GDC 13 EUROPE

Designing Games with Procedural Content and Mechanics

Joris Dormans
Gameplay Engineer @ Ludomotion

GAME DEVELOPERS CONFERENCE EUROPE
COLOGNE, GERMANY
AUGUST 19–21, 2013
EXPO DATES: AUGUST 19–20
2013

About me

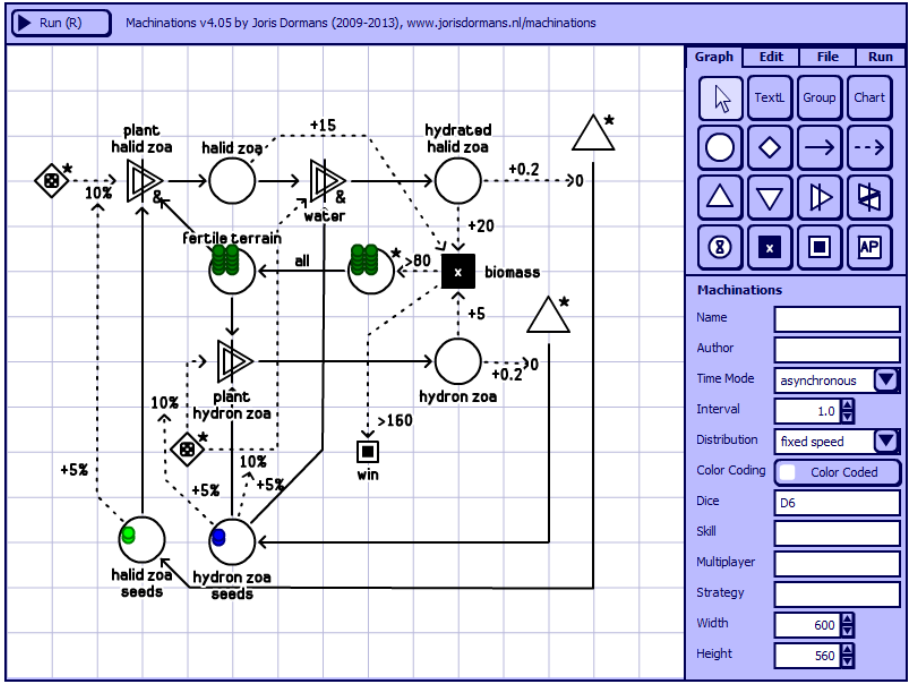
- Gameplay Engineer @ Ludomotion

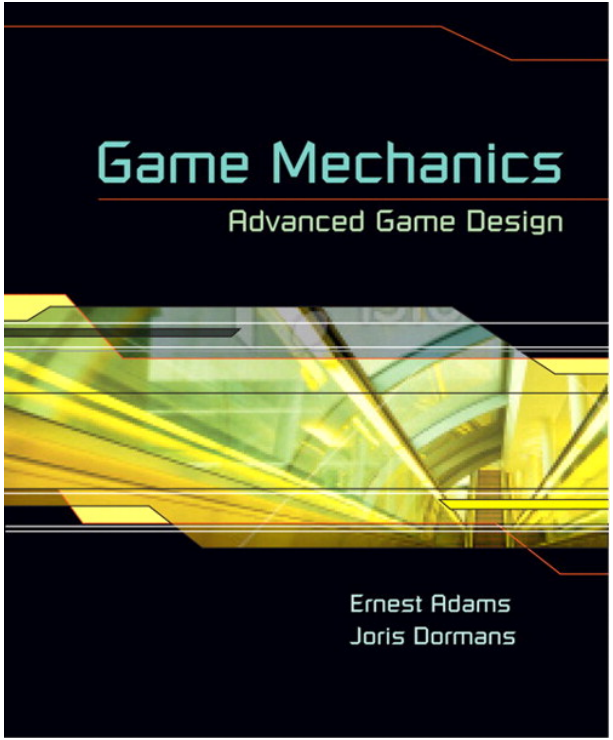
  www.ludomotion.com

- Senior Researcher @ Amsterdam University of Applied Sciences

  www.jorisdormans.nl

My biggest claim to fame





What to expect from this talk

- PCG we did for a game about to release
- PCG we are doing for a game in development
- Resent research trends that inspired the way we do PCG
- Relationship between PCG and game design

What (do I think) is PCG?

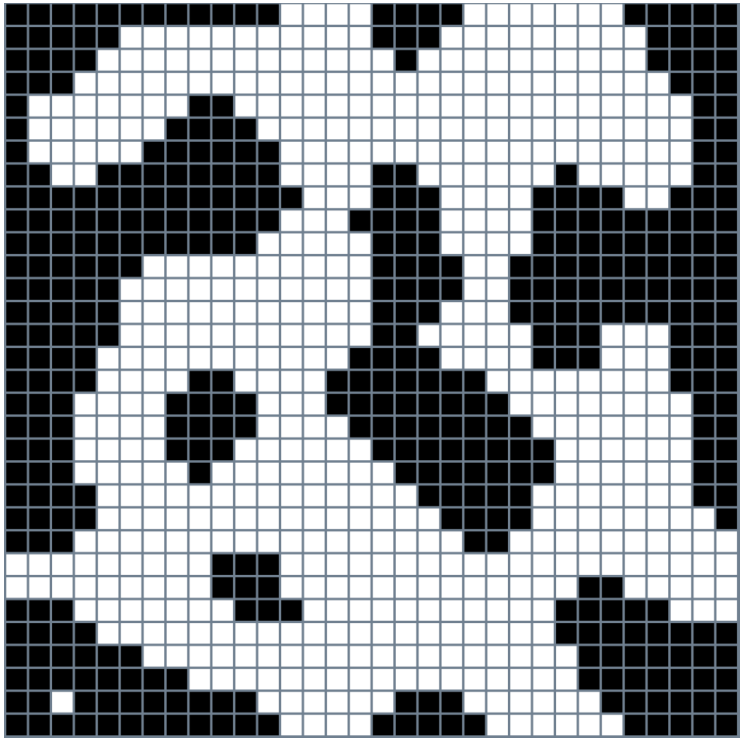
- Generative techniques to produce or adapt content for games
- In this case “Content” is a contested term: it includes visuals, levels, mechanics, and so on

Quick observations about PCG

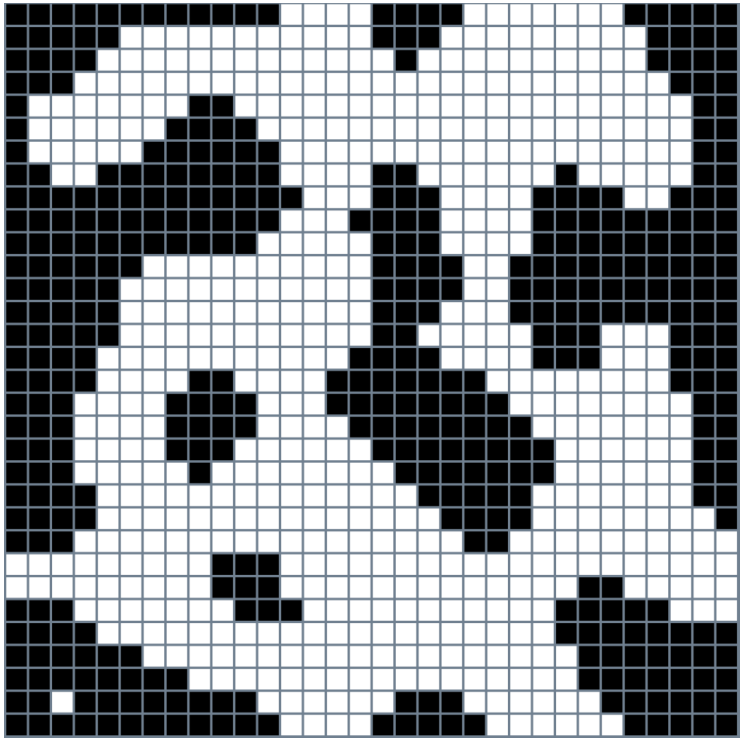
- Some game genres have more affordance for PCG (e.g. Rogue-likes)
- Much work is done in the visual domain (e.g. generated trees)
- Many techniques rely on brute force random algorithms

