Evolutionary Computation and Games

Tutorial at GECCO 2016

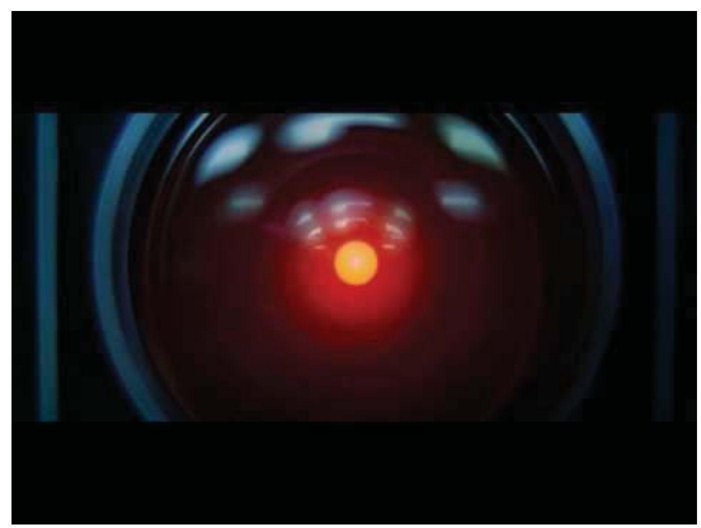
Julian Togelius
New York University

GECCO'16 Companion, July 20-24, 2016, Denver, CO, USA.
ACM 978-1-4503-4323-7/16/07.
DOI: http://dx.doi.org/10.1145/2908961.2927001

Who am I?

- From Malmö, Sweden
- Studied: Lund (Sweden) >> Sussex >> Essex (UK)
- Worked: Lugano (Switzerland) >> Copenhagen >> NYU
- philosophy + psychology >> artificial intelligence + robotics >> games
- Current research focus: AI in games (player modeling, procedural content generation, evolutionary computation)

Artificial Intelligence



Artificial Intelligence

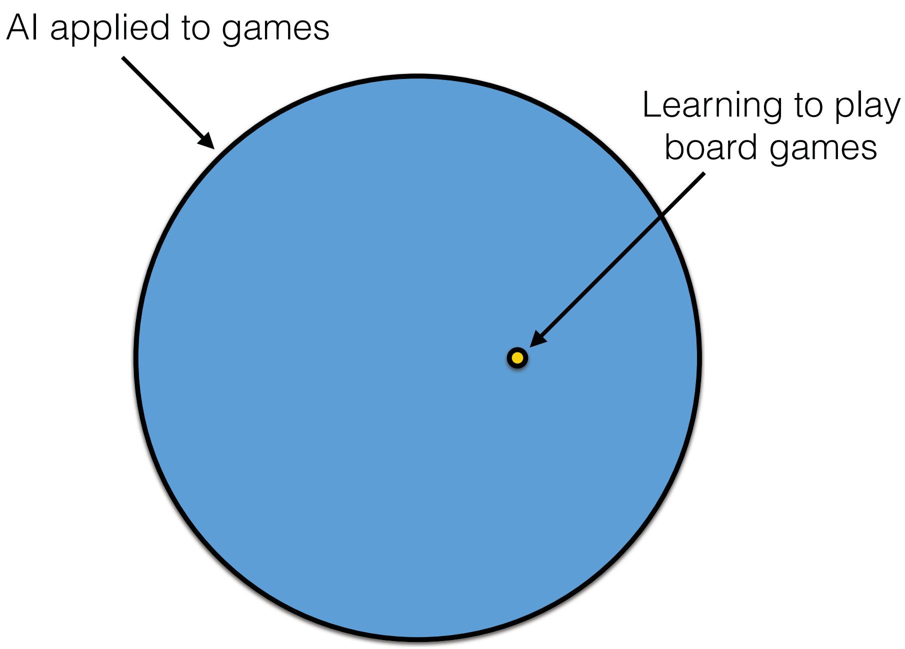
Making computers able to do things which currently only humans can do.

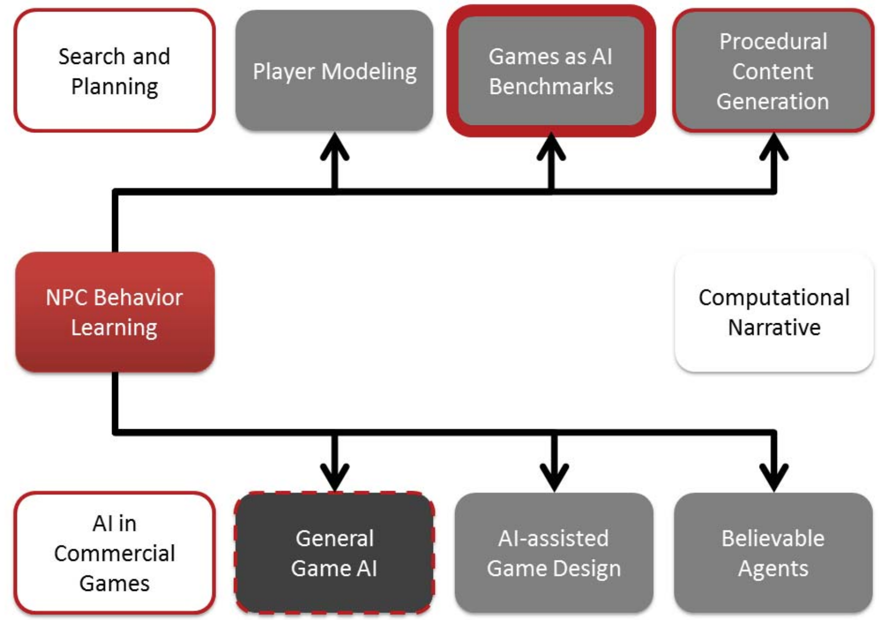
What do humans do with games?

What do humans do with games?

- Play them
- Study them
- Build content for them - levels, maps, art, characters, missions…
- Design and develop them

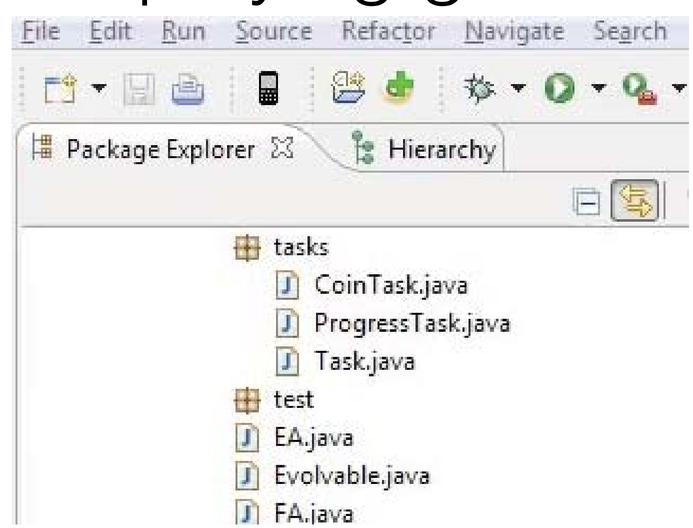
Like this, then?





