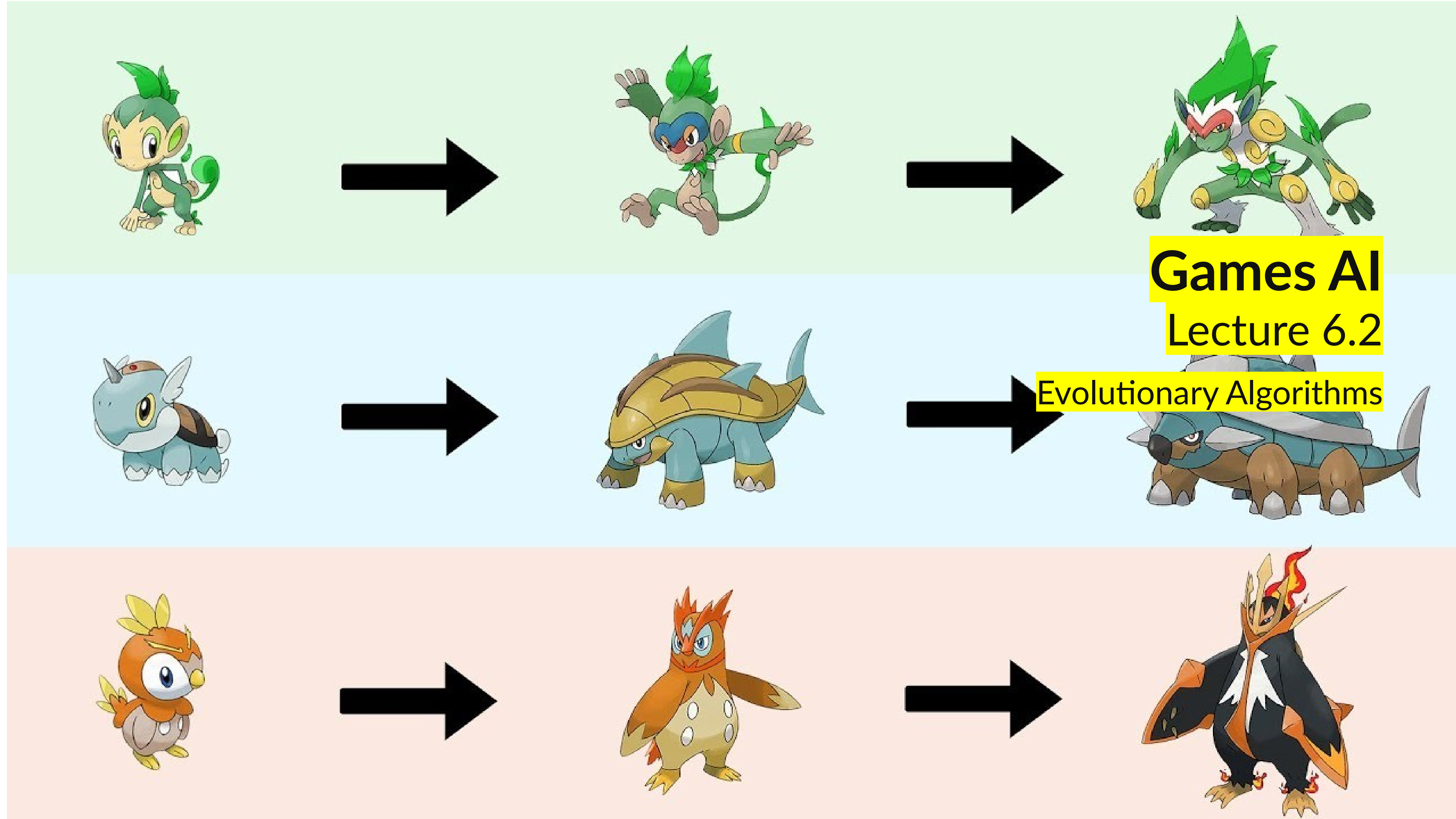


Games AI

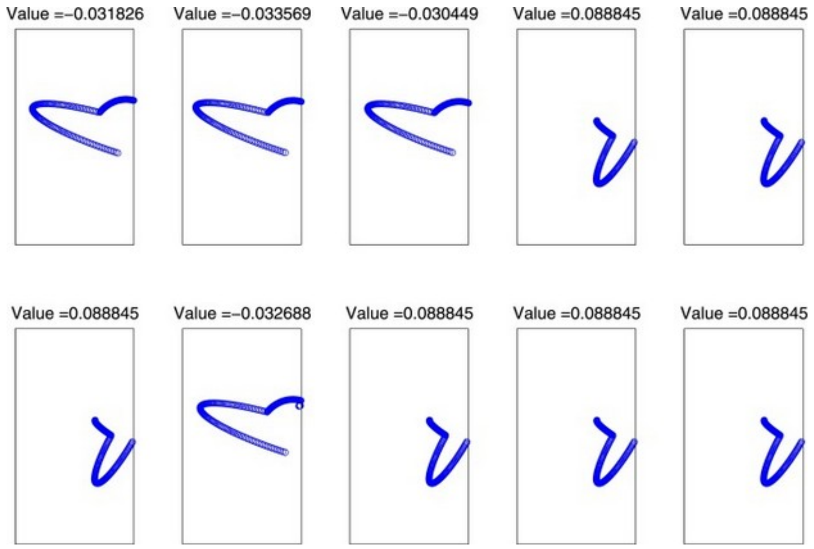
Lecture 6.2

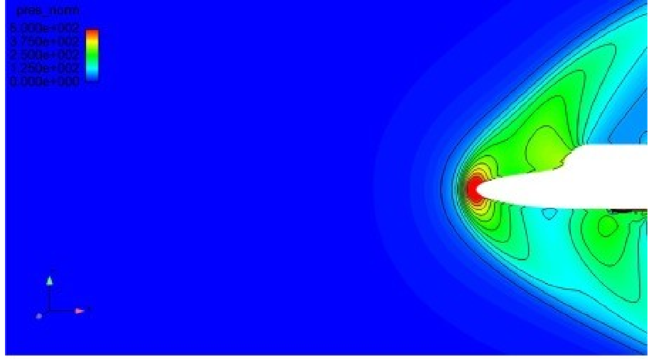
Evolutionary Algorithms

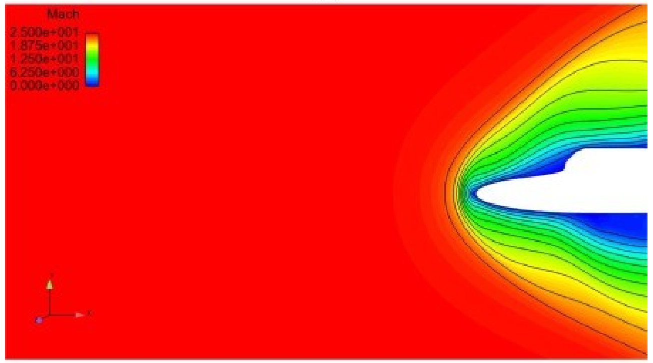
- Last lecture we described PCG as a search problem
  1) You have a **generative space** defined by your algorithm
  2) Sometimes this space includes both good and bad content.
  3) If you can score how "good" content is with an **evaluation function** then you can **search** for high-scoring content.
     - This is an **optimisation** problem

- Why evolution?
