Procedural Level Generation in Games

Kim Pedersen

raywenderlich.com

Tech talk brought to you by ...

Ray Wenderlich

Wes Paugh

Kim Pedersen

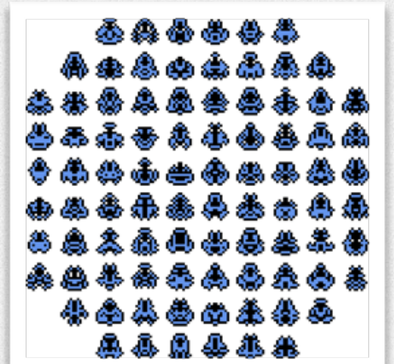
Definition

“ 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 “

Source: Procedural Content Generation Wiki, http://pcg.wikidot.com

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

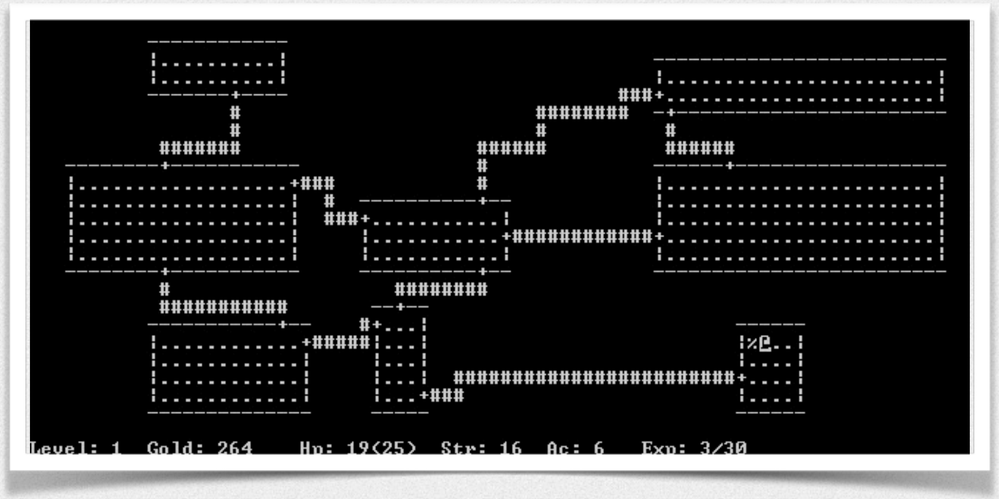
100 Rogues

Diablo

Minecraft

Rogue

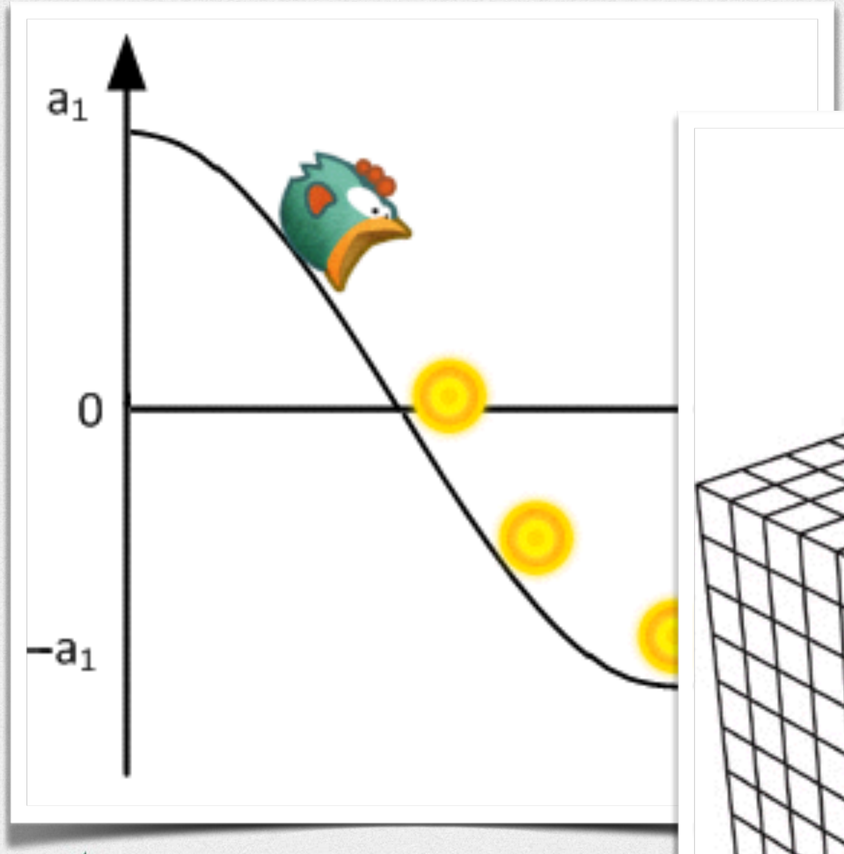
- Developed on Unix by Michael Toy and Glenn Wichman around 1980
- One of the first games to use procedural levels
- Has lead to a class of games called “roguelikes” characterized by procedural level generation, tile-based graphics and permadeath



