Using procedural content generation to create video game levels

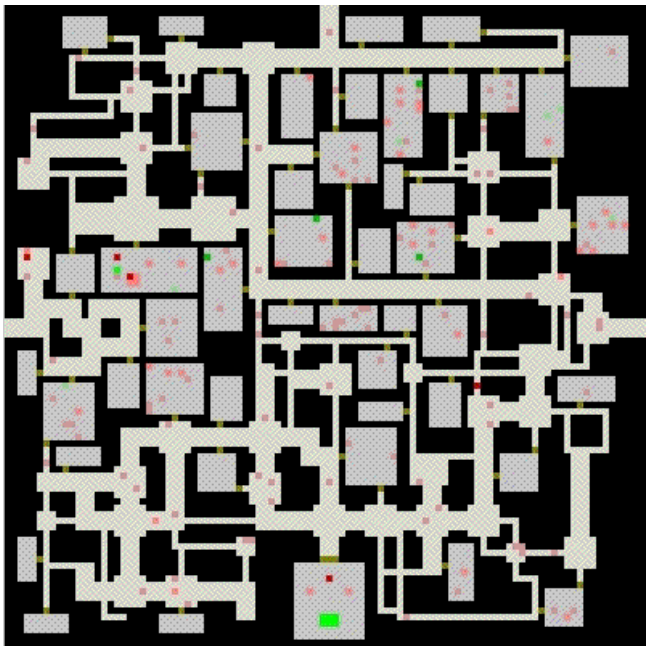
Richard Stangl

Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

15 April 2017
Senior Seminar Conference, Morris

The big picture

- Procedural Content Generation: a game developer's best friend
- Single room/screen generation
- Generating an entire dungeon map



Dungeon Maker
http://bit.ly/2ovjLjD

Outline

Outline

Definition

From *Procedural Content Generation in Games*:

> *PCG is the algorithmic creation of game content with limited or indirect user input.*

Definition

From *Procedural Content Generation in Games*:

> *PCG is the algorithmic creation of game content with limited or indirect user input.*

"Content" is:

> *...most of what is contained in a game: levels, maps, game rules, textures, stories, items, quests, music, weapons, vehicles, characters, etc.*

"Content" excludes the game engine itself and non-player character (NPC) behavior

Definition

PCG can be used to

- Generate one or more aspect of a game
- Create an entire game from the ground up (theoretically)

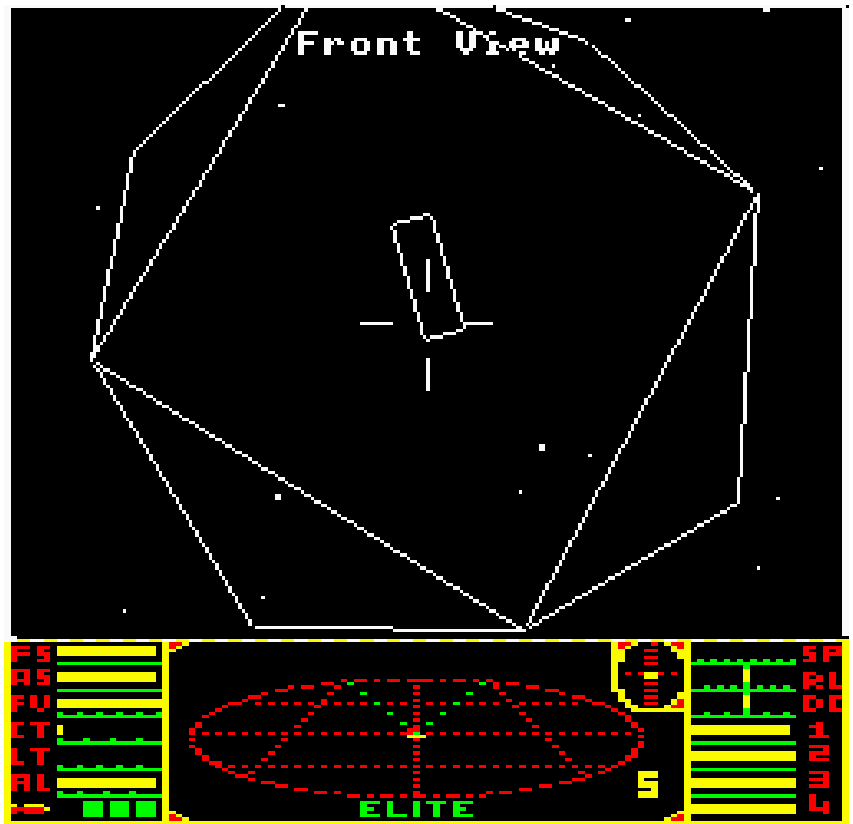
History of PCG

PCG was originally conceived as a method to overcome storage limitations in the 1980s.