  - Engineers often copy solutions in nature
  - What is the best problem solver in nature?
    - **The evolutionary process**

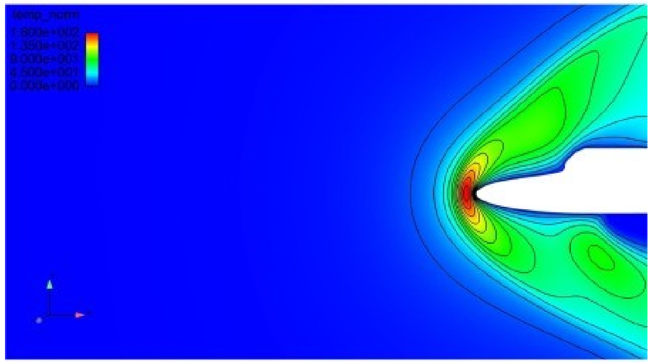
- So you can… find the optimal design for hypersonic re-entry vehicle nosecone




(a)

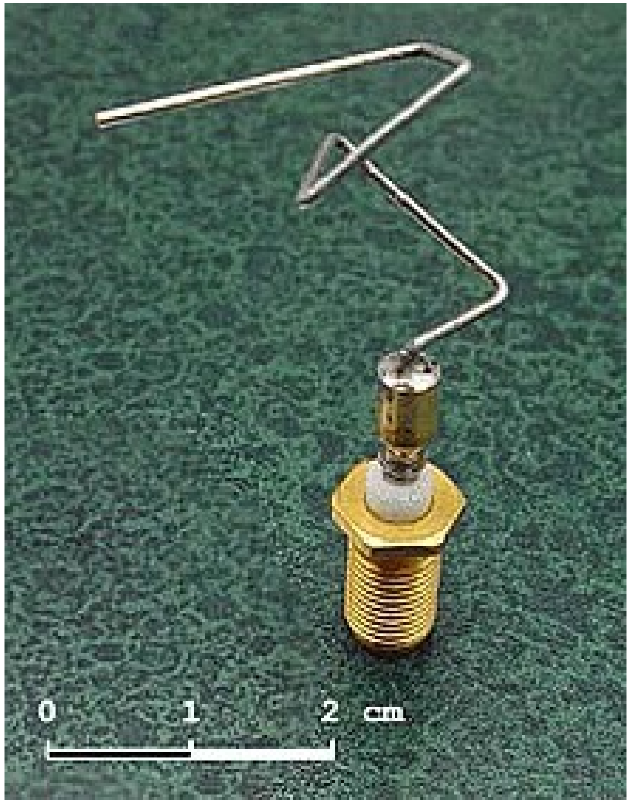


(b)


(c)

Evans, B., Walton, S., Evans, B., Walton, S. 2017 Aerodynamic optimisation of a hypersonic reentry vehicle based on solution of the Boltzmann–BGK equation and evolutionary optimisation Applied Mathematical Modelling 52 215 240 doi: 10.1016/j.apm.2017.07.024

