

Procedural Level Generation in Games

Kim Pedersen

Tech talk brought to you by ...



Ray Wenderlich



Wes Paugh



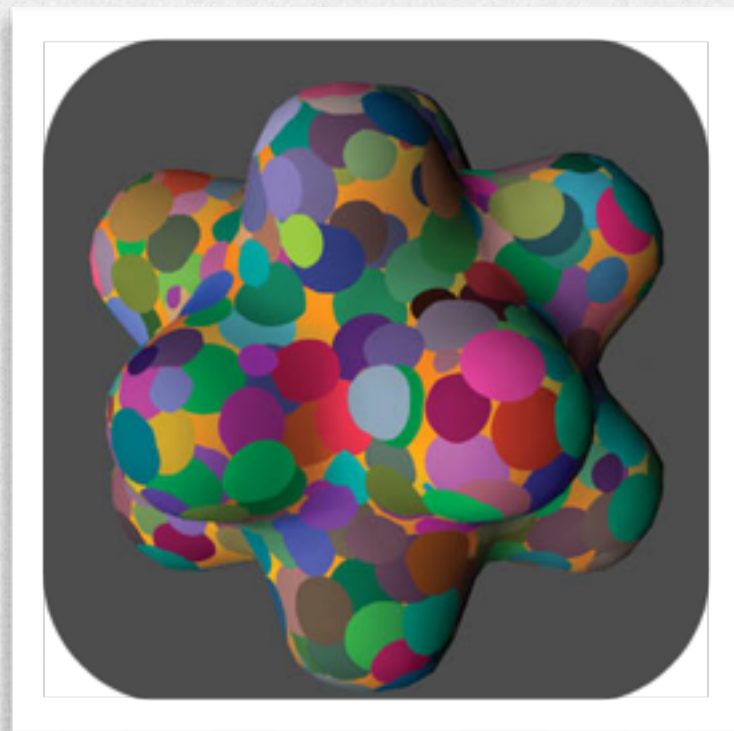
Kim Pedersen



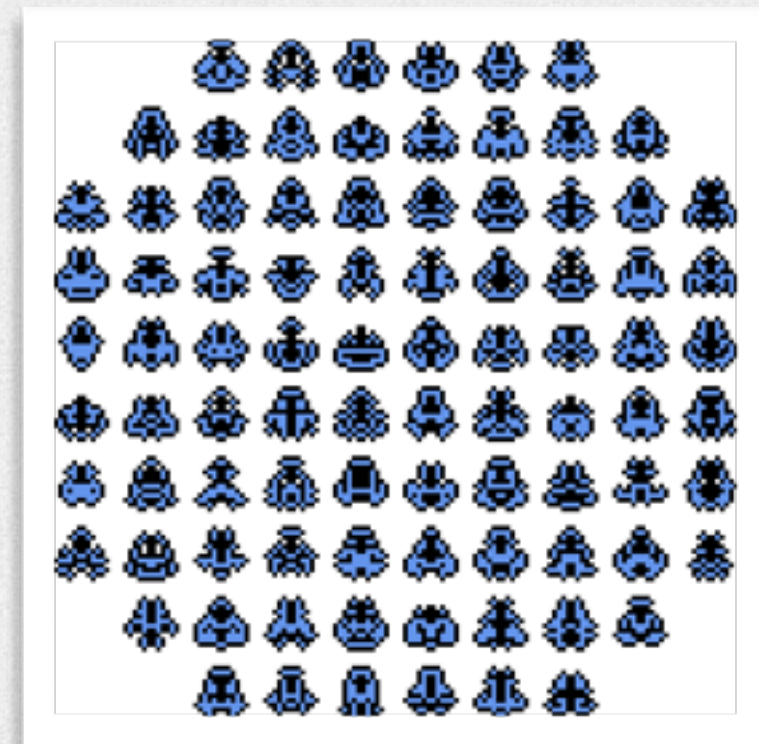
Definition

“ Procedural Content Generation (PCG) is the **programmatic generation** of game **content** using a **random or pseudo-random process** that results in an **unpredictable range** of **possible** game play spaces “

Examples of procedural content



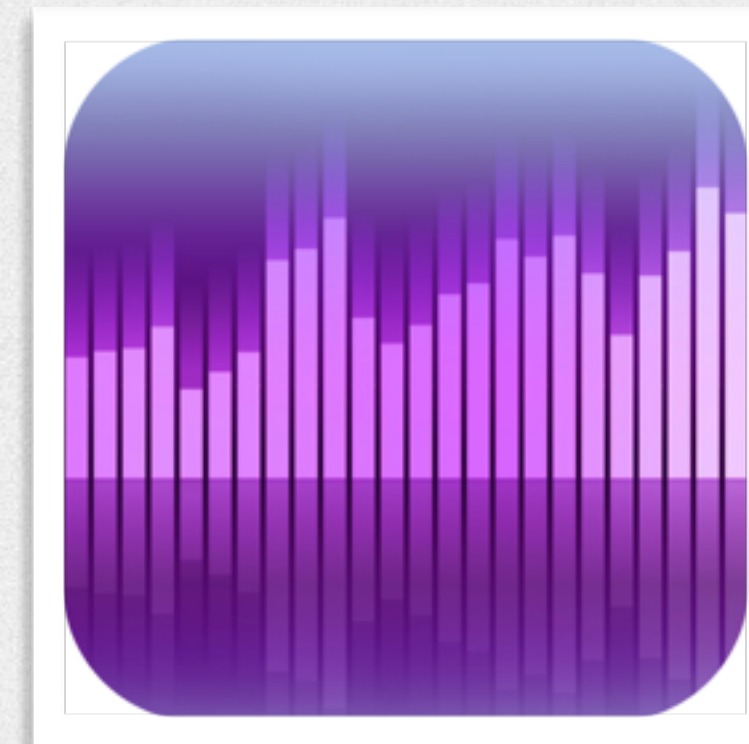
Textures



Sprites



3D models



Sound



Levels etc...

Why all the fuzz about Procedural Content Generation?

- ⚙ Reduce time generating content
- ⚙ Reduce game footprint
- ⚙ Greater variety in content.
- ⚙ Content that would otherwise be impossible/impractical to create by hand
- ⚙ Enhance replayability