Early games used stored seed codes to generate pre-defined content during user gameplay.

Examples of games that took advantage of this storage solution include:

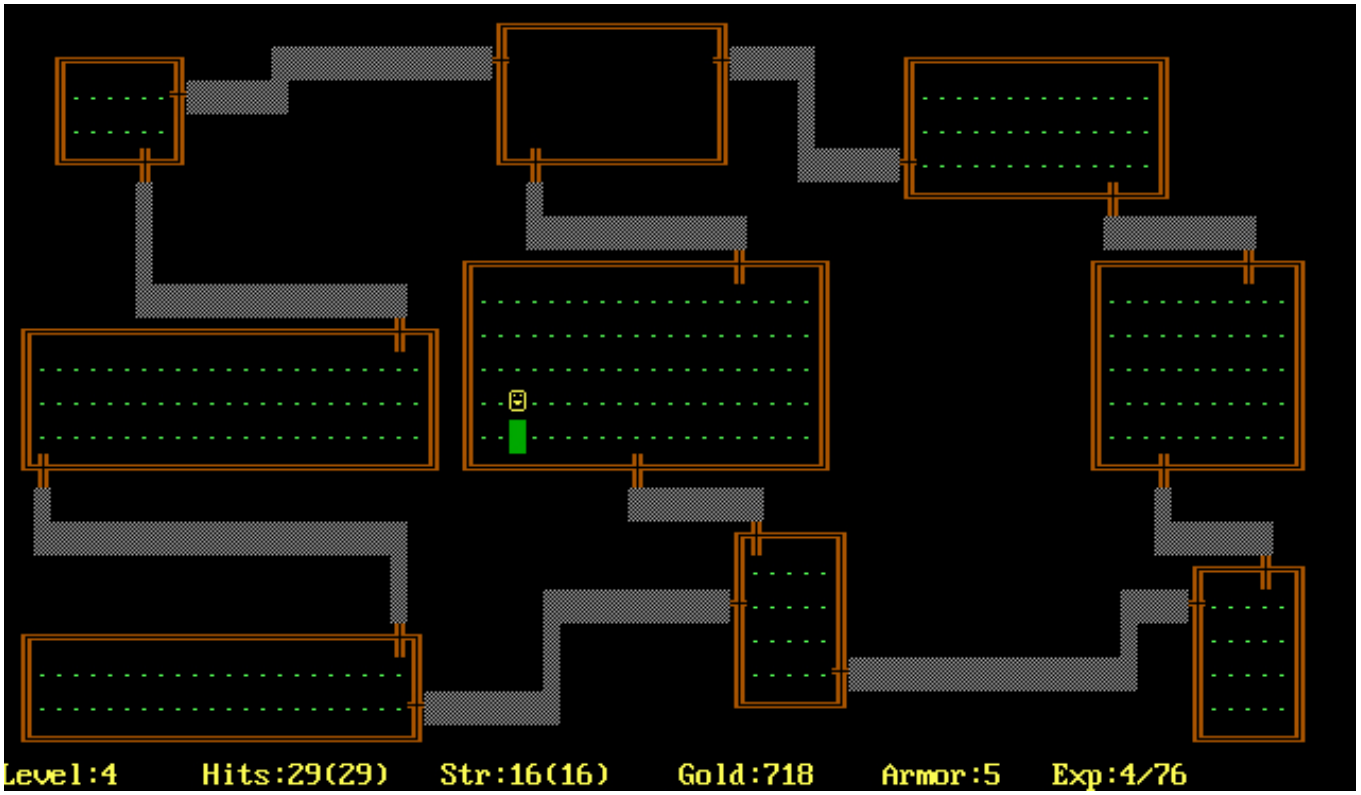
- *Elite*



"Elite" screenshot
http://bit.ly/2oJ9yDF

Examples of games that took advantage of this storage solution include:

- *Rogue*



"Rogue" screenshot
http://bit.ly/2oJm1XY

Examples of games that took advantage of this storage solution include:

- *Dwarf Fortress*

"Dwarf Fortress" screenshot
`http://bit.ly/2oAA2qA`

Modern Applications

PCG can be used to reduce the human effort required to build a game.