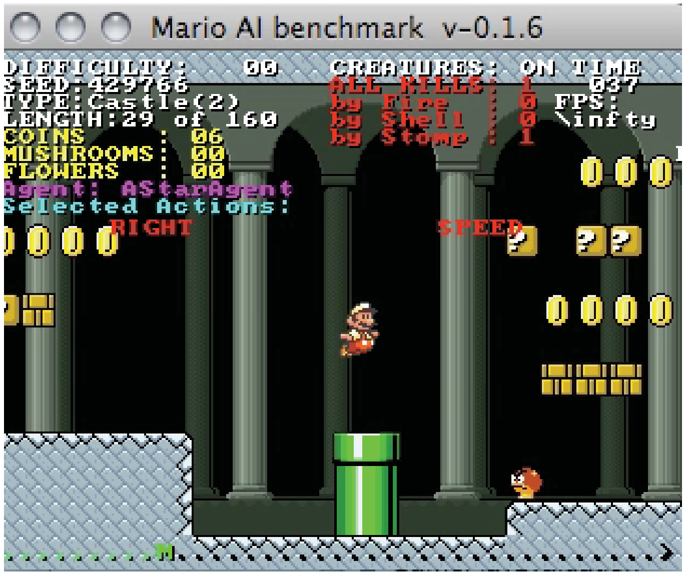
Video games as AI testbeds / benchmarks

AI playing games



Julian Togelius, Sergey Karakovskiy and Robin Baumgarten: **The 2009 Mario AI Competition**. CEC 2010.

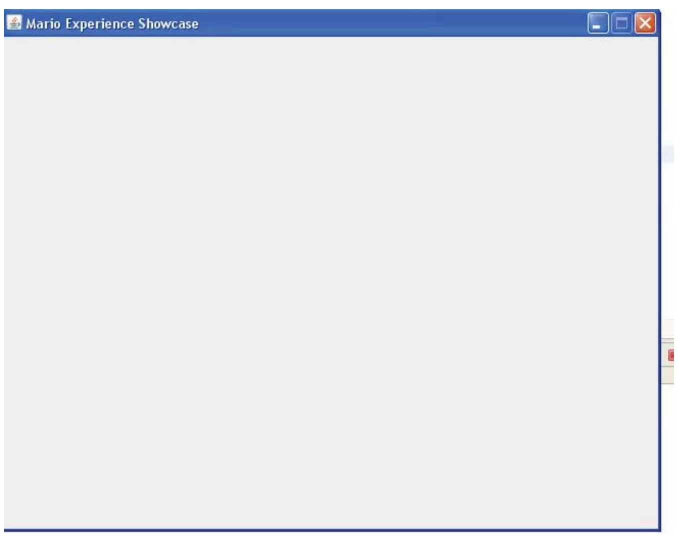
All methods have limits



REALM: Evolution to the rescue

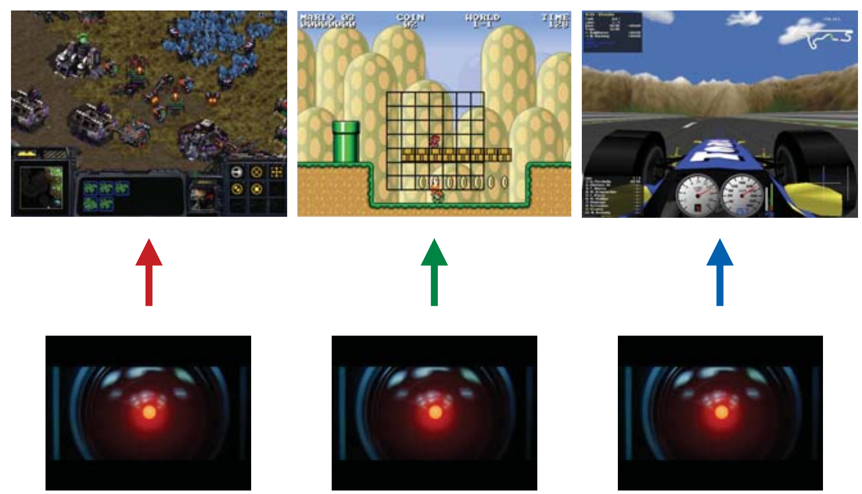
Slawomir Bojarski and Clare Bates Congdon: **REALM: A Rule-Based Evolutionary Computation Agent that Learns to Play Mario**.CIG 2010.

Human-like (?) playing



Julian Togelius, Georgios N. Yannakakis, Noor Shaker and Sergey Karakovskiy (2012): **Assessing Believability**. Believable Bots.

AI can be used for playing specific games

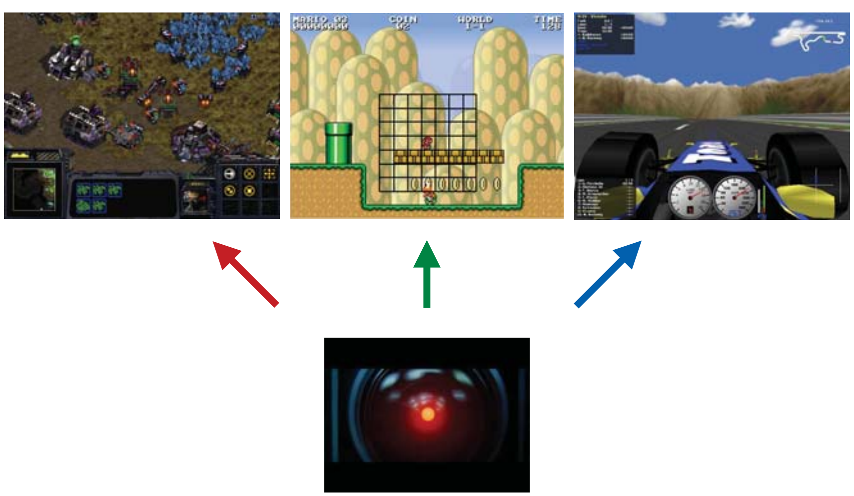


Car racing

- Driving a car fast requires fine motor control (in both senses)
- Optimizing lap times requires planning
- Overtaking requires adversarial planning

Daniele Loiacono et al: **The 2009 Simulated Car Racing Championship**. TCIAIG 2010.