Procedural Level Generation

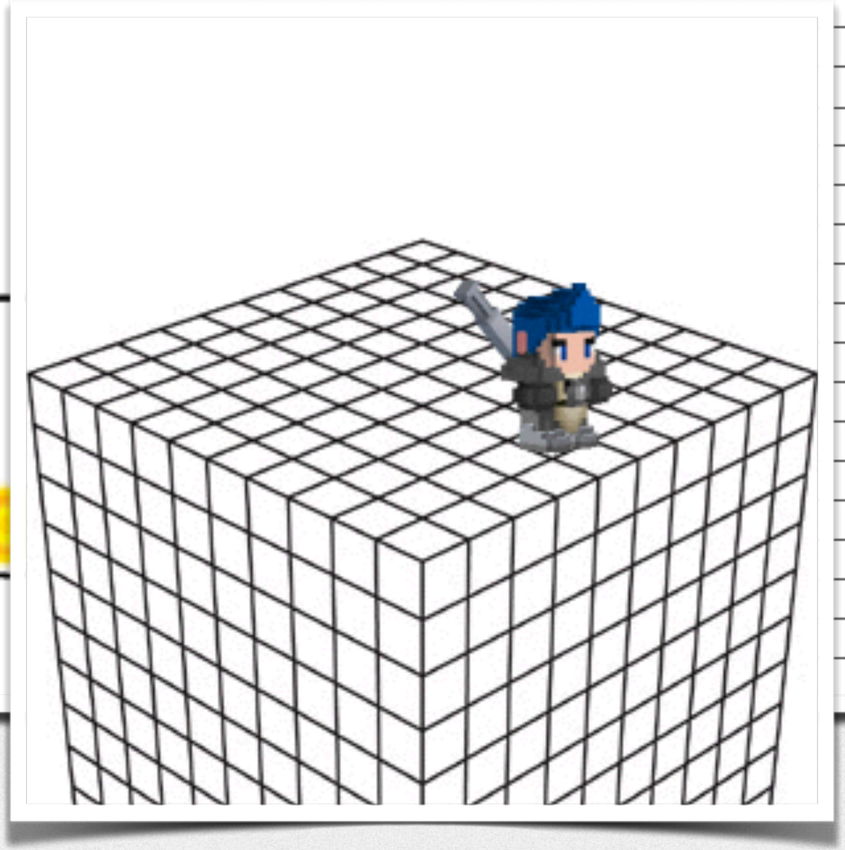
- Procedural levels can be generated in 1d, 2d, 3d

