Lecture 7: Procedural Content Generation

- ECS7002P – Artificial Intelligence in Games
- Raluca D. Gaina – r.d.gaina@qmul.ac.uk
- Office: CS.335

Game AI Group

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

Queen Mary University of London

Outline

- Introduction
- Applications
- PCG Types
    - Level Generation
    - Rules & Game Mechanics
    - Others
- Methods
    - Constructive

Procedural Content Generation

Introduction

Procedural Content Generation

Procedural Content Generation (PCG) can be defined as the *algorithmic creation of game content with limited or indirect user input*. From a developer perspective, this is *making things that make other things*.

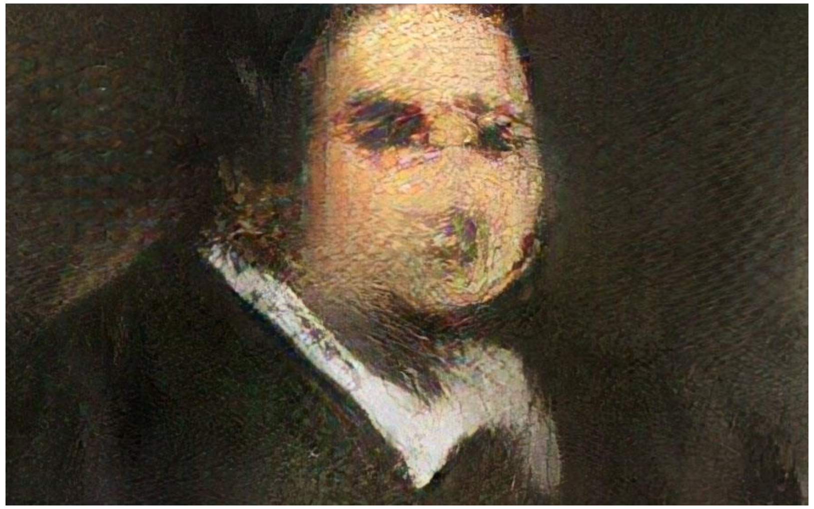
What is and (what is not) content?

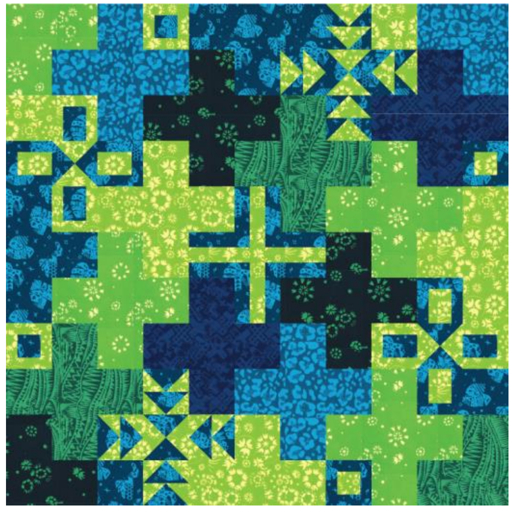
- Content is: levels, maps, game rules, textures, stories, items quests, music, weapons, vehicles, characters, etc.
- Content is not: the game engine, NPC AI.

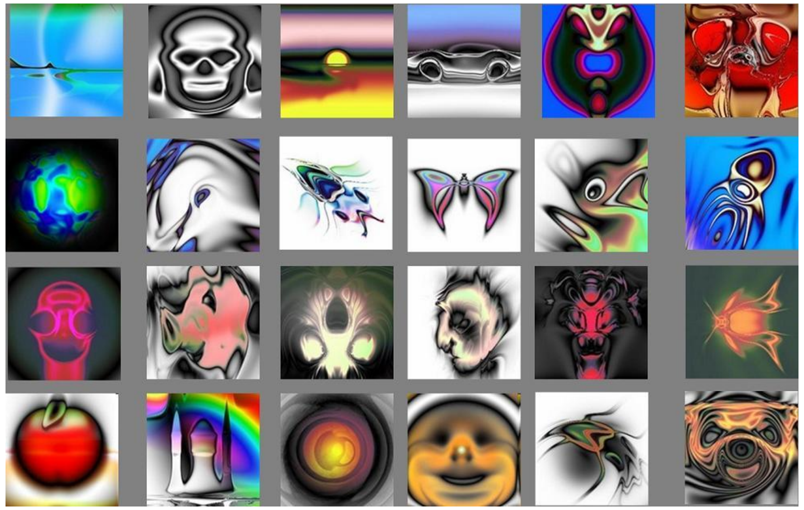
Why do we want PCG?

- No human artist/designed needed (partially: aids for design, inspiration, reachability).
- New types of games.
- Player-tailored design.
- Understanding manual design.

Not Only for Games?


Painting by GAN sold for $432,000


Anne Sullivan, Georgia Tech


Secretan, Jimmy, et al. **Picbreeder**: evolving pictures collaboratively online, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, 2008.

- Paintings
- Quilts
- Images
- Music: DADABOTS `https://youtu.be/MwtVkPKx3RA`
- And more!

Taxonomy

- Degree and dimensions of control

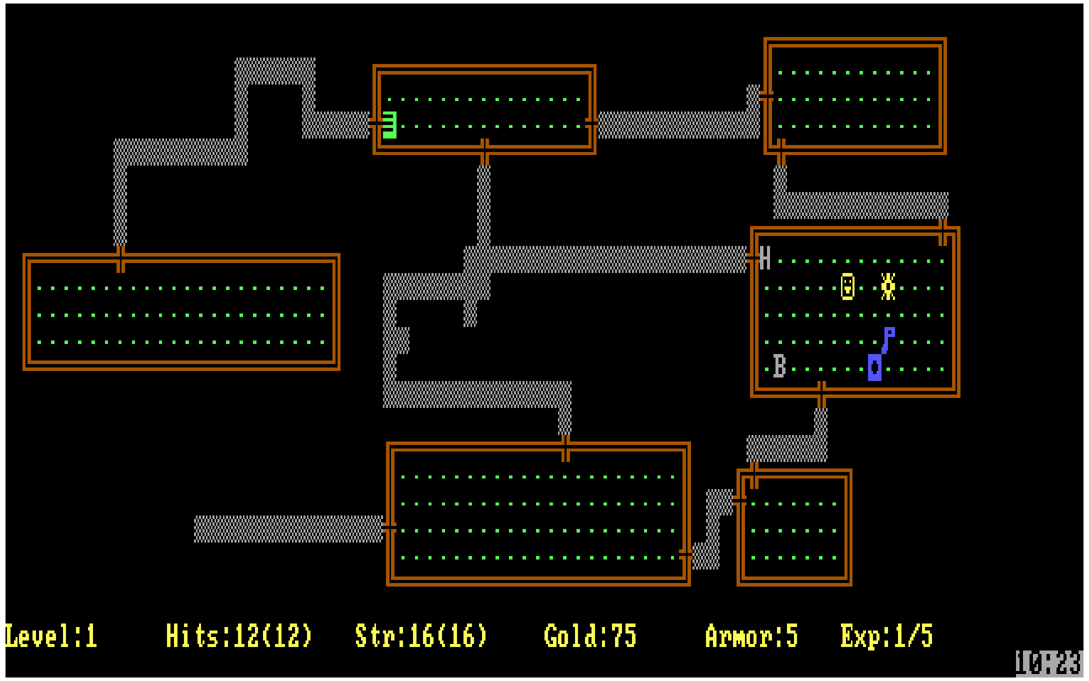
• Online	↔	Offline
• Necessary	↔	Optional
• Generic	↔	Adaptive generation of content while adaptive learns from invironment and generate
• Stochastic	↔	Deterministic
• Constructive	↔	~~Generative~~ search based
• Automatic generation	↔	Mixed authorship

Outline

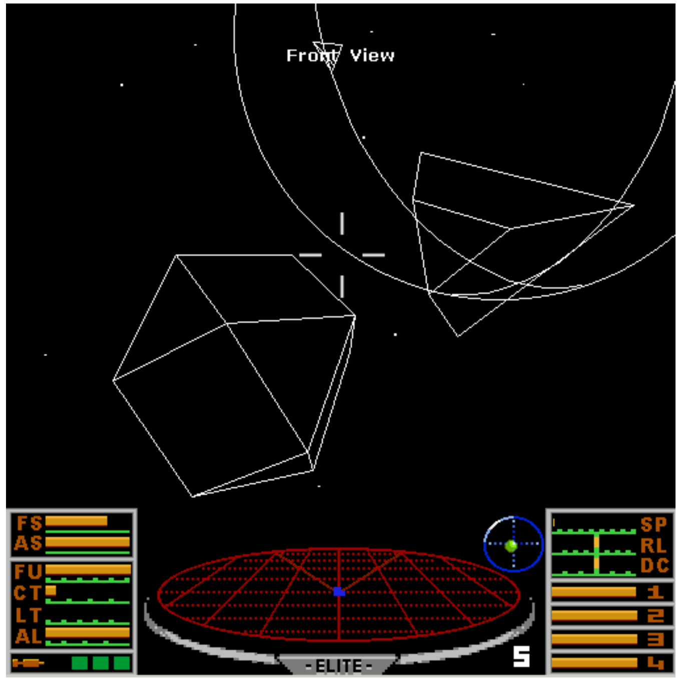
Procedural Content Generation

Applications

The Beginning

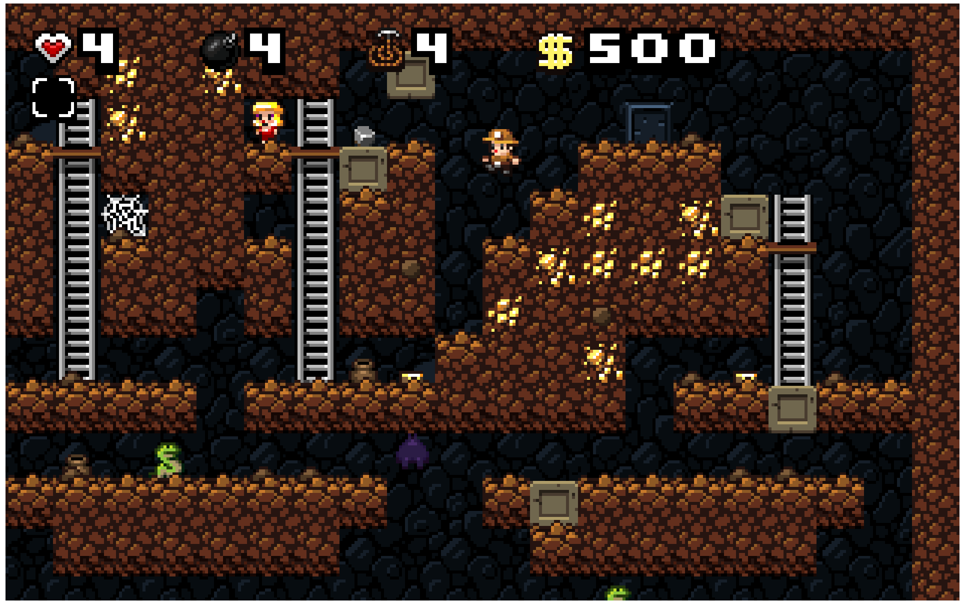


Rogue (Epyx, 1983): dungeon crawling with levels automatically generated every time the game starts.