- Or… evolve a more efficient antenna for an unusual purpose

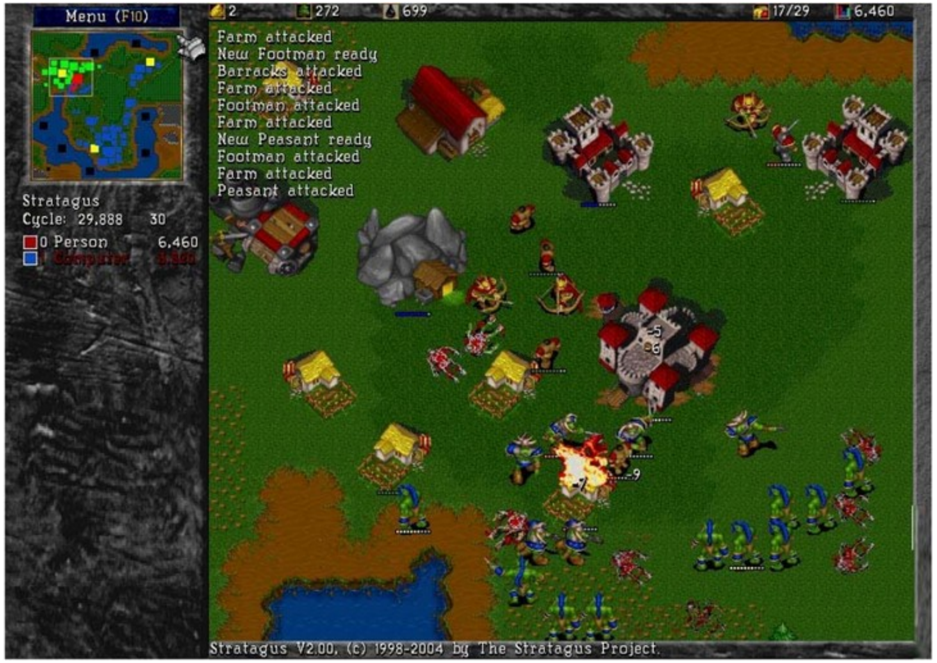


- Or… tune parameters for **rules and FSMs**

```
if (enemy.distance <= 5)
    attackWithKnife()
else if (enemy.distance > 5 AND enemy.distance <=30)
    attackWithSubmachineGun()
else attackWithRifle()
```

- Or… tune parameters that **define AI personality** (e.g. unit preference, scientific advance preference, offense vs. defense, etc.)

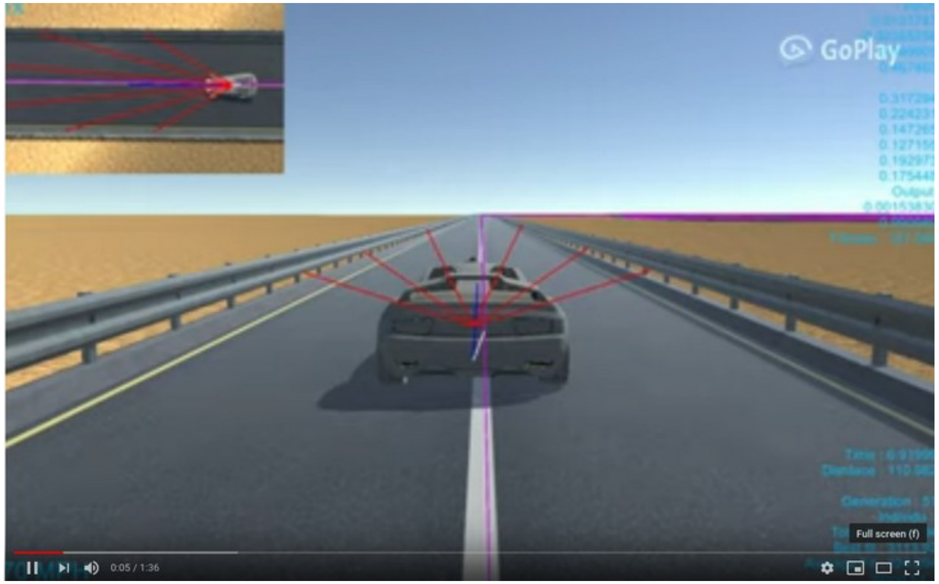
- Ponsen, Marc, and Pieter Spronck. Improving adaptive game AI with evolutionary learning. Diss. Masters Thesis, Delft University of Technology, 2004. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.109.6055&rep=rep1&type=pdf



Farm attacked
New Footman ready
Barracks attacked
Farm attacked
Footman attacked
Farm attacked
New Peasant ready
Footman attacked
Farm attacked
Peasant attacked

Stratagus V2.00, (C) 1998-2004 by The Stratagus Project.

