Procedural Content Generation for Computer Games

Lecture 2 – Search and Evaluation

Vojtěch Černý
cerny@gamedev.cuni.cz

Sampling the space of possibilities

- Desired contents form a (often many-dimensional) possibility space, which we want to draw samples from

32	98	94	80	77	26	67	66	81	23
13	40	29	36	90	55	14	3	96	76
53	87	92	97	58	88	43	93	49	59
16	74	95	100	79	86	69	11	85	51
19	37	72	82	39	12	57	30	60	41
7	25	64	83	54	1	75	48	50	2
73	91	33	99	31	4	8	52	63	61
34	71	68	28	45	24	70	15	78	6
17	89	27	35	21	9	65	47	10	22
46	18	44	42	56	20	84	62	38	5

Attributes of the space

- Some points are preferable to others
- The space may be (heavily) constrained
  - Realistic?
  - Playable?
- We want a multitude of quite different samples

Agenda

- Different viewpoint on PCG
- Using search based methods:
  - Content representation
  - Search algorithms
  - Evaluation
- How to measure quality of PCG
- Case studies: Spelunky and Yavalath

Search based methods

- Natural approach then is to use search algorithms
- Why is it good?
  - Applicable in a wide variety of contexts
  - Tries LOTS of combinations, and finds the “best”
  - Tackles hard problems

Where is it used?

- Teleological / **Ontogenetic**
  - Could be both, but we mostly describe results
- **Design-time** / runtime
  - Runtime use of search PCG is often frowned upon due to lack of guarantees

Yavalath & Pentalath

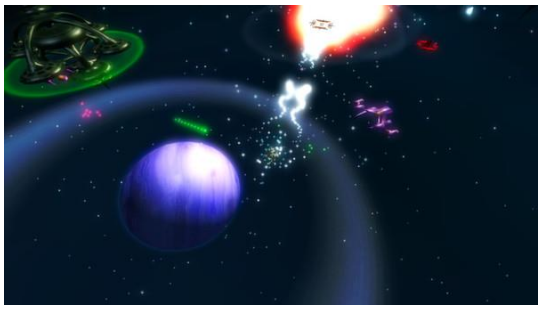
- Cameron Browne

https://boardgamegeek.com/image/1330137/pentalath

Yavalath is 4-in-a-row win, 3-in-a-row loss

Pentalath is 5-in-a-row, but with Go-like capturing of pieces

Galactic Arms Race



A variant of NEAT algorithm runs in the background.

Breaking it apart

- A search algorithm
- Content representation
- Evaluation functions

Content representation

Why it matters?

Direct representation
(binary)

Positive representation
(adding obstacles)

Negative representation
(adding tunnels)

Picture from Search-Based Procedural Generation of Maze-Like Levels by Ashlock, Lee and McGuinness

Content representation

- Pick the space you are working on
- Critical, decide well and early
- 5 viable options with Mario Levels
  - Bitmap of level layout (most direct)
  - List of positions and properties of game entities
  - List of positions of reusable patterns (gaps, hills, obstacles…)
  - List of desirable properties (# of gaps, # of enemies, …)
  - Level seed (most indirect)
- Inspired by nature: Genotype-to-Phenotype mapping

Some PCG actually is only genotype-to-phenotype mapping. E.g. some RPGs have a fixed graph of the dungeon. Of course, you could just generate the graph itself too.

Content representation

- So which is best?

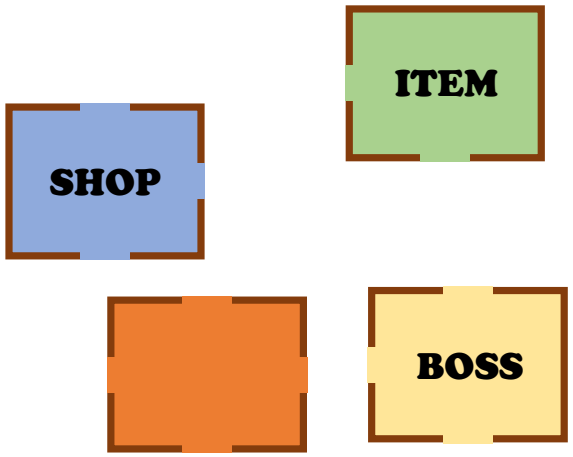
It depends. Usually neither of the extremes works too well.