Elite (Acornsoft, 1984): procedurally generates 8 galaxies with 256 planets each.

Spelunky (Derek Yu, 2008)



16 rooms in 4x4 grid, with different types (and several room templates for each type, with static and probabilistic tiles):

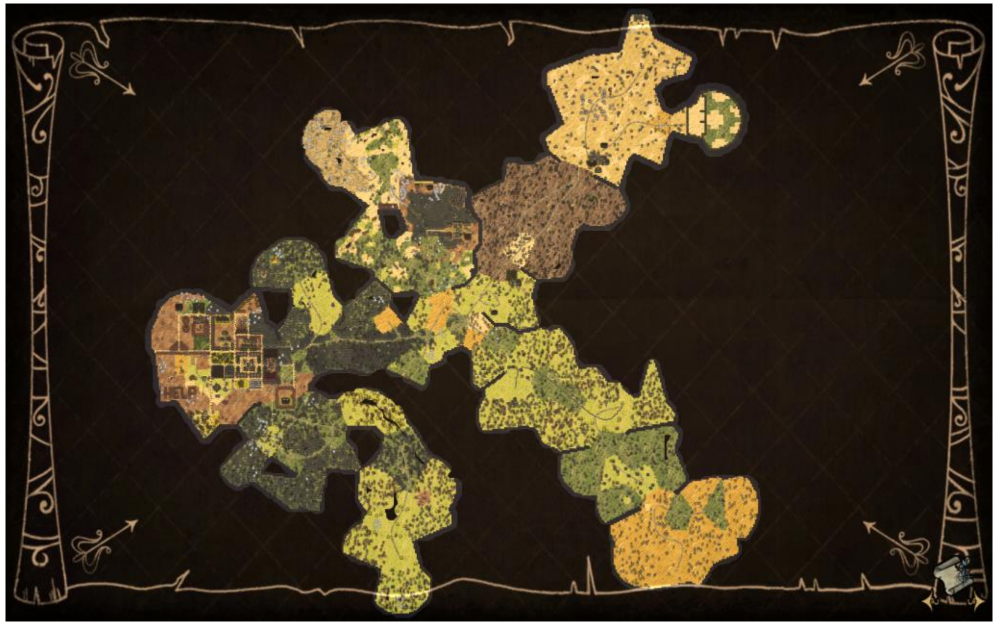
- 0: side room not on the solution path.
- 1: room guaranteed to have right and left exits.
- 2: room guaranteed to have right, left and bottom exits (+ top if room above is also type 2).
- 3: room guaranteed to have right, left and top exits.

Starts by creating solution path from start to exit room, then fill in the rooms and finally add decorations.

Minecraft (Mojang, 2011)

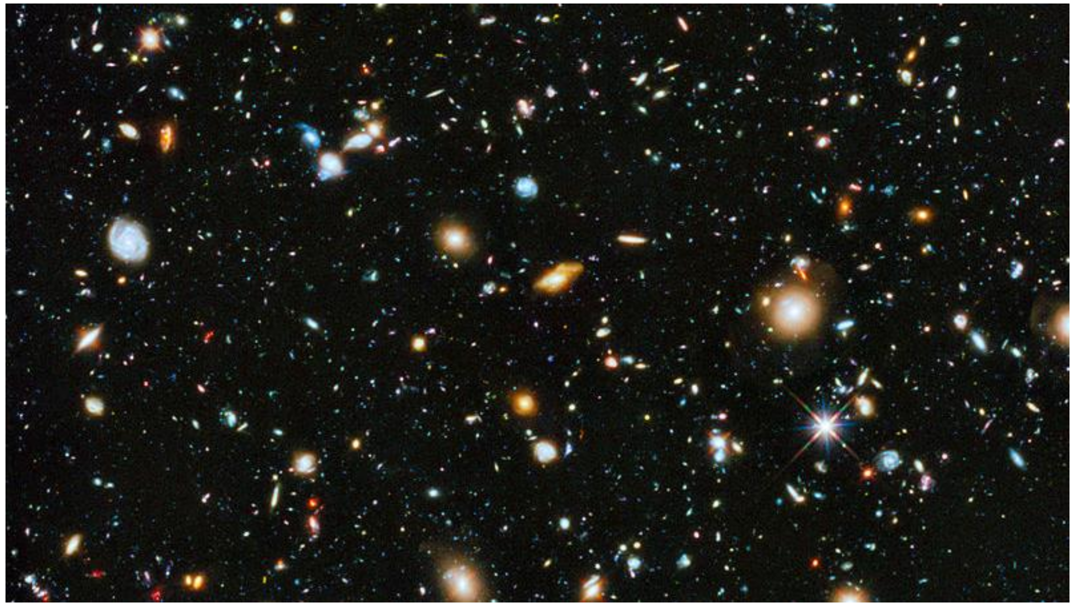
Perlin noise + detailing (lakes, interesting structures etc.).

Don't Starve (Klei Entertainment, 2013)



- Grow a tree of biomes.
- Generate each biome according to templates, add noise to biome shape and decorate with biome-specific game objects (e.g. trees in forest biomes, or reeds and tentacle monsters in swamp biomes).
- Add paths between all biomes.

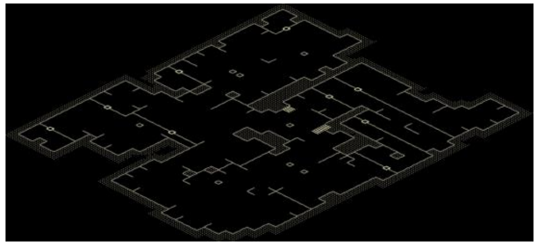
No Man’s Sky (Hello Games, 2016)



https://youtu.be/nmwG6Sj1Yfg

- Run-time generation of a universe with 18 quintillion spherical planets.
- Generates terrains, ecosystems, flora, fauna, ships, textures, animations, audio etc. from templates created by human artists.
- L-Systems.

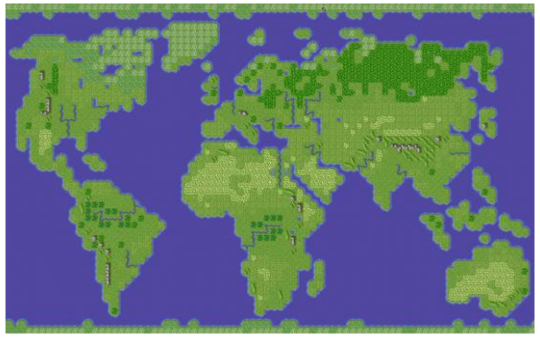
Others



Diablo (Blizzard Entertainment, 1996): maps, items, enemies.

Spore (Maxis, 2008): avatar's design.

Galactic Arms Race (Evolutionary Games, 2010): particle system weapons, adapts to player preference.
https://youtu.be/JkniSUXa8-I



Civilization (Sid Meier, 1991): random maps.

Outline

- ✓ Introduction
- ✓ Applications
- ❑ PCG Types
  - ❑ Level Generation
  - ❑ Rules & Game Mechanics
  - ❑ Others
- ❑ Methods
  - ❑ Constructive

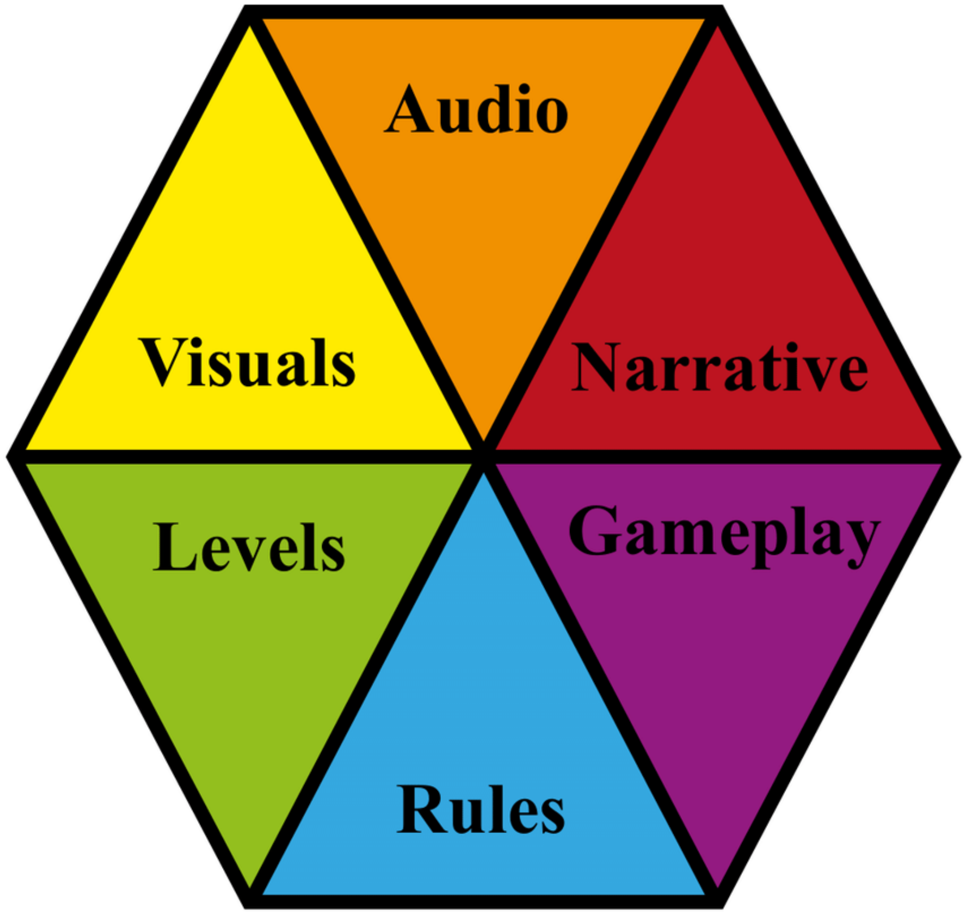
Procedural Content Generation

PCG Types

Desirable Properties of PCG

- **Speed**: generation of content must be according to the time limitations.
- **Reliability**: is the generated content viable?
- **Controllability**: how can a human make decision on the generated content?
- **Expressivity and diversity**: balance between completely random generation and diversity within a narrow scope.
- **Creativity and believability**: can the algorithm be creative within the bounds of reason?