- Or… **evolve agents** using e.g neuroevolution (evolution of neural networks)
  - https://youtu.be/_1TOKKgAock

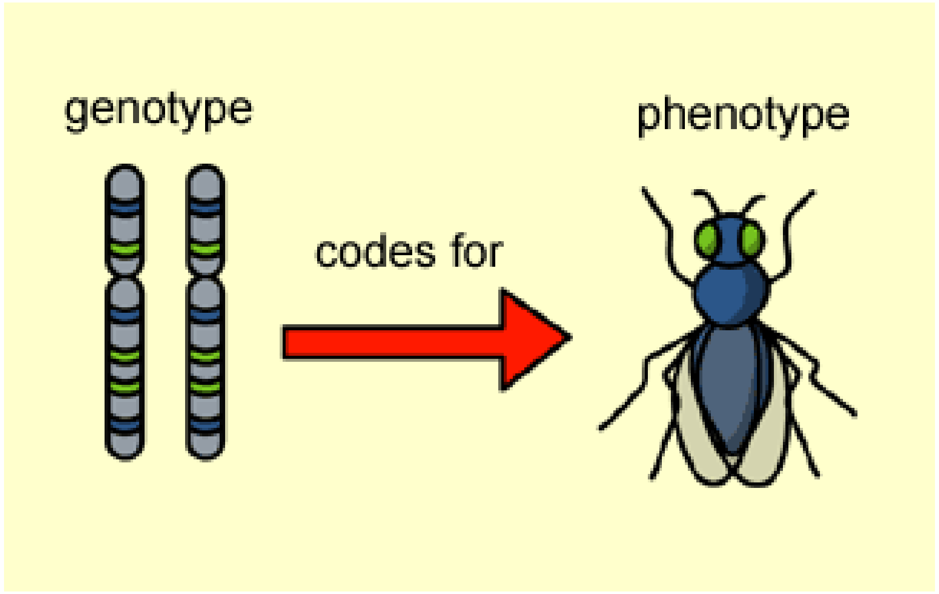


Natural evolution

- Darwinian natural selection:
  - **Competition-based selection**
  - **Phenotypic variation**
    - Behavioural and physical traits that affect an individual’s fitness

Natural evolution

- Success in survival and reproduction is determined by **phenotypical** properties
- **Phenotypic variations** are always caused by **genotypic variations**
- Phenotypes never influence genetic information
  - **No learning within individuals**



Natural evolution

- Change happens through reproduction by the mechanisms of:
  - Mutation
  - Recombination

- **Adaptive landscape**
  - Metaphor of a space where height corresponds to fitness
    - **Global Optimum =** best
    - **Local Optimum =** better than neighbours

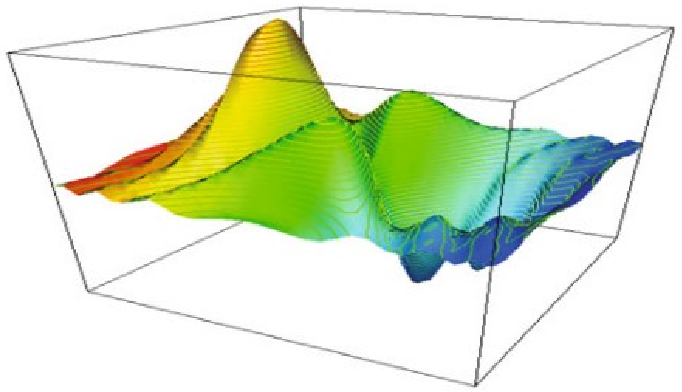
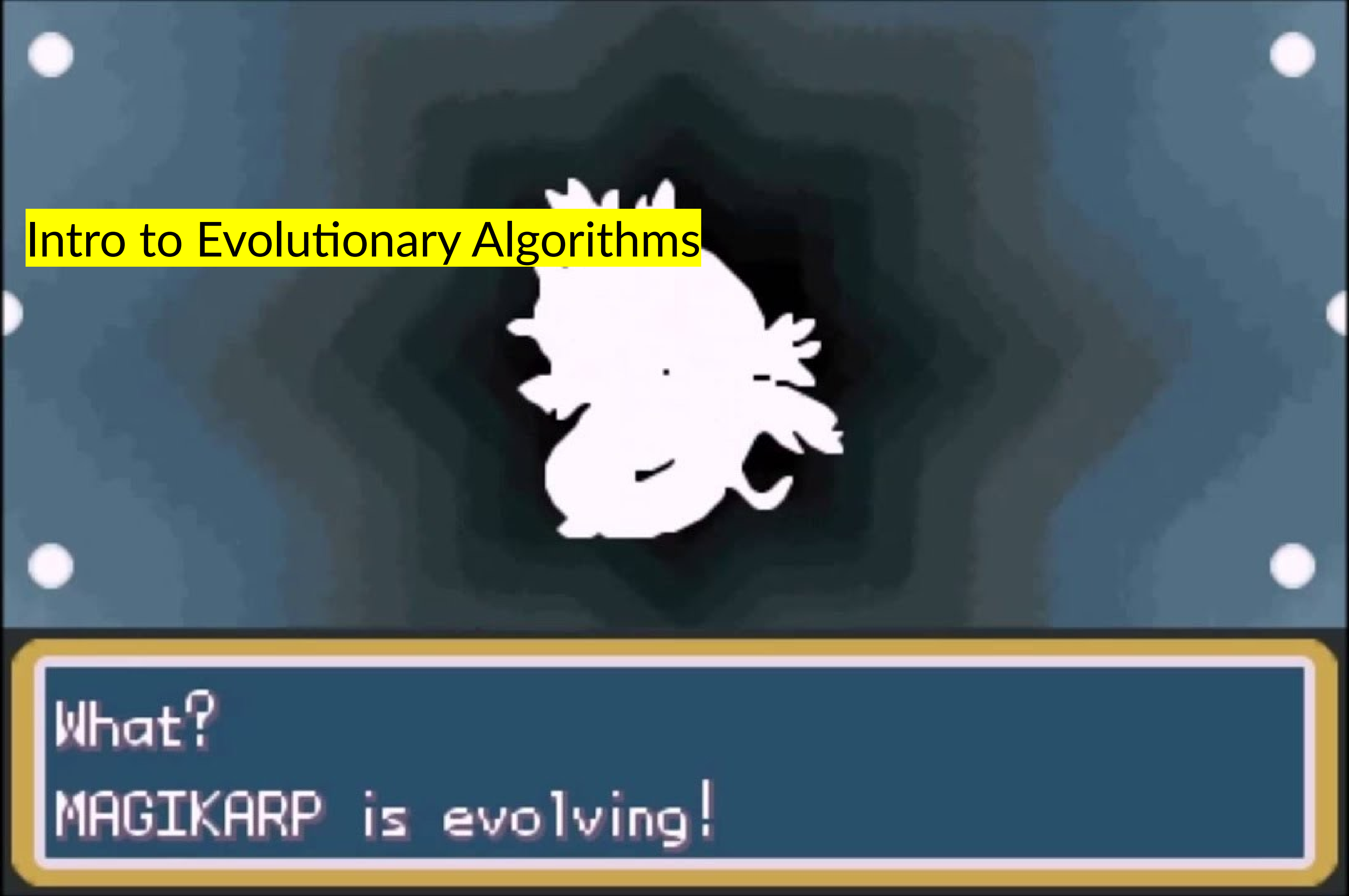