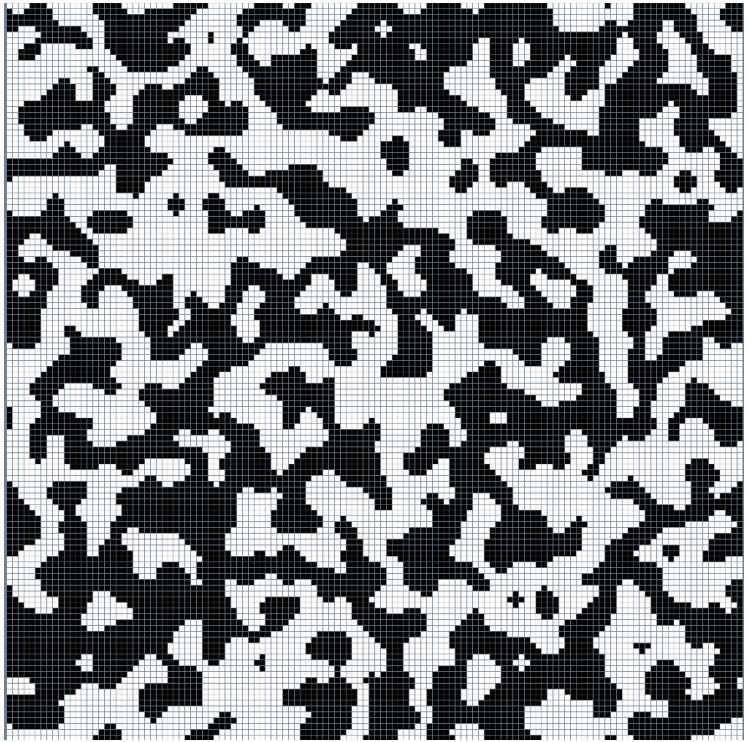
"Bottom-up PCG"

- Starts from an interesting algorithm
- Usually quick
- Little control over result
- Poor scaling



"4-5 rule" cellular automata cave generator





What I really want to talk about

- PCG that generates and supports gameplay and game structure

- PCG that mimics and supports game design

PCG is game design?

Settlers of Catan

PCG in ***Bezircle***

Bezircle Trailer


Back
Start
Time
120 sec
Arena
Standard


Back
Start
Time
120 sec
Arena
Zircle


Back
Start
Time
120 sec
Arena
Treacherous


Back
Start
Time
120 sec
Arena
Wild

Emergent Gameplay & PCG

- ***Bezircle***'s gameplay is highly emergent.
- This makes PCG easy, but not unnecessary!
- Not any random combination of orbs works.

Constraints for PCG in ***Bezircle***

- Perform
  - <0.5 seconds on mobile device
- Consistent quality
  - Controllable size, etc.
- Consistent generation
  - Same seed should generate same level on different platforms

Bezircle uses a two step generation process:

- Step 1: Generate a level ‘recipe’
- Step 2: Use that recipe to generate the level

- Think of the recipe as the plan for the level; it is a “top-down” approach to PCG

Step 1: Generating Level Recipes

- Recipes describe how levels should be generated
- Transformational grammars are used generate recipes
- Each level type has a different recipe grammar

Example ***Casa*** Recipe Grammar

```
S > Start Players Grow1 Grow2 Options Setsize PowerUps Finish
  | StartCircle PlayersCircle Grow1 SpaceCircle Grow2
    EndCircle Setsize PowerUps Finish
...
Grow1 > AddPower AddOrb AddOrb | AddCenOrb AddPower AddOrb
Grow2 > ExtraCenOrb ExtraOrb | ExtraOrb
Options > AddVampire | AddKing | AddExploder | Ø | Ø
...
```

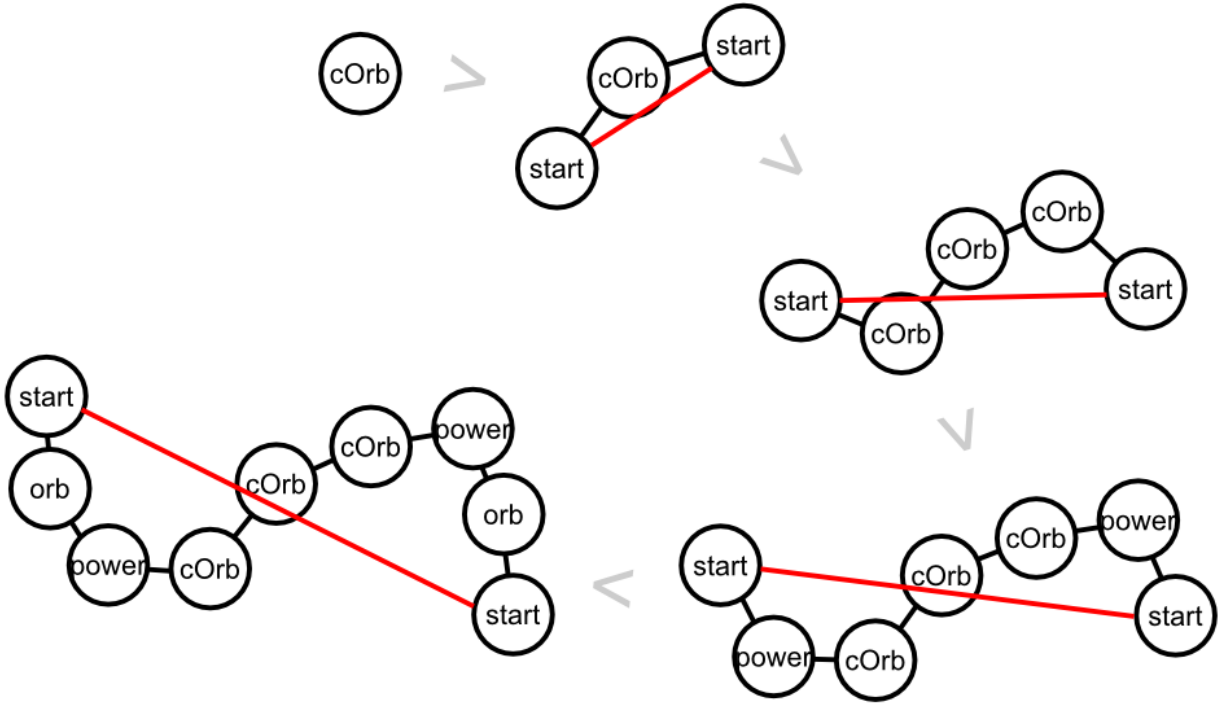
```
            Files["casa.grm"] =
@"start: STRING S
rule: S > Start Players Grow1 Grow2 Options SetSize PowerUps Finish
rule: CircleVariant(probability=0.4) = S > StartCircle PlayersCircle Grow1 SpaceCircle Grow2 EndCircle SetSize PowerUps F
rule: StartCircle > ""openGrammar(circleGeneration.grm)"" ""clear()"" ""iterateRule(StartCircle)""
rule: PlayersCircle > ""executeRule(AddCirclePlayer,@players-1)"" ""doLayout(10)"" ""executeRule(AddCircleDistancer)"" ""
rule: SpaceCircle > ""if(@players<3)"" ""addNode(spacer(x=0,y=0,size=80))"" ""else"" ""if(@players<4)"" ""addNode(spacer(
rule: EndCircle > ""openGrammar(basic.grm)""
rule: Start > ""openGrammar(basic.grm)"" ""clear()""
rule: Players > ""executeRule(AddPlayer,@players)"" ""doLayout(10)"" ""executeRule(AddDistancer)"" ""doLayout(10)""
rule: Grow1 > ""iterateRuleLSystem(GrowPowerUp)"" ""untangle()"" ""iterateRuleLSystem(GrowOrb)"" ""untangle()"" ""iterateL
rule: Grow1 > ""iterateRuleLSystem(GrowCentralOrb)"" ""untangle()"" ""iterateRuleLSystem(GrowPowerUp)"" ""untangle()"" ""
rule: Grow2 > ""executeRule(AddExtraCentralOrb,3)"" ""executeRule(AddExtraOrb,2D@players)"" ""untangle()"" ""doLayout(20)
rule: Grow2 > ""executeRule(AddExtraOrb,3+2D@players)"" ""untangle()"" ""doLayout(20)""
rule: AddVampire > ""openGrammar(specialPowerUps.grm)"" ""executeRule(CentralOrbToVampire,D@players)""
rule: AddKing > ""openGrammar(specialPowerUps.grm)"" ""executeRule(CentralOrbToKing,1)""
rule: CentralEnergizer > ""changeSymbols(CentralOrb,energizer)""
rule: AddExploders > ""openGrammar(specialPowerUps.grm)"" ""iterateRuleLSystem(OrbToExploderChance)""
rule: Energize > ""doLayout(20)"" ""untangle()"" ""connectNeighbors(Edge,20)"" ""openGrammar(specialPowerUps.grm)"" ""exe
rule: SetSize > ""openGrammar(setSizes.grm)"" ""executeRuleCellular(EqualStartOrbs)"" ""executeRuleCellular(LargeCentralO
rule: SetSize > ""openGrammar(setSizes.grm)"" ""executeRuleCellular(EqualStartOrbs)"" ""executeRuleCellular(MediumCentral
rule: SetSize > ""openGrammar(setSizes.grm)"" ""executeRuleCellular(EqualStartOrbs)"" ""executeRuleCellular(MediumCentral
rule: PowerUps > ""openGrammar(basicPowerUps.grm)"" ""executeCellular()""
rule: Finish > ""openGrammar(specialPowerUps.grm)"" ""iterateRuleLSystem(SetCasa)"" ""iterateRuleLSystem(CasaHasInputsOnl
rule: CentralSpacers > ""executeRule(AddCentralSpacer,3)"" ""doLayout(5)"" |
rule: Spacers > ""executeRule(AddSpacer,2+D@players)"" ""doLayout(5)"" |
rule: Options > Spacers | Spacers Spacers | CentralSpacers Spacers | """";
```