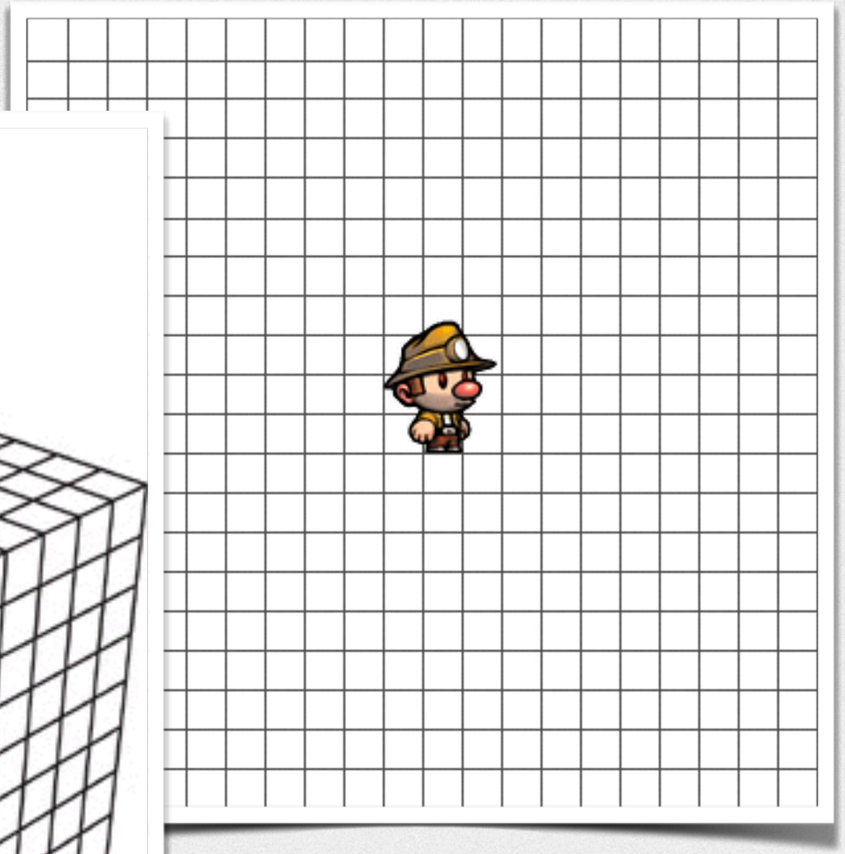
Procedural Level Generation



Tiny Wings

Cube World





Spelunky

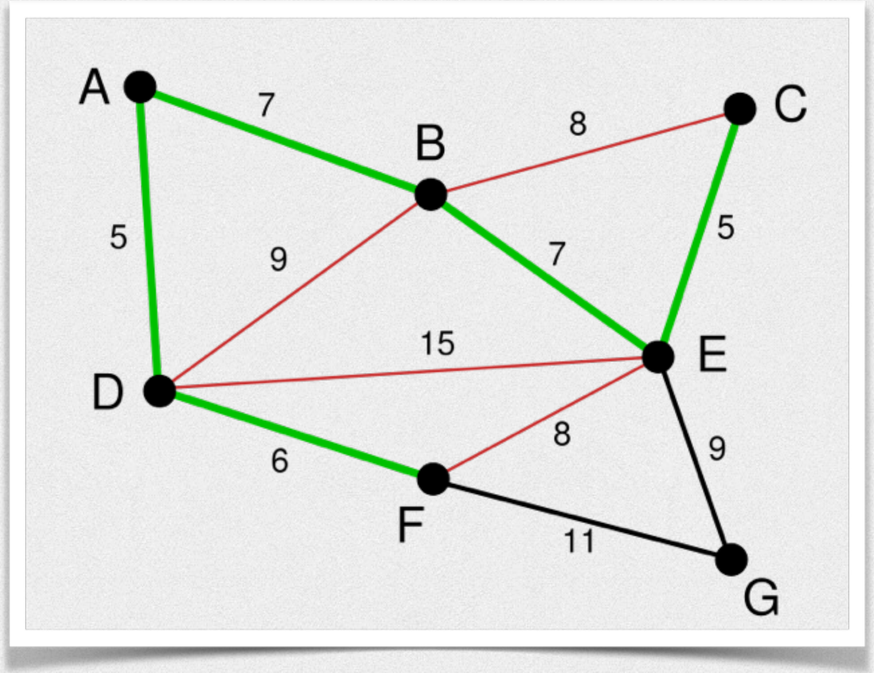
Procedural Level Generation

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

Algorithms in procedural level generation

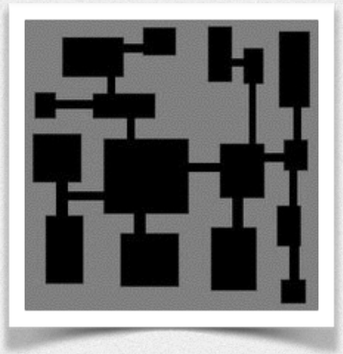
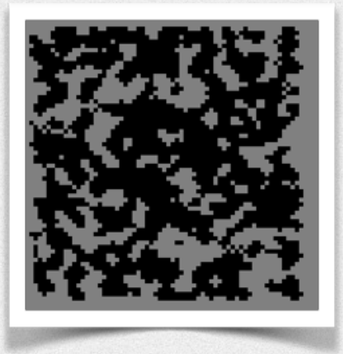
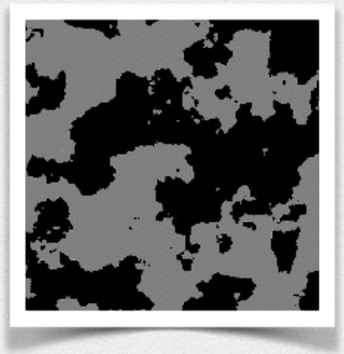
