Procedural Content Generation in Games
Perspectives from the Ivory Tower

Julian Togelius

IT University of Copenhagen / New York University
Who am I?

• From Malmö, Sweden
• Studied in Lund, Sussex, Essex
• Worked in Lugano, Copenhagen, New York
• philosophy + psychology >> artificial intelligence + robotics >> games
• Current research focus: AI in games (player modelling, procedural content generation, evolutionary computation)
Further reading

pcgbook.com

julian.togelius.com
What is PCG in games?

- Procedural Generation: with no or limited human intervention, algorithmically
- of Content: not NPC behaviour, not the game engine, things that affect gameplay
- in Games: computer games, board games... any kind of games
Game content, e.g.

- Levels, tracks, maps, terrains, dungeons, puzzles, buildings, trees, grass, fire, plots, descriptions, scenarios, dialogue, quests, characters, rules, boards, parameters, camera viewpoint, dynamics, weapons, clothing, vehicles, personalities...
Elite
Elite
Elite

[Image of a game interface with various icons and numbers, including 'Buy Cargo']
Elite

Mission No. 1

From: Commander Gerocher

A prototype model of the new top-secret military ship, the Constrictor, has been stolen by unknown agents from the Navy Research Base at Riaadi.

Due to a capabilities of this ship, the Galactic Co-operative of Worlds is offering a large reward to anyone who destroys the Constrictor before it falls into enemy hands.

Launch my ship Earth, fly to local planet, price item bank status file and who knows.

Accept mission? [ ]
Elite

Fits in memory on a Commodore 64!
Rogue
Diablo III
Dwarf Fortress
Spelunky
Far Cry 2
SpeedTree
Civilization IV
Borderlands
Galactic Arms Race
Ludi / Yavalath
• Can we drastically cut game development costs by creating content automatically from designers’ intentions?

• Can we create games that adapt their game worlds to the preferences of the player?

• Can we create endless games?

• Can the computer circumvent or augment limited human creativity and create new types of games?

• Can we understand game design through formalising the design process?
What are the problems?

• *Speed*
  Real-time? Or design-time?

• *Reliability*
  Catastrophic failures break gameplay

• *Controllability*
  Allow specification of constraints and goals

• *Diversity*
  Content looks like variations on a theme

• *Creativity*
  Content looks “computer-generated”
Search-based PCG

• Currently widely researched in academia
• Making its way into commercial games, e.g. City Conquest
• Doing PCG with evolutionary algorithms or similar methods
Evolutionary computation?

- Keep a population of candidates
- Measure the fitness of each candidate
- Remove the worst candidates
- Replace with copies of the best (least bad) candidates
- Mutate/crossover the copies
And of course, the algorithm!

- Lots of different types of evolutionary algorithms: Genetic Algorithms, Evolution Strategies, Evolutionary Programming
- And evolution-like algorithms: Particle Swarm Optimisation, Differential Evolution
- Keep It Simple, Stupid!
  - Often, simple $\mu+\lambda$ ES with no crossover and no self-adaptation works well enough
Simple $\mu+\lambda$ ES

- Create a population of $\mu+\lambda$ individuals
- Each generation
  - Evaluate all individuals in the population
  - Sort by fitness
  - Remove the worst $\lambda$ individuals
  - Replace with mutated copies of the $\mu$ best
The fitness landscape
How would we generate levels for Super Mario Bros?
The Mario AI Benchmark

- Reasonably faithful clone of SMB 1/3
- APIs for level generators and AI controllers
Representation

- A number of “vertical slices” are identified from the original SMB levels
- Levels are represented as strings, where each character corresponds to a pattern
Evaluation

- 25 patterns are identified in the original SMB levels
- e.g. enemy hordes, pipe valleys, 3-paths…
- The fitness function counts the number of patterns found in the level
The one-point crossover is illustrated in figure 7. A rhythmic variation between exciting parts and calm parts is achieved by injecting new expressions generated from this search-based approach. The evolutionary approach together with vertical slices of patterns allows for a full complex sequence. However, if a beginning, a full pattern and perhaps another beginning or ending is in a different order, the fitness function that in principle, punished any form of pattern or beginning or end of a pattern. Level two-level (see section 8) in figure 8 is overfilling the game space and level two-level showing tendencies to stack patterns on top of each other. A beginning or ending is typically rewarded less than a full complex sequence. However, if a beginning, a full pattern and perhaps another beginning or ending is in a different order, the fitness function that in principle, punished any form of pattern or beginning or end of a pattern. Level two-level showing tendencies to overfill game space.
Procedural map generation for RTS games