Direct representation spaces are just too high-dimensional to explore well.

Indirect representation spaces lose continuousness and are hard to search reasonably.

Building blocks

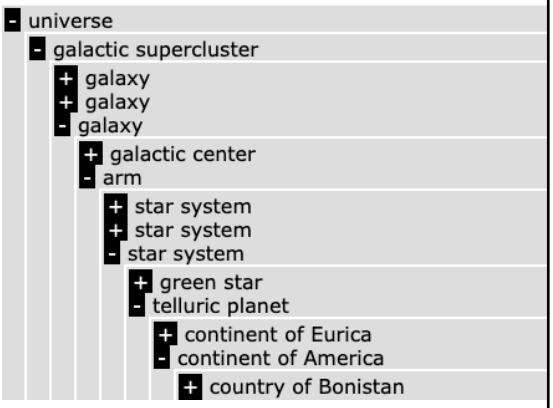
- Build content incrementally from building blocks
- Usually fixed set of blocks



- A simple way to generate content. But lacking variety.

Hierarchical representation

- Your space contains subspaces
- Divide and conquer approach
- You may use a different method in each sublevel



http://orteil.dashnet.org/nested

Template search

- Combining building blocks and hierarchical representation
- Specify possible variants within the blocks
- This is essentially a form of grammar

Pros:

- Can nicely use human created bits
- May be simpler to test and validate

Cons:

- Limited by human creation

Shape grammar is for shapes (useful in e.g. architecture)

