Search-Based Procedural Content Generation: A Taxonomy and Survey
http://ccg.doc.gold.ac.uk/wp-content/uploads/2016/03/togelius_tciaig12.pdf

- search-based-procedural content generation -> applying evolutionary and other metaheuristic search algorithms to automatically generate content for games
- AI focuses on creative and artistic side of game design
- Algorithms that can produce content on their own can potentially save significant expense
- PCG refers to creating game content automatically through algorithmic means -> all aspects of game other than NPC behavior and engine itself
- -> terrain, maps, levels, stories, dialogue, quests, characters, rule-sets, dynamics and weapons
- NPC AI already well documented
- rich literature on pcg of textures
- PCG algorithms create specific content based on short description (seed)
- generation process often (not always) random
- pcg desirable because
    - content can be compressed until needed -> low memory consumption
    - avoid expense of manually creating game content
    - PCG might allow emergence of completely new types of games with game mechanics built around content generation -> games with infinite replay value
    - PCG can augment limited human imagination -> inspiration for human designers
- **PCG WIKI http://pcg.wikidot.com/**
- **PCG BOOK http://pcgbook.com/**
- **OFFLINE VS ONLINE**
- Online
    - instantly generates, fast, predictable runtime & quality
- Offline
    - algorithm suggest layouts which are then edited by human designer before shipping
- **NECESSARY VS OPTIONAL CONTENT**
- Necessary
    - required by the player to progress
    - always needs to be correct -> intractable dungeon / unplayable rule set / unbeatable monsters not acceptable
    - appropriate difficulty
- Optional
    - content that the player can choose to avoid
    - acceptable if unusable weapons and floor layouts are generated
- -> analysis of what content is optional should be done on a game-for-game basis
- Borderlands randomly generated weapons
- **RANDOM SEEDS VS PARAMETER VECTORS**
- Random Seed
    - seed as input for random number generator
- Parameter Vector
    - multidimensional vector of real valued parameters that specify properties of content as input

    - e.g. dungeon generator -> #rooms, branching factor of corridors, clustering of item locations, etc.
    - degrees of control
- **STOCHASTIC VS DETERMINISTIC GENERATION**
- amount of randomness in content generation
- right amount of variation
- Stochastic
    - mutation, random seed rely on stochasticity -> search-based classified as stochastic
- Deterministic
    - always produce same content given same parameters
- **CONSTRUCTIVE VS GENERATE-AND-TEST**
- Constructive
    - generates content once and is done with it
    - needs to make sure that content is correct / good enough while being constructed
    - e.g. use of fractals to generate terrains
    - markov chain algorithm
- Generate-and-Test
    - generate and test mechanism
    - candidate content instance is generated and then tested according to some criteria
    - process continues on until content is good enough
    - evolutionary algorithms
- **Search-Based Content Procedural Content Generation**
    - test function does not simply accept or reject, rather grades using one or a vector of real number
    - fitness / evaluation function
    - aim to produce new content with higher value through previously assigned fitness value
- genotype instructions for creating a level
- phenotype actual game level
- direct / indirect encodings
- well studied case: candidates represented as vectors of real numbers
    - dimensionality -> too short / curse of dimensionality
    - high locality -> small change to genotype -> small change to phenotype & fitness
    - representation capable of representing all interesting solutions
- **EVALUATION FUNCTIONS**
    - what should be optimized? measures e.g. emotions
- Direct
    - features extracted from generated content and directly mapped to fitness
    - element of personalization possible
    - theory-driven -> designer guided by intuition and/or theory of player experience
    - data-driven -> based on data collected e.g. questionnaires or physiological measurements
- Simulated
    - based on artificial agent playing through parts of a game involving content being evaluated
    - features extracted from observed gameplay

    - static -> static agent
    - dynamic -> dynamic agent -> incorporates change / learning
    - generally more computationally expensive than direct
    - dynamic time consuming, ruling out online content generation
- Interactive
    - score content based on interaction with a player in the game
    - fitness evaluated during actual gameplay
    - data collected explicitly (questionnaires) / implicitly by measuring
    - explicit needs to be well integrated into the game
    - implicit noisy, inaccurate, low resolution and/or delayed, multimodal data -> may be technically infeasible and/or expensive
- **SURVEY**
- **Rules and Mechanics**
- Hom & Marks evolved two-player board game rules
    - rule represented directly as expression trees ZOG game description language
    - evaluation function simulation-based, static and theory-driven
- Tongelius & Schmidhuber rulesets evolved offline for grid-based games in which player moves an agent (Pac-Man)
    - Rulesets represented as fixed-length parameter vectors
    - evaluation function simulation-based, dynamic and theory-driven
    - evolutionary reinforcement learning algorithm learns ruleset
- Browne developed Ludi system for offline design of rules for board games using genetic programming
    - game rules represented directly as expression trees
    - evaluation function complex combination of direct measures and static simulation-based measures
- Smith & Mateas provide representation of game rules for simple games
    - game rules and ontologies represented indirectly as answer sets in answer set programming (ASP) -> form of constraint programming
    - generate-and-test procedure
- Salge & Mahlmann propose simulation-based evaluation function based on information-theoretic concept of relevant information
    - relevant information of game mechanic defined as minimum amount of information needed to realize optimal strategy
- **Puzzles**
- Oranchak constructed genetic algorithm-based puzzle generator for Shinro (Japanese number puzzle)
    - puzzles directly encoded as matrices
    - evaluation function simulation-based through solver
- Ashlock generated chess mazes and chromatic puzzles using evolutionary computation
    - puzzles represented directly
    - evaluation function: simulation-based and theory-driven
    - dynamic programming approach tries to solve each puzzle
- **Tracks and Levels**
- Togelius et al. designed system for offline/online generation of tracks for simple racing game
    - Tracks represented as fixed-length parameter vectors
    - racing track created by interpreting parameter vector as parameters for b-spline
    - evaluation function simulation-based, static and personalized