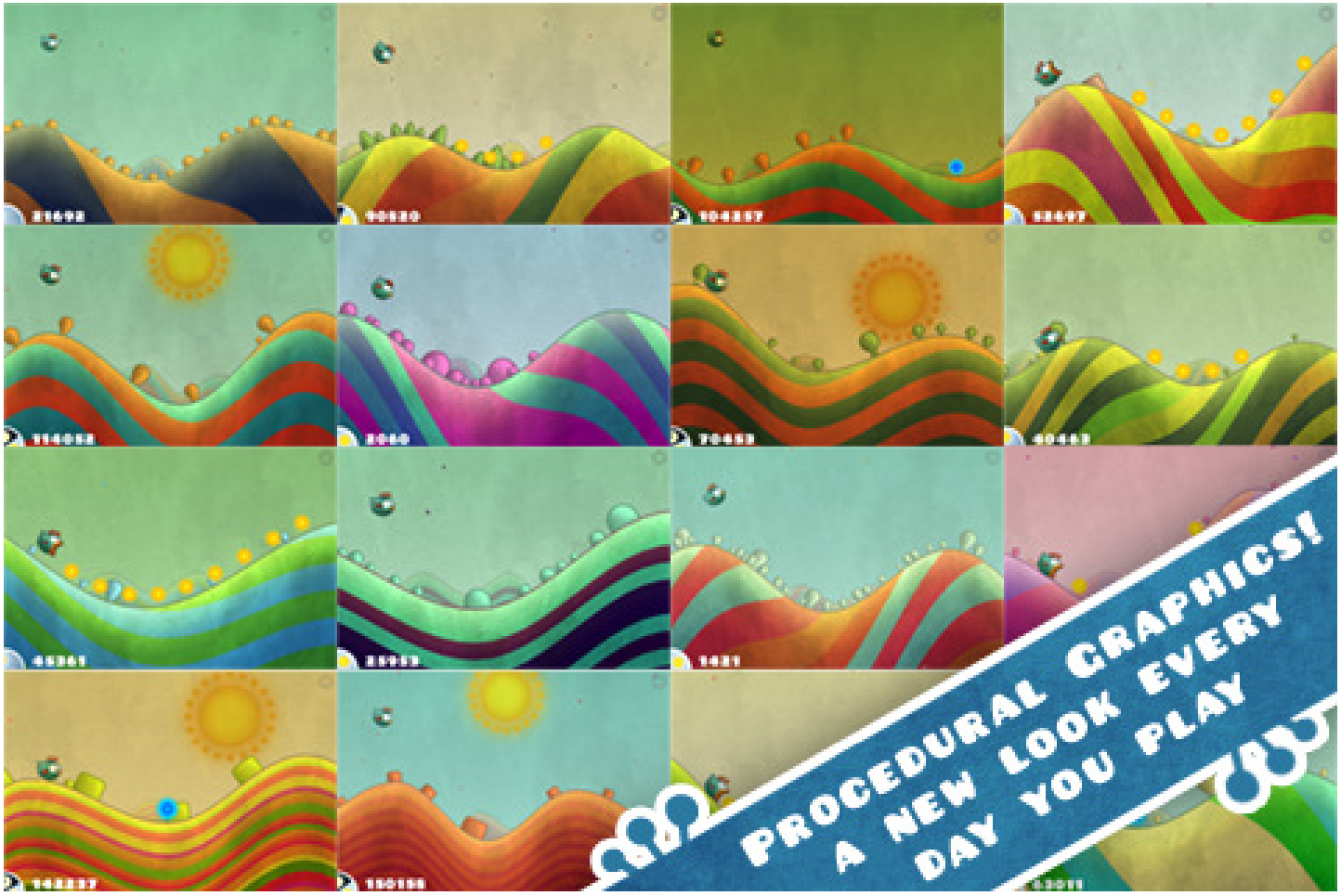
- Elaborate (and expensive) tentpole game releases can use PCG for cost-reduction.
- Small developer teams can use PCG to supplement the human-generated content to create more expansive or creative games than possible otherwise.