Step 2: Generating Levels

- Graph grammars execute the steps in the recipe to generate the level

Example

```
Start
Players
AddCenOrb
AddPower
AddOrb
...
```



Learning Points

- PCG blends easily into emergent gameplay
- Nobody notices PCG done right
- Embrace PCG as an aesthetic for your game
- Embrace the opportunities PCG brings

Bezircle challenges

- Daily, Weekly, Monthly
- Local

Research trends

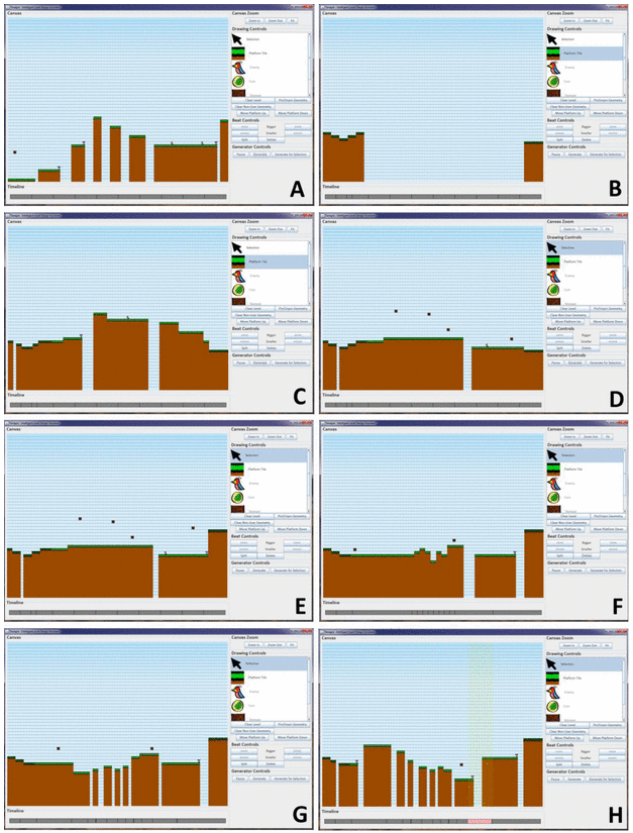
- Increased interest in generating gameplay critical ‘content’
- Mixed-initiative PCG
- Model-driven engineering
- Adaptive games

Research trends

- Increased interest in generating gameplay critical ‘content’
- **Mixed-initiative PCG**
- **Model-driven engineering**
- Adaptive games

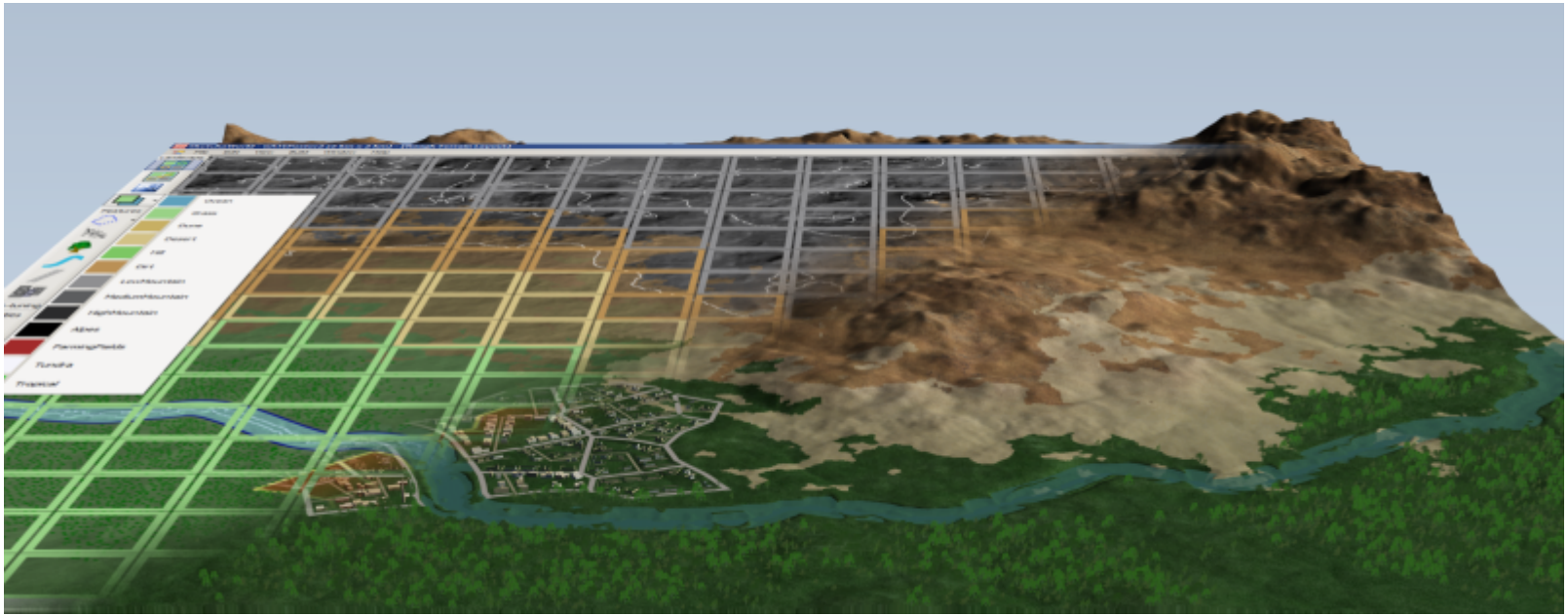
Mixed-Initiative PCG

- Designers use smart design tools to create content



Tanagra (Gillian Smith)

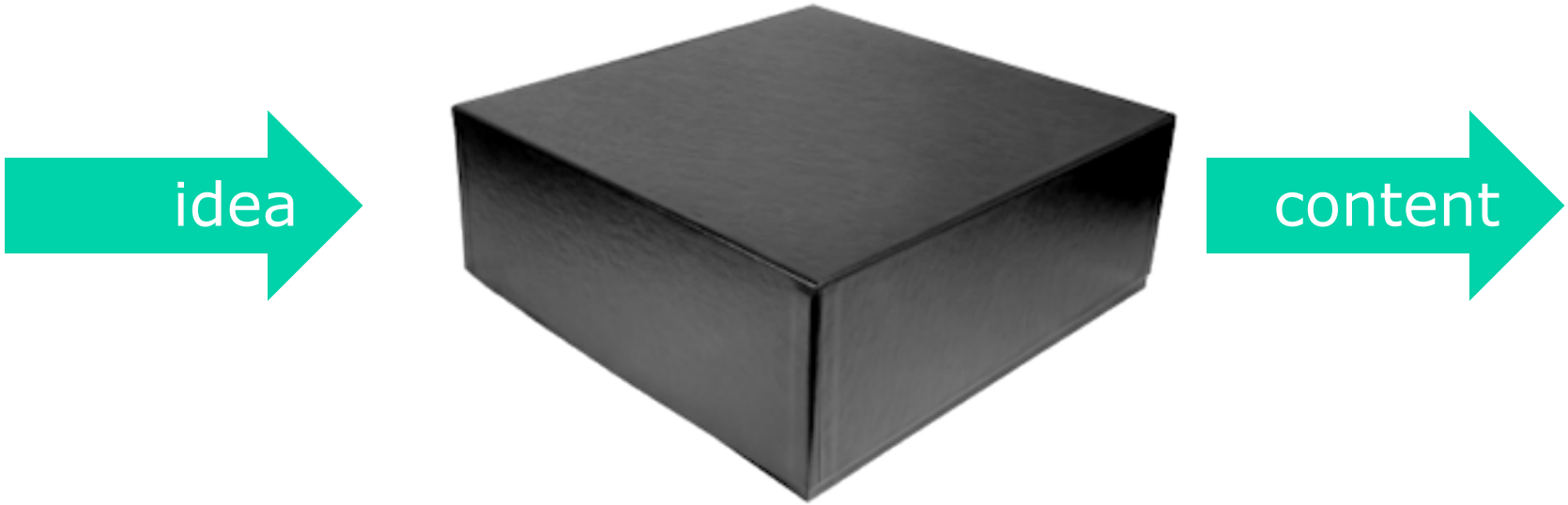
Mixed-Initiative PCG



SketchaWorld (Ruben Smelik, et al.)