Examples of games using procedural content



Tiny Wings



Spelunky



100 Rogues



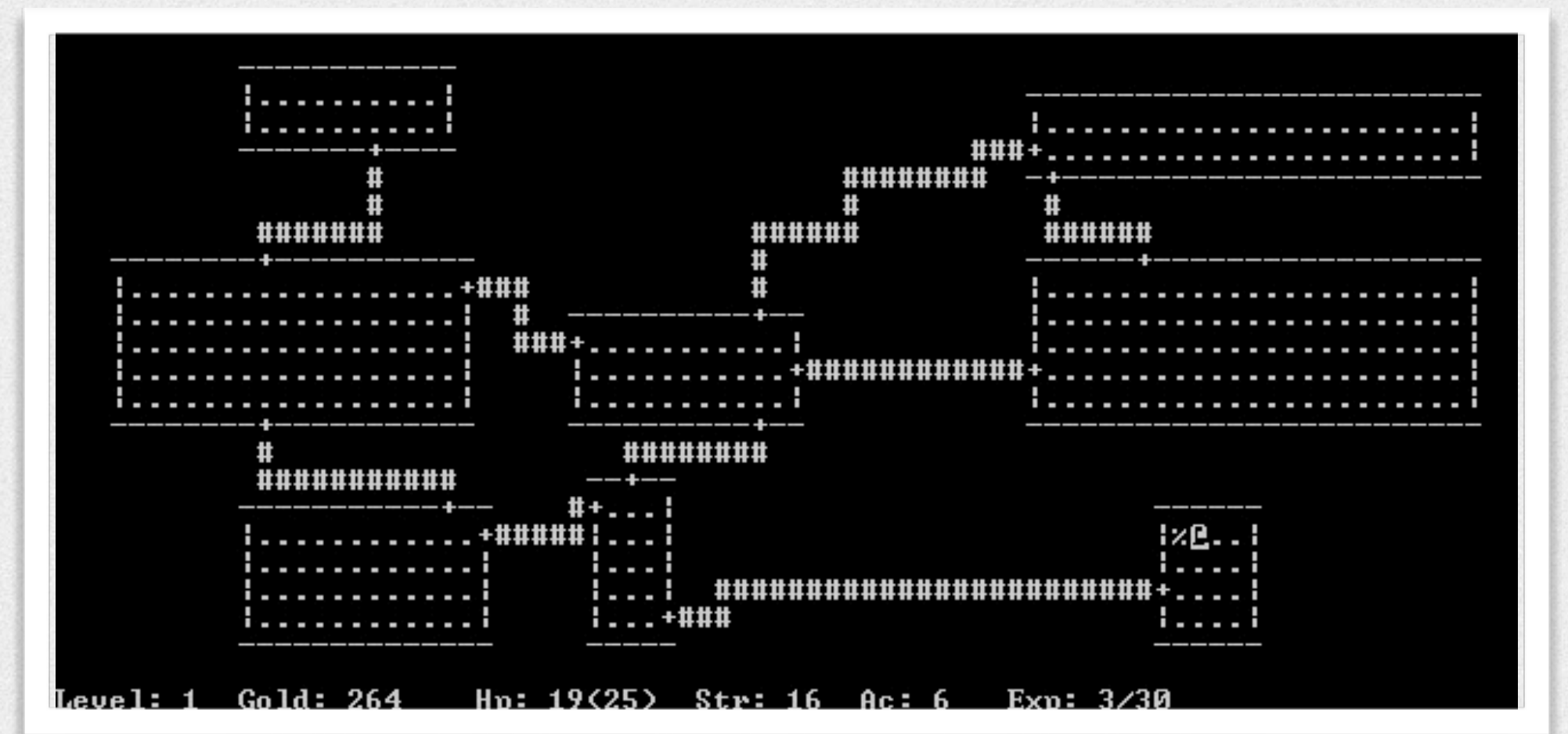
Diablo



Minecraft

Rogue

- ⚙️ Developed on Unix by Michael Toy and Glenn Wichman around 1980
- ⚙️ One of the first games to use procedural levels
- ⚙️ Has lead to a class of games called “roguelikes” characterized by procedural level generation, tile-based graphics and permadeath