Can we construct an AI that can play many games?



General intelligence

According to Legg and Hutter: sum of the performance of an agent on all possible problems, weighted by their simplicity

$$\Upsilon(\pi) := \sum_{\mu \in E} 2^{-K(\mu)} V_{\mu}^{\pi}.$$

The general video game playing competition

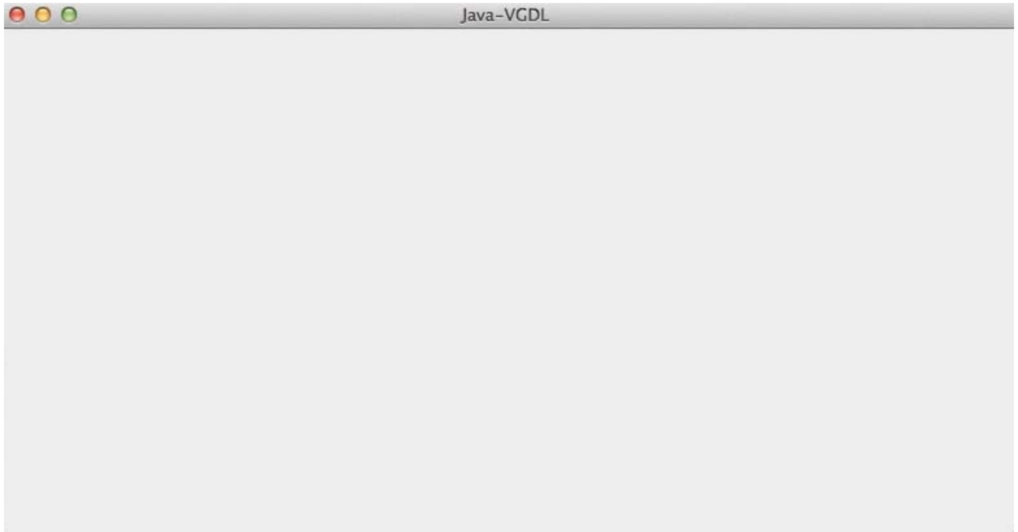
- Competitors submit controllers (AI programs written in Java)
- The game engine lets these controllers play a number of *unseen* games, and scores them
- The games are written in the ***Video Game Description Language***

The Video Game Description Language

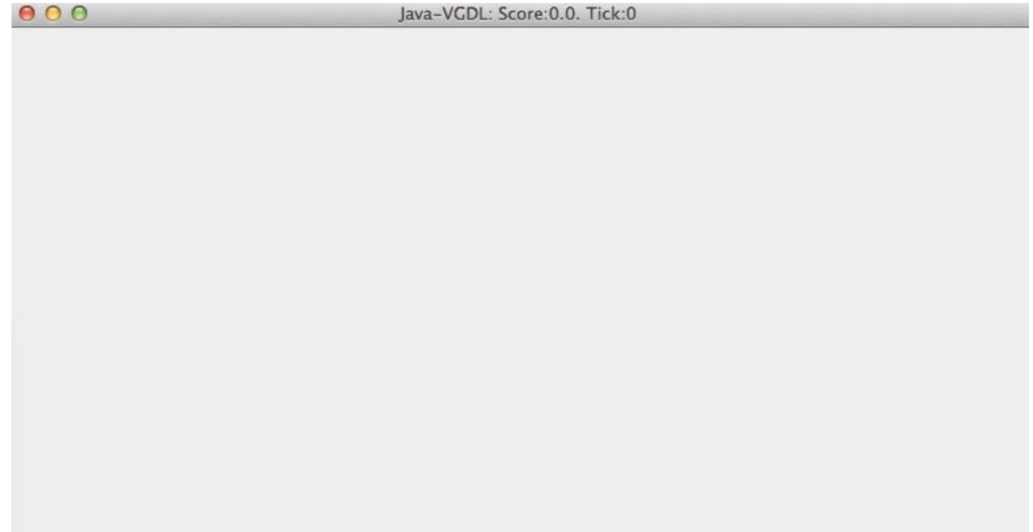
- Developed in order to be able to represent most games from the Atari 2600 era (and many from the C64 era)
- Assumes 2D movement and graphical logic
- Compact and human-readable
- Game engines in Java and Python

```
BasicGame
    SpriteSet
        sword > Flicker color=LIGHTGRAY limit=1 singleton=True img=sword.png
        dirt > Immovable color=BROWN img=dirt.png
        exitdoor > 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   stype=sword img=avatar.png
            enemy > RandomNPC
                crab > color=RED img=camel.png
                butterfly > color=PINK img=butterfly.png
    LevelMapping
        . > dirt
        E > exitdoor
        o > 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 stype=diamond scoreChange=1
        exitdoor avatar > killIfOtherHasMore resource=diamond limit=9

    TerminationSet
        SpriteCounter stype=avatar limit=0 win=False
        SpriteCounter stype=exitdoor limit=0 win=True
```