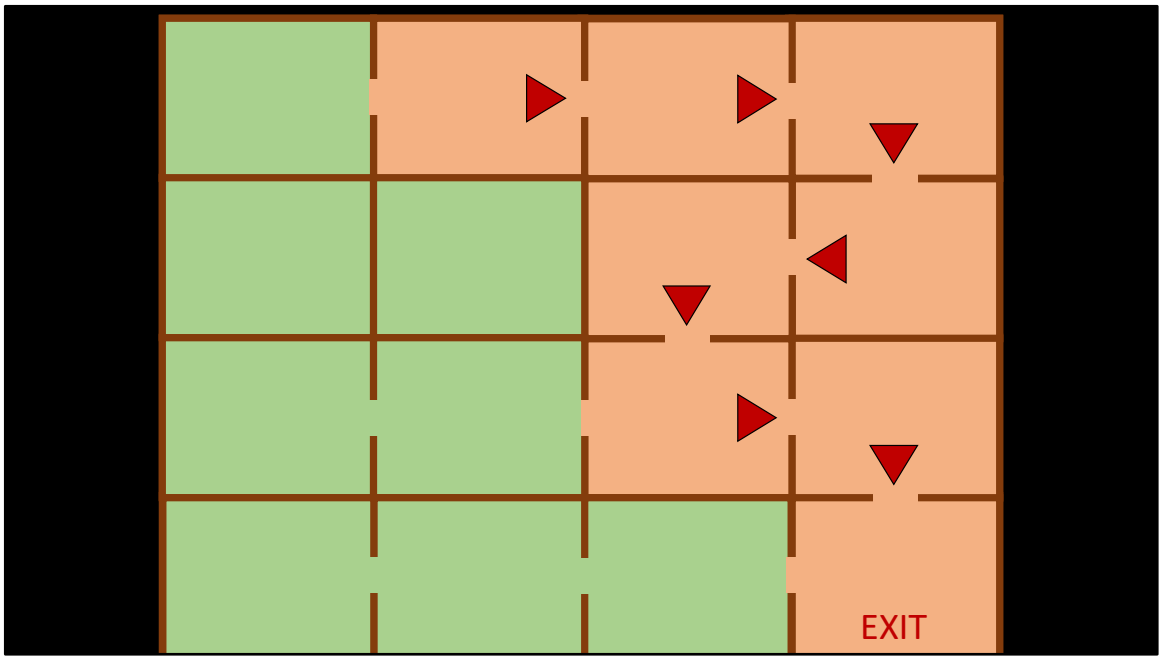
Case Study: Spelunky

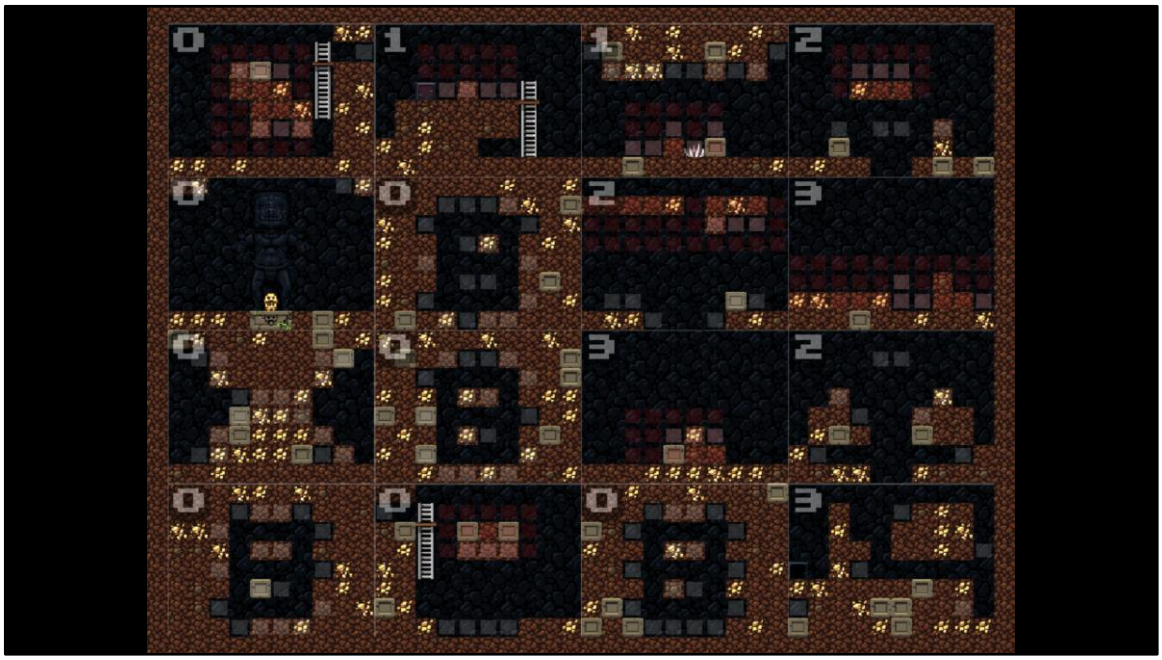
- Derek Yu, 2008
  - enhanced version 2013
- Critically acclaimed platformer with PCG levels

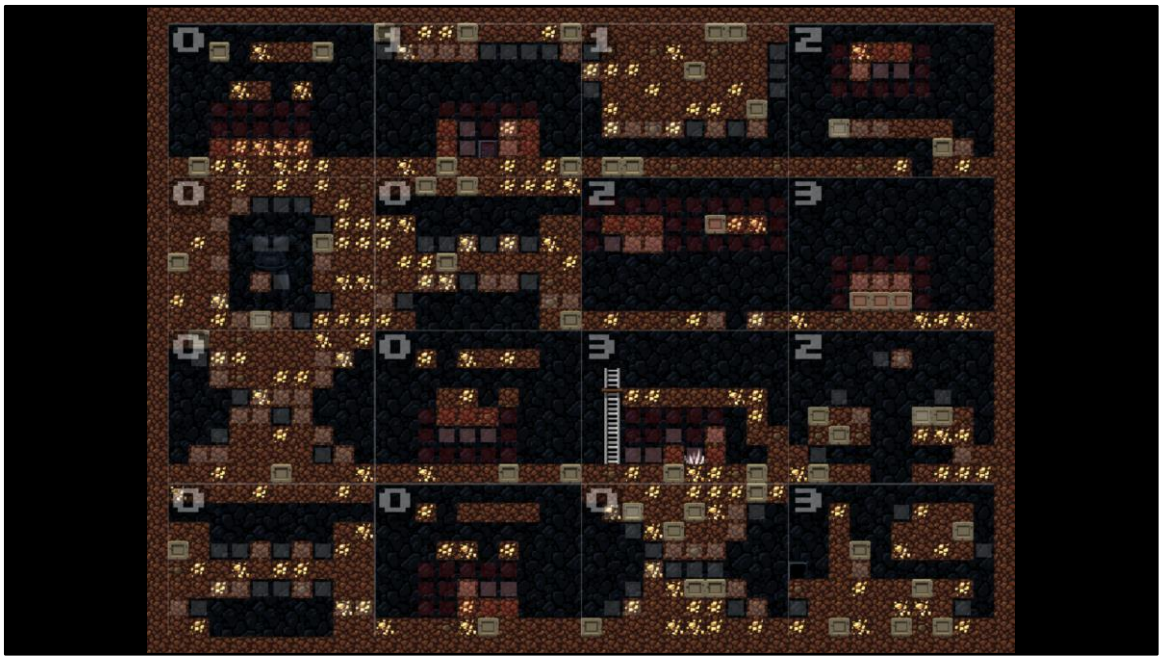
Each brief life in Spelunky tells an extraordinary story, and by the time you're willing to take a break you'll have hundreds to share.