    - candidate tracks evaluated by neural network-based car controller
- Pedersen et al. modified open-source clone of Super Mario Bros to allow personalized level generation
    - levels represented as short parameter vector describing number, size and placement of gaps
    - evaluation function direct, data-driven personalized using neural network converting level information to emotional state predictors
- Sorenson & Pasquier devised indirect game level representation for number of different but related genres
    - representation based on “design elements” (e.g. platforms, enemies)
    - design elements laid out spatially in genome to simplify crossover
    - Levels first tested on general validity -> if test fails relegated to second population using FI-2Pop evolutionary algorithm designed for constraint satisfaction problems
    - evaluation function direct or weakly simulation-based theory-driven
- Jennings-Teats et al. describe initial work towards creating platform game levels that adapt difficulty to human player during runtime
    - generation process based on generate-and-test with optimization mechanisms
    - level parts evaluated by a number of critics based on acquired difficulty ratings of level segments
- **Terrains and Maps**
- Frade et al. evolved terrains for Chapas
    - terrain represented indirectly as expression trees which were evolved with genetic programming using approach similar to CPPN encoding
    - evaluation function direct and theory-driven scores maps depending on largest connected area of flat terrain
- Togelius et al. designed method for generating maps for RTS games
    - positions of bases and resources directly represented as coordinates
    - other terrain features represented more indirectly
    - collection of direct and lightly simulation-based theory-driven evaluation functions used
    - functions directly motivated by gameplay considerations and to a large extent based on A* search algorithm
    - multiobjective evolutionary algorithm MOEA
- constructive methods not search based
    - generation based on fractals
    - diamond-square algorithm
    - agent-based simulated erosion
    - cellular automata
    - Diorama map generator using answer set programming
- Ashlock et al. proposed indirect search-based method for landscape generation based on evolvable L-system representation
    - uses approach to evolve fractal landscapes to fit specific shapes
- **Narrative and Storytelling**
- constructive and generate-and-test approaches
- core mechanism version of classical AI planning
- Wardrip-Fruin’s book
- **Optional Content**
- **Weapons**
- Hastings et al. developed multiplayer game built on search-based PCG Galactic Arms Race

    - Weapons represented indirectly as variable-size vectors of real values
    - evaluation function interactive, implicit and distributed
    - weapon evaluated based on user usage
- Same authors added a method that enables more directly player-controlled weapon generation
    - players allow to perturb individual genes of their choice
- **Buildings**
- Marin et al. designed system for interactively evolving buildings for Subversion
    - buildings represented indirectly in custom mark-up language, describing each building from bottom up as stack of three-dimensional objects
    - explicit interactive evaluation function user selects two parents -> system produces 16 offspring buildings
- **Camera Control**
- Burelli & Yannakakis devised method for controlling in-game camera movements to keep specified objects or characters in view / out of view
    - camera configurations evaluated by calculating visibility of selected objects
- Yannakakis et al. introduce notion of affect-driven camera control within games, associating player affective to camera profiles and player physiology
    - affective models constructed using neuroevolutionary preference learning on questionnaire data
    - camera profiles represented as set of 3 parameters: distance height from player character and frame-to-frame coherence
    - direct, data-driven evaluation function represented by neural network predictor of emotion
- **Trees**
- Talton et al. developed system for offline manual exploration of design spaces of 3D-models
    - system lets user view and navigate space
    - interactive evaluation function
    - tree models represented as fixed-length vectors of real-numbers
    - underlying algorithm comparable to estimation of distribution algorithm EDA
- **Major research challenges in SBPCG**
- Which type of content are suitable to generate?
    - which types can be generated well, fast and reliable enough
- How can we avoid catastrophic failure?
    - PCG reliability
- How can we speed up generation?
    - speed up generation process as much as possible
- How is game content represented?
    - useful to have a set of principles and best practices for content representation
- How can player models be incorporated into evaluation functions?
    - advantageous to move to data-driven evaluation functions based on experiments on real players
- Can we combine interactive and theory-driven evaluation functions?
    - combine interactive evaluation of content with data-driven evaluation functions, allowing models to inform each other