Several popular algorithms are used in procedural level generation:

- **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

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

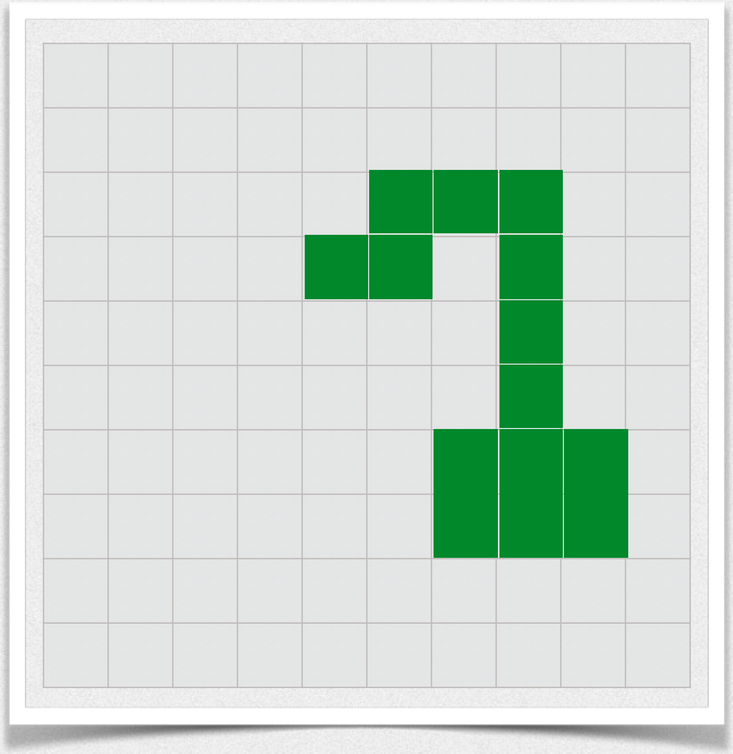
Agent based dungeon growing

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



Ray Wenderlich tutorial: http://www.raywenderlich.com/49502/procedural-level-generation-in-games-tutorial-part-1