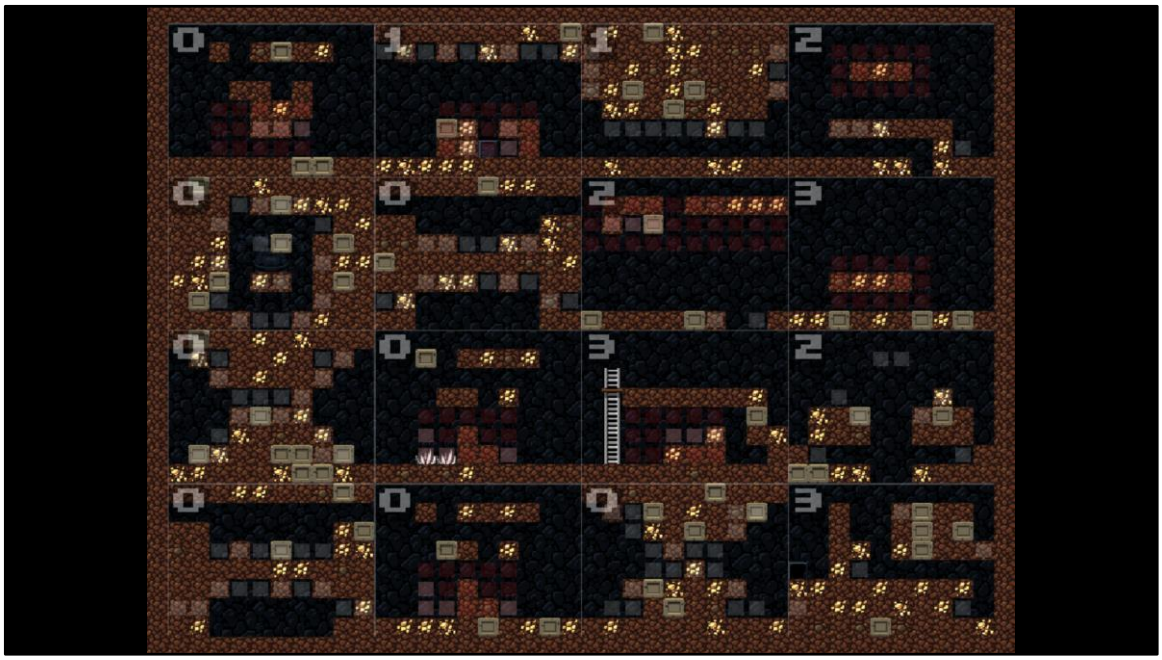
Mitch Dyer, IGN. (9/10)

http://tinysubversions.com/spelunkyGen/


EXIT




0
1
1
2
0
0
2
3
0
0
3
2
0
0
0
3



Spelunky generation, pros/cons

- *Note: there follows another sweep for monster / items, etc.*

- Content feels human-authored
- Fresh challenge every time
- Unexpected, though some patterns will be noticed

- Slightly limited possibilities

A good mix between purely PCG and human-designed approach.