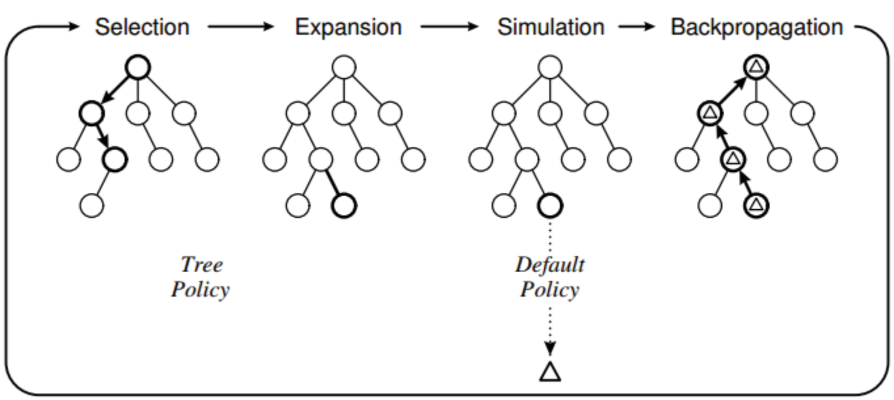


Human player in Boulder Dash



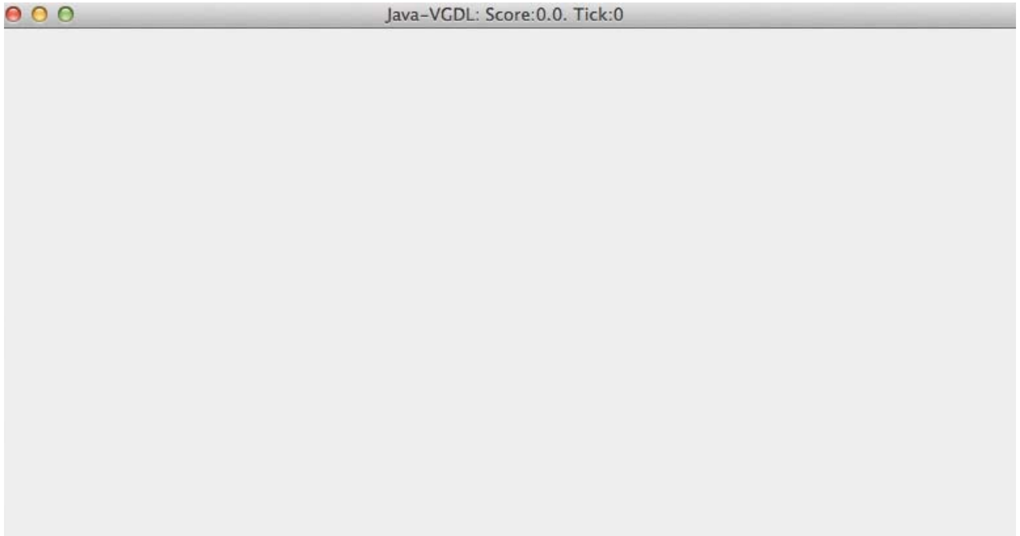
Random controller on Boulder Dash

Monte Carlo Tree Search

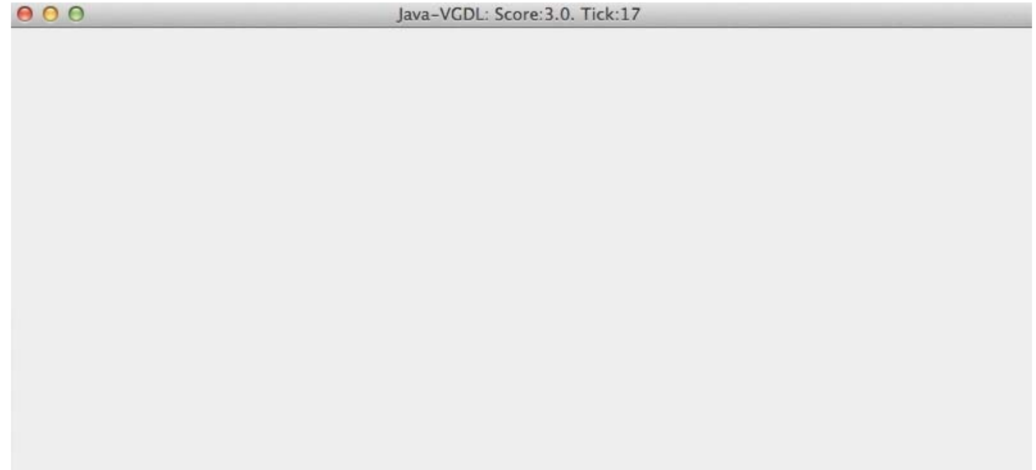


Rolling horizon evolution

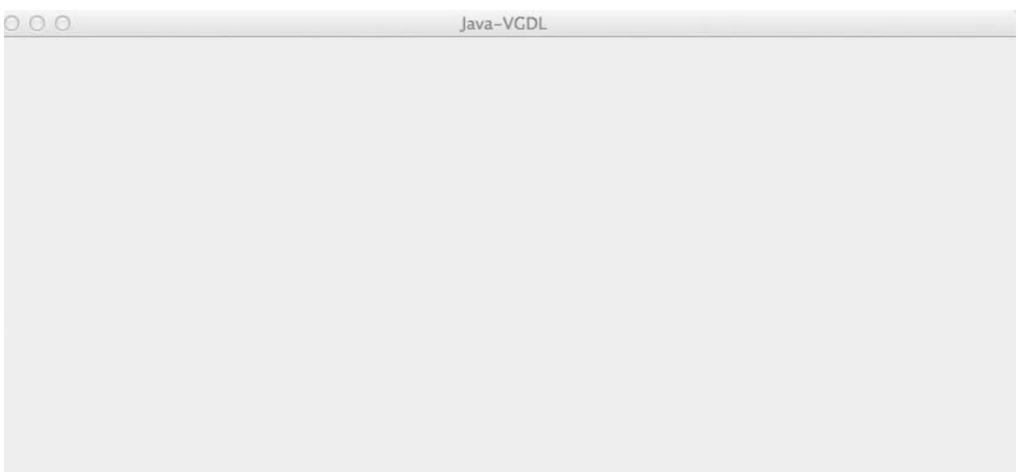
- Instead of evolving a controller and then testing it…
- …evolve an *action sequence* every action!
- Fitness: rollouts or evaluation function to estimate the value of the state this sequence brings you to
- Perez, Samothrakis, Lucas



MCTS controller on Boulder Dash



Random controller on “Aliens” (Space Invaders)



MCTS controller on “Aliens” (Space Invaders)

Rank	Username	G-1	G-2	G-3	G-4	G-5	G-6	G-7	G-8	G-9	G-10	Total Points	Victories
1	adrienctx	**25**	0	**25**	0	**25**	10	15	**25**	**25**	8	158	256/500
2	JinJerry	18	6	18	**25**	15	6	18	18	12	12	148	216/500
3	SampleMCTS†	10	18	6	4	18	**25**	6	12	0	0	99	158/500
4	Shmokin	6	**25**	0	12	10	8	0	10	6	0	77	127/500
5	Normal_MCTS	12	0	4	15	4	15	10	4	4	0	68	102/500
6	culim	2	12	8	1	8	4	8	6	10	2	61	124/500
7	MMbot	15	0	1	2	12	12	2	15	0	0	59	130/500
8	TESTGAG	0	8	15	0	0	1	1	0	2	**25**	52	68/500
9	Yraid	0	6	10	0	0	0	12	0	15	6	49	93/500
10	T2Thompson	0	0	0	10	0	0	0	1	18	18	47	87/500
11	MnMCTS	8	8	0	0	1	18	4	8	0	0	47	109/500
12	SampleGA†	4	10	12	0	0	2	0	0	8	4	40	76/500
13	IdealStandard	1	6	0	0	6	0	**25**	0	0	1	39	134/500
14	Random†	0	15	0	18	2	0	0	0	0	0	35	78/500
15	Tichau	0	6	0	8	0	0	0	0	1	15	30	55/500
16	OneStepLookAhead†	0	6	0	0	0	0	0	0	1	10	17	51/500
17	levis501	0	0	2	6	0	0	0	2	1	0	11	50/500
18	LCU_14	0	4	0	0	0	0	0	0	0	0	4	54/500

TABLE IV

FINAL RESULTS OF THE GVGAI COMPETITION. †DENOTES A SAMPLE CONTROLLER.