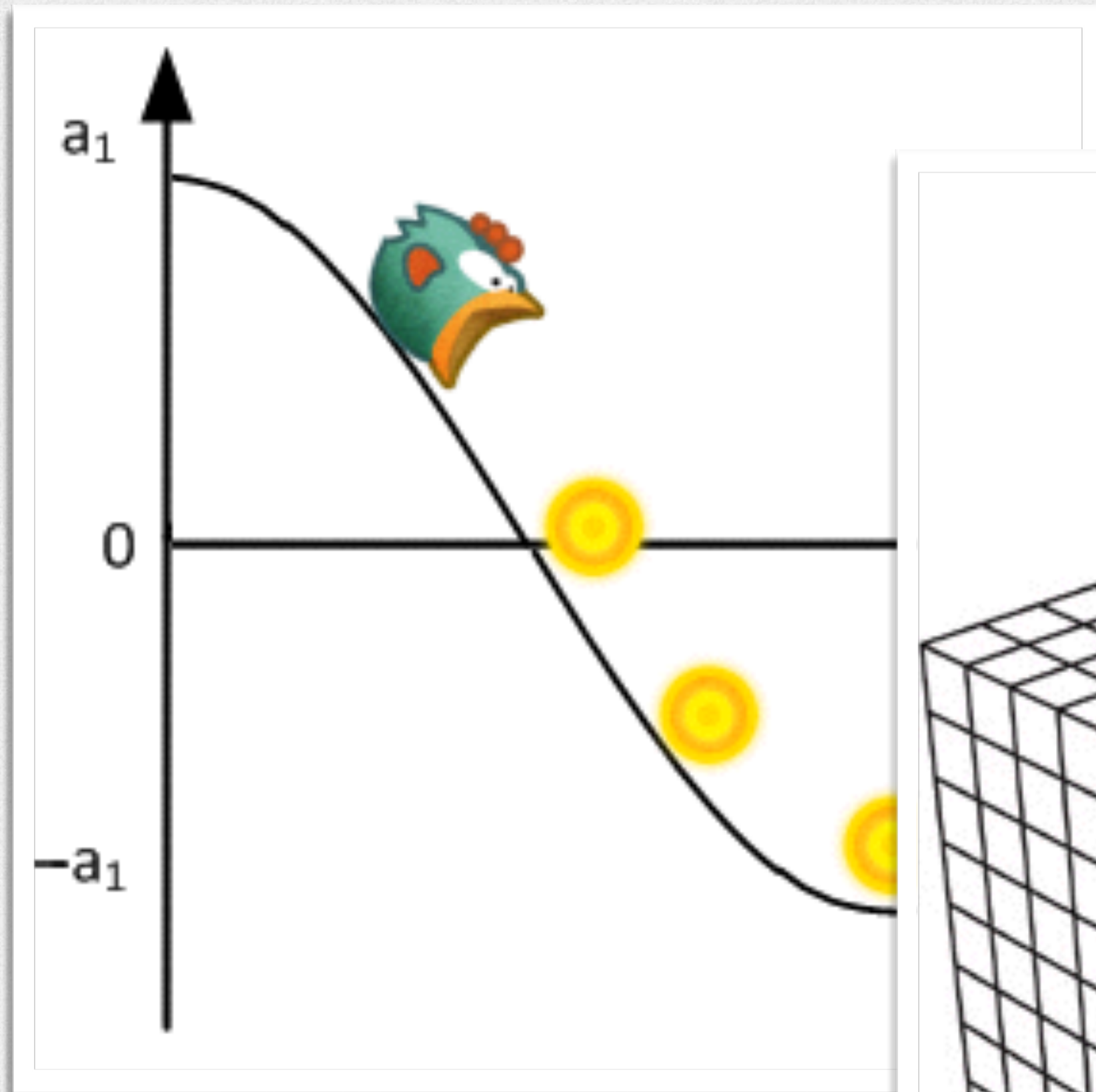


Procedural Level Generation

- ⚙ Procedural levels can be generated in 1d, 2d, 3d

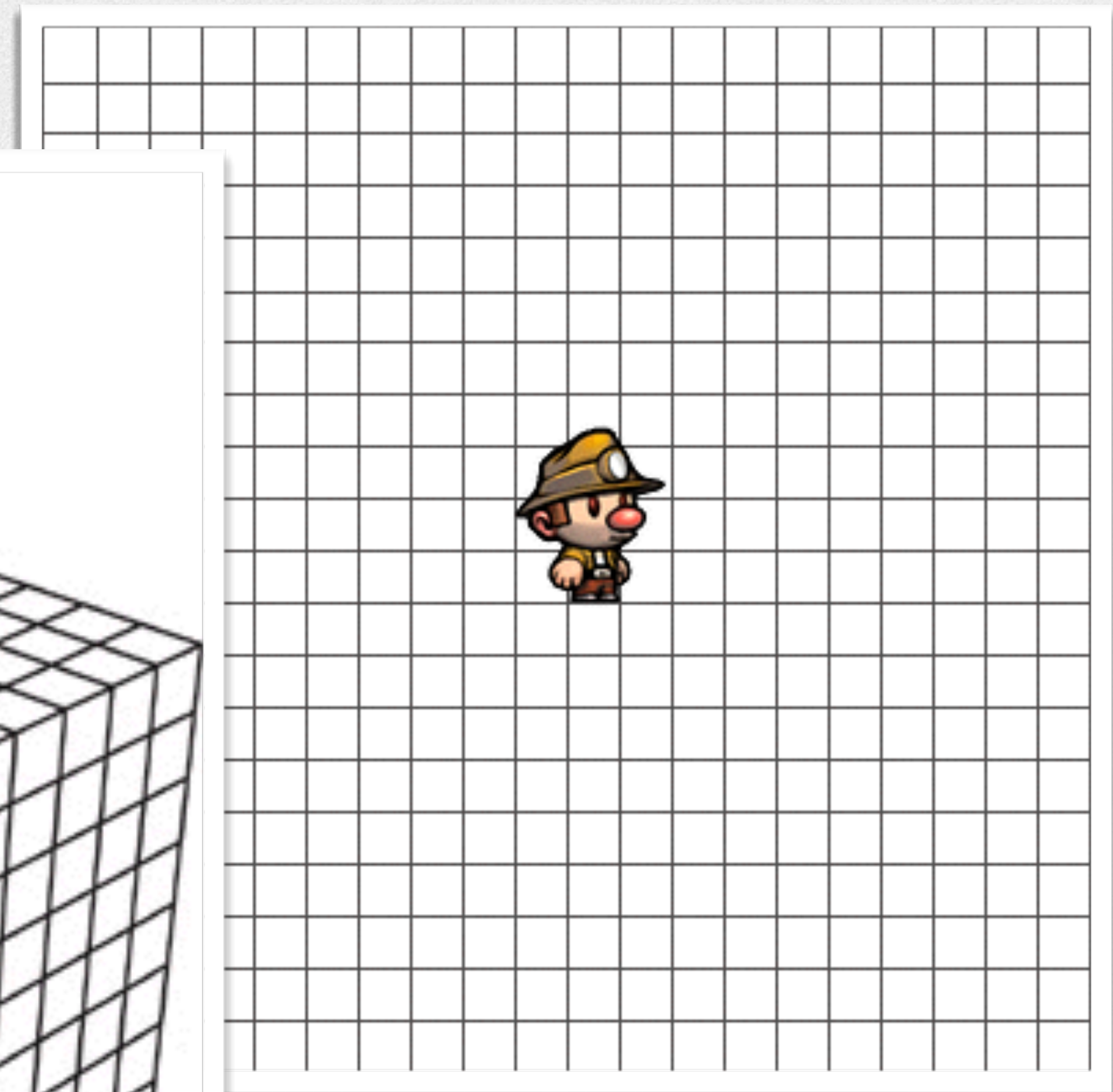
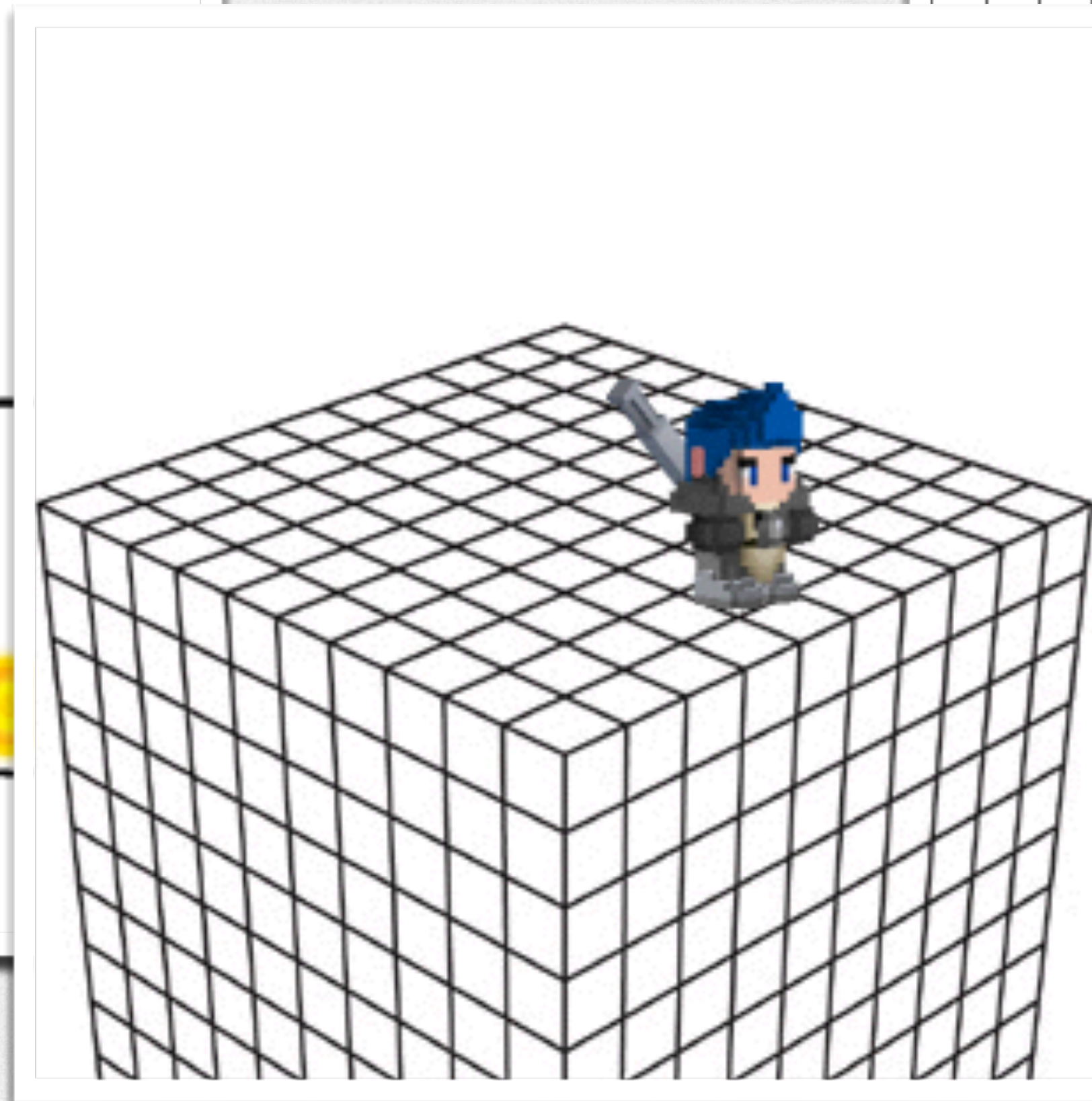


Procedural Level Generation



Tiny Wings

Cube World



Spelunky

Procedural Level Generation

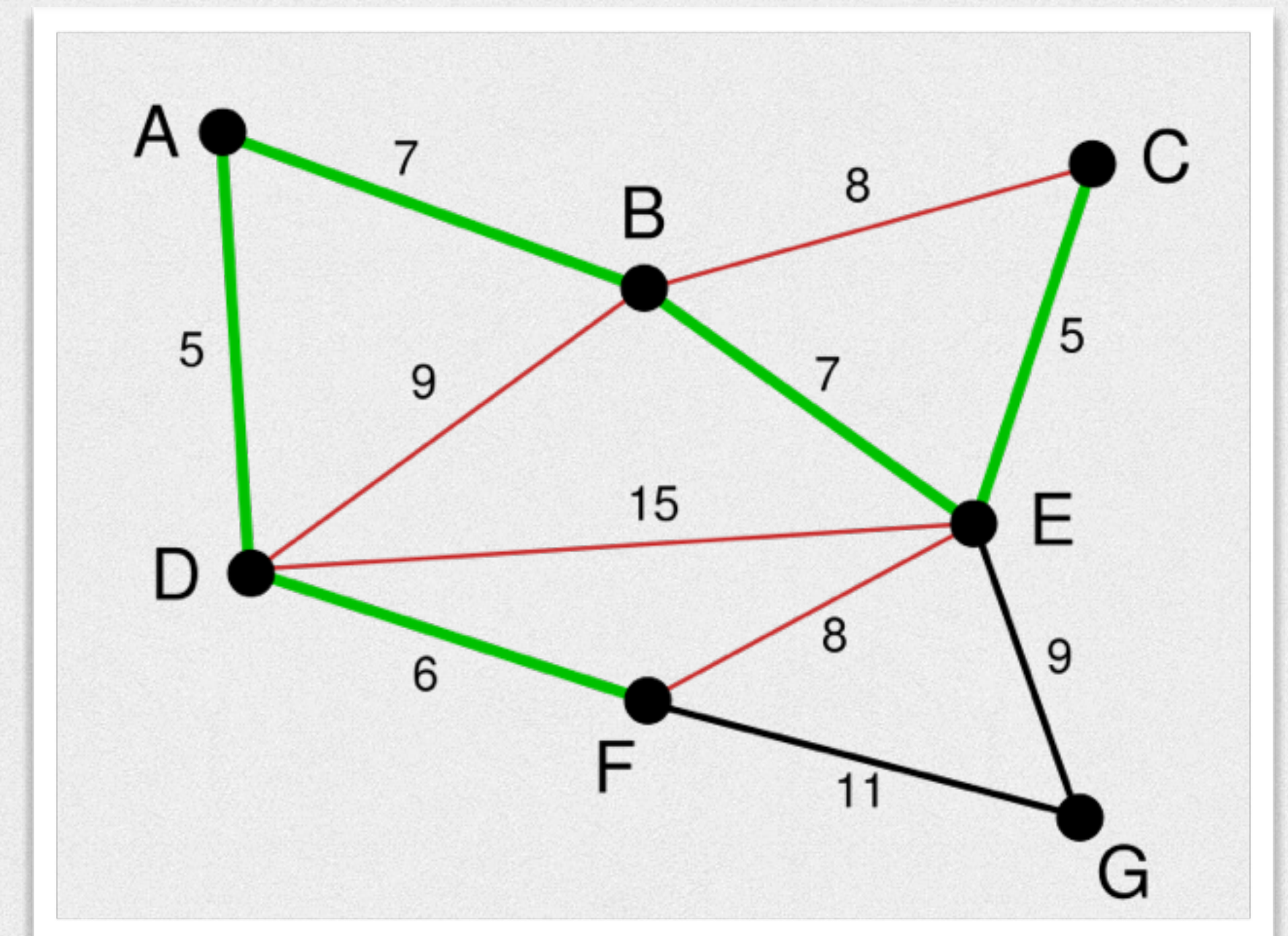
- ⚙ Procedural levels can be generated in 1d, 2d, 3d
- ⚙ Generated programmatically using an algorithm and pseudo-random parameters



Algorithms in procedural level generation

Several popular algorithms are used in procedural level generation:

- ⚙ **Agent-based dungeon growing** (Drunkard Walk)
- ⚙ **Space Partitioning** (Binary Space Partitioning (BSP))
- ⚙ **Cellular automata**
- ⚙ **Noise** (Perlin noise, Simplex noise)
- ⚙ **.. or roll your own algorithm.**



Get more inspiration from RogueBasin: <http://pcg.wikidot.com/pcg-algorithm:map-generation>

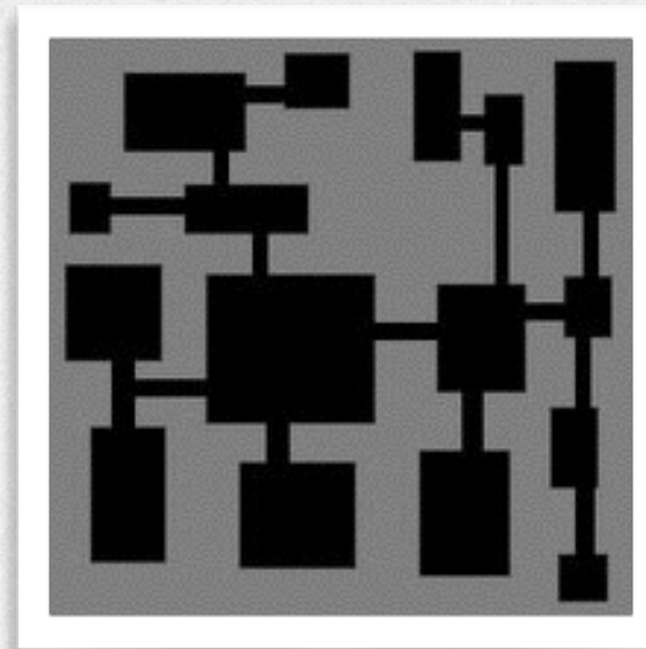
raywenderlich.com

Algorithms in procedural level generation

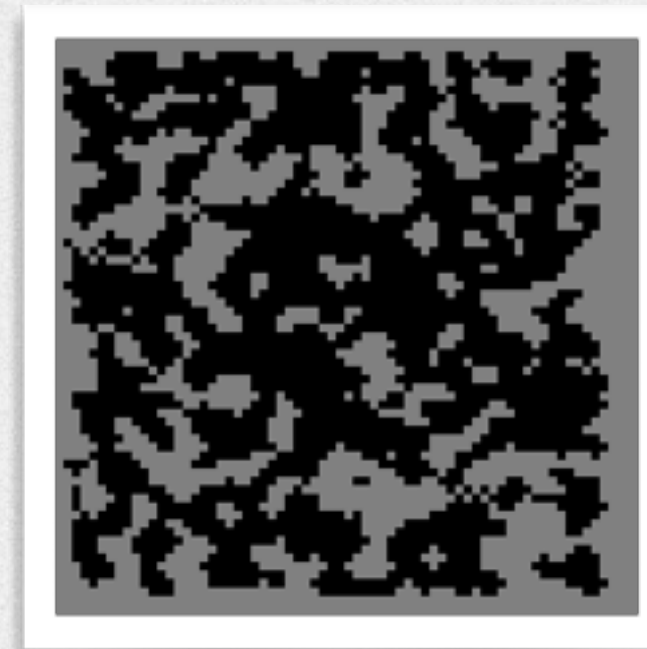
**Agent based
dungeon growing**



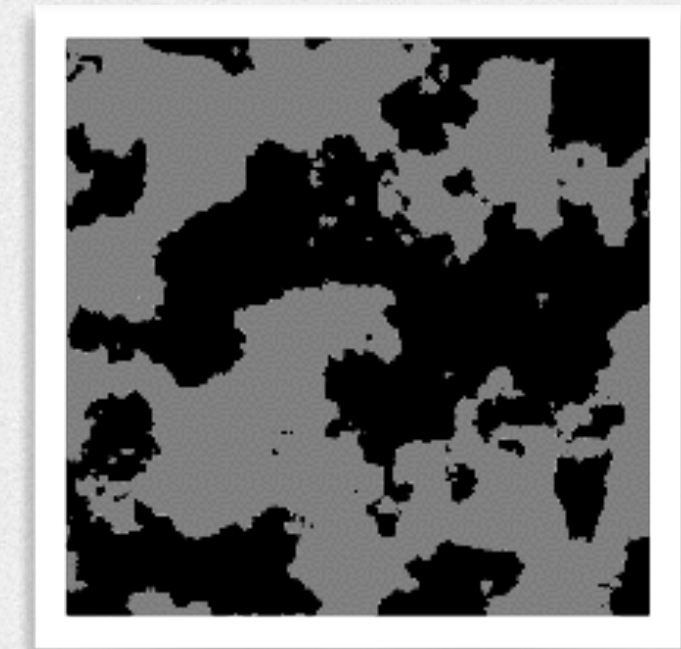
**Space
Partitioning**



**Cellular
Automata**



Noise



Useful for:

Organic or chaotic dungeons

Neatly laid out dungeons with
rooms connected by
corridors

Organic looking caves or
islands

Maps and terrain

Complexity:

Simple

Simple

Medium

Hard

Connected:

Yes

Yes

No

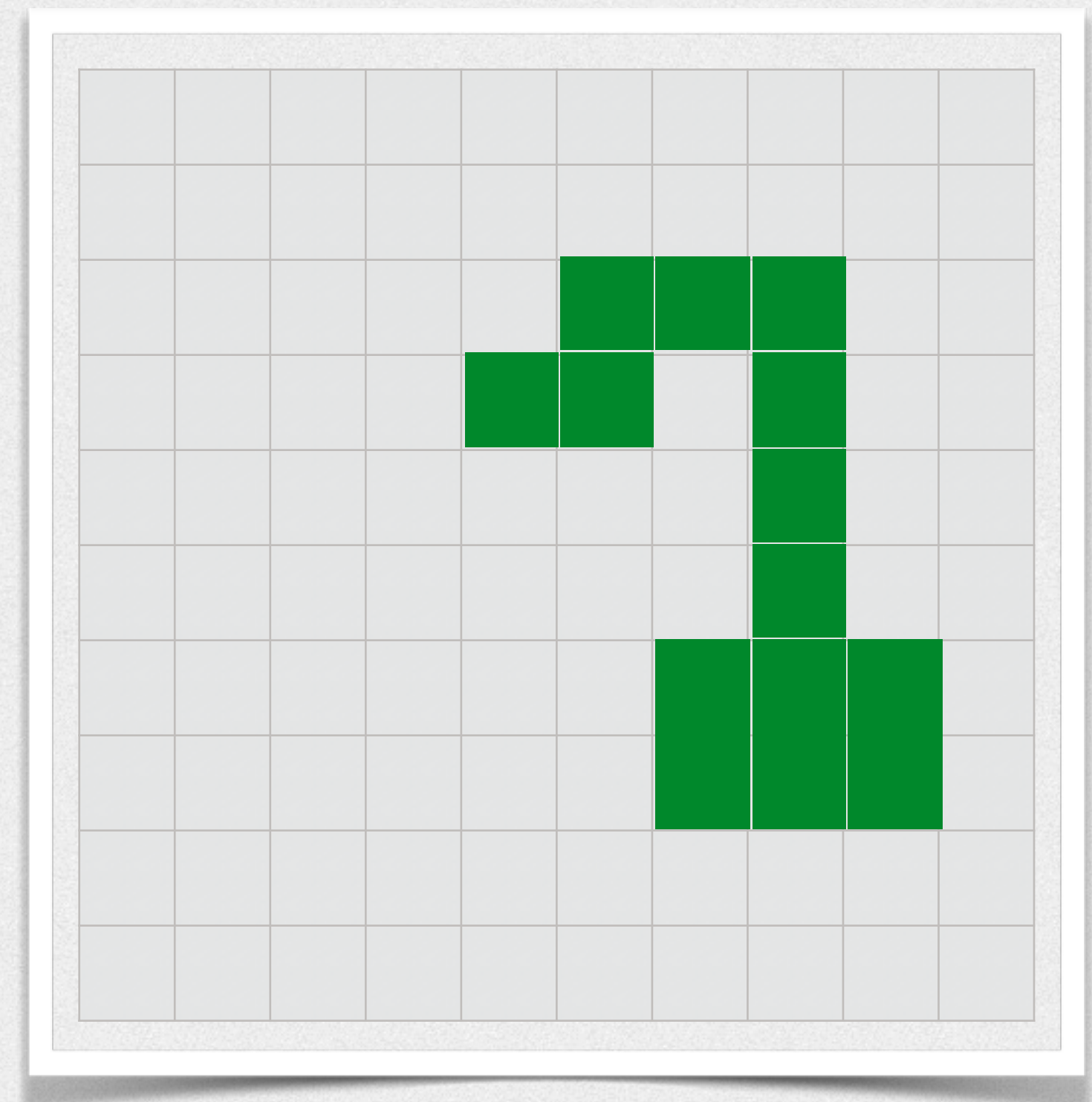
No

Get more inspiration from RogueBasin: <http://pcg.wikidot.com/pcg-algorithm:map-generation>