Modern Applications

PCG can also be used to generate unique and/or endless game content.

- PCG algorithms may generate novel content that a developer might not have come up with on their own.
- Games that generate almost infinite content are possible with PCG. These include endless runners like *Tiny Wings* or *Temple Run* or expansive world explorers like *Minecraft* or *No Man's Sky*.

Modern games that use PCG:



Tiny Wings
http://bit.ly/2ovsiDo

Modern games that use PCG:

Temple Run
http://bit.ly/2o3opcg

Modern games that use PCG:

Minecraft (note: Herobrine has been removed from the current version)
`http://bit.ly/2ngtV7r`

Modern games that use PCG:

No Man's Sky
http://bit.ly/2ovPeC6

Outline

Definition and description of GVG-LG

- GVG-LG is an attempt to create a general generator that works for multiple or even all video games (Khalifa 2016)
- Built on the General Video Game Artificial Intelligence (GVG-AI) framework
  - GVG-AI was created for the 2014 General Video Game Playing Competition
  - Developers tested AI game players–"controllers"–on provided games and game levels
  - GVG-LG uses certain submitted controllers to evaluate generated game levels

Definition and description of GVG-LG

Game descriptions are made up of

- Sprites
  - Avatar, NPC, Resource, Portal, Static, Moving
- Termination conditions
- Interactions
- Level Mapping

Included generators

Random Level Generator

- Type of sprite and sprite location generated randomly based on a probability selected by the developer
- Will contain at least one of each kind of sprite.

Included generators

Constructive Level Generator

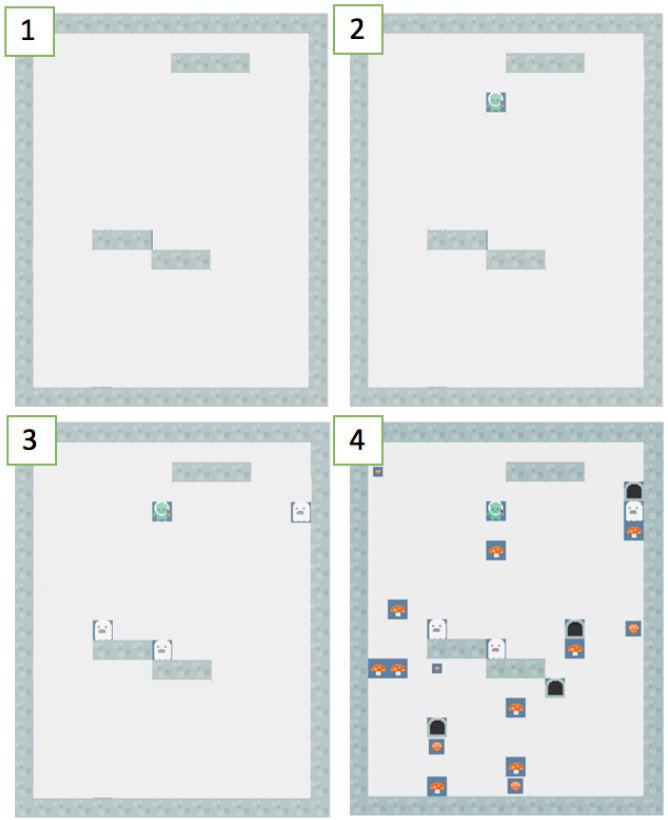
- First sorts sprites into
  - Avatar Sprites
  - Solid Sprites (walls, obstacles)
  - Harmful Sprites
  - Collectible Sprites
  - Other Sprites

Included generators

Constructive Level Generator

- Pre-processing:
  - Calculates how many tiles (locations) should have sprites on them
  - Calculates the ratio for each type of sprite
  - Calculates the total number of sprites
- Designed for difficulty balancing
- For example, if there should be a large ratio of collectible sprites, the total number of sprites is increased. If there is a large ratio of harmful sprites, the total is decreased.