Evolution instead of tree search

- Evolution can be used not only to search for game-playing agents, but also to play games
- *Online evolution*: search for sets or sequences of moves at each turn
  - Fitness function: state evaluation
- Especially useful for large branching factors

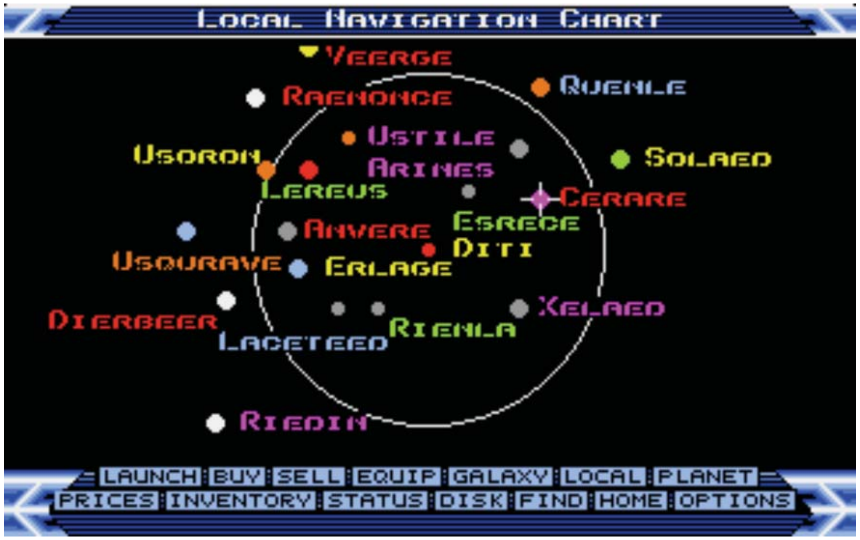
Niels Justesen, Tobias Mahlmann and Julian Togelius (2016): Online Evolution for Multi-Action Adversarial Games. Proceedings of EvoApplications.

Modern game development

Procedural content generation in games

Elite

Elite



Elite

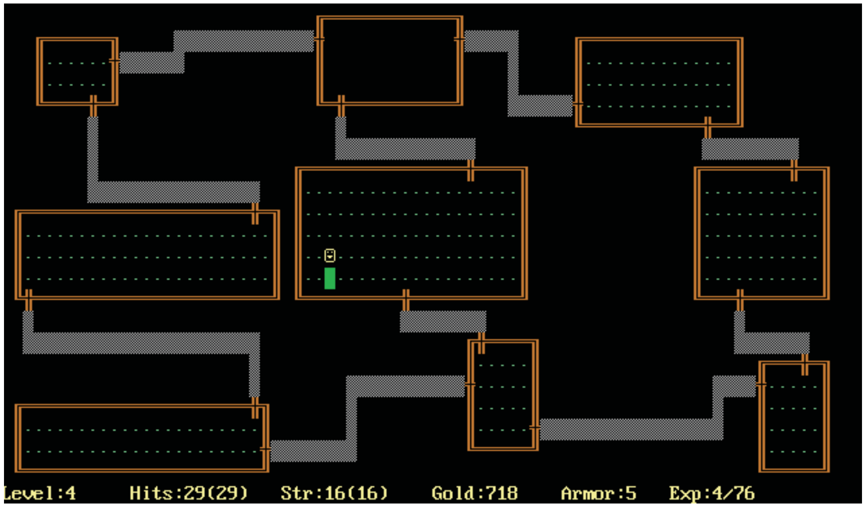
Elite

Elite

Elite

Fits in memory on a Commodore 64!

Rogue



Diablo III

Spelunky

Civilization IV

Why PCG?

- Save development time and effort (money)
- Unleash non-human creativity
- Create endless games
- Create player-adaptive games
- Study game design by formalizing it

What are the challenges?

- *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

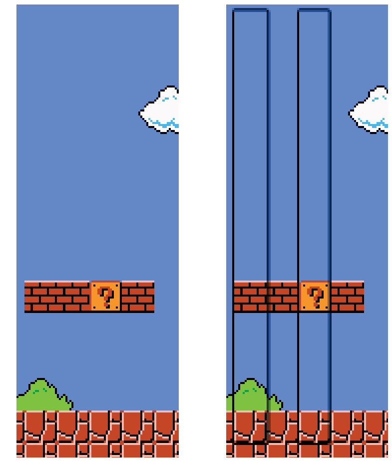
- Use evolution (or similar algorithms) to search for good content
- Main issues:
  - How to represent the content so that the content space can be searched effectively
  - How to evaluate the quality of content

J. Togelius, G. Yannakakis, K. O. Stanley and C. Browne
Search-based Procedural Content Generation: a Taxonomy and Survey
IEEE TCIAIG 2011

Let's evolve levels for Super Mario Bros!