raywenderlich.com

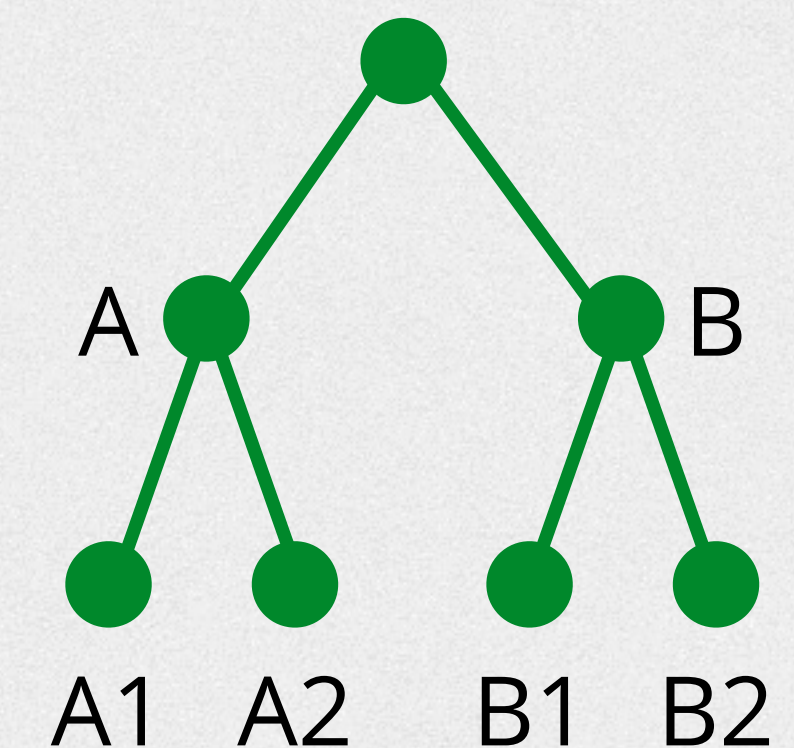
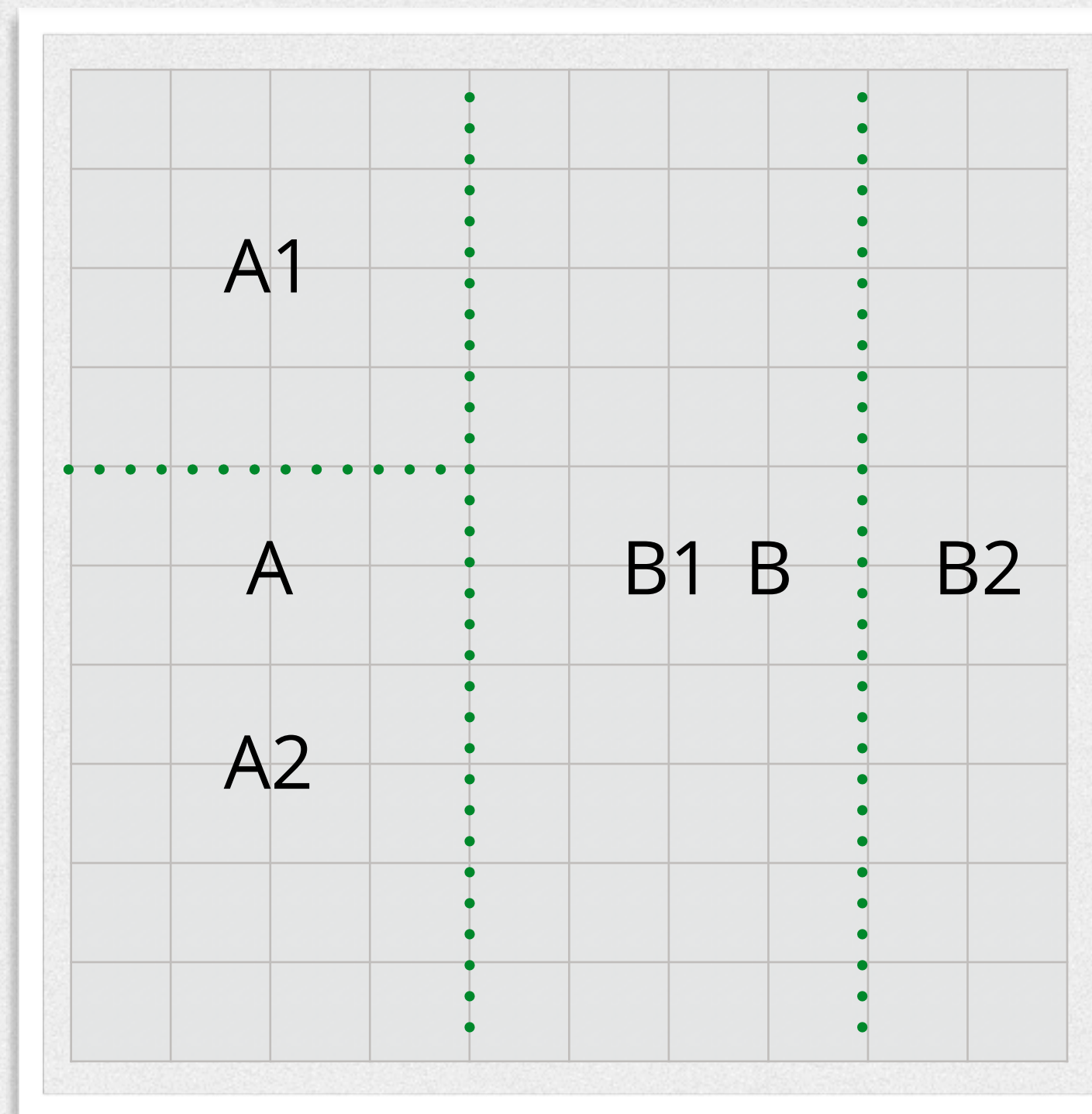
Agent based dungeon growing

1. Choose a random start position
2. Pick a random direction to move
3. Move in that direction and mark the position as a floor or room, unless it already is a floor.
4. Repeat steps 2 and 3