Sprite Placement



Four steps of sprite placement

Included generators

Constructive Level Generator

- Post-processing:
  - Checks number of goal sprites
  - Corrects if needed
  - For example, if the game ends when five enemies are defeated, the generator checks that there are at least five enemies present

Included generators

Search-Based Level Generator

- Uses a genetic algorithm called Feasible Infeasible 2 Population Genetic Algorithm (FI2Pop) to evolve two populations, feasible and infeasible, of initial game states.
  - Feasible populations focus on improving game states to fit the desired outcome
  - Infeasible population are meant to remove game levels that violate the problem constraints

Included generators

Search-Based Level Generator

- FI2Pop is an example of a genetic algorithm, which is a form of evolutionary computation
- Evolutionary computation (EC):
  - Uses evolutionary principles to gradually shape a population of programs or program objects to a desired outcome
  - A population of programs that fits an initial state is created
  - Population is changed/mutated over successive generations
  - Each member of each generation is evaluated by a heuristic function that encourages development towards a desired outcome

Included generators

Search-Based Level Generator

- Initial population members are created via the Constructive Level Generator
- Members are mutated by adding or removing a random sprite to a random tile or by swapping two existing sprites
- Mutated members are evaluated using three player AI from the GVG-AI competition
  - Adrienctx, the winner of the 2014 competition
  - OneStepLookAhead, which chooses a beneficial next step from the available steps
  - DoNothing, which does nothing

Included generators

Search-Based Level Generator

- Two methods of evaluation by the three controllers for the feasible population
  - Score Difference Fitness is the difference between the Adrienctx score and the OneStepLookAhead score after 50 runs, meant to favor levels that reward skill with a higher score
  - Unique Rule Fitness generates a score based on the number of novel events that took place
- The final heuristic uses a score based on an average of the two evaluated scores to balance difficulty with novelty

Included generators

Search-Based Level Generator

- Evaluation of the infeasible population is based on seven factors
  - Must have one avatar and one of each type of sprite
  - Must be more goal sprites than the terminating factor requires
  - 5-30% of tiles must have sprites on them
  - Levels must not be solvable in under 200 steps
  - Levels must have been completed by Adrienctx
  - DoNothing must last at least 40 steps in 50 different runs
  - DoNothing cannot win in the same number of steps as Adrienctx

Examples of levels generated by each generator

Levels generated using a) random level generation, b) constructive level generation, and c) search-based level generator

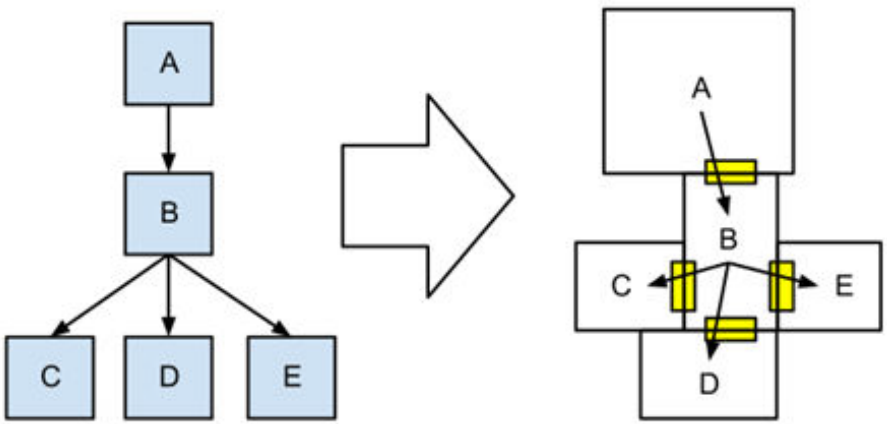
Outline

Overview

- This implementation of PCG also uses a genetic algorithm
- Instead of placing sprites in a single room, Valtchanov and Brown (2012) create a series of connected rooms to be used as a game dungeon

Element representation

- Rooms are represented by nodes which contain data about the size and shape of the rooms,the location and number of doors, and special characteristics such as hallway or event room
- Room connections are represented by edges in the tree; parent nodes are connected to all leaf nodes



Translation of a tree to a room layout

Operators

Crossover

- Copies a random subtree and attaches it to a random parent node
- Allows a subtree to possibly find a better location
- Creates the illusion of design through symmetry

Operators

Mutation

- mutation_Grow selects a few rooms and adds leaf nodes to them
- mutation_Trim removes leaf nodes from a few rooms or deletes them entirely
- mutation_Change picks a few random rooms and changes their shape and/or number of doors

