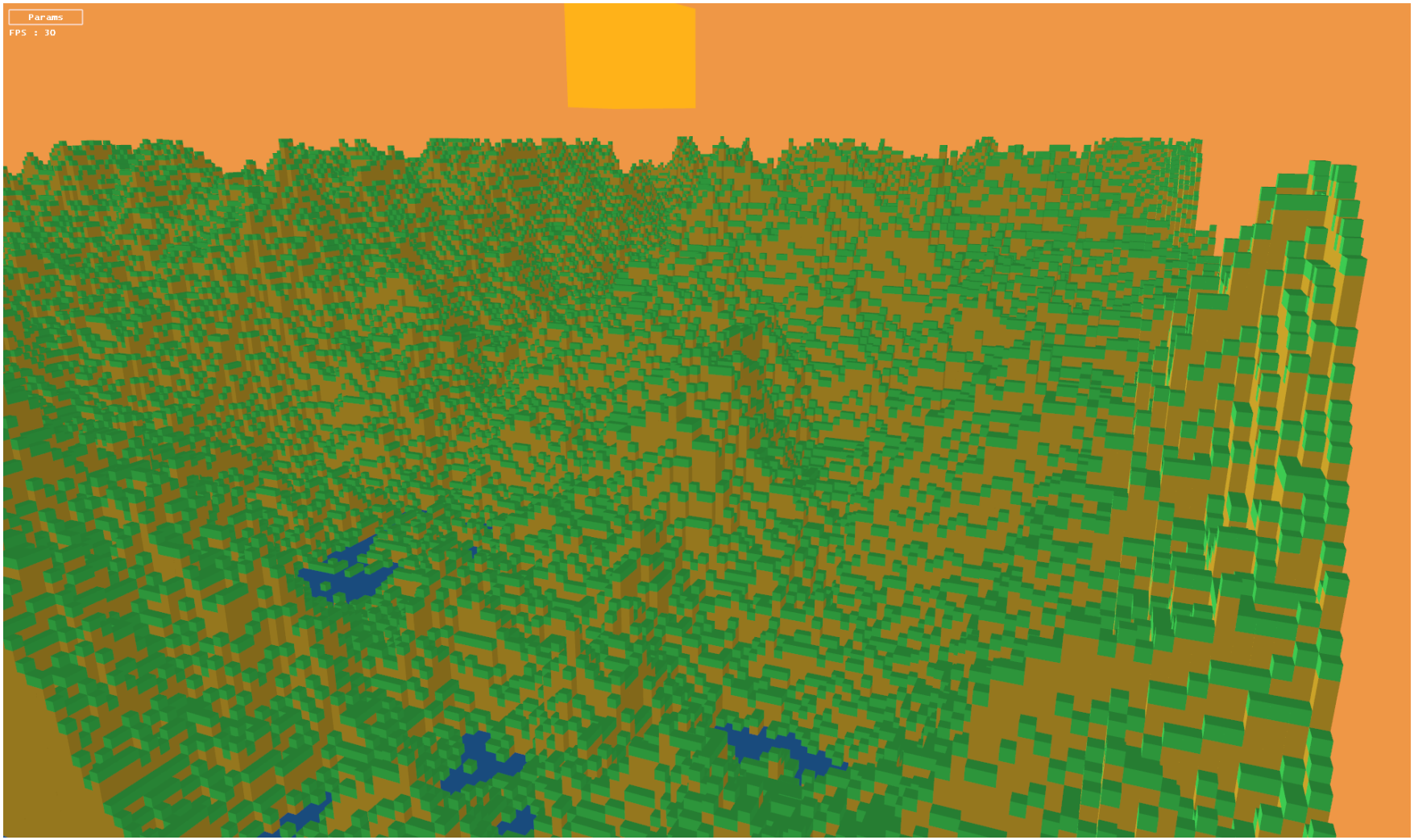


Phase du carré avec un pas de 2

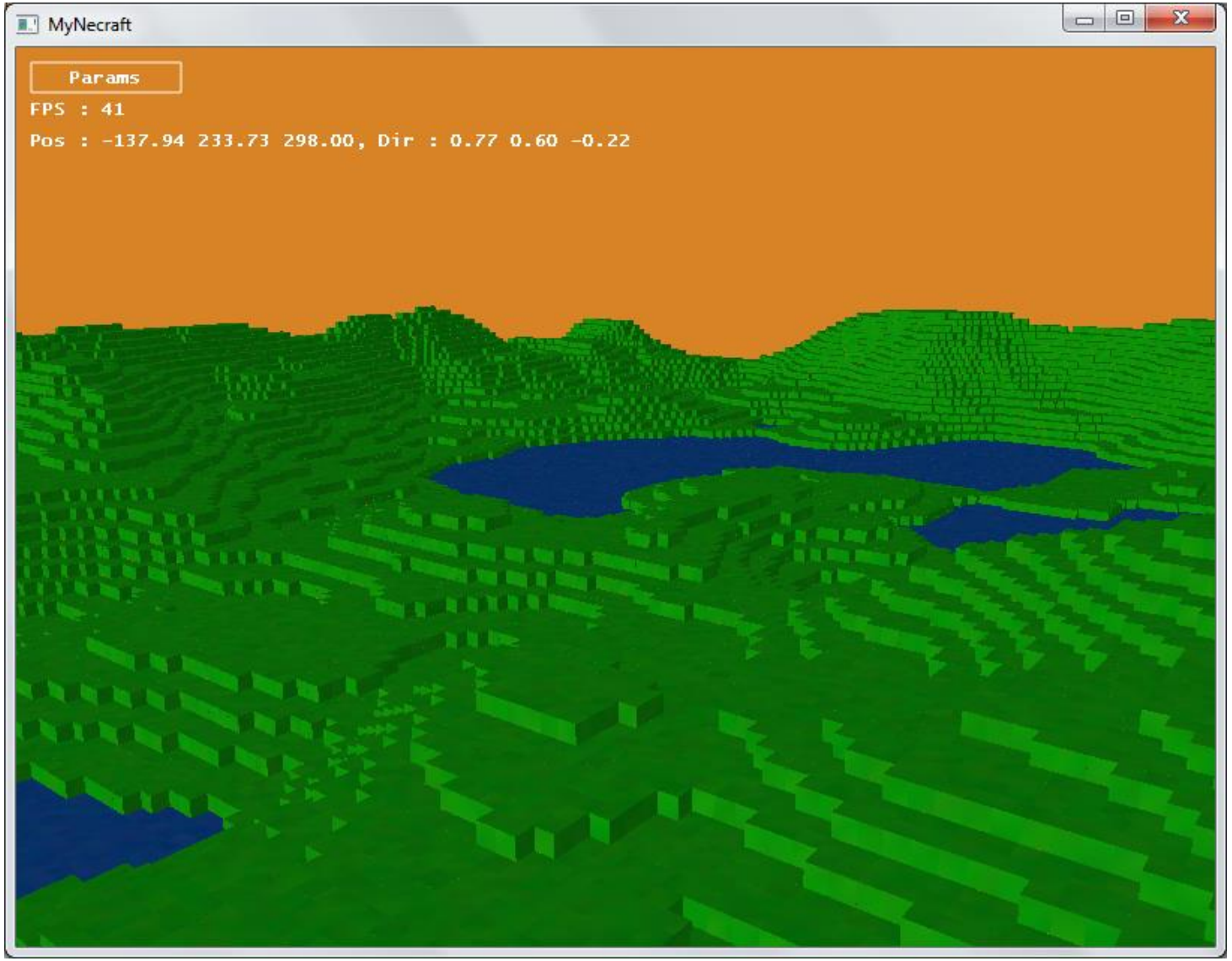


Matrice entièrment remplie

Diamond Square Algorithm



Diamond Square Algorithm



WFC