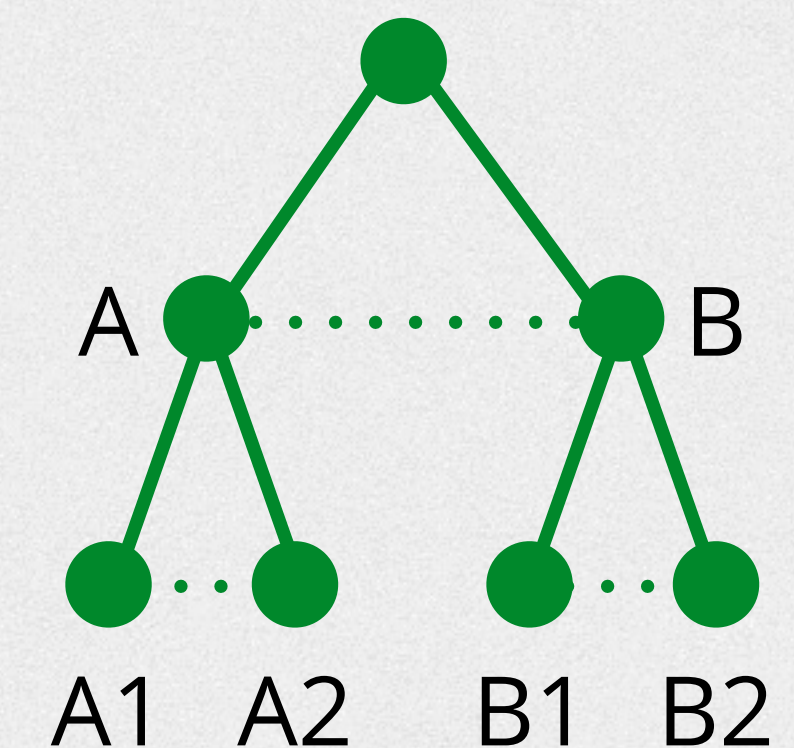
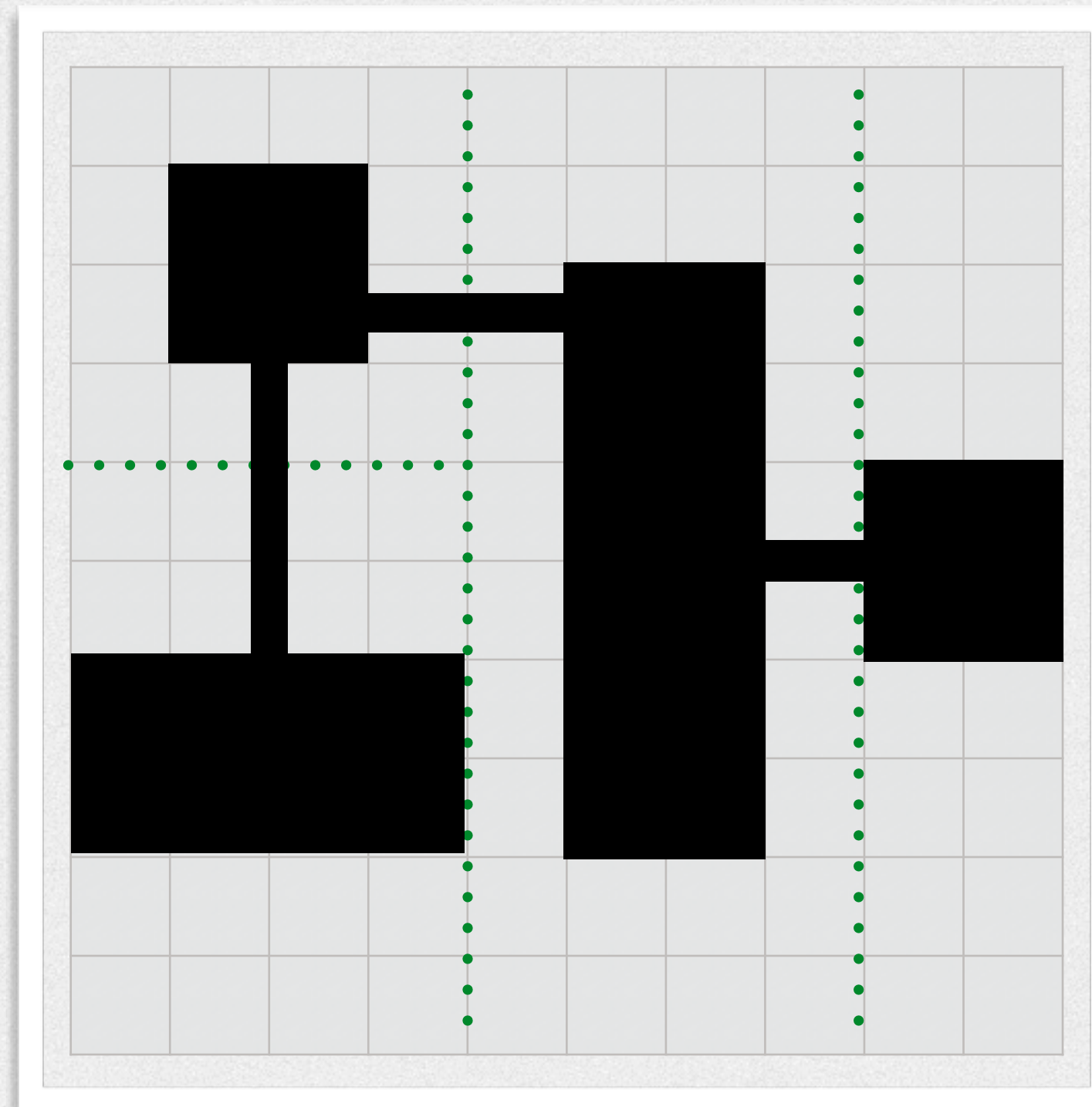
Space Partitioning - Step 1-4

1. Choose to split horizontally or vertically
2. Choose a random position (x for vertical, y for horizontal)
3. Split the dungeon into two sub-dungeons (leafs)
4. Repeat steps 1 and 3 for each leaf



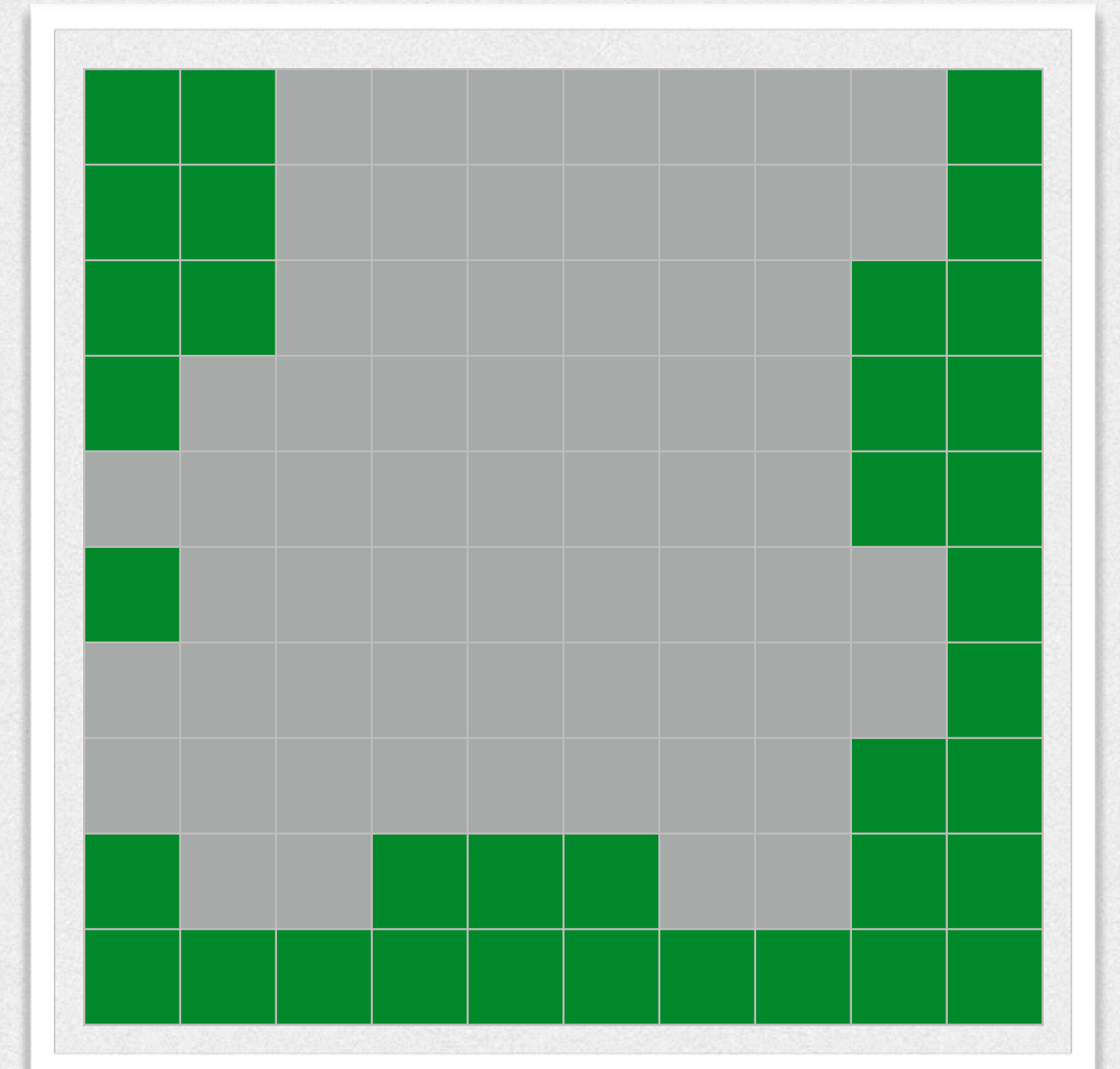
Space Partitioning - Step 5-8

5. Create a room with random size in each leaf of the tree
6. Loop through all the leafs of the tree, connecting each leaf to its sister
7. Go up one level in the tree and repeat the process for the parent sub-regions
8. Repeat the process until the first two sub-dungeons A and B are connected