StarCraft

- Classic real-time strategy game
- Korea’s unofficial national sport
- Two or three player competitive matches
- Three distinct races
StarCraft map features
Our approach

- Define desirable traits of RTS maps
- Operationalise these traits as fitness functions
- Define a search space for maps
- Search for maps that satisfy the fitness functions as well as possible, using multiobjective evolution
Desirable traits of an RTS map

• Playability
• Fairness
• Skill differentiation
• Interestingness
Playability
fitness functions

• Base space: minimum amount of space around bases

• Base distance: minimum distance between bases (via A*)
Fairness fitness functions

- Distance from base to closest resource
- Resource ownership
- Resource safety
- Resource fairness

![Diagram showing resource fairness with unsafe and safe resources.](image)
Skill differentiation fitness functions

(also contribute to interestingness)

• Choke points
  (narrowest width of shortest path)

• Path overlapping
Evolved map

Resource fairness vs. choke points
Three-player map
Another three-player map
What about the human designer?

- PCG sometimes seen as a one-way process
- Let’s face it: we cannot replace all aspects of the designer or even developer… yet
- Some people don’t want to be replaced
Mixed-initiative

- Human designer
- Computational designer
- Both take initiatives
- sliding scale of initiative
Sliding scale of initiative

- Human primary, computer as “slave”: Computer-aided design

- Computer creating content, human as “guide”: Interactive evolution
Examples of smart CAD tools

• Tanagra (Smith et al)
• Sentient Sketchbook (Liapis et al)
• Ropossum (Shaker et al)
Tanagra

G. Smith, J. Whitehead. Tanagra. TCI AIG 2012
Tanagra

- 2D platformers
- Playability ensured via constraint solving
- Evaluate and change level according to pacing
- User can “lock” geometry, computer changes rest
Sentient Sketchbook

Sentient Sketchbook

- Map Sketches (strategy game, dungeon, FPS level)
- multiple solutions evolved & shown in real-time
- fitnesses on area influence, exploration and balance
- constraints on playability handled with FI-2pop GA
A user can select among a predefined set of map sizes. Map size determines the number of allowed bases and resources.
Ropossum

- Physics-based puzzle: “cut the rope”
- Grammatical genetic programming for creating new puzzles
- Playability module for testing how (if?) to solve puzzles using constrained tree search
- Using the designer’s input in complete or partial designs

M. Shaker, N. Shaker, J. Togelius. Evolving Playable Content for Cut the Rope through a Simulation-Based Approach

CIG 2013
Click the air cushion to blow object
Generate Level Samples
How about creating the game itself?
An example: *Ludi* creating board games

- Construct a language that can describe games…
- …and a game engine that can play any game described in the language
- Then, use *evolution* to design games!
The Ludi Game Description Language

- In practice limited to board games
- *Ludeme*: Fundamental units of independently transferable game information ("game meme")
  - (tiling square)
  - (size 3 3)
Tic-Tac-Toe

(game Tic-Tac-Toe
  (players White Black)
  (board
    (tiling square i-nbors)
    (size 3 3)
  )
  (end (All win (in-a-row 3)))
)
The term game shall henceforth refer to a two-player combinatorial game throughout this paper. Such games are an ideal test bed for the experiments as they are typically deep but described by simple, well defined rule sets.

Note that this is not a work in combinatorial game theory (CGT), which is concerned with the analysis of games with a view to solving them or at least finding optimal strategies [3] and developing artificial players able to challenge human experts. Within the context of this study, the artificial player is of little interest except as a means for providing self-play simulations. While it must be of sufficient strength to provide meaningful playouts, we are concerned primarily with the quality of the game itself rather than the quality of the player.

B. Ludemes

Just as a meme is a unit of information that replicates from one person to another [4], a ludeme is a game meme or unit of game information. First coined by Borvo [5], this term describes a fundamental unit of play often equivalent to a rule; ludemes are the conceptual equivalent of a game's components – both material and non-material – and are notable for their ability to pass from one game or game class to another [6].

Ludemes may be single units of information, such as the following items that describe aspects of the game board shown in Fig. 1(a):

(size 3 3)

Conceptually related items may be encapsulated to form higher level compound ludemes as follows:

(board (tiling square) (size 3 3))

Collecting rules into such compound ludemes is a convenient way to describe games. For example, the essence of Tic-Tac-Toe may be succinctly described as follows (assuming a two-player combinatorial model):

(game Tic-Tac-Toe (board (tiling square) (size 3 3)) (win (in-a-row 3)))

The concept of an entire game as an item of information may seem odd but it is valid; there exist many examples of identical games being discovered, fully formed, at similar times. The most famous case is the independent discovery of Hex by mathematicians Piet Hein and John Nash in the 1940s [201x107]. A more recent example is Chameleon, discovered by New Zealand and USA designers within a week of each other in 2003. Such cases may be examples of “memetic convergence” in action towards optimal designs.