Spelunky-common

Common things in PCG:

- Template instantiation
- Multiple sweeps with different approaches
  - Generate path to goal
  - Place templated rooms
  - Instantiate them
  - Place enemies, items, …

Searching

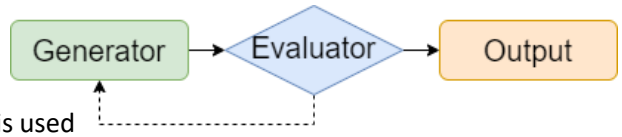
- Spelunky doesn't search, every result is used
- Why not?
  - Search can reduce the variability
  - Search adds more complexity
  - Search takes some time
- Why do it?
  - Your space is huge and you want to constrain it
  - Your space contains boring or even unplayable content
  - You have enough time (e.g. design time)
  - You don't care about variability (e.g. Yavalath)

Search algorithm

Two basic levels

- Constructive
  - Every output of the generator is used
  - Spelunky

- Generate and test (G&T)
  - Discard bad examples
  - Usually, some heuristic search is used
  - Galactic Arms Race



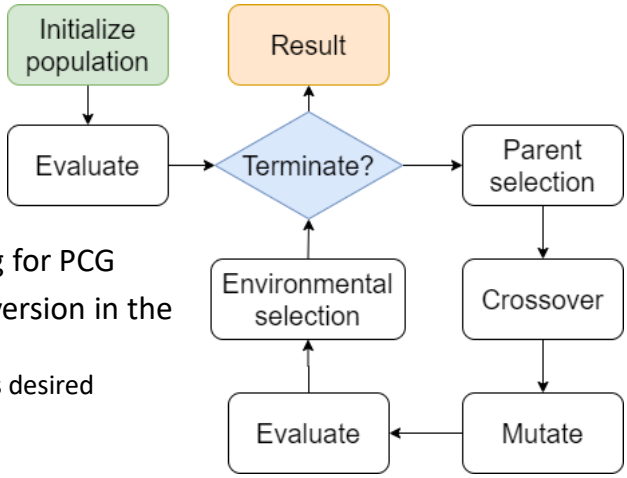
Differentiation used by Togelius. I don't like it, I feel heuristic search shouldn't be fitted under generate-and-test (which means something completely different in AIMA)

Very simple approach of G&T is to throw away bad examples and start over. This is used e.g. in Dungeonmans crypts and some versions of Angband.
Check chapter 7 – Handcrafted integration in Procedural Generation in Game Design.

Search algorithm

- *Usually* Some evolutionary algorithm (EA)
  - For more on EAs, see NAIL025 + NAIL086 courses by Roman Neruda and Martin Pilát
- Start simple!
  - Basic EA may work well enough
- Go multiobjective if you need to (most times you don't)

Basic structure of EA



- Might require different tuning for PCG
- More attention to keeping diversion in the population
  - Usually more than one result is desired

Evaluation function

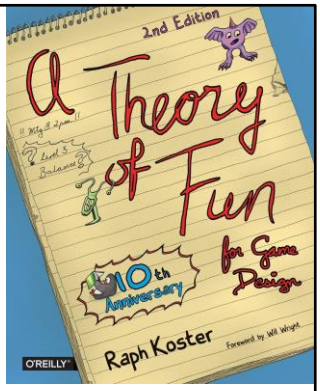
Evaluation function

- Direct
- Simulation-based
- Interactive

Direct evaluation functions

- Theory-driven
  - Models of Fun? (Koster, Malone, …)
  - Whatever game-designer feels like
- Data-driven
  - Collect lots of data and extrapolate

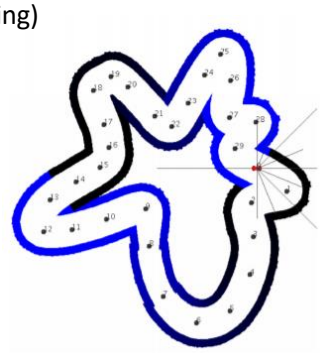
- Examples:
  - Terrain – similiarity to real life