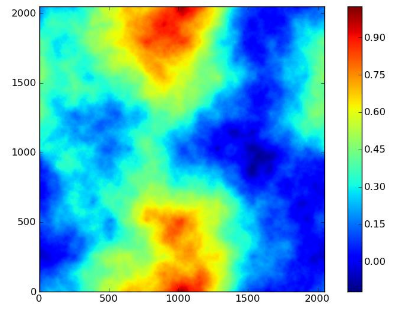
Types of PCG



Liapis, A., Yannakakis, G.N., Nelson, M.J., Preuss, M. and Bidarra, R., 2018. Orchestrating game generation. *IEEE Transactions on Games*, *11*(1), pp.48-68.

Levels

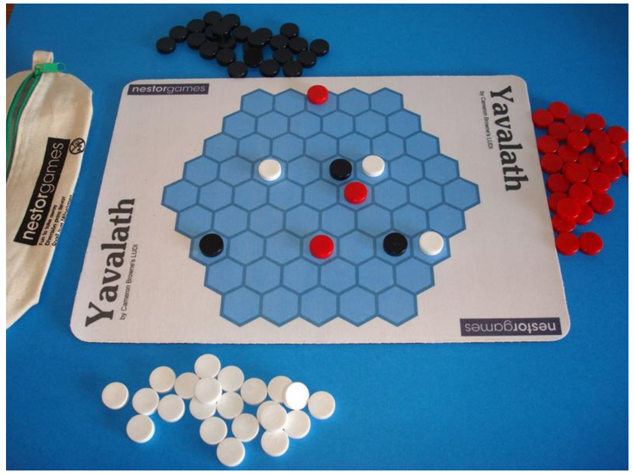
- Most popular form of PCG
- **Where** do things happen?
- How can you create the virtual space a game takes place in?
  - Terrains
  - Maps
  - Dungeons
  - Platforms
- Aesthetic vs functional
  - Memorable visible landmarks
  - Constraining paths
- Size and detail
- Representation
  - Genotype vs Phenotype
  - 2D matrix
  - Height intensity maps for terrain



The Witcher 3: Wild Hunt (2015, CD Projekt)

Rules & Mechanics

Yavalath (2007, nestorgames, Cameron Browne)



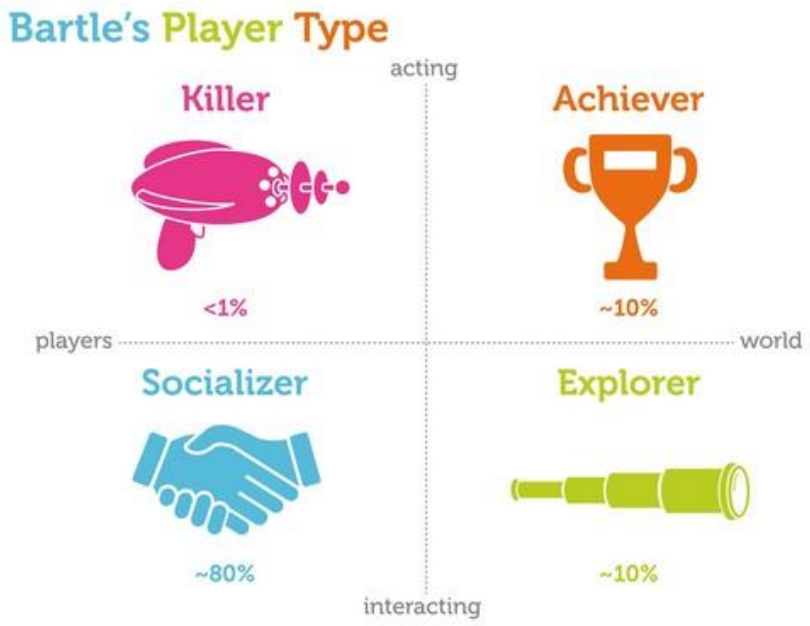
```
(game Yavalath
   (players White Black)
   (board (tiling hex) (shape hex) (size 5))
   (end (All win (in-a-row 4)) (All lose (in-a-row 3)))
)
```

- **What** happens if different objects interact?
- How can you create rules defining how the world works?
    - Movement schemes
    - Collisions
    - Player interaction effects
    - Game parameters
    - Winning / losing conditions

- Some problems:
    - Is the game turn-based, or real-time? more graphical, or more mathematical?
    - The generated games must be playable.
    - The generated games must be able to be evaluated. What makes it good, or fun?
    - How do we generate levels for these games, if needed?

Gameplay

- **How** or **Why** do players play / do things happen the way they do?
- How can you create specific player experiences in a game?
  - Playability
  - Fairness
  - Memorability
  - Uniqueness



Audio

https://youtu.be/gWs_RKXkyu0

- Often ignored, but key feature in games.
- Context, theme, immersion.
- How can you create sounds for a game, or change them based on the player's interactions with the world?
  - Proteus (Key and Kanaga, 2013): adapt sound to player location and viewpoint.
  - Adapt sounds to specific events or areas in the game.
  - Generate music in real-time to foreshadow upcoming game events based on narrative.

Narrative

- Motivation, context, theme, immersion.
- How can you create the story of a game?
  - Plot
  - Dialogue
  - Character backstory
  - World story

Façade (Procedural Arts, 2005) by Michael Mateas, Andrew Stern

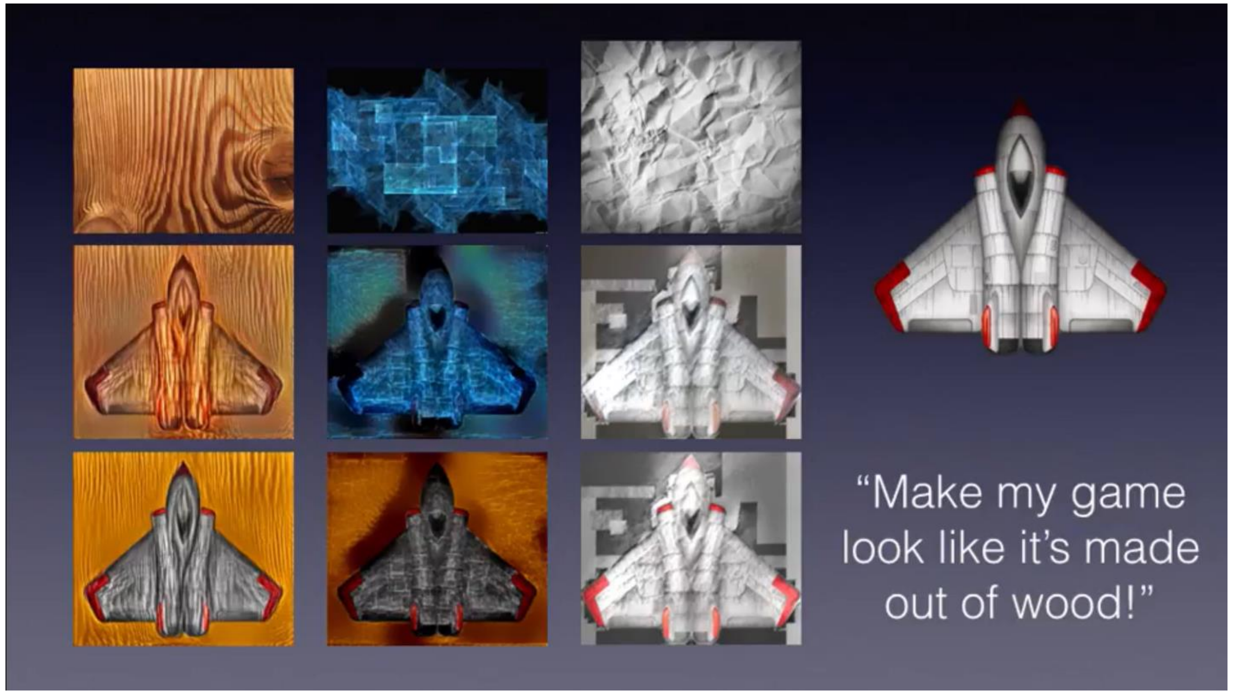
Visuals (1)

- Context, theme, immersion.

- How can you create a look for the game?
  - Landscapes
  - Character looks
  - Objects in the level
  - Etc.