Cellular Automata

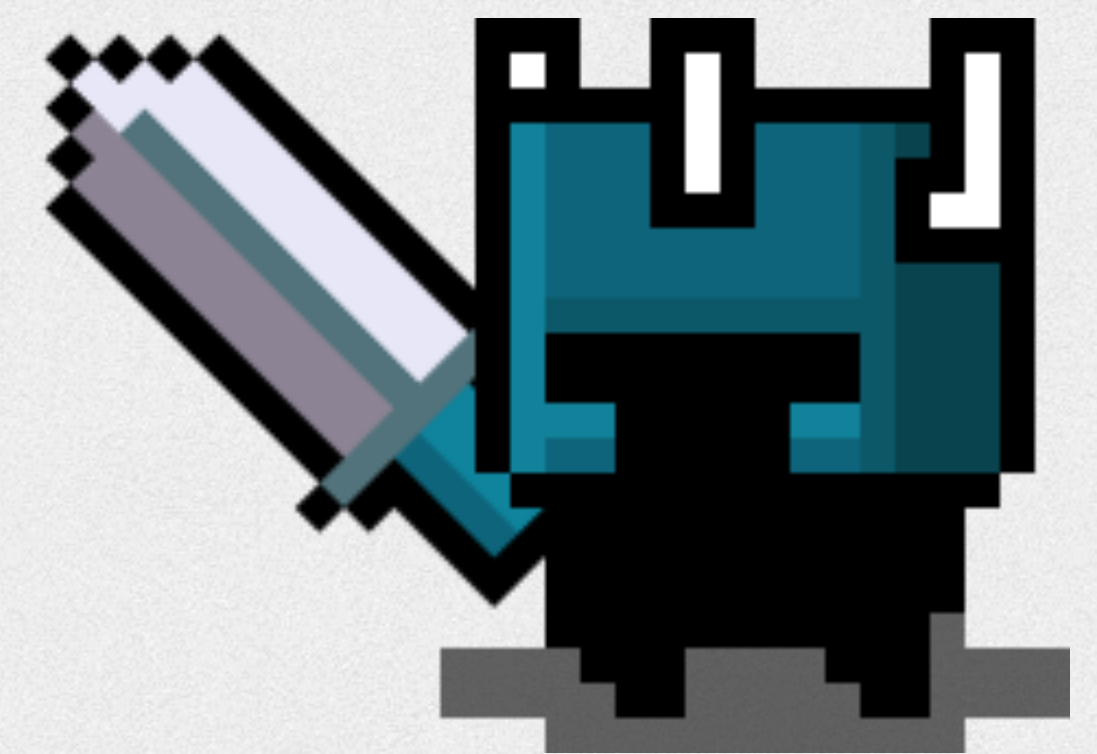
1. Set the state of the cells in the grid to either on or off
2. Apply transition rules to all grid cells simultaneously
3. Repeat step 2 a desired number of times



Building a procedural level

This tech talk will examine implementing a cave-like level for a top-down 2D RPG type game.

- ⚙ Cave should appear to be organic
- ⚙ All parts of the level should be reachable
- ⚙ The exit must be reachable from the entrance



Steps in Procedural Level Generation

A simple approach to procedural level generation in games:

- ⚙️ **Create an abstract level class**

This class will handle generic level rendering, stores level data and has methods to query the level for helpful information.

- ⚙️ **Implement procedural algorithm to generate the level**

There are many algorithms with varying difficulty to implement that will give different visual results. Pick your algorithm wisely.

- ⚙️ **Place loot, add enemies and generate obstacles/quests etc** *(not in scope for this tech talk)*

Adding the content to the level is often the most difficult part of generating interesting levels.

- ⚙️ **Determine fitness of level**

All levels in a game should be feasible, interesting and target an appropriate skill level.



Abstract level class

The screenshot shows the Xcode IDE with the following components:

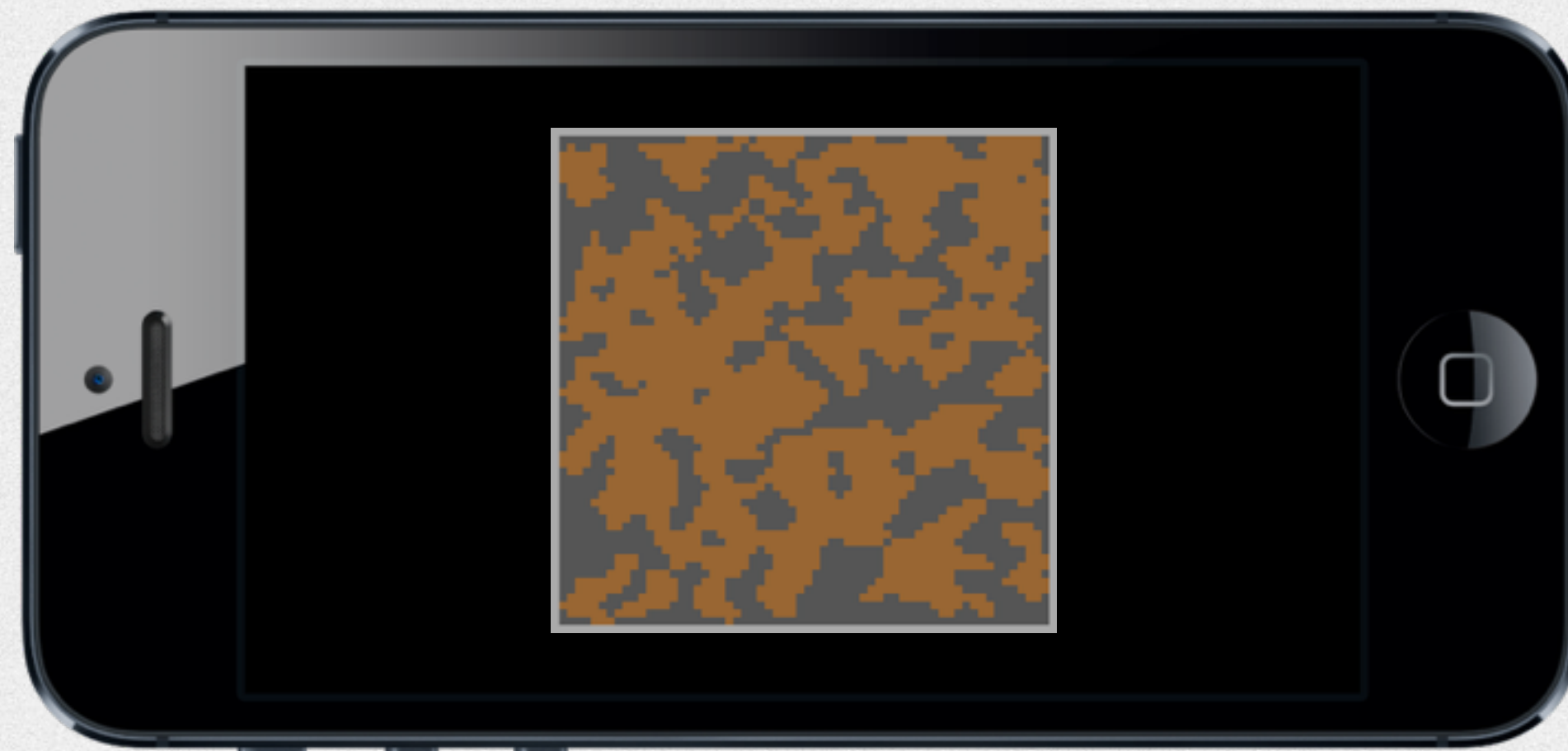
- Project Navigator (Left):** Shows the project structure for 'CellularAutomataTechTalkVersion'. The file 'AbstractLevel.h' is selected.
- Editor (Center):** Displays the code for 'AbstractLevel.h'. The code includes a header comment, an import for 'SpriteKit', a typedef for 'LevelCellType', an '@interface LevelCell' block with properties and methods, and an '@interface AbstractLevel' block with properties and methods.
- Inspector (Right):** Shows the 'Identity and Type' panel for 'AbstractLevel.h', including its name, type, location, and full path.
- Console (Bottom):** Displays the output of the application, showing the generation of a cave and the number of caverns in the map.

```
1 //
2 // AbstractLevel.h
3 // CellularAutomataTechTalkVersion
4 //
5 // Created by Kim Pedersen on 06/05/14.
6 // Copyright (c) 2014 Kim Pedersen. All rights reserved.
7 //
8
9 #import <SpriteKit/SpriteKit.h>
10
11 typedef NSInteger LevelCellType
12 {
13     LevelCellTypeInvalid = -1,
14     LevelCellTypeWall,
15     LevelCellTypeFloor,
16     LevelCellTypeExit,
17     LevelCellTypeMax,
18 };
19
20 @interface LevelCell : NSObject <NSCopying>
21
22 @property (assign, nonatomic) CGPoint coordinate;
23 @property (assign, nonatomic) LevelCellType type;
24 @property (assign, nonatomic) BOOL isDestructible;
25 @property (weak, nonatomic) SKSpriteNode *tile;
26
27 - (instancetype) initWithCoordinate:(CGPoint)coordinate type:(LevelCellType)type;
28
29 @end
30
31 @interface AbstractLevel : SKNode
32
33 @property (readonly, assign, nonatomic) CGSize gridSize;
34 @property (readonly, assign, nonatomic) CGSize tileSize;
35 @property (strong, nonatomic) NSMutableArray *grid;
36
37 - (instancetype) initWithGridSize:(CGSize)size;
38 - (void) generate;
39 - (BOOL) isValidGridCoordinate:(CGPoint)coordinate;
40 - (BOOL) isEdgeAtGridCoordinate:(CGPoint)coordinate;
41 - (NSInteger) indexForGridCoordinate:(CGPoint)coordinate;
42 - (CGPoint) positionForGridCoordinate:(CGPoint)coordinate;
43 - (CGPoint) gridCoordinateForPosition:(CGPoint)position;
44 - (CGRect) cellRectFromGridCoordinate:(CGPoint)coordinate;
45 - (LevelCell *) cellForGridCoordinate:(CGPoint)coordinate;
46
47 @end
```