- Can we combine search-based PCG with top-down approaches?
    - hybrid approaches
- How can we best assess the quality and potential of content generators?
    - assess other properties of content generators

PCGRL: Procedural Content Generation via Reinforcement Learning
https://arxiv.org/pdf/2001.09212.pdf

- search policies that generate game content
- at each step policy asked to take action that leads to highest expected final level quality
- advantages over existing methods
    - generating a new level once a model has been trained is much faster since no search needs to be performed
    - comes at cost of having a long training phase
    - no training data necessary
    - incremental nature of trained policies makes them more suitable for interactive and mixed initiative approaches
- iterative content generation
    - start with a level populated by random tiles
    - at each step agents is allowed to make a small change
    - change will be judged by the system with respect to a target goal for the level and assigned a reward
    - system also determines termination
- 3 part framework
- Problem module
    - stores information about generated level such as goal, reward function
- Representation module
    - responsible for transforming current level into a viable observation for the agent
- Change percentage
    - limits number of changes agent can affect in the content over the course of an episode

Search-based procedural content generation for GVG-LG
https://reader.elsevier.com/reader/sd/pii/S1568494619306908?token=4EC17E933FD0626A5C9DC4F76D3708E02ABA89DF97BE1DAFDC3746CCA2F045567958F86E88D5A403EC58CF6C9719E2D8

- PCG refers to the algorithmic generation of game content with limited human intervention
- constructive, generate and test, machine learning based
- level generation requires variation, coherent style, speed and correctness
- re-usability and cost of generators are problematic
- FI2PO algorithm
    - maintains set of two populations feasible / infeasible
    - feasible -> solutions that satisfy constraint, infeasible vice versa
    - levels represented as 2D array of tiles
    - each tile consists of array of strings representing all sprites
    - generator uses one-point crossover which swaps 2 chromosomes around a random tile

        - create, destroy and swap mutations
        - simulation-based fitness and constraint evaluation
- Fitness metrics
    - Aesthetics
    - Density
    - Reachability
    - Number of room structures
    - Symmetry
    - Balance
    - Difficulty
    - Number of enemies in a level
    - Enemy sparsity
    - Percentage of empty spaces
    - Number of isolated elements
    - Number of goal sprites

Computational Creativity for Valid Rube Goldberg Machines - http://computationalcreativity.net/iccc2018/sites/default/files/papers/ICCC_2018_paper_19.pdf

- “RGMs should work but they also need to capture attention. The more theatrical and funny your machine is, the better it will score!”
- artificial intelligence system to create novel science experiments
- RGM good at engaging students and helping them gain deeper understanding of difficult concepts

**An Angry Birds Level Generator with Rube Goldberg Machine Mechanisms**
http://www.ice.ci.ritsumei.ac.jp/~ruck/PAP/cog2019-abdullah.pdf

- method for generating levels that feature a domino effect, allowing them to be completed by one shot of a bird
- RGM machine intentionally designed to perform a simple task in an indirect and over-complicated fashion
- machines consist of a series of simple devices that are linked together to produce a domino effect
- each device triggers next one and original goal is achieved after many steps
- RGM educational effectiveness in motivation
- innovative humorous unconventional nature
- RGM have strong potential in entertaining people who watch them
- levels generated using predefined set of objects -> segments
- segments chained using information of previous segments output destination
- process repeated until desired number of segments or max level boundary reached
- level generated as combination of segments containing various objects
- input object that eventually results in movement of output object
- position of segment of interest determined by output destination of preceding one
- structures generated as set of several layers stacked on top of each other
- layer consists of blocks of same type but with different materials
- blocks in a layer are arranged symmetrically according to its base layer

- evaluation factors: stability, running time, expressivity, perfect-shot rate
- dynamic measure

**Generating Angry Birds-Like Levels With Domino Effects Using Constrained Novelty Search**

- generate Angry Birds levels featuring RGM mechanism using Constrained Novelty Search (CNS)
- level must satisfy several rules
    - structure stable
    - all pigs must be destroyed when structure collapses
    - when collapsing one object outside boundary flying
- variables for structures
    - structure boundary
    - input / output object
    - input / output direction
    - output position
- genotype 2-dim array with integer-type identifiers corresponding to objects
- simulation process to evaluate individuals
- feasible individual scored upon novelty score
- diversity measured through visual distance
- Fitness function consisting of
    - stability, destroyed pigs and object flying outside boundary
- structure selection based on connecting sections in- and outputs
- **-> higher diversity to previous algorithm**

Thesis Proposal

- eng?
- für wen?

- PCG
- mehr Nebenbedingungen?
- RGM wie genau generieren?
- objekte?

- andere arbeiten
- framework oder selber -> UNITY
- paper mit ideen suchen ähnliche themenrichtung