- Computer graphics techniques

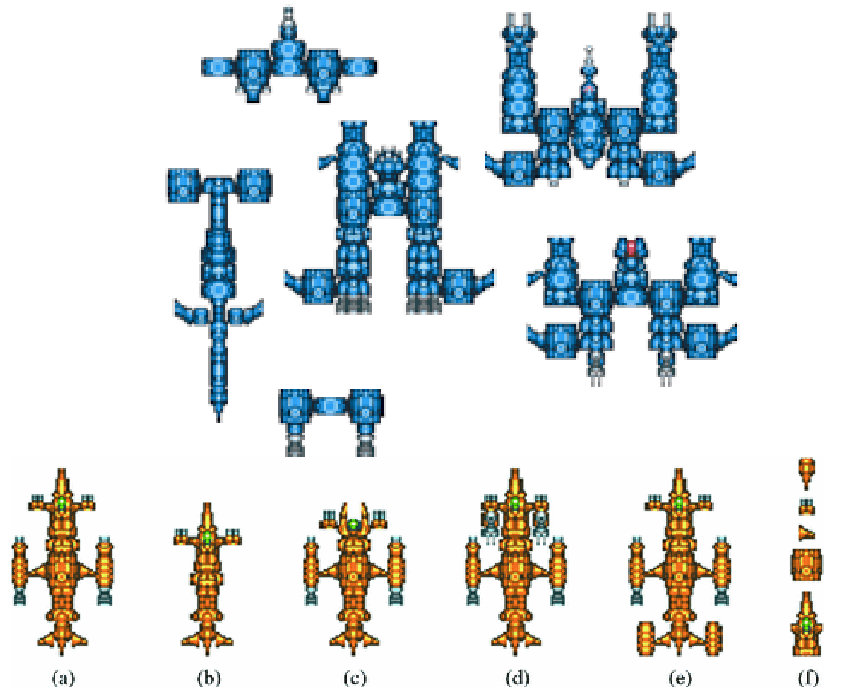
- AI based generation:
  - Evolution of graphic shaders.
  - Procedural scene filters.
  - Evolution of arcade-style spaceships for symmetry, simplicity, patterns.
  - Style transfer.



Simon Colton, Queen Mary University of London

Visuals (2)

Tutenel, T., Van Der Linden, R., Kraus, M., Bollen, B. and Bidarra, R., 2011, June. Procedural filters for customization of virtual worlds. In *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games* (p. 5). ACM.



Liapis, A., 2016, March. Exploring the visual styles of arcade game assets. In *International Conference on Computational Intelligence in Music, Sound, Art and Design* (pp. 92-109). Springer, Cham.

Outline

- ✓ Introduction
- ✓ Applications
- ✓ PCG Types
  - ✓ Level Generation
  - ✓ Rules & Game Mechanics
  - ✓ Others
- ❑ Methods
  - ❑ Constructive

Procedural Content Generation

Constructive Methods

Generative Methods

- Find the right tool for the job!

- **Constructive**
  - Put together different pieces so as to form content.
  - Respecting rules / constraints.
  - Following specific logic.
  - Automatic generation.
  - **Advantages**:
    - (Mostly) easy to implement and interpret.
    - High controllability and reliability.
    - High speed.
  - **Disadvantages**:
    - Lower expressivity and diversity, not much variation in generated spaces.

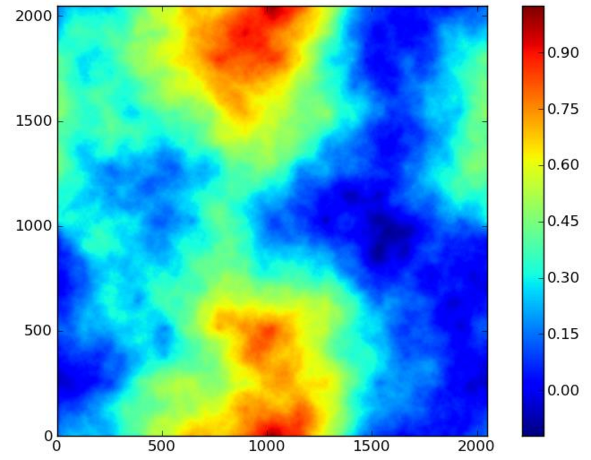
Rules and Constraints Based Generation

Design rules, procedures, logic the generation process should follow. Assign probabilities to rules triggering to increase the possibility space.

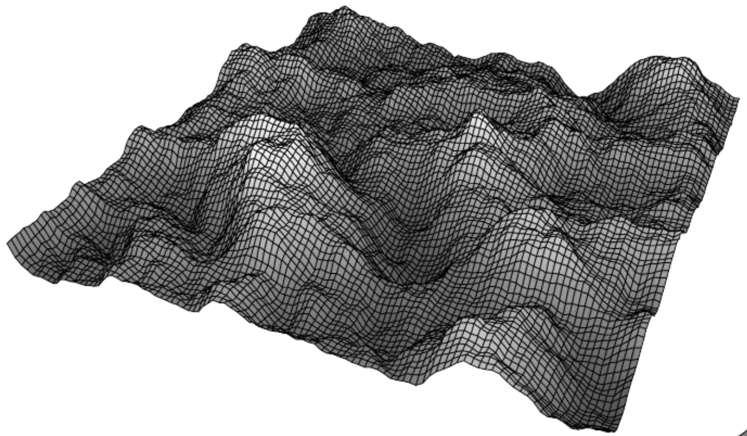
For example:

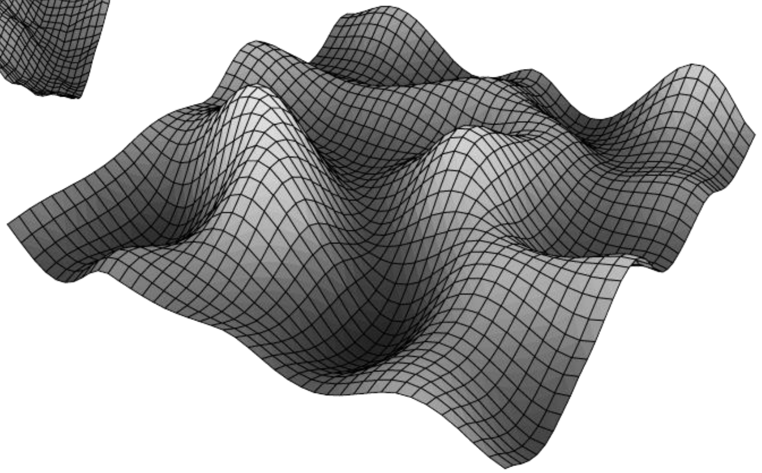
1. Place floor tiles all along the bottom of the screen, with a probability of 0.95 of placing a tile in each column.
2. Place platforms of varying lengths (mean 3, sd 1) at varying heights (mean 10, sd 3), with a probability of 0.42 of placing a platform starting in each column. Do not overlap platforms.
3. Place enemies on each platform, with probability distribution over enemy type, and probability 0.78 of placing an enemy on each platform, and probability 0.45 of placing a second enemy on platforms with 1 enemy.

Perlin Noise



joecrossdevelopment.wordpress.com





- Generate height map: 2D grid -> 3D terrain.
- Loop over all cells in a grid and apply Perlin noise given the cell coordinates:

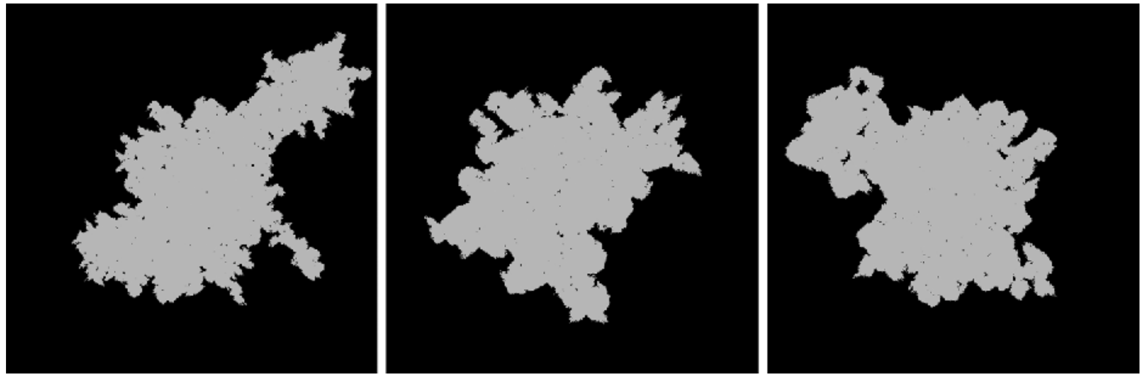
```
height[x, y] = Mathf.PerlinNoise((x / terrain.width) * tileSize,
                                 (y / terrain.height) * tileSize);
```

Doran and Parberry's Algorithm

Agents with different tasks act in the environment simultaneously:

- **Coastline**: outline and shape of terrain.
- **Landform**: terrain features, hills, mountains, beaches, rivers.
- **Erosion**.

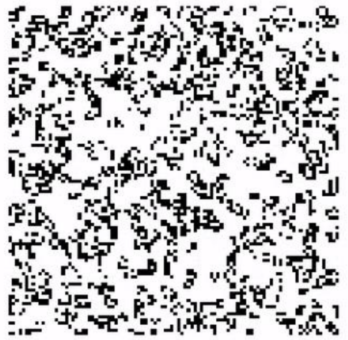
```
COASTLINE-GENERATE(agent)
if tokens(agent) $\geq$ limit
    then
        create 2 child agents
        for each child
            do
                child $\leftarrow$ a random seed point on parent's border
                child $\leftarrow$ 1/2 of the parent's tokens
                child $\leftarrow$ a random direction
                COASTLINE-GENERATE(child)
    else
        for each token
            do
                point $\leftarrow$ random border point
                for each point p adjacent to point
                    do
                        score p
                fill in the point with the highest score
```

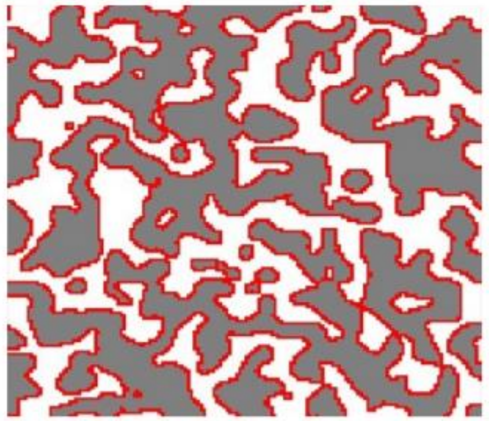


Doran, J., Parberry, I.: Controlled procedural terrain generation using software agents. IEEE Transactions on Computational Intelligence and AI in Games 2 (2), 111–119 (2010)