Fig. 2.2. Illustration of Wright's adaptive landscape with two traits

https://dantelore.com/posts/evolutionary-algorithm-playable-demo/

- **Each individual is a sample** of the space of all possible living things
  - Produced by forces of **variation**
  - Evaluated by forces of **selection**
    - Prove viable to live
    - Prove viable to reproduce
- In algorithmic terms this is a “**generate and test**” method.

Intro to Evolutionary Algorithms


What?
MAGIKARP is evolving!

- In an Evolutionary Algorithm, a solution is described by a **genotype**
  - This might be a string, enum[], number[]…
  - e.g. **parameter values** for a PCG algorithm
  - e.g. **weights** for a neural network
- This data (genotype) is interpreted to create the **phenotype** (e.g. level map, AI behaviour, game audio)

- Evolutionary Algorithms (EAs) are:
  - **Population based**
    - (You keep hold of a bunch of possible solutions at the same time)
  - **Stochastic**
    - (Involves randomness)
  - Most use **recombination**
    - (You mix together different solutions to try and find a better one)

- Why EAs work
  - **Variation** (recombination and mutation) create diverity and novelty
  - **Selection** acts as a force to increase the mean quality of solutions

Intro to EAs

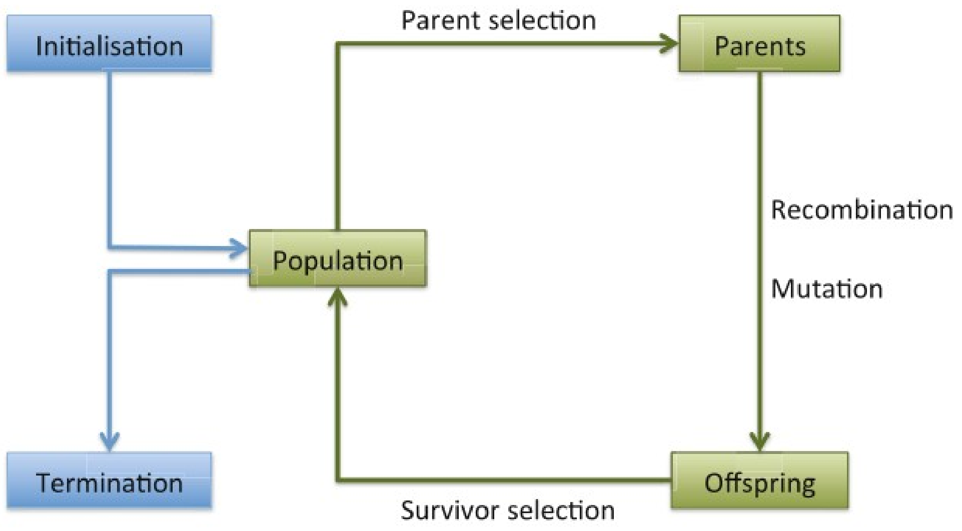
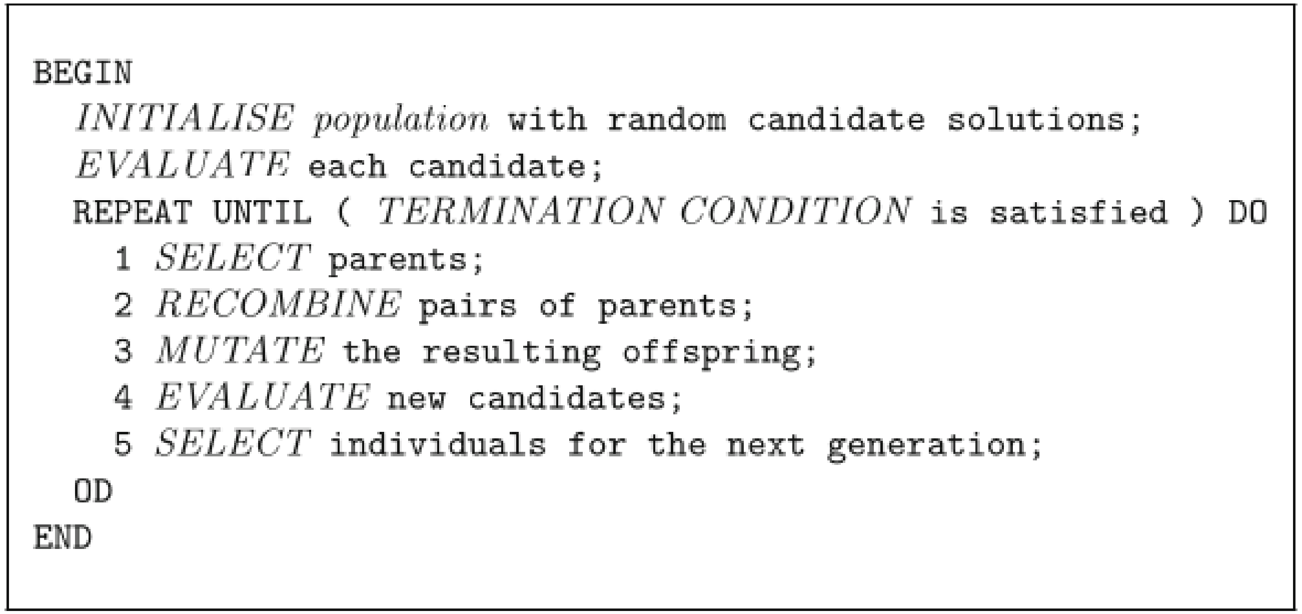