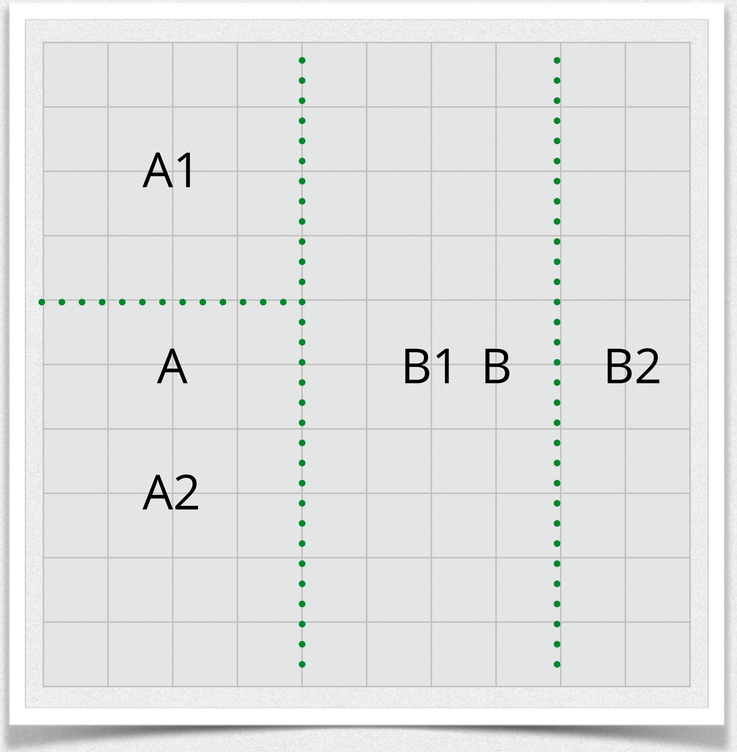
Space Partitioning - Step 1-4

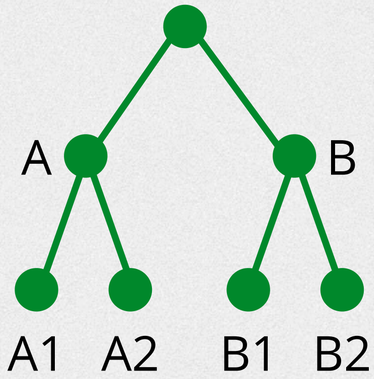
1. Choose to split horizontally or vertically

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





Source: www.roguebasin.com/index.php?title=Basic_BSP_Dungeon_generation

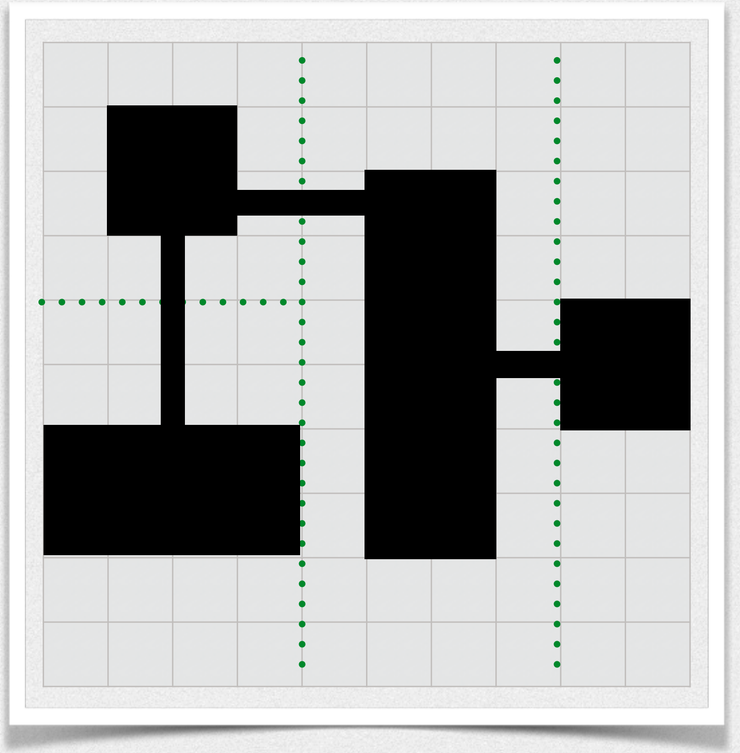
Space Partitioning - Step 5-8

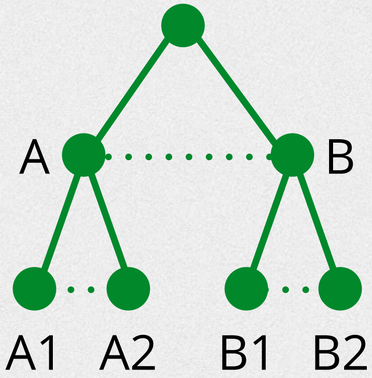
5. Create a room with random size in each leaf of the tree