Cellular Automata

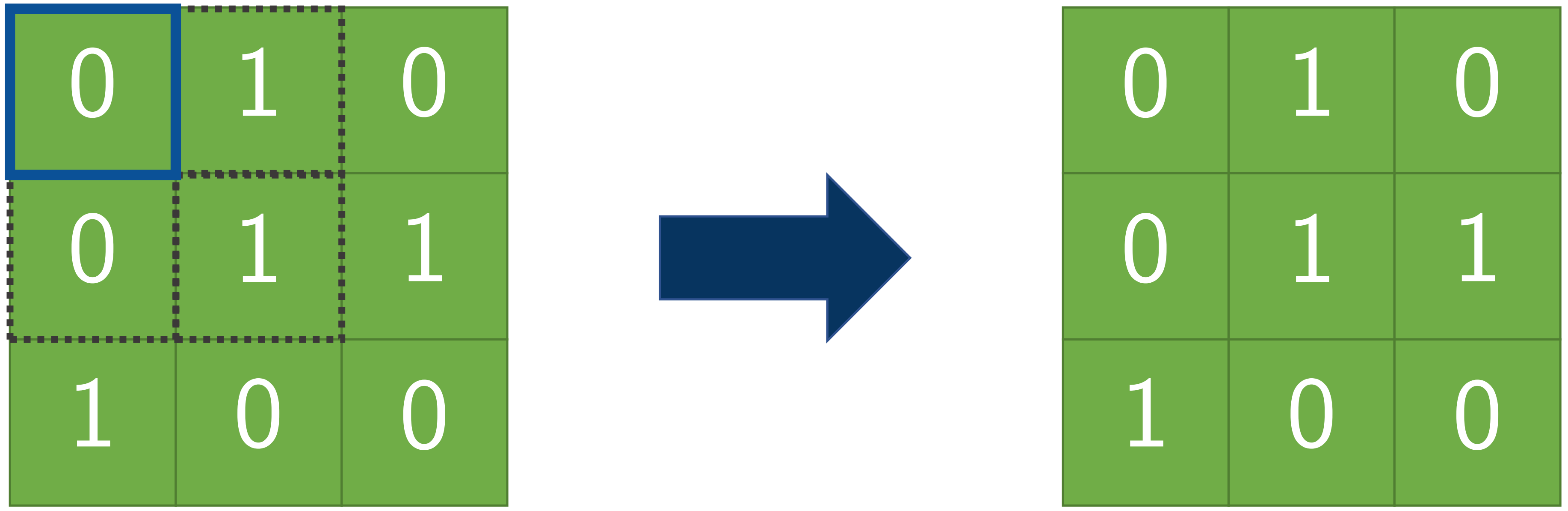
- Content representation: grid, where each cell has several neighbours.
- Algorithm:
  1. Loop over all cells in the grid and modify each according to defined rules.
  2. Repeat for several iterations.
- **Conway's Game of life** (8 neighbours, Moore neighbourhood):
  - Cell becomes 0 if cell is 1 and $(neighbours == 1) \leq 1$
  - Cell becomes 1 if cell is 0 and $(neighbours == 1) = 3$
- **Cave generator** (8 neighbours):
  - Cell becomes *floor* if *rock* cells in neighbourhood ≥ 5.
  - Cell becomes *rock* if *rock* cells in neighbourhood < 5.
  - At the end, *rock* becomes *wall* if at least 1 neighbour is *floor*.
  - At the end, create tunnels to connect all cave sections.





Johnson, L., Yannakakis, G.N. and Togelius, J., 2010, June. Cellular automata for real-time generation of infinite cave levels. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games* (p. 10). ACM.

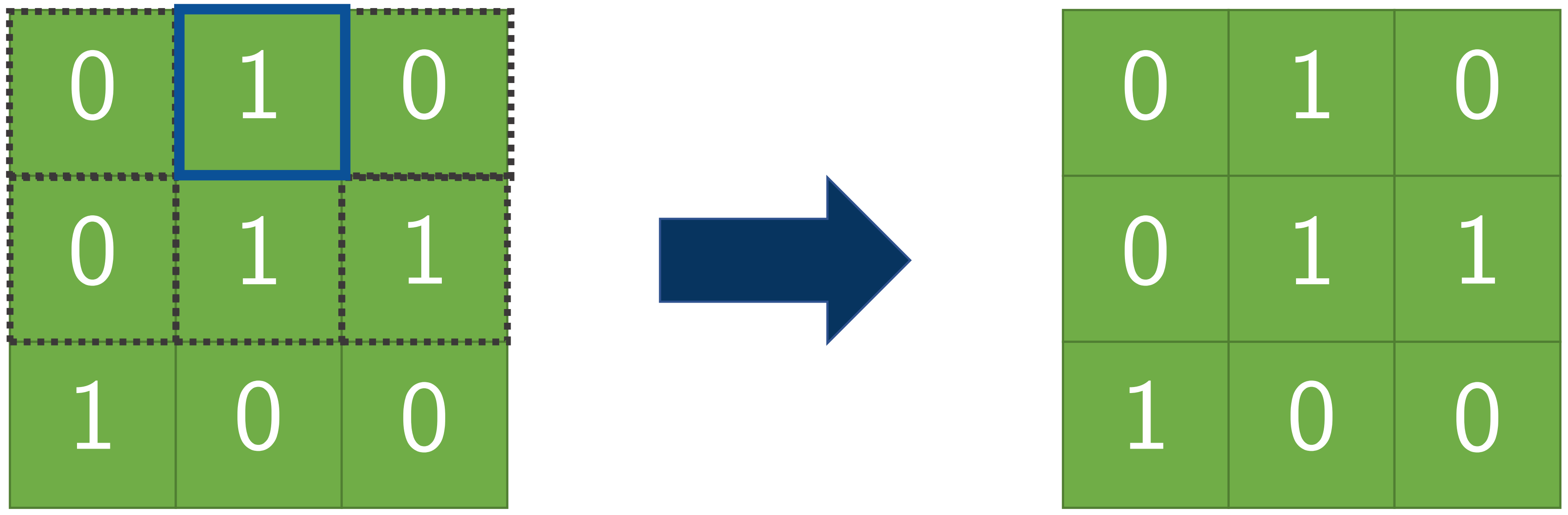
Cellular Automata Example



Conway's Game of life:

- If cell is 1, cell becomes 0 if (neighbours == 1) ≤ 1
- If cell is 0, cell becomes 1 if (neighbours == 1) $= 3$

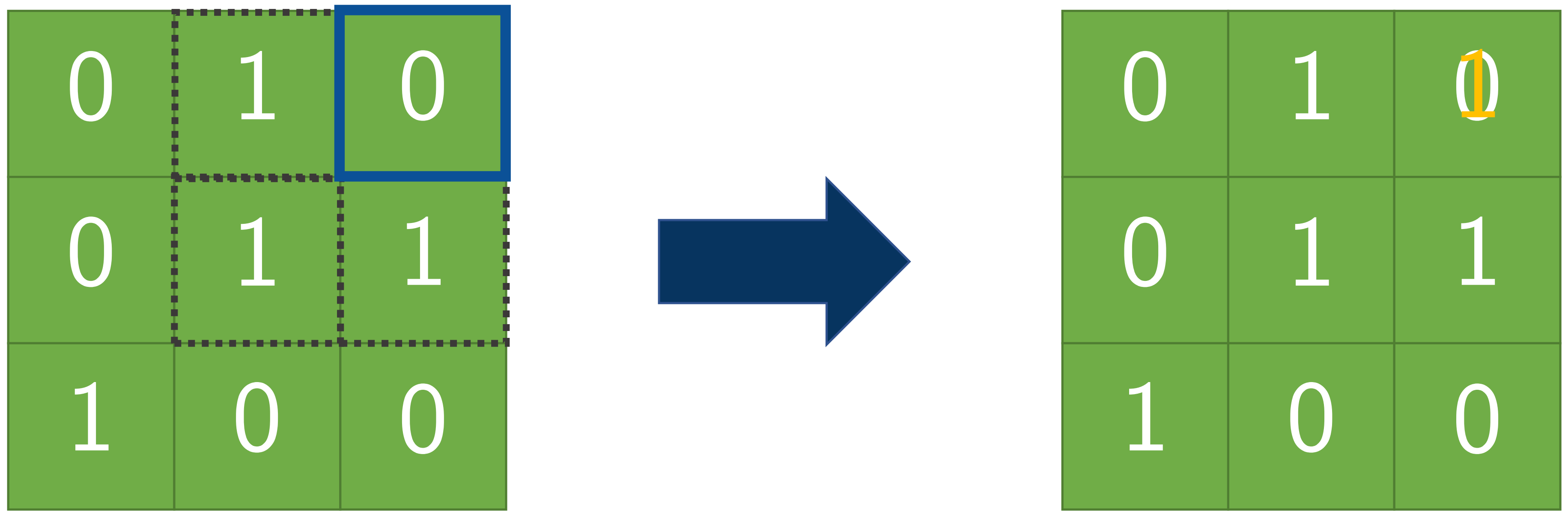
Cellular Automata Example



Conway's Game of life:

- If cell is 1, cell becomes 0 if (neighbours == 1) ≤ 1
- If cell is 0, cell becomes 1 if (neighbours == 1) $= 3$