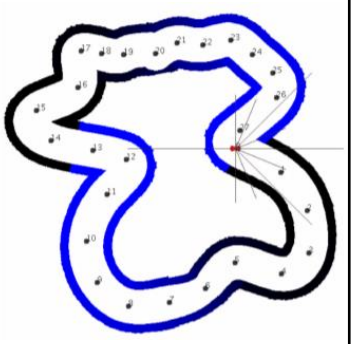
Theory driven – you look at the map and by some game design intent you know what is bad.
Theory of Fun for Game Design, Raph Koster.
Psychology of optimal experience, Mihály Csíkszentmihályi
Data driven – different kinds – (human rated data, human designed data, real life data)

Simulation-based evaluation functions

- Model a player
  - Directly (supervised learning)
  - Indirect (similar metrics)



Strong player



Weak player

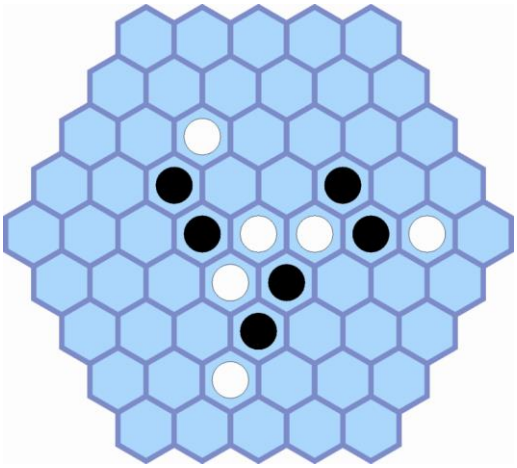
Togelius: Towards automatic personalised content creation for racing games
Direct learning: possible difficulties with generalization
Indirect learning: generalizes better, but it's a reinforcement learning task

Interactive function

- Let a human do *some* of the ratings

Empowering Quality Diversity in Dungeon Design with Interactive Constrained MAP-Elites (A. Alvarez et al)
Also, a simple thing is to just let players rate the content.

Case study: Yavalath



White wins in three moves

- PhD thesis, LUDI system
- Criteria:
  - Intrinsic (direct)
  - Extrinsic (simulation-based)
    - Quality
    - Viability
  - 16 intrinsic and 41 extrinsic criteria
- Weighted into fitness by human-trials (data-driven direct)
- Examples of criteria
  - Drama (game looked lost, but was won)
  - Length of the game

https://bitcoinmagazine.com/articles/rise-of-the-machines-1383576469

Case study: Yavalath

- 18 games hand picked out of the results
- Yavalath was the best and most innovative
  (interactive)

- Also made Pentalath (generated name was **Ndengrod**)
  - A combination of five-in-a-row and Go

End of search-based approach

Quality of PCG

The bad (yet common) approach

- *No Man's Sky* has 18,446,744,073,709,616 planets
- *Borderlands 3* has one billion gun variations
- …

a) Nobody will ever browse millions of instances.
b) Are all of them noticeably different? (10.000 bowls of oatmeal)

Better options

- Try to visualize the generated space
- However, dimensionality is often just too large and unplottable
- Lecture 1 and first HW of NAIL109 Applications of Computational Intelligence Methods is about data visualization
  - https://github.com/martinpilat/CImethods
  - (IPython notebooks, but you can open them in GitHub directly)
- Dimensionality reduction?

Better options

- Design metrics of content (capabilities of PCG)
- Examples:
  - Linearity
  - Symmetry
  - Leniency
  - Features of AI playthroughs
    - # / variety of used actions
- Preferably, these should be “emergent properties” from the view of the generator