C. Recombination Games

Given a game in its ludemic form, it is a simple matter to manipulate its rules to create variants and new games. For Tic-Tac-Toe, such modifications might include the board size:

(size 2 2)

or the target line length:

(win (in-a-row 2))

However, a moment’s reflection will reveal that each of these changes break the game, by making it unwinnable in the first case and trivially winnable in the second. Other manipulations might involve extending the board to three dimensions, as shown in Fig. 1(b):

(size 3 3 3)

or inverting the end condition to give a misère version:

(lose (in-a-row 3))

These variants are both more interesting but still trivially solvable, and are more notable for their novelty value than any inherent value as games. There is much room for improvement in this branch of the N-in-a-row family.

The difficulty of deriving an interesting game from Tic-Tac-Toe does not just stem from the fact that it is itself flawed (it is drawish if played correctly). There is the serious problem that rule sets for combinatorial games tend to be highly optimised and fragile; authors strive for the simplest rule sets that give the deepest playing experience, and the slightest change will generally break a game. As in most creative fields, it is easy to generate artificial content but much more difficult to generate artificial content of human expert quality.
EVALUATE

INBRED? (Y/N)

DRAWISH? (Y/N)

CHOOSE POLICY

TOO SLOW? (Y/N)

BAPTISE

WELL FORMED? (Y/N)

RULE CHECK

CROSSOVER

MUTATE

SELECT PARENTS

POPULATION

BIN
Yavalath

`Yavalath`

```lisp
(game Yavalath
  (players White Black)
  (board (tiling hex) (shape hex) (size 5))
  (end
    (All win (in-a-row 4))
    (All lose (and (in-a-row 3) (not (in-a-row 4)))))
)
```
Beyond bored games
Automatic Game Design

• Simple Pac-Man like games
• Rule encoding: what happens when things collide
• Fitness function: learnability

(Togelius and Schmidhuber 2008)
The Video Game Description Language

- Developed in order to be able to represent most games from the Atari 2600 era (and many from the C64 era)
- Assumes 2D movement and graphical logic
- Compact and human-readable
- Not (explicitly) made for reasoning
- Game engines in Java and Python
BasicGame

SpriteSet
sword > Flicker color=LIGHTGRAY limit=1 singleton=True img=sword.png
dirt > Immovable color=Brown img=dirt.png
door > Door color=GREEN img=door.png
diamond > Resource color=yellow limit=10 shrinkFactor=0.75 img=diamond.png
boulder > Missile orientation=down color=GRAY speed=0.2 img=boulder.png

moving =
    avatar > ShootAvatar style=sword img=avatar.png
    enemy = RandomNPC
crab > color=RED img=camel.png
butterfly > color=PINK img=butterfly.png

LevelMapping
.
E > exitDoor
0 > boulder
x > diamond
c = crab
b > butterfly

InteractionSet
dirt avatar > killSprite
dirt sword > killSprite
diamond avatar > collectResource
diamond avatar > killSprite scoreChange=2
moving wall > stepBack
moving boulder > stepBack
avatar boulder > killIfFromAbove scoreChange=-1
avatar butterfly > killSprite scoreChange=-1
avatar crab > killSprite scoreChange=-1
boulder dirt > stepBack
boulder wall > stepBack
boulder diamond > stepBack
boulder boulder > stepBack
enemy dirt > stepBack
enemy diamond > stepBack
 crab butterfly > killSprite
butterfly crab > transformTo style=diamond scoreChange=1
exitDoor avatar > killIfOtherHasMore resource=diamond limit=0

TerminationSet
SpriteCounter style=avatar limit=0 win=False
SpriteCounter style=exitDoor limit=0 win=True
Random controller on Boulder Dash
MCTS controller on Boulder Dash
Could we generate any type of game?
Characteristics of generatable games

• Short - or at least short game atoms
• One player, one and a half or maybe two
• Easy to acquire heuristics (learnable)
• Clear outcomes and ending conditions
• Not too high branching factor
• Preferably deterministic
The end of game design?

• No - our AI is nowhere near good enough yet

• No - there will always be need for human supervision

• Yes - lots of repetitive and low skilled design tasks will become automated
The end of this talk

- More about me: 
  http://julian.togelius.com
- More about PCG: 
  http://pcgbook.com
- More about GVG/VGDL: 
  http://www.gvgai.net