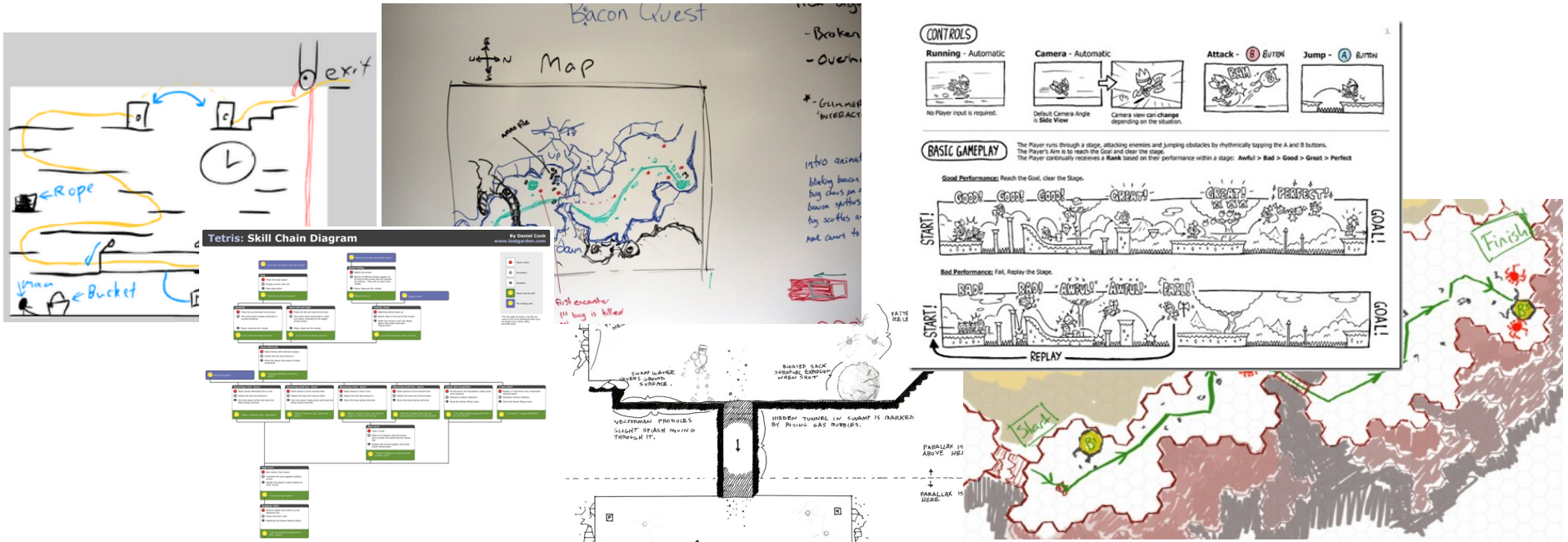
Model-Driven Engineering

- Game design doesn’t work like this:



Model-Driven Engineering

- Instead it looks something like this:

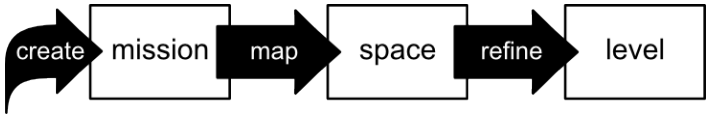


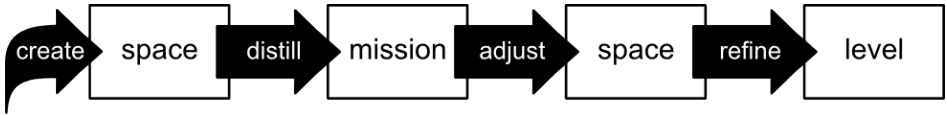
Model-Driven Engineering

- Likewise, PCG shouldn't look like this:

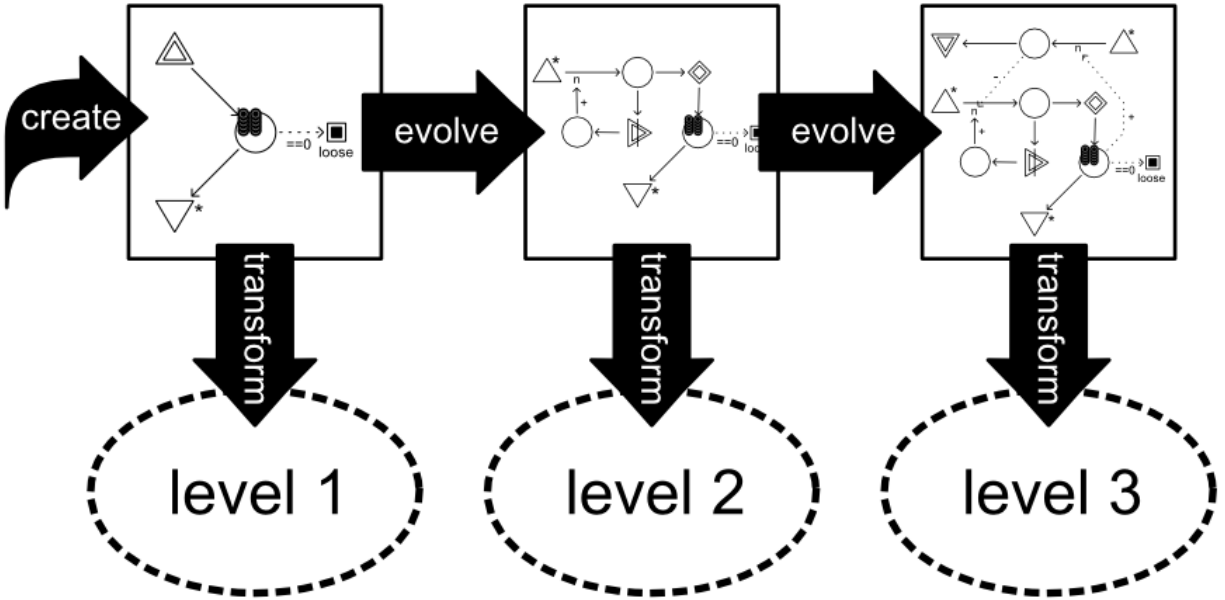
Model Driven Engineering

- Make use of separate models to represent different aspects of the game.
- Use model transformations to go from model to model





Model Driven Engineering



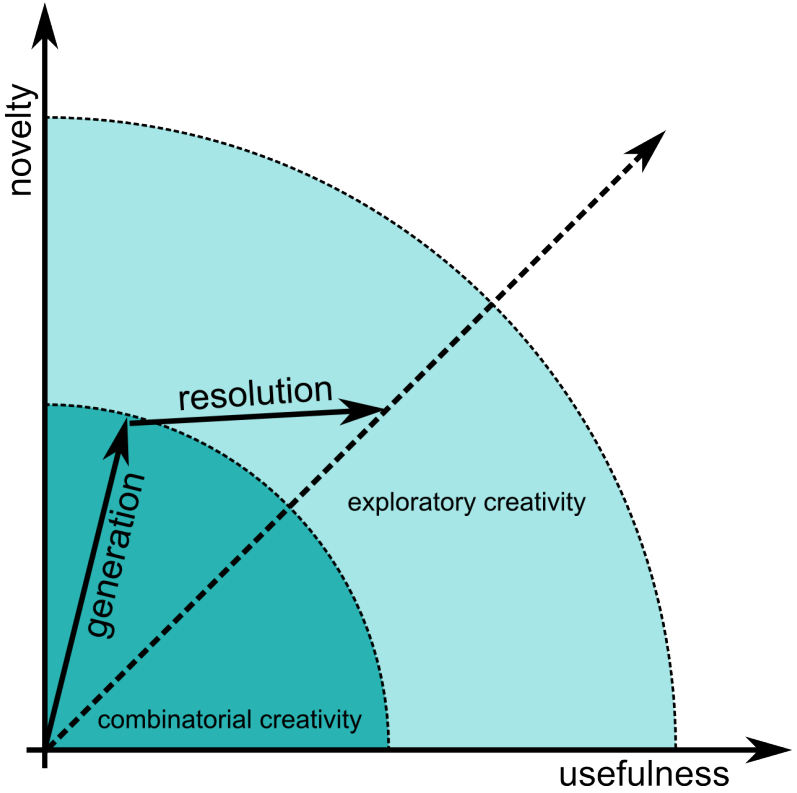
Model-Driven Engineering

- Breaks down the PCG problem in small manageable steps
- Creates a flexible and versatile process
- Ties in well with the Mixed-Initiative PCG

- Requires steps in the design process to be represented as models
- Transformations between models are non-trivial

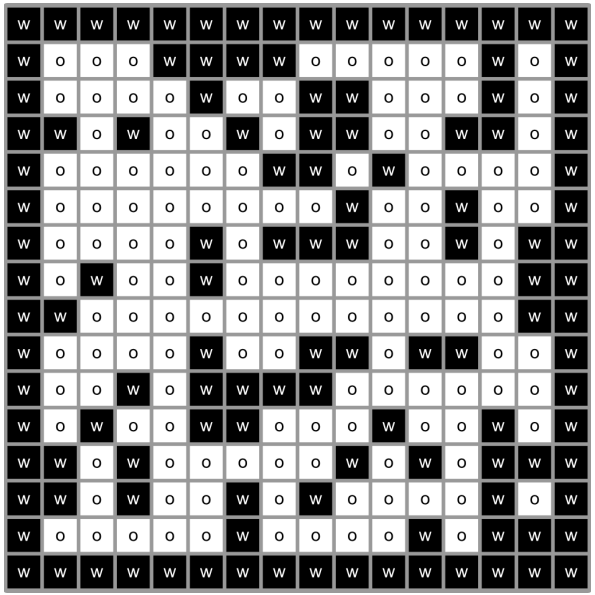
What steps to produce creative solutions?