Fig. 3.2. The general scheme of an evolutionary algorithm as a flowchart

Intro to EAs



```
BEGIN
  INITIALISE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
    1 SELECT parents;
    2 RECOMBINE pairs of parents;
    3 MUTATE the resulting offspring;
    4 EVALUATE new candidates;
    5 SELECT individuals for the next generation;
  OD
END
```

Fig. 3.1. The general scheme of an evolutionary algorithm in pseudocode

- Components of EAs
  - **representation** (definition of individuals)
  - **evaluation function** (or fitness function)
  - **population**
  - **parent selection mechanism**
  - **variation operators**, recombination and mutation
  - **survivor selection mechanism** (replacement)
  - **termination condition**

- Components of EAs
  - **representation** (definition of individuals)
  - **evaluation function** (or fitness function)
  - **population**
  - **parent selection mechanism**
  - **variation operators**, recombination and mutation
  - **survivor selection mechanism** (replacement)
  - **termination condition**

- **Representation**: What is your genome?
  - **Strings** in a finite alphabet (genetic algorithms)
  - **Real-valued vectors** (evolution strategies)
  - **Finite state machines** (classical evolutionary programming)
  - **Trees** (genetic programming)
- Different approaches may suit different problems better depending on the natural way to encode the candidate solutions

- Components of EAs
  - **representation** (definition of individuals)
  - **evaluation function** (or fitness function)
  - **population**
  - **parent selection mechanism**
  - **variation operators**, recombination and mutation
  - **survivor selection mechanism** (replacement)
  - **termination condition**

- **Evaluation Function**
  - Function that **assigns quality measure to genotypes**
    - **Evaluate phenotype** for quality.
  - Represent requirements that population should adapt to meet (i.e. defines what improvement means)

- Components of EAs
  - **representation** (definition of individuals)
  - **evaluation function** (or fitness function)
  - **population**
  - **parent selection mechanism**
  - **variation operators**, recombination and mutation
  - **survivor selection mechanism** (replacement)
  - **termination condition**

- **Population**
  - holds possible solution
    - A multiset of genotypes (almost always a constant size)
  - Individuals do not change or adapt, population does
  - Selection operators **work at the population level**
    - Best individual **of a given population** is selected

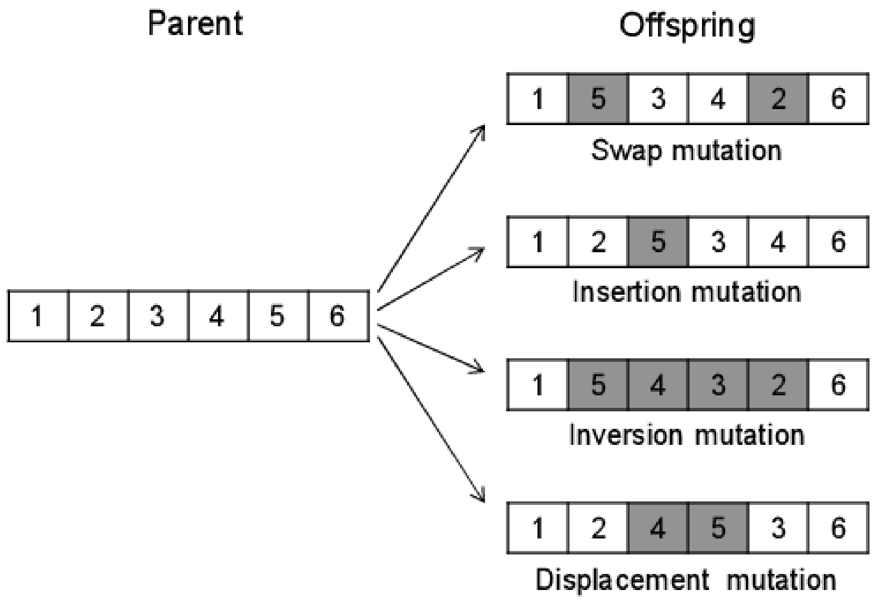
- Components of EAs
  - **representation** (definition of individuals)
  - **evaluation function** (or fitness function)
  - **population**
  - **parent selection mechanism**
  - **variation operators**, recombination and mutation
  - **survivor selection mechanism** (replacement)
  - **termination condition**