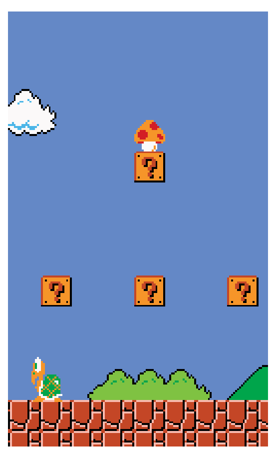
Representation

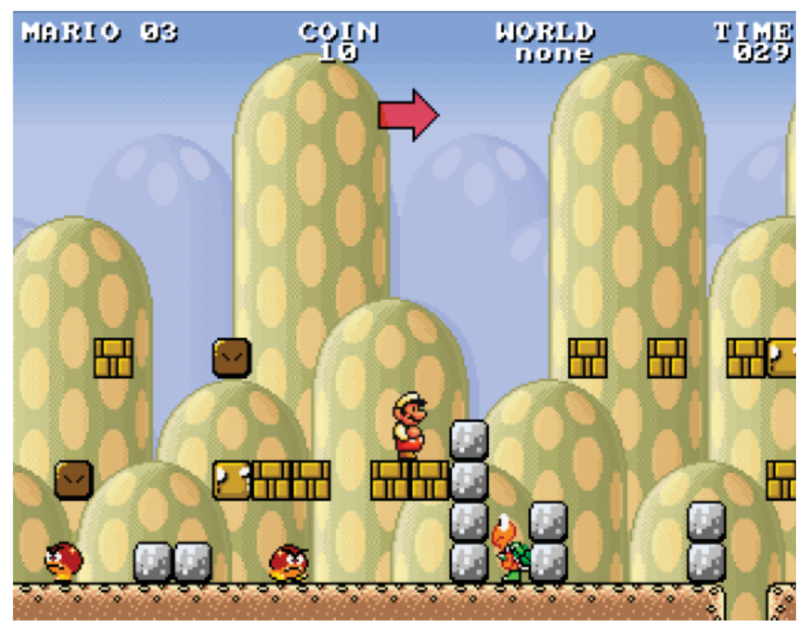
- A number of "vertical slices" are identified from the original SMB levels
- Levels are represented as strings, where each character correspond to a pattern



Evaluation

- 25 patterns are identified in the original SMB levels
- e.g. enemy hordes, pipe valleys, 3-paths…
- The evaluation function counts the number of patterns found in the level





Steve Dahlskog and Julian Togelius: Patterns as Objectives for Level Generation. PCG Workshop 2013

How would we generate rules for completely new games?

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!

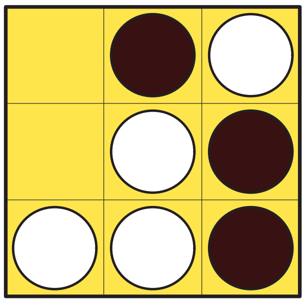
The Ludi Game Description Language

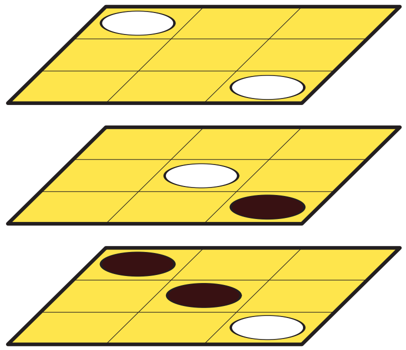
- In practice limited to board games
- *Ludeme:* Fundamental units of independently transferable game information ("game meme")
  - (tiling square)
  - (size 3 3)

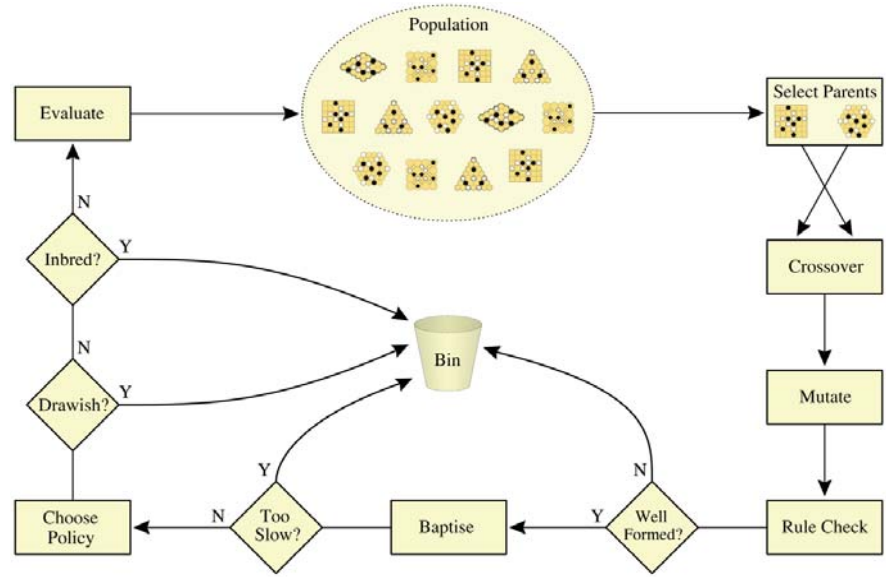
Tic-Tac-Toe

```
(game Tic-Tac-Toe
  (players White Black)
  (board
    (tiling square i-nbors)
    (size 3 3)
  )
  (end (All win (in-a-row 3)))
)
```

(size 3 3) vs (size 3 3 3)

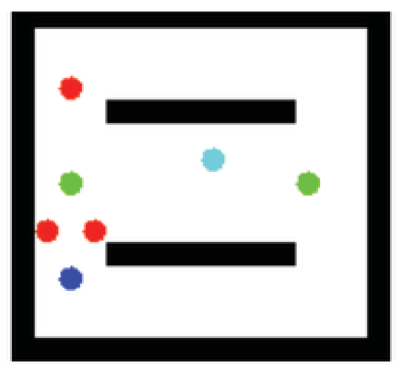






Cameron Browne: Evolutionary Game Design, 2008.

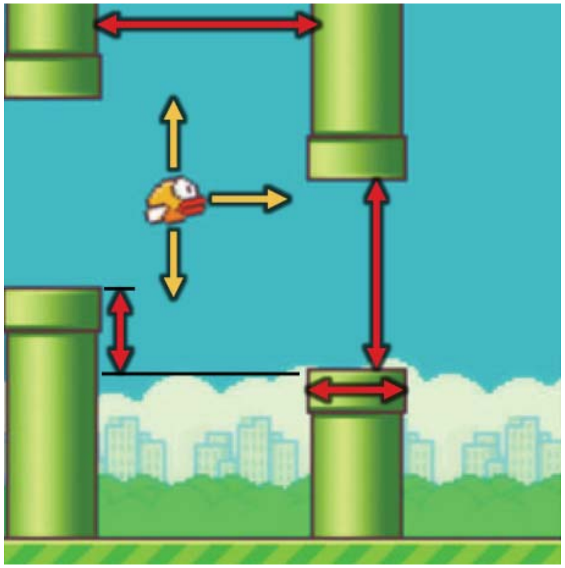
Automatic Game Design



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

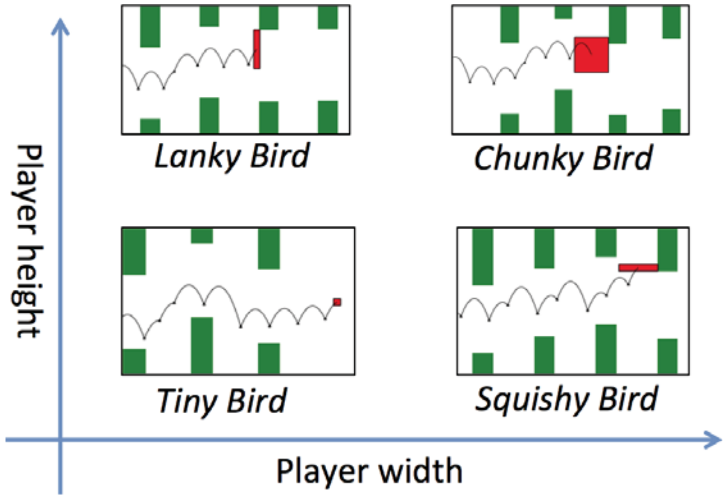
(Togelius and Schmidhuber 2008)