- Generate variety
- Resolve into something useful



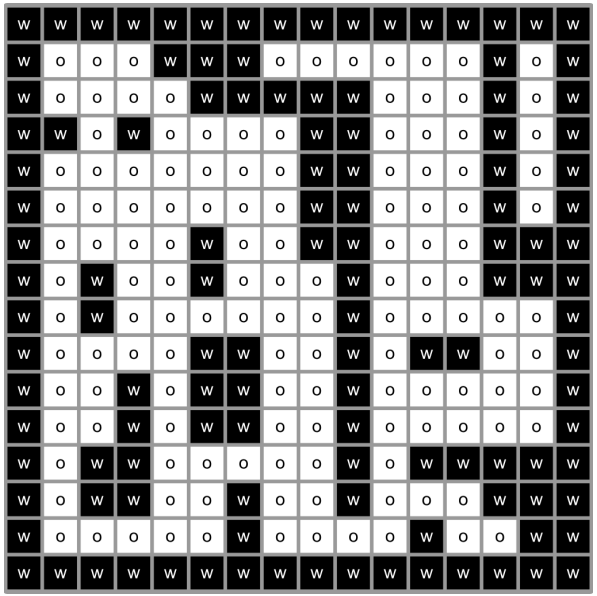
Small Generation steps

Random set

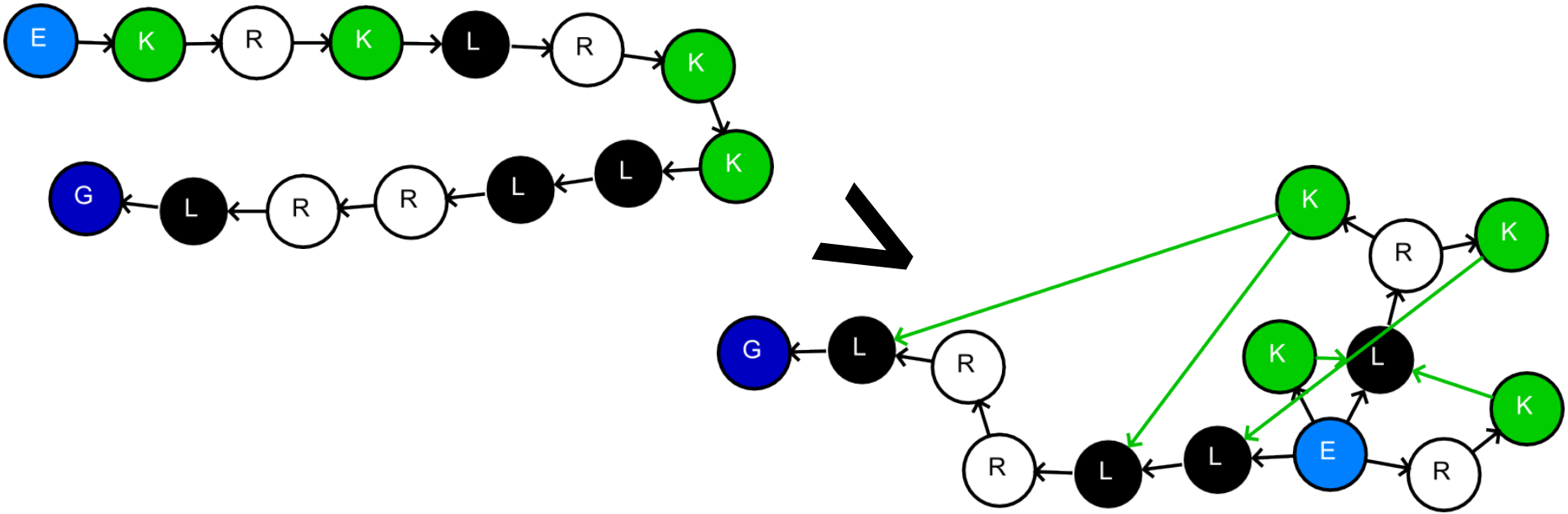


>

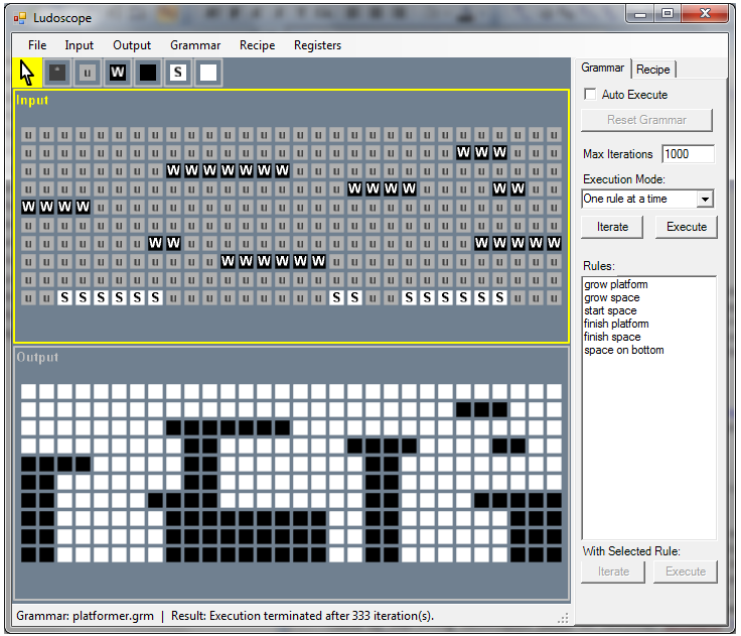
More useful



Very effective way of creating useful variety



All this could lead to powerful, experimental, automated design tools



But also ties in with certain game design aesthetics

Power Grid

The Tricky part: Correlating concrete and abstract models of the same level

Conrete:
(tilemaps, vertices)