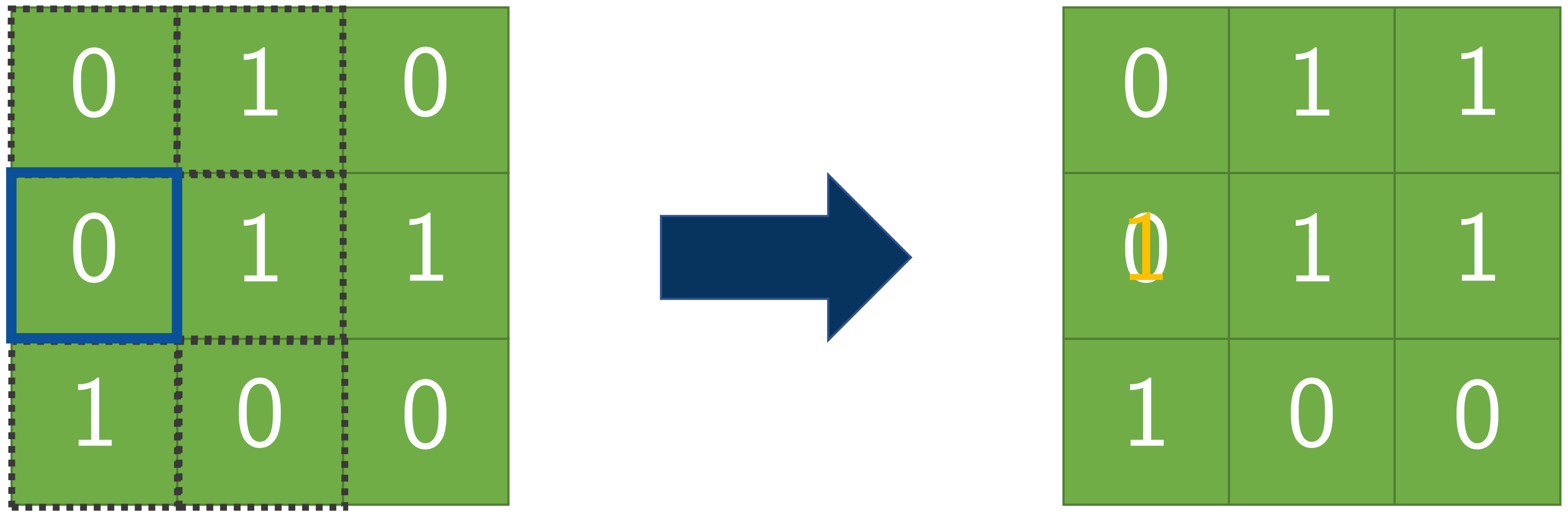
Cellular Automata Example



Conway's Game of life:

- If cell is 1, cell becomes 0 if (neighbours == 1) ≤ 1
- If cell is 0, cell becomes 1 if (neighbours == 1) $= 3$

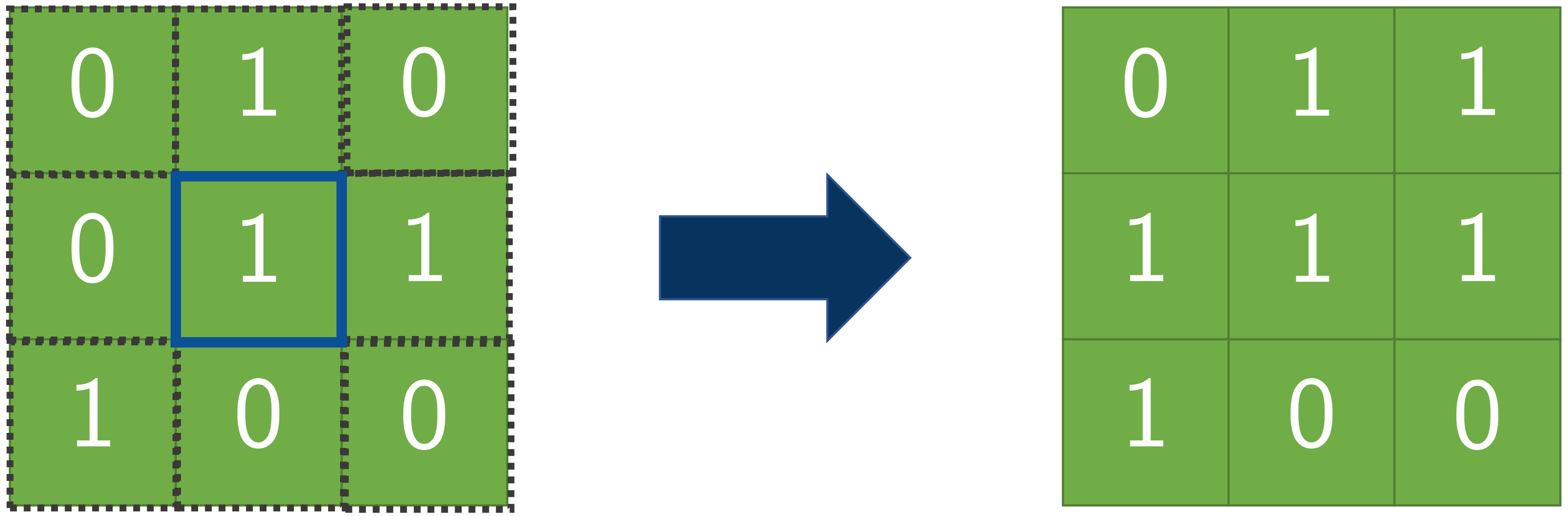
Cellular Automata Example



Conway's Game of life:

- If cell is 1, cell becomes 0 if (neighbours == 1) ≤ 1
- If cell is 0, cell becomes 1 if (neighbours == 1) = 3

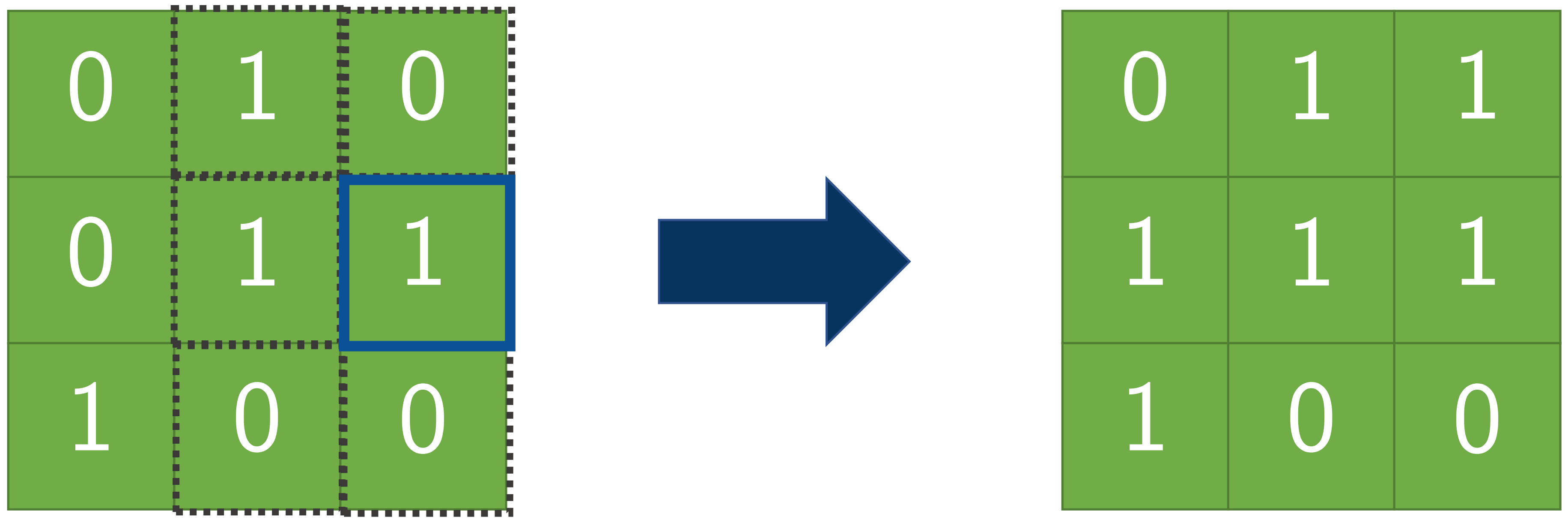
Cellular Automata Example



Conway's Game of life:

- If cell is 1, cell becomes 0 if (neighbours == 1) ≤ 1
- If cell is 0, cell becomes 1 if (neighbours == 1) $= 3$

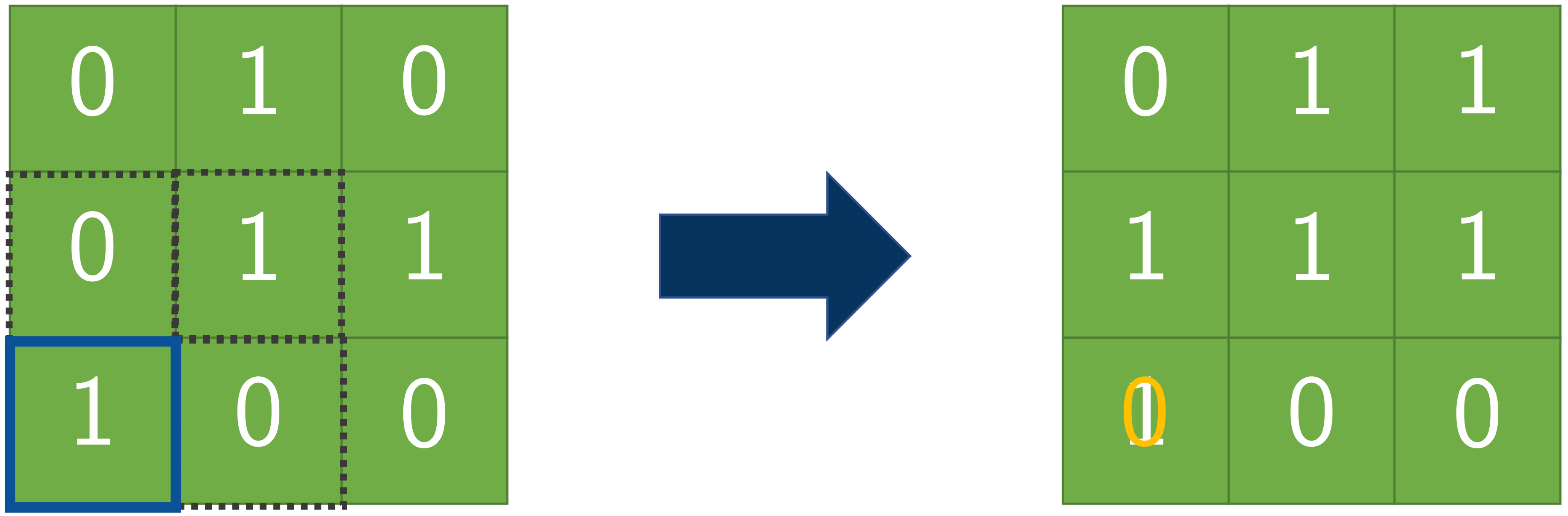
Cellular Automata Example



Conway's Game of life:

- If cell is 1, cell becomes 0 if (neighbours == 1) ≤ 1
- If cell is 0, cell becomes 1 if (neighbours == 1) = 3

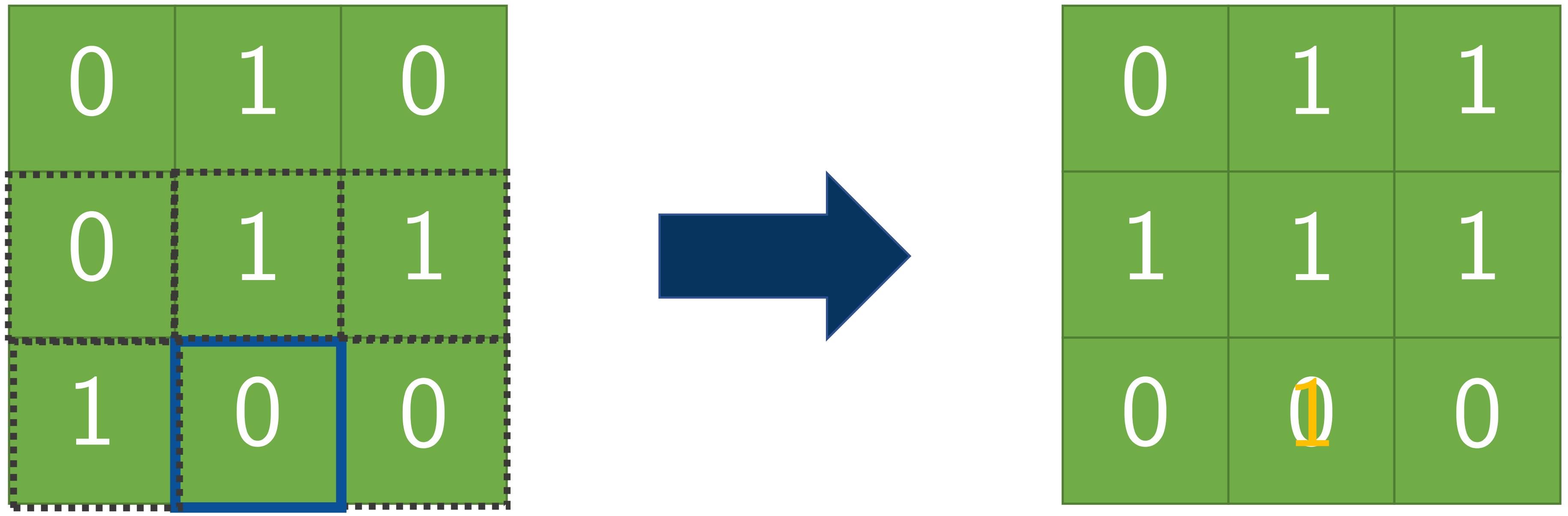
Cellular Automata Example



Conway's Game of life:

- If cell is 1, cell becomes 0 if (neighbours == 1) ≤ 1
- If cell is 0, cell becomes 1 if (neighbours == 1) $= 3$

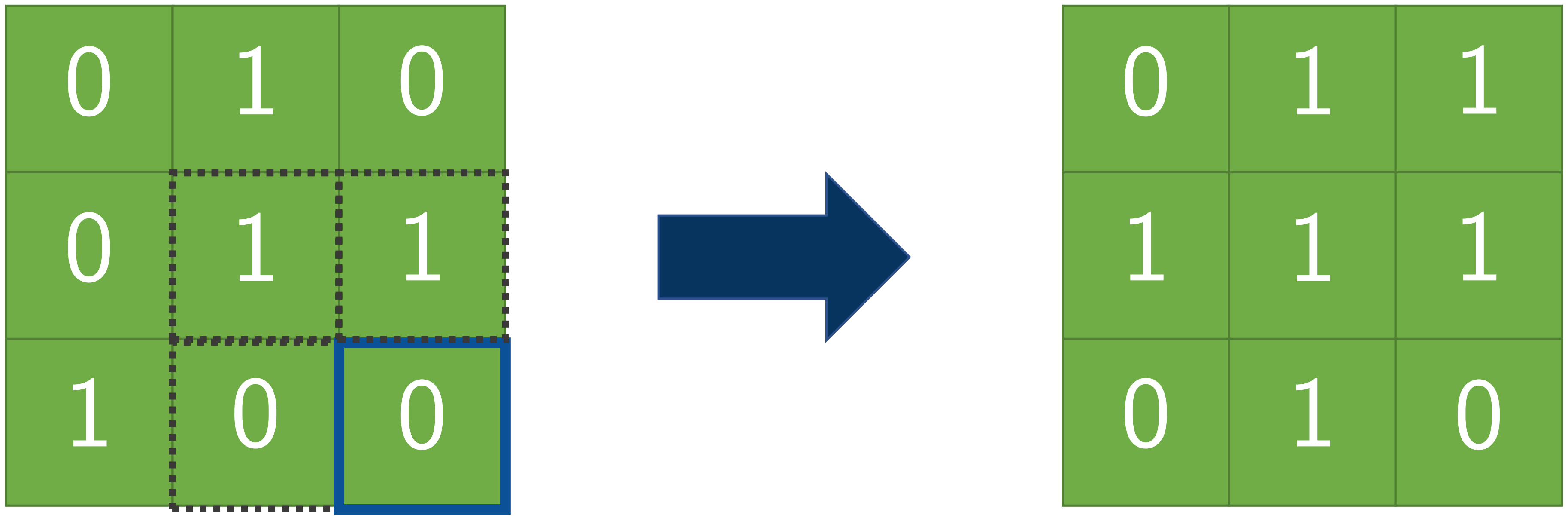
Cellular Automata Example



Conway's Game of life:

- If cell is 1, cell becomes 0 if (neighbours == 1) ≤ 1
- If cell is 0, cell becomes 1 if (neighbours == 1) $= 3$

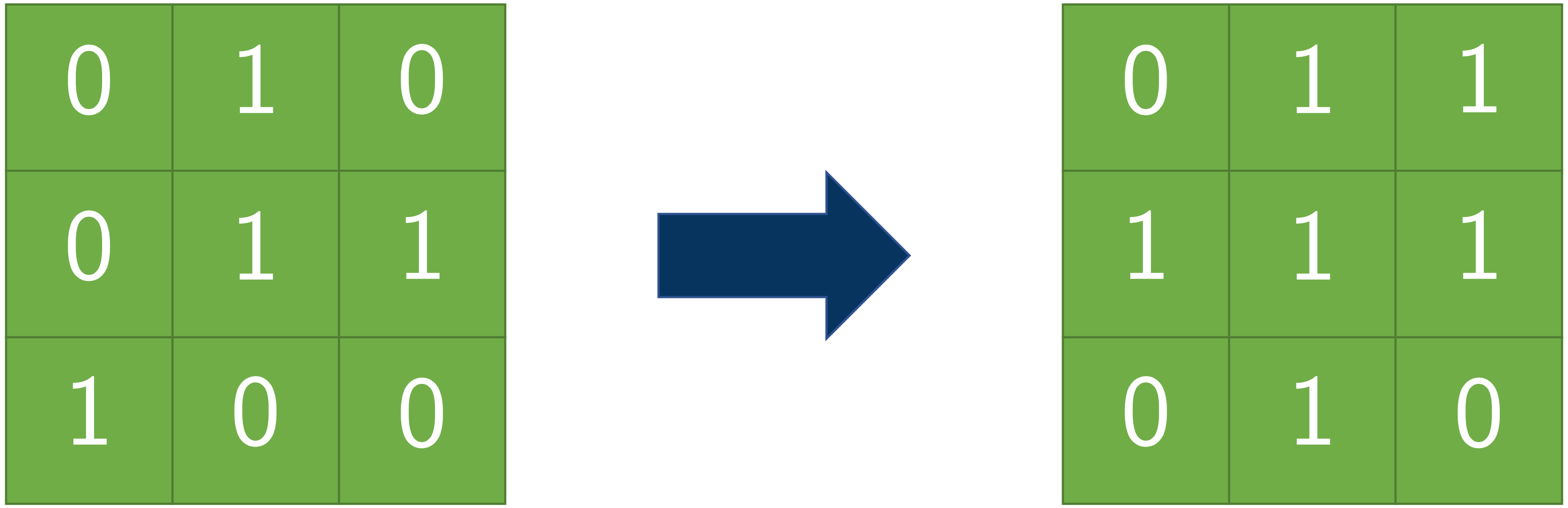
Cellular Automata Example



Conway's Game of life:

- If cell is 1, cell becomes 0 if (neighbours == 1) ≤ 1
- If cell is 0, cell becomes 1 if (neighbours == 1) = 3

Cellular Automata Example



Conway's Game of life:

- If cell is 1, cell becomes 0 if (neighbours == 1) ≤ 1
- If cell is 0, cell becomes 1 if (neighbours == 1) $= 3$

Cellular Automata Example

$$\begin{array}{|c|c|c|}\hline 0 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array} \quad \Longrightarrow \quad \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Conway's Game of life:

- If cell is 1, cell becomes 0 if (neighbours $== 1) \leq 1$
- If cell is 0, cell becomes 1 if (neighbours $== 1) = 3$

Generative Grammars (1)

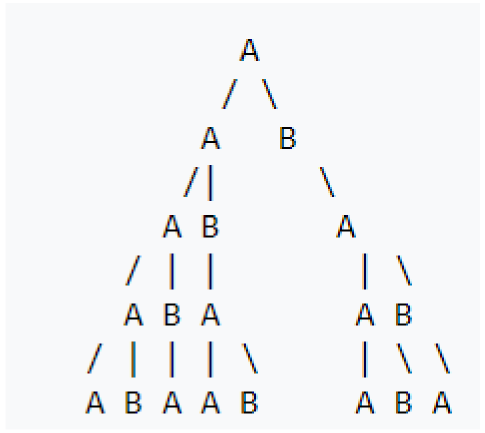
- **Alphabet**: set of symbols.
- **Rules**: rewrite operations (replacing symbols creates new strings).
  - E.g. starting symbol "S" and rule "S → aSb" => "S", "aSb", "aaSbb" etc.
- **Terminals** do not feature on the left-hand side of any rules and cannot be replaced (e.g. a and b above).
- **Non-terminals** (symbols on left-hand side of rules) should not appear in final solution unless valid symbols themselves.
- **Recursion**: the same symbol appears on both sides of the rule.