Examples of Metrics

- Levels of various linearity in a platformer game

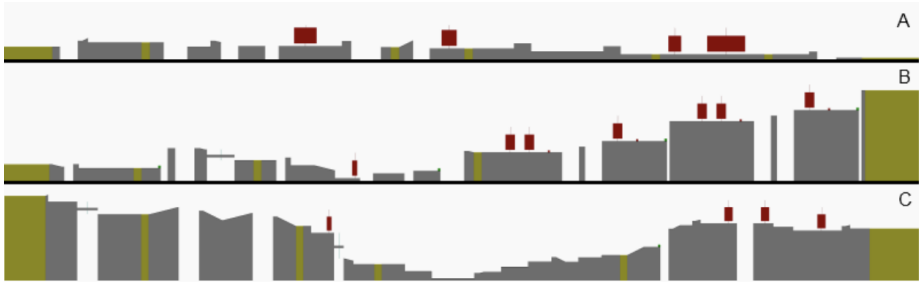


Figure 3 of **Analyzing the Expressive Range of a Level Generator** by Smith and Whitehead, 2010.

Examples of Metrics

- Levels of various leniency in a platformer game

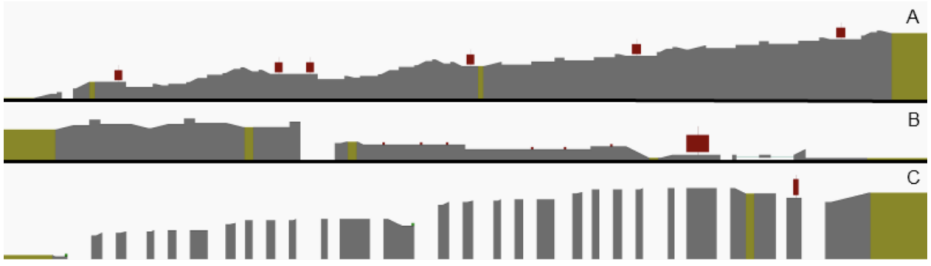


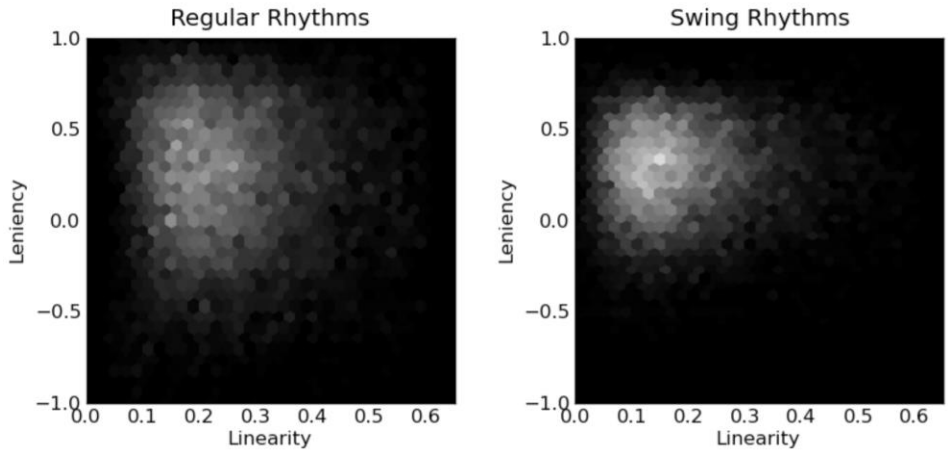
Figure 5 of **Analyzing the Expressive Range of a Level Generator** by Smith and Whitehead, 2010.

Other examples of metrics

- Density (how packed is the level)
- Compressed size ratio (measuring self-similarity / repetitiveness)

A really nice work measuring repetitiveness of music lyrics
https://pudding.cool/2017/05/song-repetition/

Plotting the generative range



(part of) Figure 7 of **Analyzing the Expressive Range of a Level Generator** by Smith and Whitehead, 2010.

Take-away points

- Search is powerful, but can be complicated
- Pick a good representation
  - “Templates” are common, powerful and human-controllable
- Consider using evolutionary algorithms
- Three possible fitness functions
  - Direct, Simulation-based, Interactive
- Don’t (!) measure your range by amount of possible content
  - Instead, use metrics
  - Plot your generative range

Q & A

cerny@gamedev.cuni.cz
discord.gg/Zts98PGw6z

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Material has been produced within and supported by the project
„Zvýšení kvality vzdělávání na UK a jeho relevance pro potřeby trhu práce“
kept under number CZ.02.2.69/0.0/0.0/16_015/0002362.