+accurate, details, representative

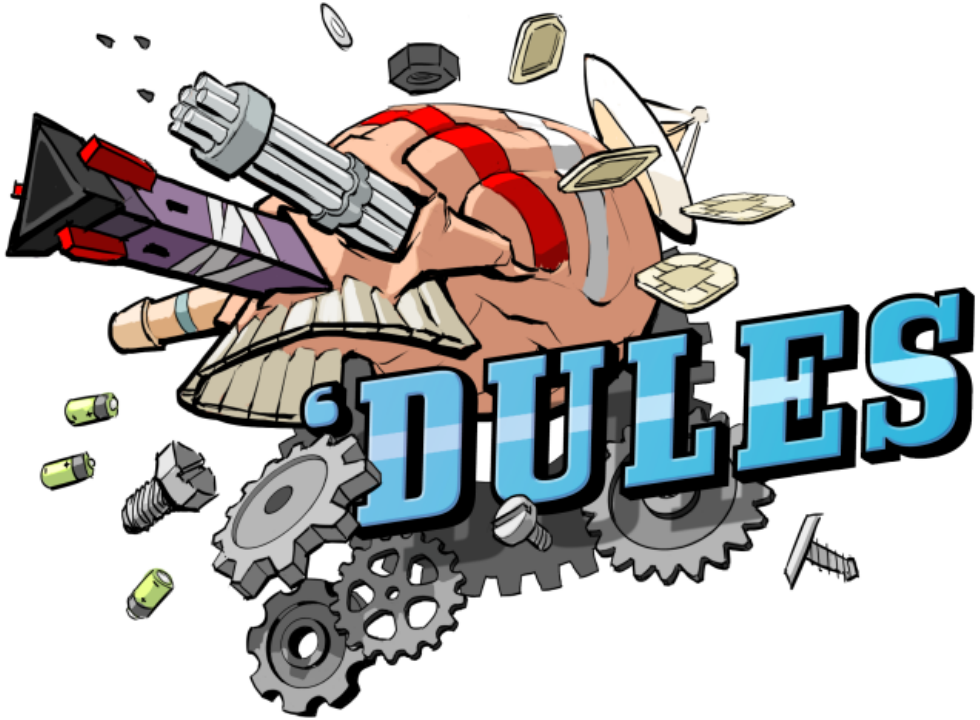
- overall shape or structure

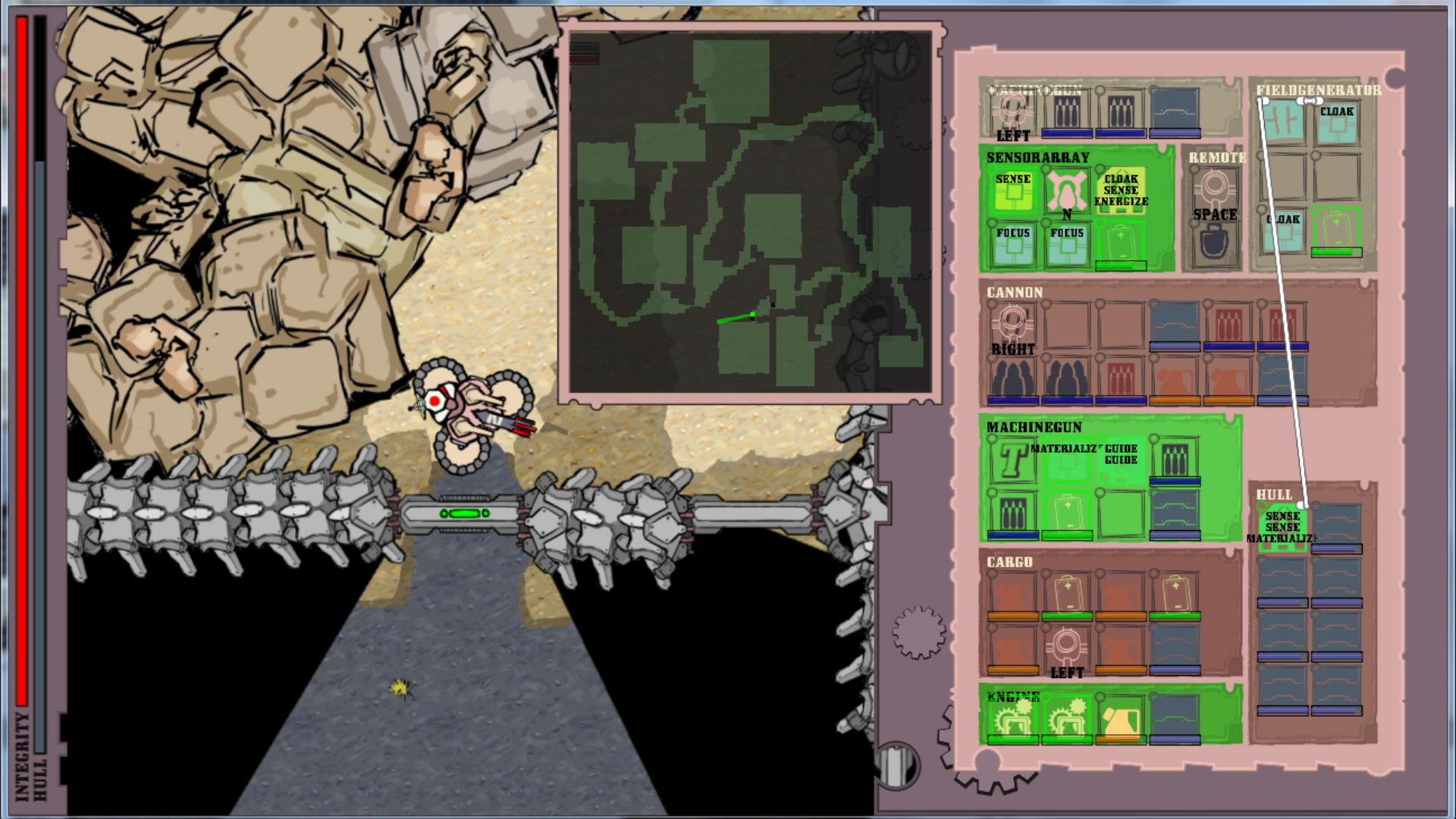
Abstract:
(graphs, strings)