Generative Grammars (2)

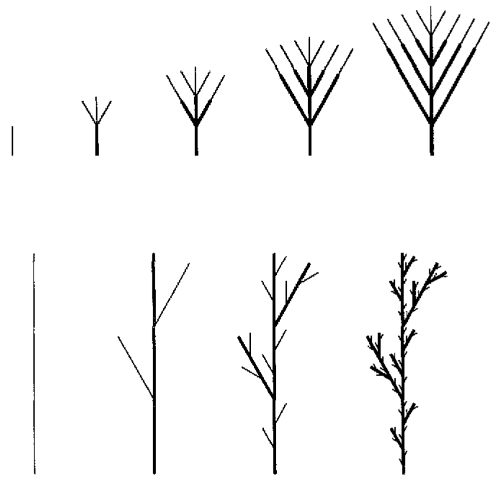
- **Game example**: symbols are game objects/tiles (obstacle, treasure, key, monster etc.)
  - Rules:
    1. **dungeon** → obstacle + treasure
    2. obstacle → key + obstacle + lock + obstacle
    3. obstacle → monster + obstacle
    4. obstacle → room
  - Solutions:
    - **dungeon**
      - (**obstacle** + treasure) (=> 2)
        - (key + **obstacle** + lock + **obstacle**) + treasure (=> 3)
          - key + (monster + **obstacle**) + lock + (monster + **obstacle**) + treasure (=> 4)
            - key + monster + (room) + lock + monster + (room) + treasure
    - **dungeon**
      - (**obstacle** + treasure) (=> 4)
        - (room) + treasure

Generative Grammars (3)

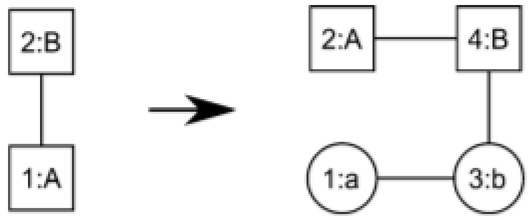
- Lindenmayer Systems (L-Systems): generate tree structures.



```
          A
         / \
        A   B
       /|    \
      A B     A
     / | |    | \
    A B A     A B
   / | | | \  | \ \
  A B A A B   A B A
```

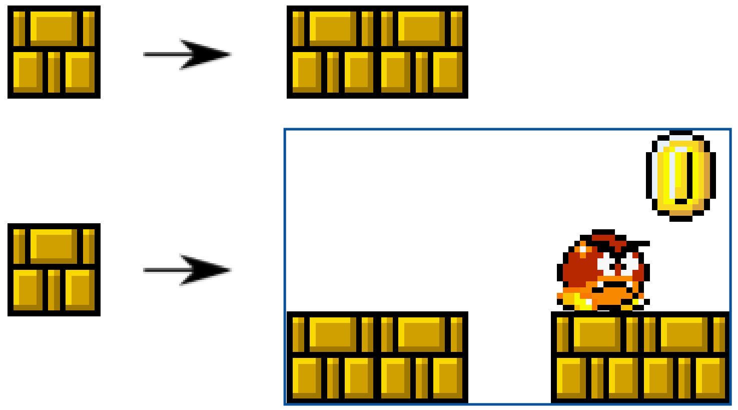


- Grammars can also be applied to graph structures, where symbols are now nodes and edges that connect the nodes.



Generative Grammars (4)

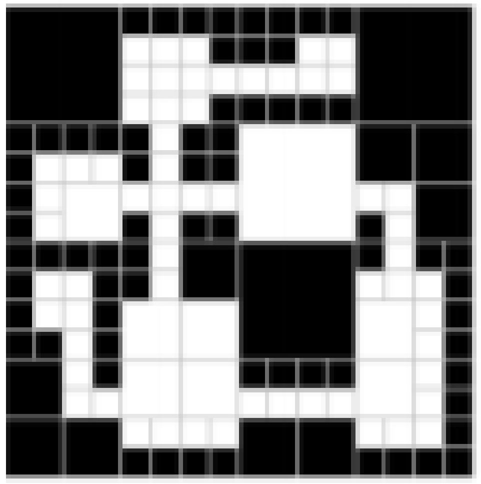
- Trees and graphs are applicable to many areas of games (anything that can be represented as a tree or a graph structure):
  - Level maps
  - Story / mission / gameplay graph
  - Game objects (actual trees!)
  - etc.

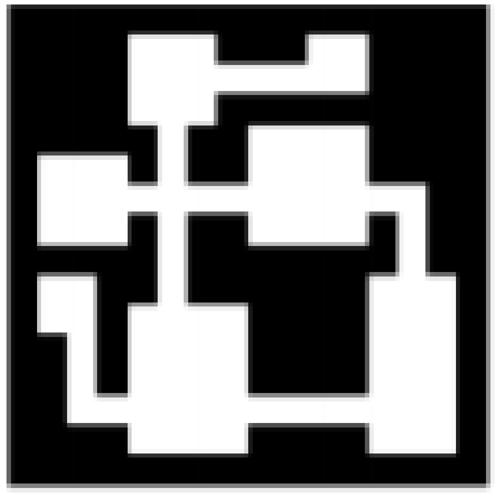


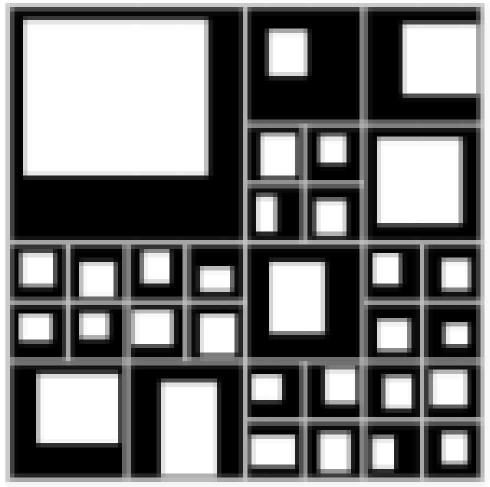
Dormans, J. and Bakkes, S., 2011. Generating missions and spaces for adaptable play experiences. *IEEE Transactions on Computational Intelligence and AI in Games*, *3*(3), pp.216-228.

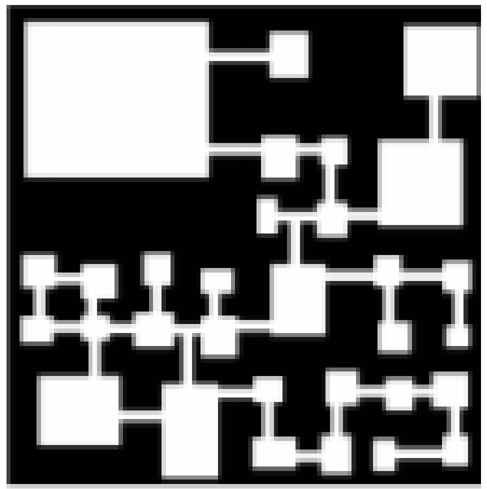
Space Partitioning

- Iteratively split the game areas into smaller sections.
- And split the sections (or not, probabilities) again.
- Dungeon application: each section becomes a room, connect the rooms with their neighbours via corridors at the end.



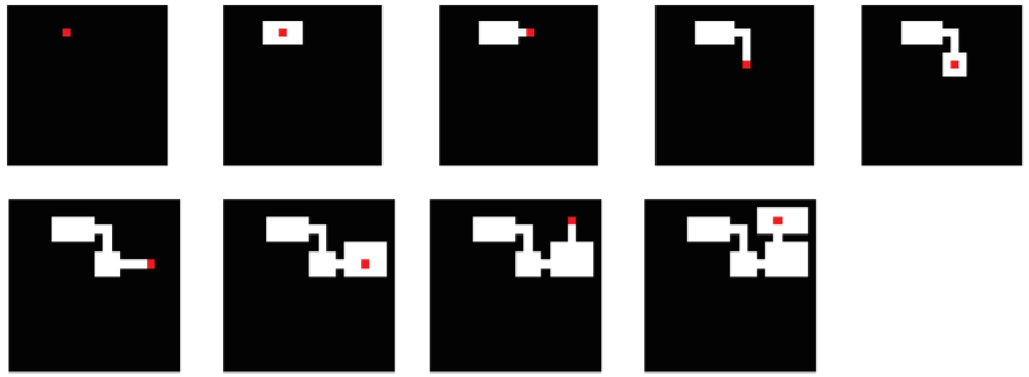






Others

- Declarative modelling:
  - List of facts describing the model.
  - *What* should be generated instead of *how.*
- Agent-based terrain shaping.



Shaker, N., Liapis, A., Togelius, J., Lopes, R. and Bidarra, R., 2016. Constructive generation methods for dungeons and levels. In *Procedural Content Generation in Games* (pp. 31-55). Springer, Cham.

Outline

- ✓ Introduction
- ✓ Applications
- ✓ PCG Types
  - ✓ Level Generation
  - ✓ Rules & Game Mechanics
  - ✓ Others
- ❑ Methods
  - ✓ Constructive

Labs and Assignment 2

- Mario AI: level generation
- 2 labs exercises + 3 labs assignment
- Groups of 3 (can be different!)

- Set: 18th November 2019
- Due: 13th December 2019

EECS Research Showcase

- **December 5th 2019 @ The Octagon**, QMUL
- 13:00-15:30

- Posters
- Demos
- Industry visitors
- Coffee & cake!

- PhD Opportunities
  - QMUL Game AI Group: http://gameai.eecs.qmul.ac.uk
  - Intelligent Games and Games Intelligence (IGGI): http://iggi.org.uk