Operators

Selection

- Uses the fitness heuristic (below) to score the maps in a population
- Replaces the bottom half scorers with children of the top half scorers
- Children are created using the other two operators

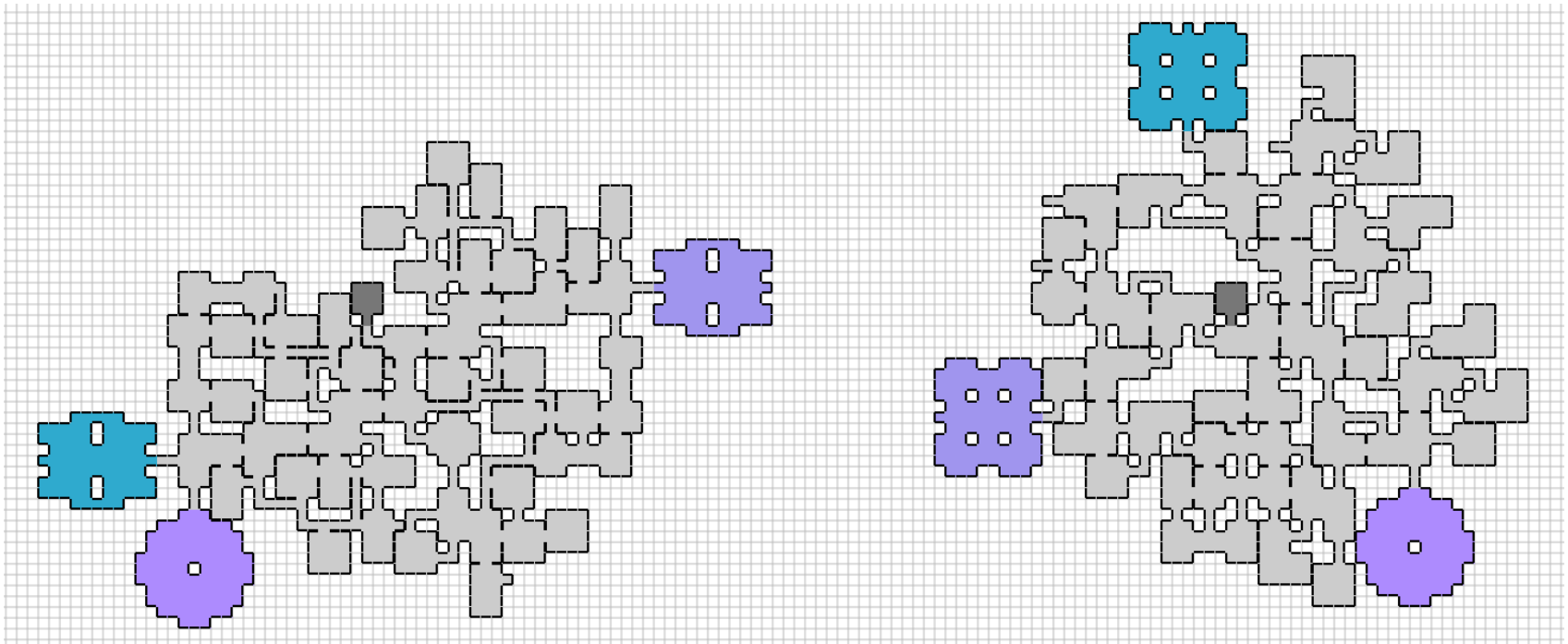
Fitness Heuristic

- To begin, all rooms with the special Hallway characteristic that are connected to each other are merged into one long hallway
- Any map with more than three special Event rooms is awarded zero points

Fitness Heuristic

- Each hallway is awarded a score based on how many rooms are connected through them but receives a small penalty based on the total number of original nodes that are made of
- Regular rooms are awarded if they are connected to a hallway and at least one other regular room
- This encourages efficient hallways and tightly packed clusters of rooms

Fitness Heuristic



Two examples of dungeon maps

Outline

Conclusions

- Using both PCG implementations in tandem would result in a large percentage of a completed game with little work on the developer's part
- Good for the game industry
  - Cost savings for large game projects
  - Allows indie developers to expand the scope of their games

  Good for gamers
  - New, original game types and genres
  - Almost limitless game content

Questions?