Discovering interesting game variants

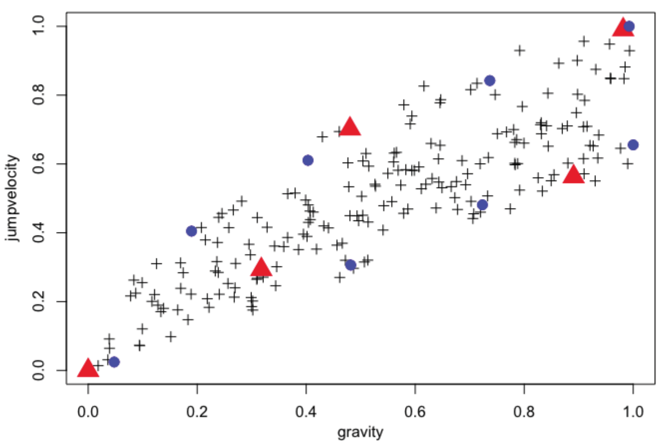


Aaron Isaksen, Dan Gopstein, Julian Togelius and Andy Nealen: **Discovering Interesting Game Variants**. ICCC 2015.

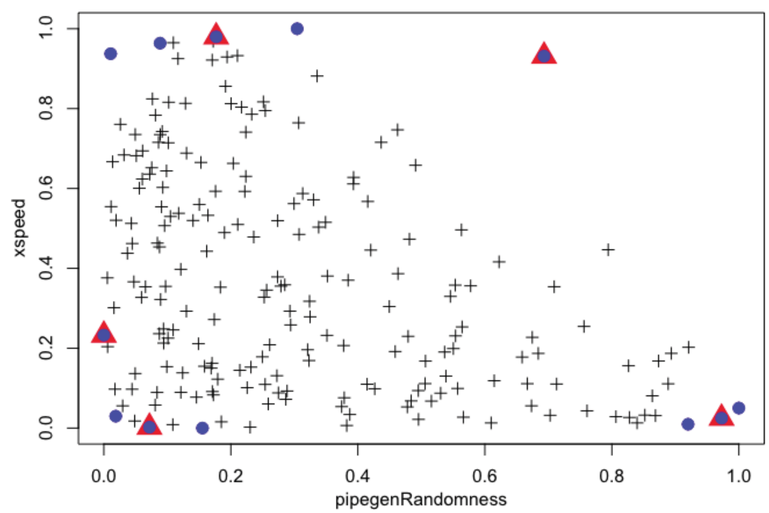
Varying two dimensions

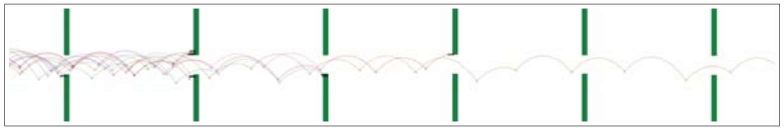


Evolving far-apart games

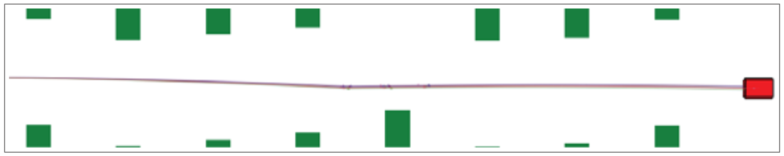


Evolving far-apart games

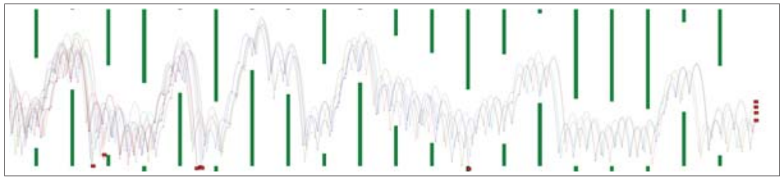




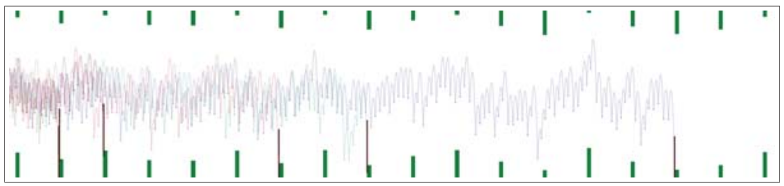
Needle Gnat



Lazy Blimp



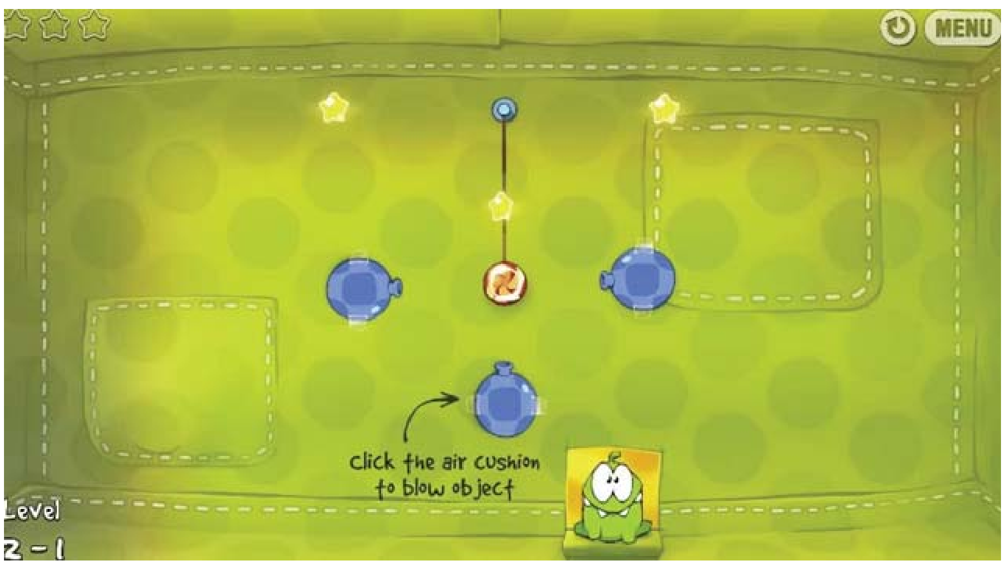
Droppy Brick



Pogo Pigeon

Collaborating with the AI

- The AI can design levels, but so can you
- Maybe you have different strengths and can work together?