2014-05-06 14:25:51.918 CellularAutomataTechTalkVersion[4467:60b] Generating cave...
2014-05-06 14:25:51.929 CellularAutomataTechTalkVersion[4467:60b] Number of caverns in map: 184
2014-05-06 14:25:51.938 CellularAutomataTechTalkVersion[4467:60b] Number of caverns in map: 34
2014-05-06 14:25:51.961 CellularAutomataTechTalkVersion[4467:60b] Generated cave in 0.041367 seconds

Picking the right algorithm

A cellular automaton would be a good choice for this game as we would like the level to look like an organic cave



Implementing the algorithm

The screenshot shows the Xcode IDE with the following components:

- Project Navigator (Left):** Shows the project structure for 'CellularAutomataTechTalkVersion'. The file 'Cave.m' is selected.
- Editor (Center):** Displays the code for 'Cave.m'. The code includes a header file, an interface, and an implementation with methods for initialization, generation, and identifying caverns.
- Inspector (Right):** Shows the 'Identity and Type' panel for the selected file, including its name, type, location, and full path.
- Console (Bottom):** Displays the output of the application, showing the time taken to generate a cave and the number of caverns found.

```
1 //
2 // Cave.m
3 // CellularAutomataTechTalkVersion
4 //
5 // Created by Kim Pedersen on 06/05/14.
6 // Copyright (c) 2014 Kim Pedersen. All rights reserved.
7 //
8
9 #import "Cave.h"
10
11 @interface Cave()
12 @property (strong, nonatomic) NSMutableArray *caverns;
13 @end
14
15 @implementation Cave
16
17 - (instancetype) initWithGridSize:(CGSize)size
18 {
19     if (( self = [super initWithGridSize:size] ))
20     {
21         _chanceToBecomeWall = 0.45f;
22         _floorsToWallConversion = 4;
23         _minCavernCellCount = 10;
24         _numberOfTransitions = 3;
25         _wallsToFloorConversion = 3;
26     }
27     return self;
28 }
29
30 - (void) generate
31 {
32     NSLog(@"Generating cave...");
33     NSDate *startDate = [NSDate date];
34
35     [self initialize];
36
37     for ( NSUInteger i = 0; i < _numberOfTransitions; i++ )
38         [self doTransition];
39
40     [self identifyCaverns];
41     [self removeSmallCaverns];
42     [self identifyCaverns];
43     [super generate];
44
45     NSLog(@"Generated cave in %f seconds", [[NSDate date] timeIntervalSinceDate:startDate]);
46 }
47
48 - (void) initialize
49 {
50     // Create grid
51     self.grid = [NSMutableArray arrayWithCapacity:(NSUInteger)self.gridSize.height];
52
53     for ( NSUInteger y = 0; y < self.gridSize.height; y++ )
54     {
55
```

Console Output:

```
2014-05-06 14:25:51.918 CellularAutomataTechTalkVersion[4467:60b] Generating cave...
2014-05-06 14:25:51.929 CellularAutomataTechTalkVersion[4467:60b] Number of caverns in map: 184
2014-05-06 14:25:51.938 CellularAutomataTechTalkVersion[4467:60b] Number of caverns in map: 34
2014-05-06 14:25:51.961 CellularAutomataTechTalkVersion[4467:60b] Generated cave in 0.041367 seconds
```

Procedural algorithms

A good procedural algorithm needs to nail:

- ⚙️ **Feasibility:** Can you beat the level?
- ⚙️ **Interesting design:** Do you want to beat the level?
- ⚙️ **Appropriate skill level:** Is it a good challenge?

Procedural algorithms - Feasibility

If you want more than a 1-star rating on the AppStore, your levels should be feasible.

- ⚙️ **Is it possible to reach the exit from the entrance?**

Use A* path-finding to determine if a passage exists

- ⚙️ **What restrictions will the algorithm have on the feasibility of the level?**

For example, a cellular automaton will create disconnected caverns.

- ⚙️ **Are there any constraints on the game world?**

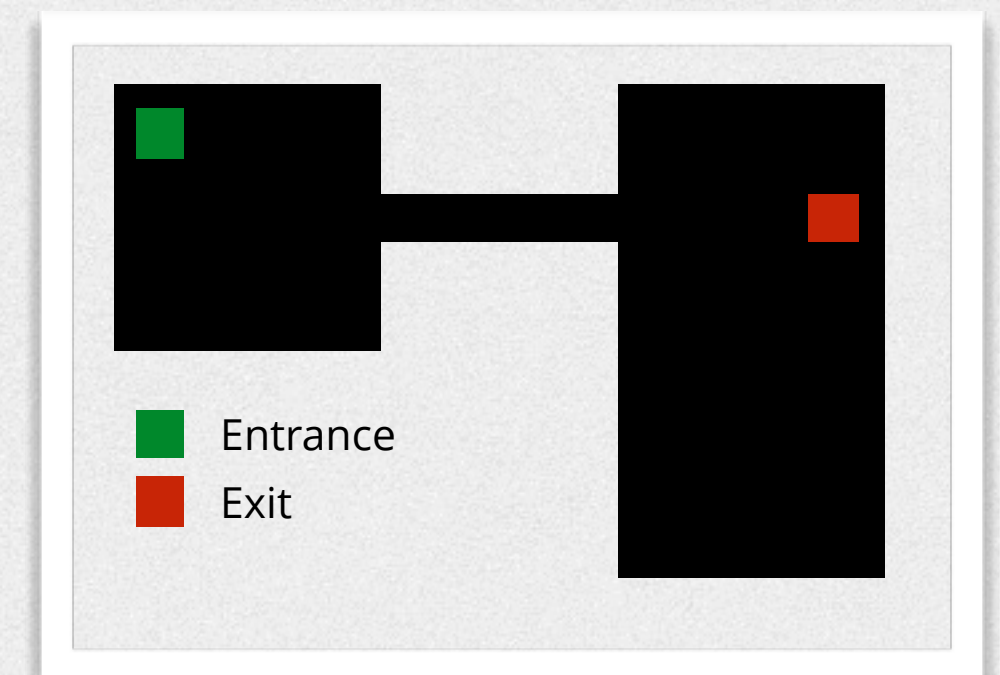
Consider the constraints on the game world like gravity, collision shapes, doors.



Procedural algorithms - Interesting Design

Making the level interesting is one of the most difficult parts of procedural level generation.

- ⚙️ Make any parameter that has an impact on the algorithm a property to allow more flexibility in generating the level
- ⚙️ Make the level destructible
- ⚙️ Include keys and locks to ensure the player explores the level
- ⚙️ Include special items to make the player want to explore the level
- ⚙️ Make some rooms “hand-made” like “temple”, “treasure room” etc
- ⚙️ Ensure variability in room layouts. That makes the levels seem less repetitive.



A not so interesting level

Procedural algorithms - Appropriate Skill Level

Do not go overboard with complexity - especially for mobile games it is sufficient to control difficulty with:

- ⚙ The longer the distance between entrance and exit the harder the level
- ⚙ More enemies increase difficulty
- ⚙ More powerful enemies increase difficulty
- ⚙ Smarter enemies increase difficulty
- ⚙ A locked door blocking the path to the exit increases difficulty
- ⚙ Etc.

No need to make it more complex than what is needed (KISS)

Resources

⚙️ Ray Wenderlich tutorials on procedural level generation:

- ⚙️ <http://www.raywenderlich.com/49502/procedural-level-generation-in-games-tutorial-part-1>
- ⚙️ <http://www.raywenderlich.com/51786/procedural-level-generation-in-games-part-2>
- ⚙️ <http://www.raywenderlich.com/66062/procedural-level-generation-games-using-cellular-automaton-part-1>
- ⚙️ <http://www.raywenderlich.com/70610/procedural-level-generation-games-using-cellular-automaton-part-2>

⚙️ RogueBasin

- ⚙️ <http://www.roguebasin.com>

⚙️ Procedural Content Generation Wiki

- ⚙️ <http://pcg.wikidot.com/>

⚙️ Roguelike Radio (podcasts about roguelikes)

- ⚙️ <http://www.roguelikeradio.com>

Source: Jordan Fisher, How to make insane, procedural platformer levels, Gamasutra, May 10, 2012