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





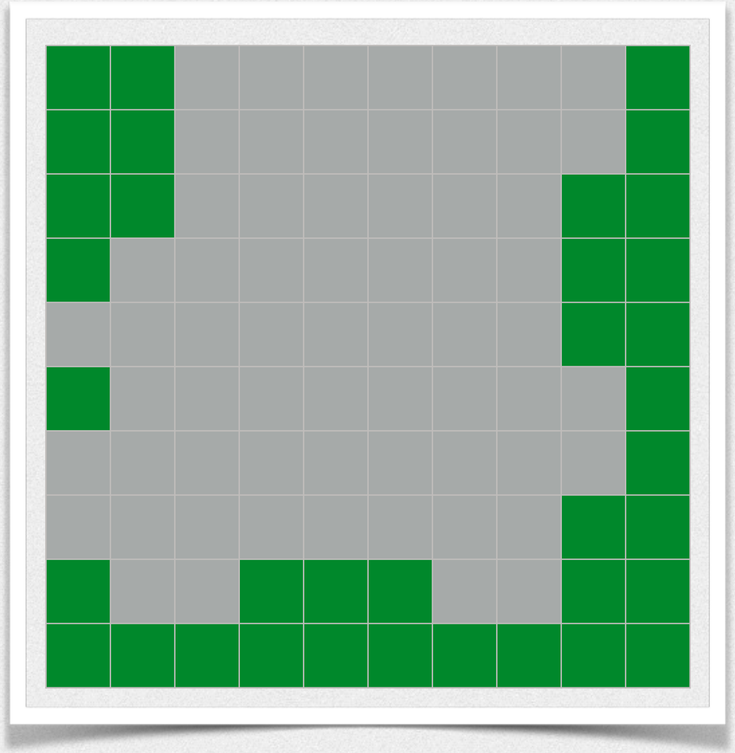
Source: www.roguebasin.com/index.php?title=Basic_BSP_Dungeon_generation

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

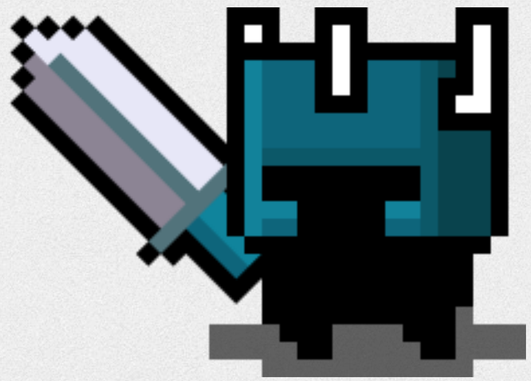


Ray Wenderlich tutorial: http://www.raywenderlich.com/66062/procedural-level-generation-games-using-cellular-automaton-part-1

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.