+ consistency, flow, overall structure

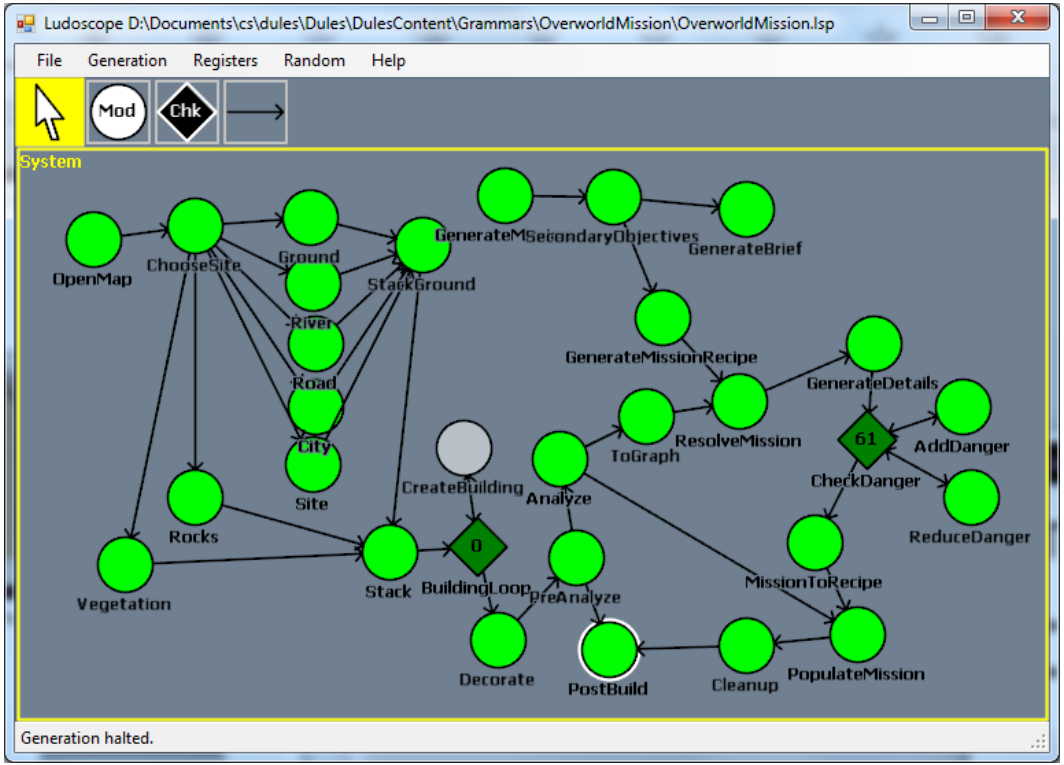
- hard to map to actual game space

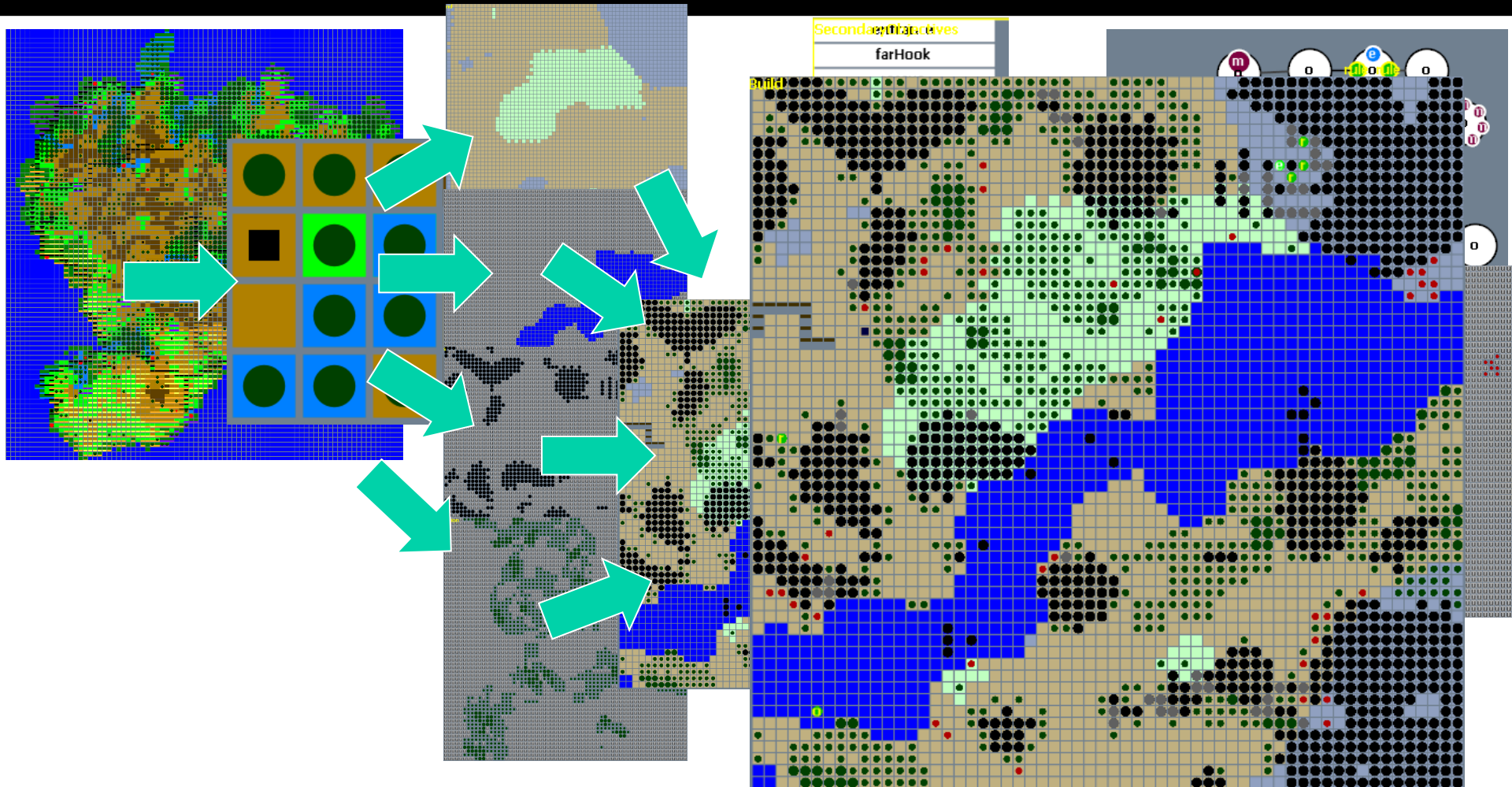
PCG in *'Dules*



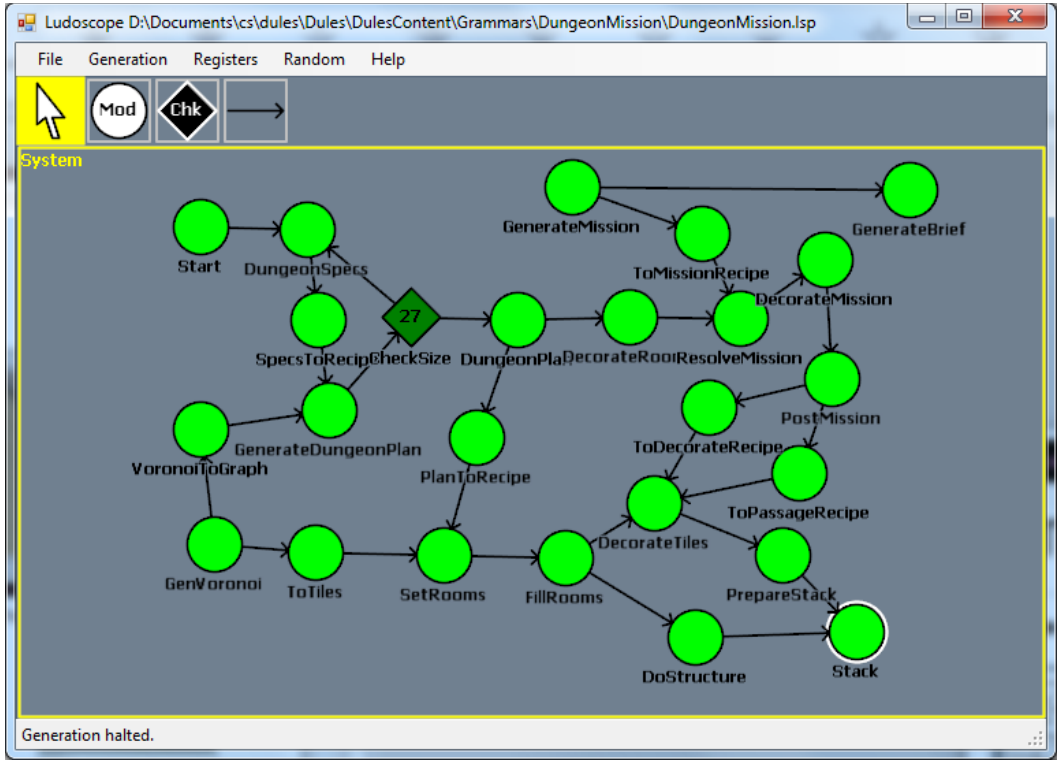

INTEGRITY
HULL
LEFT
SENSORARRAY
SENSE
FOCUS
N
FOCUS
CLOAK
SENSE
ENERGIZE
REMOTE
SPACE
FIELDGENERATOR
CLOAK
CLOAK
CANNON
RIGHT
MACHINEGUN
MATERIALIZ
GUIDE
GUIDE
HULL
SENSE
SENSE
MATERIALIZ
CARGO
LEFT
ENGINE