- **Parent selection** / mate selection
  - Distinguish among individuals based on quality
  - Allow better individuals to become parents of next generation
  - Probabilistic – fitter individuals more likely to be selected as parents
    - Often still give low-fitness individuals a small chance, otherwise search becomes too greedy and gets stuck in **local optima**.

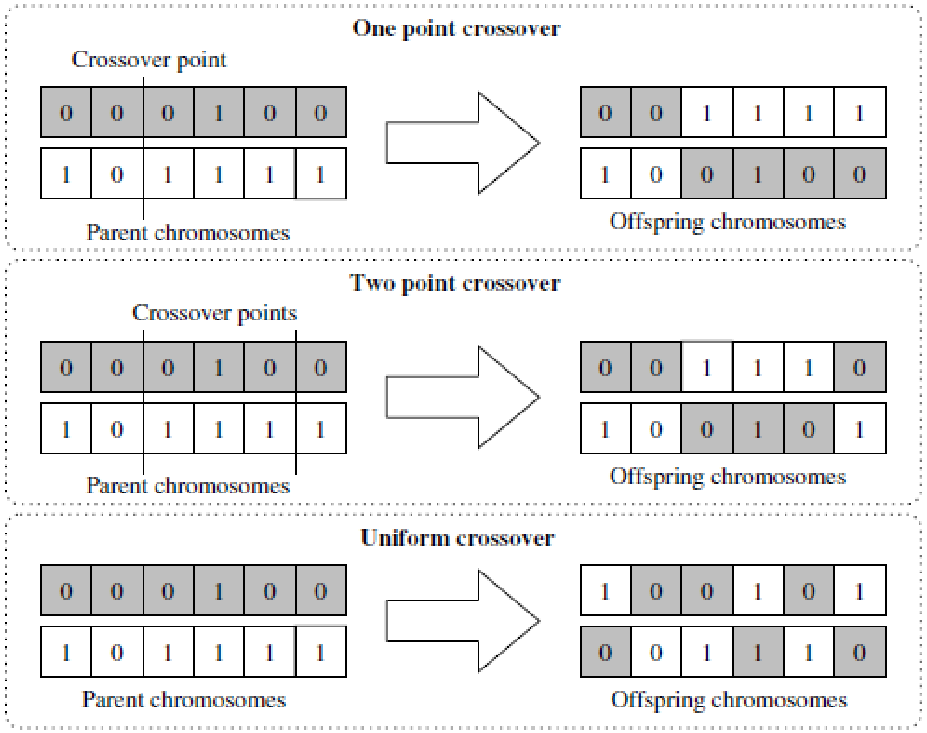
- Components of EAs
  - **representation** (definition of individuals)
  - **evaluation function** (or fitness function)
  - **population**
  - **parent selection mechanism**
  - **variation operators**, recombination and mutation
  - **survivor selection mechanism** (replacement)
  - **termination condition**

- **Variation operators**
  - Create new individuals from old ones
  - The “generate” in “generate and test”
  - **Mutation** and **Crossover** (recombination)

- **Mutation**
  - Randomly change part of the genotype
  - Stochastic
  - Unbiased
  - Provides the gene-pool with “fresh blood”



- **Crossover**
  - Merges information in two or more parents
  - Random recombinations
  - Hopefully take good qualities of each parent



- Components of EAs
  - **representation** (definition of individuals)
  - **evaluation function** (or fitness function)
  - **population**
  - **parent selection mechanism**
  - **variation operators**, recombination and mutation
  - **survivor selection mechanism** (replacement)
  - **termination condition**

- **Survivior selection** / replacement strategy
  - Decide which of older generations to keep
  - Based on **fitness values**
    - e.g. keep top % of new and old generation combined
  - Or based on **age**
    - e.g. only keep newest generation

- Components of EAs
  - **representation** (definition of individuals)
  - **evaluation function** (or fitness function)
  - **population**
  - **parent selection mechanism**
  - **variation operators**, recombination and mutation
  - **survivor selection mechanism** (replacement)
  - **termination condition**

- **Termination condition**
  - Stop when we find **optimum fitness** (if you guarantee reaching it)
  - Otherwise
    1) Limit **CPU time**
    2) Limit total number of **fitness evaluations**
    3) **Check if fitness improvement remains low** for a given period of time (i.e. for a number of generations or fitness evaluations)
    4) **Check if the population diversity drops** below a given threshold

How do EAs behaive?