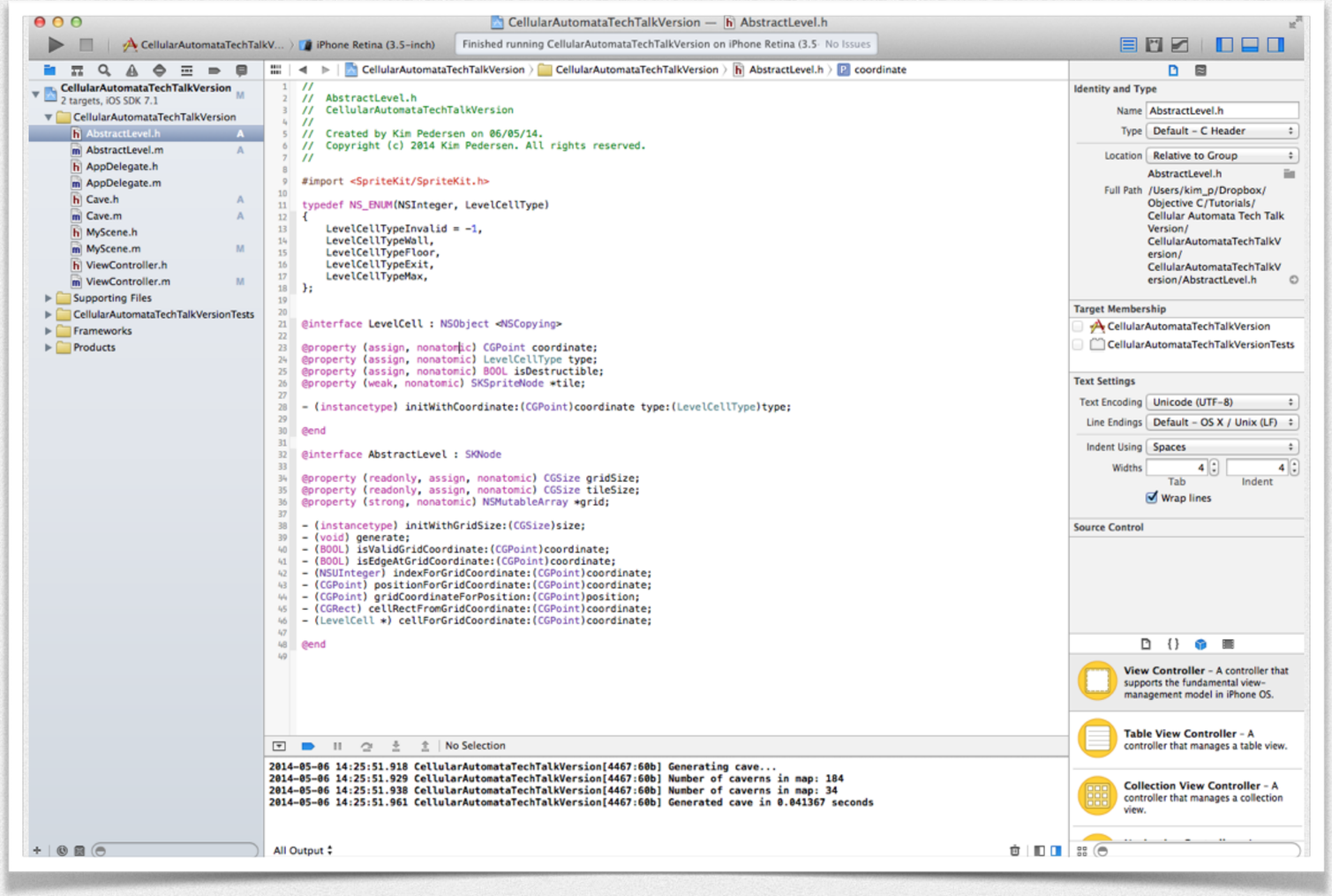
- **Implement procedural algorithm to generate the level**
  There are many algorithms with varying difficulty to implement that will give different visual results. Pick your algorithm wisely.

- **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**

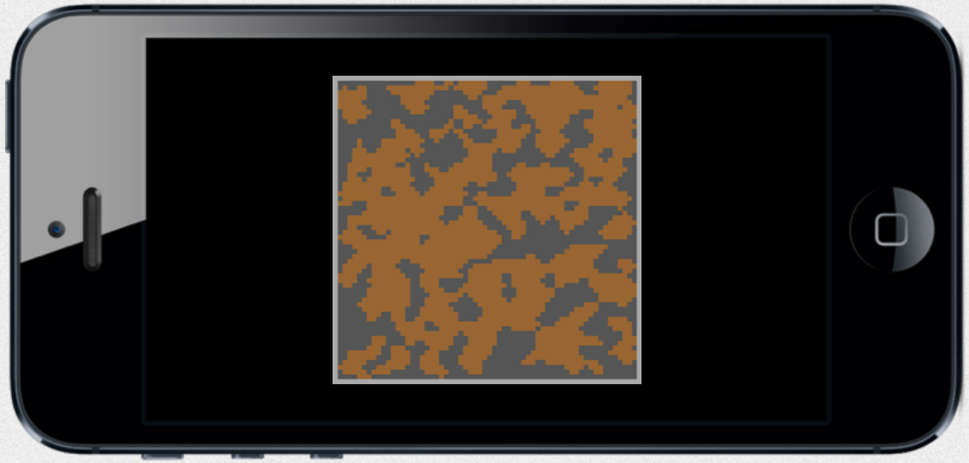
  All levels in a game should be feasible, interesting and target an appropriate skill level.