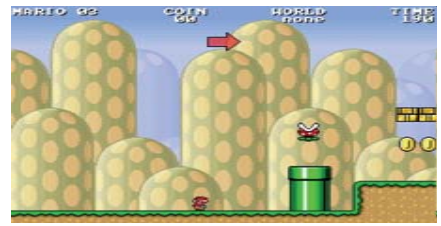


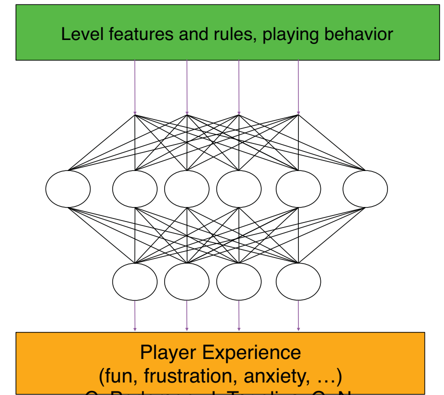
Adaptive games

- Can we use PCG to create games that adapt to the player?
- Adapt to what? Skill, preferences, strategy, playing style…

Player level preferences in Super Mario Bros

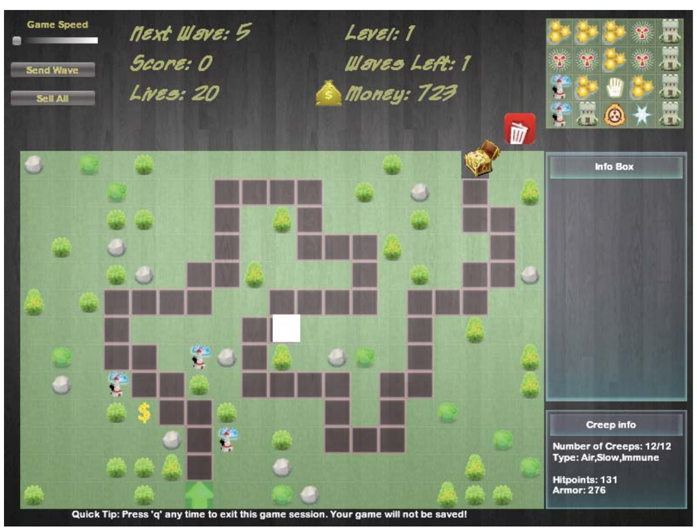
- Neuroevolutionary preference learning
- Player experience model 73-92%





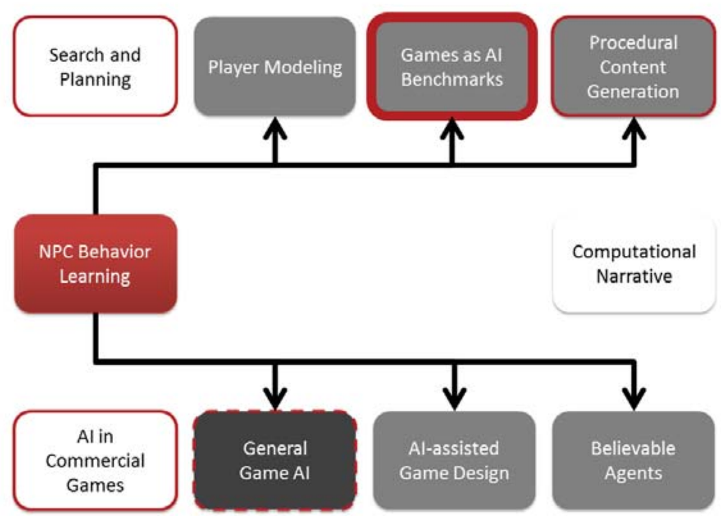
C. Pedersen, J. Togelius, G. N. Yannakakis., **Modeling Player Experience for Content Creation** *IEEE TCIAG,* 2010

Infinite tower defense



Avery, Togelius, Alistar, van Leeuwen: Computational Intelligence and Tower Defence Games. CEC 2011

It all comes together



Game evaluation

- Game rules need to evaluated procedurally, by playing them
- Learnability or heuristics acquisition potential is one possibility
- Current work in progress: active learning, i.e. algorithms that choose their own training data

Curious agents

- Curiosity can be formalized as constantly seeking environments where one can learn to predict as quickly as possible
- A curious agent selects training instances to maximize its learning
- Goes for *natural* as well as *artificial* curiosity (and creativity)

Curious games playing their players

- If the game is choosing training instances so as to maximize its knowledge about its player, it is curious about the player
- Are levels that are interesting for the game also interesting for the player?
- Design idea: curious game master?
- Can we have AI if the AI can't have fun?

What can computational intelligence do for games?

- Generate complete games, which requires…
- generating game content, which requires…
- evaluating content and game quality, which requires…
- modeling player preference and style, and…
- learning to play arbitrary games

What can games do for computational intelligence?

- Provide superb testbeds, that are varied and human-relevant
- Show us how we think
- Teach us how to create AI that has fun

Further reading

- Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley and Cameron Browne (2011): Search-based Procedural Content Generation: A Taxonomy and Survey. IEEE Transactions on Computational Intelligence and AI in Games (TCIAIG), volume 3 issue 3, 172-186.
- Georgios N. Yannakakis and Julian Togelius (2014): A Panorama of Artificial and Computational Intelligence in Games. IEEE Transactions on Computational Intelligence and AI in Games (TCIAIG).
- Diego Perez, Spyridon Samothrakis, Julian Togelius, Tom Schaul, Simon Lucas, Adrien Couetoux, Jerry Lee, Chong-U Lim and Tommy Thompson (2015): The 2014 General Game Playing Competition. IEEE Transactions on Computational Intelligence and AI in Games (TCIAIG).
- Sebastian Risi and Julian Togelius (2016): Neuroevolution in Games. Transactions on Computational Intelligence and AI in Games (TCIAIG).