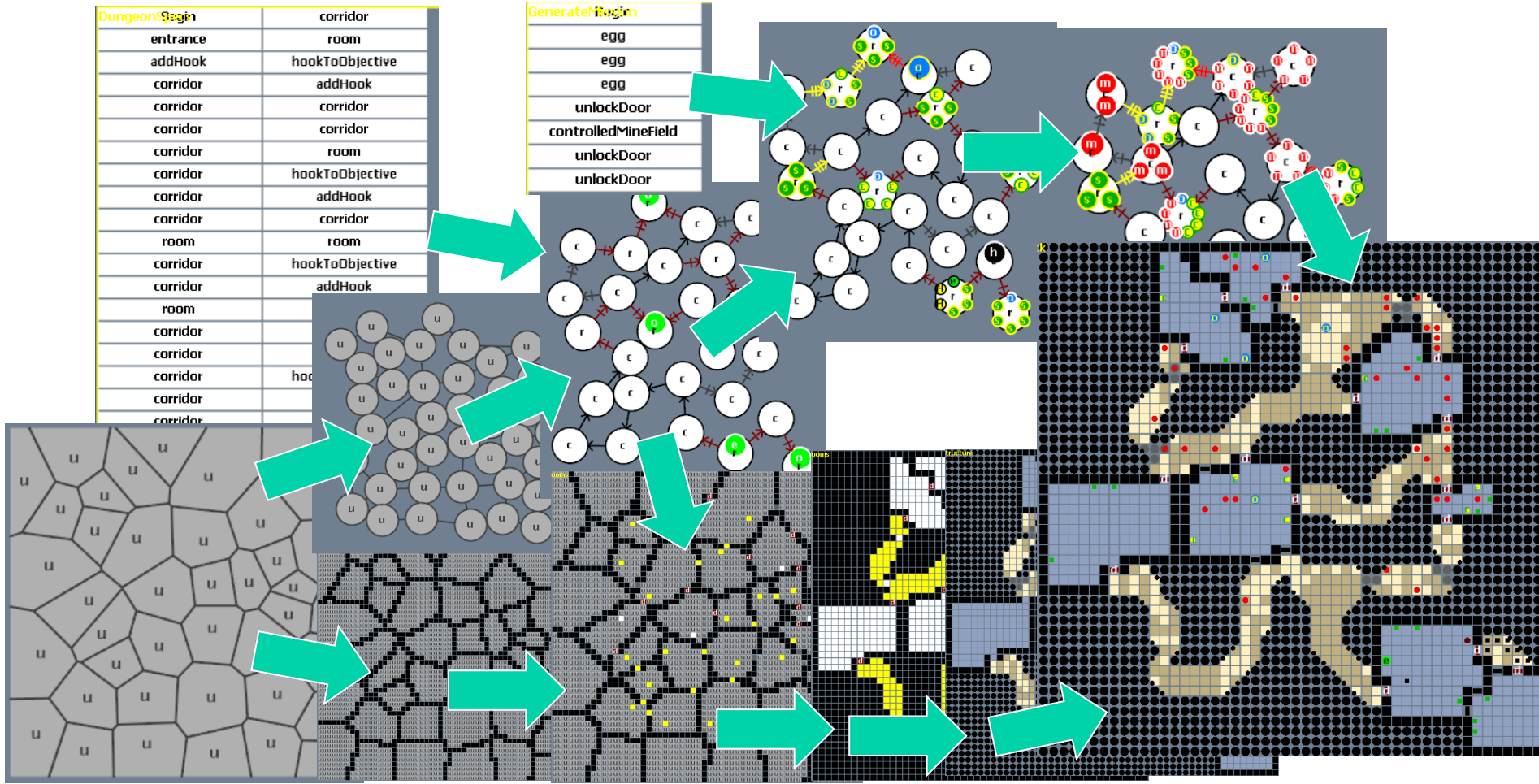
The Generation Proces in *'Dules*



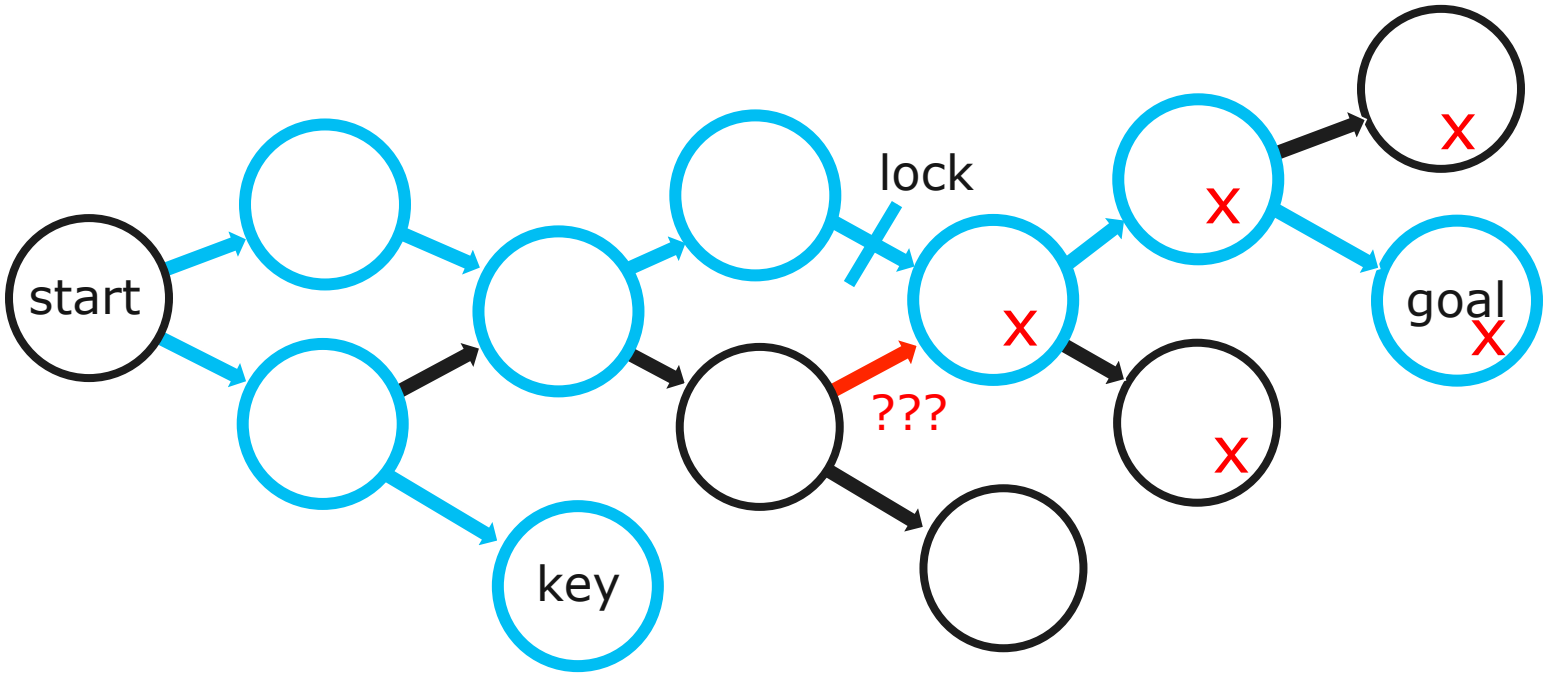

farHook

Generating dungeon missions

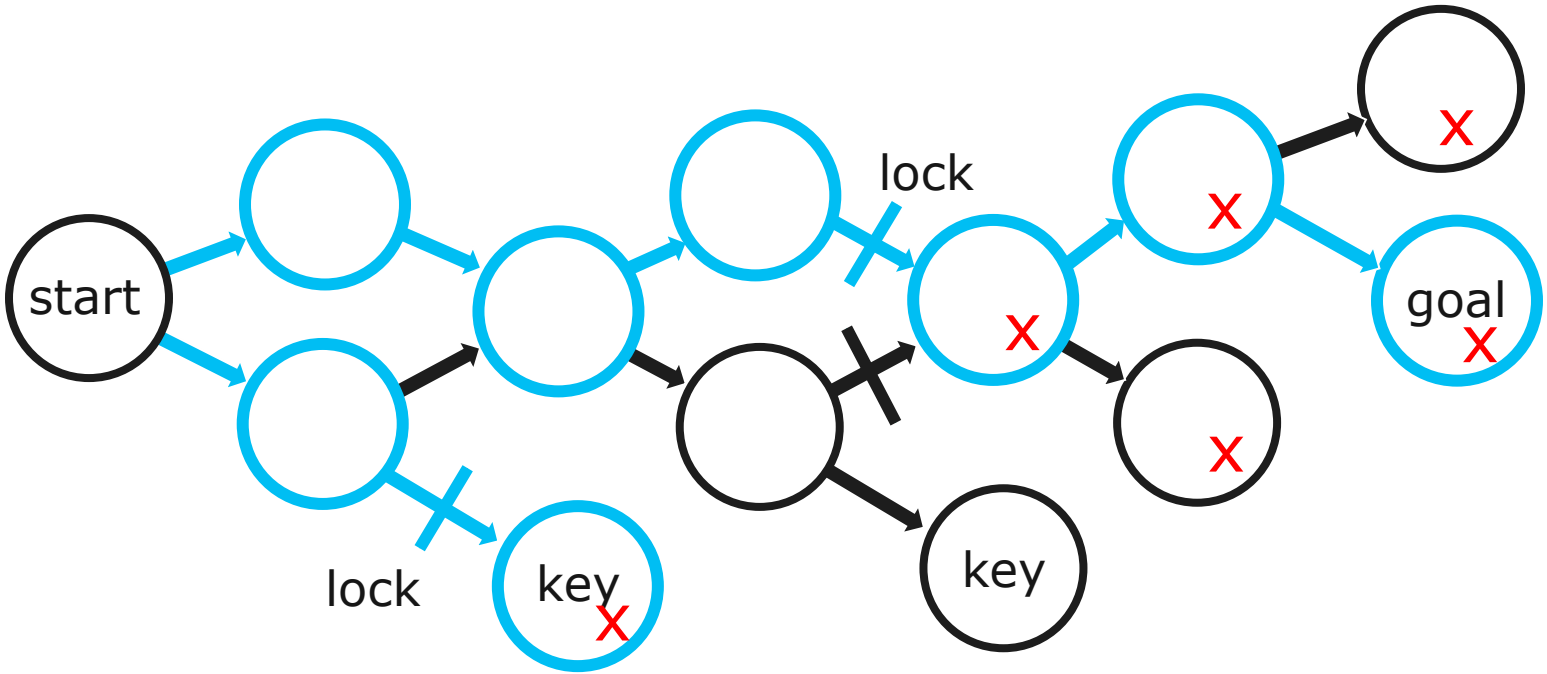



DungeonSpawn
Begin
corridor
entrance
room
addHook
hookToObjective
corridor
addHook
corridor
corridor
corridor
room
corridor
hookToObjective
corridor
addHook
corridor
corridor
room
room
corridor
hookToObjective
corridor
addHook
corridor
corridor
GenerateMain
Begin
egg
egg
egg
unlockDoor
controlledMineField
unlockDoor
unlockDoor

PCG to reflect the design process



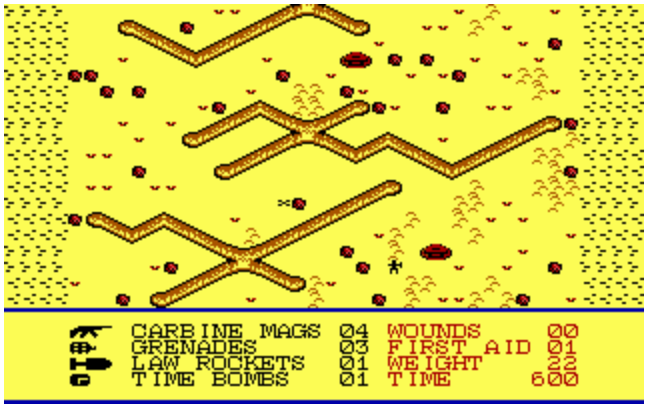
PCG to reflect the design process



Overworld Missions

Dungeon Missions

PCG as a game design aesthetics



Airborne Ranger

PCG gives a different meaning to failure

It counters repetitiveness and loss of time
(as put forward by Jesper Juul)

Spelunky

PCG gives a different meaning to winning

Each win is unique.

No win comes after endless repeat tries.

Players don’t always expect to win

Take-Aways

- Break down the PCG into multiple steps
- To be successful PCG should model the steps in the design process (top-down, not only bottom-up)
- Emergent gameplay creates much affordance for PCG
- Embrace the aesthetic implications PCG brings.

Thank you!

Joris Dormans
www.jorisdormans.nl
jd@jorisdormans.nl
www.ludomotion.com

Some references

Dormans (2010) “Adventures in Level Design”

Dormans (2011) “Level Design as Model Transformations”

Dormans & Leijnen (2012) “Combinatorial and Exploratory Creativity in Procedural Content Generation”

Juul (2010) "In search of Lost Time: on Game Goals and Failure Costs".

Smelik, et al. (2010) “Integrating procedural generation and manual editing of virtual worlds”.

Smith, et. al (2010) “Tanagra: A mixed initiative level design tool”

Smith & Mateas (2010) “Answer Set Programming for Procedural Content Generation: A Design Space Approach”

Togelius, et. al. (2010) “What is procedural content generation?”

Togelius, et. al. (2010) “Towards multiobjective procedural map generation”