Abstract level class

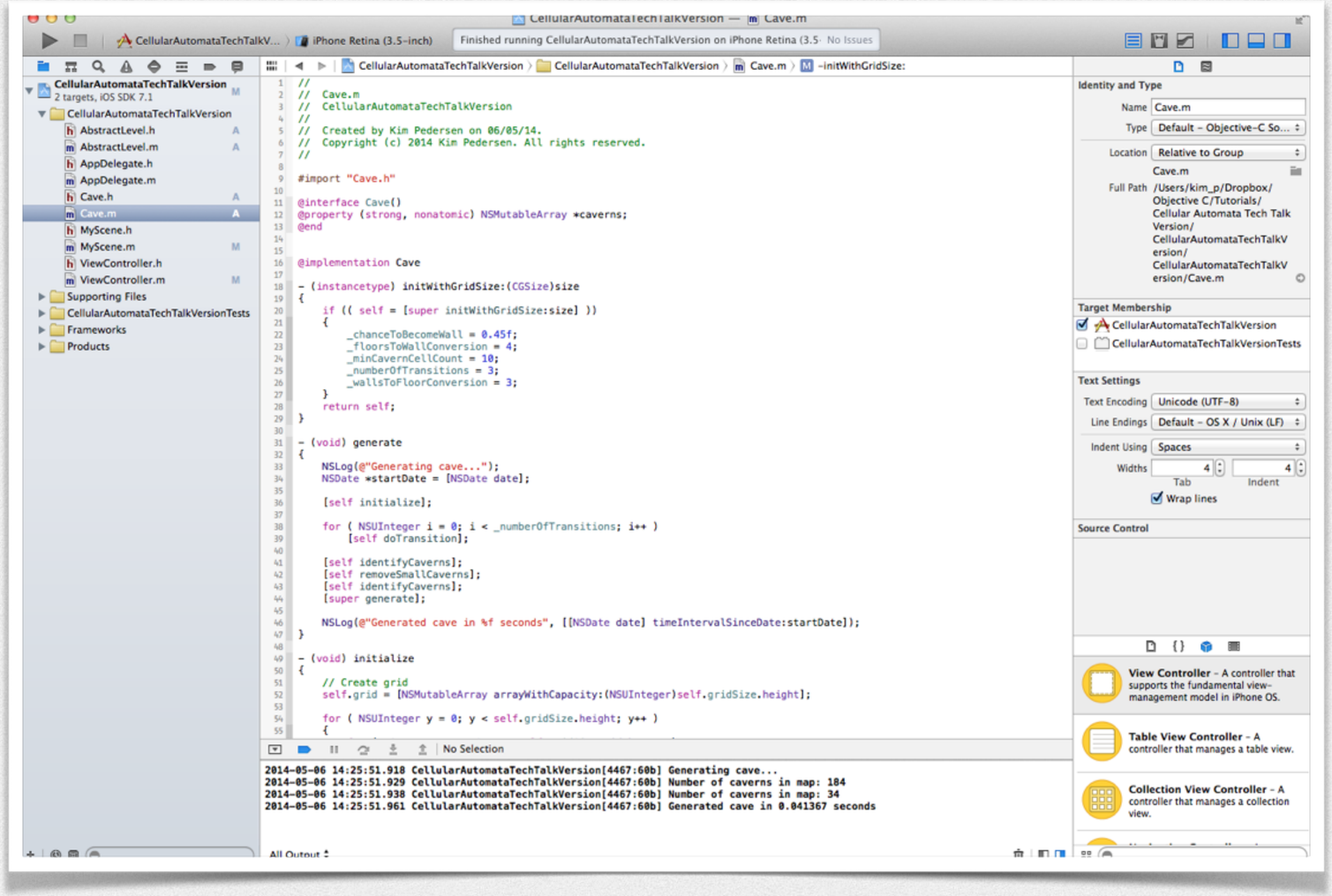


Picking the right algorithm

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



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?

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

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

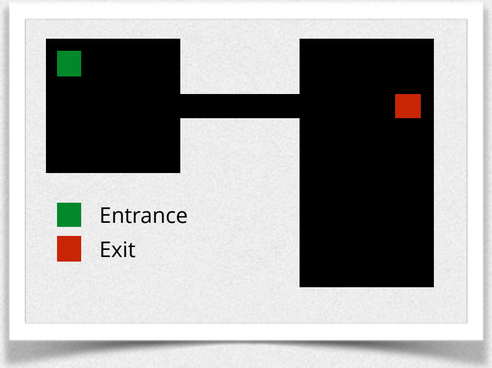
- **What restrictions will the algorithm have on the feasibility of the level?**
  For example, a cellular automaton will create disconnected caverns.

- **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 looked door blocking the path to the exit increases difficulty
- Etc.

No need to make it more complex that 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