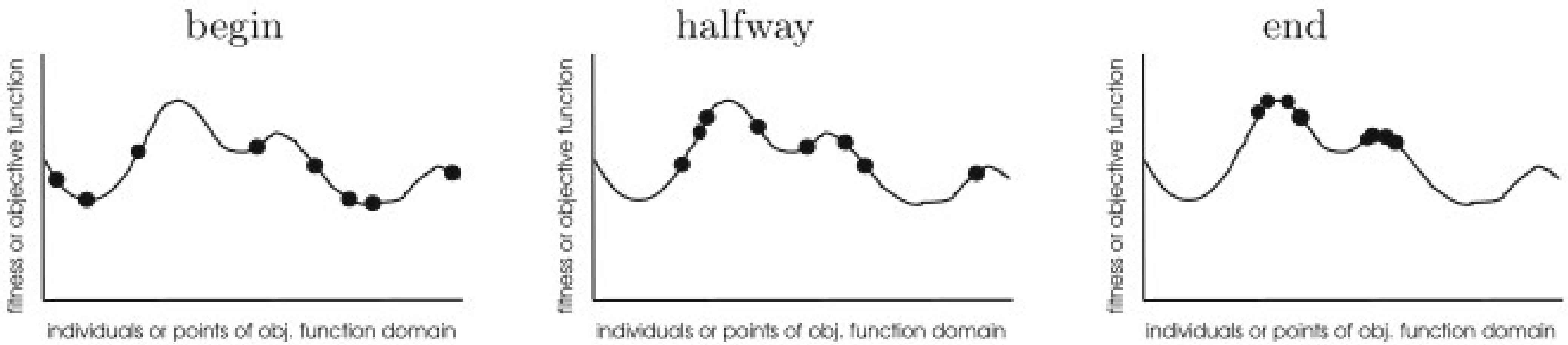

begin
halfway
end
fitness or objective function
individuals or points of obj. function domain

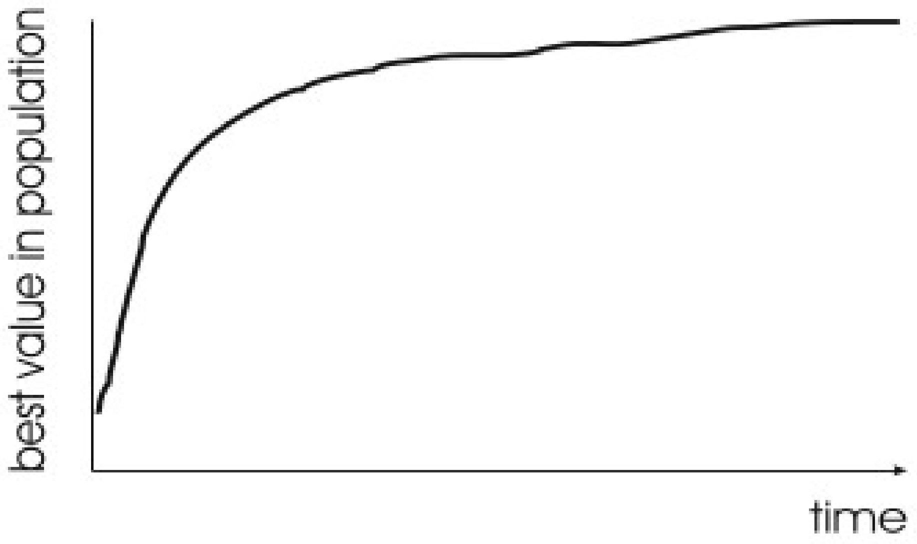
Fig. 3.4. Typical progress of an EA illustrated in terms of population distribution. For each point x in the search space y shows the corresponding fitness value.

EA behaviour

- **Exploration**: generation of individuals in untested space
- **Exploitation**: concentrating in vicinity of known good solutions
- **Premature convergence**: losing population diversity too quickly and getting stuck in a local optimum.

EA behaviour

- EAs are **anytime**
  - You can interrupt them at… any time
  - Get the best solution so far, even if suboptimal



- **Improvement is often rapid at start**
- You can initialise your population with **heuristics**, but it might not be worth the effort

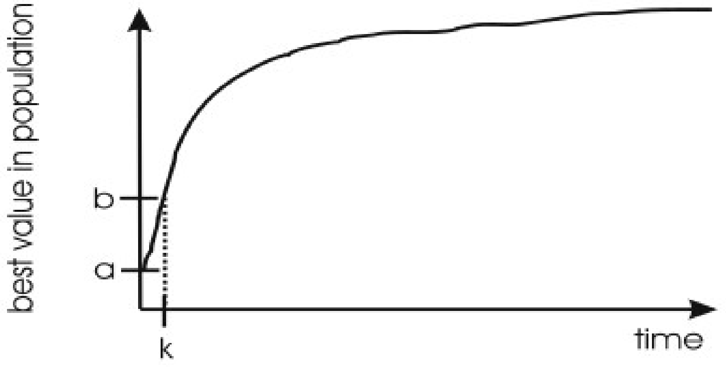


Fig. 3.6. Illustration of why heuristic initialisation might not be worth additional effort. Level a shows the best fitness in a randomly initialised population; level b belongs to heuristic initialisation

- **Improvement slows down**, so long runs may be not worth i t

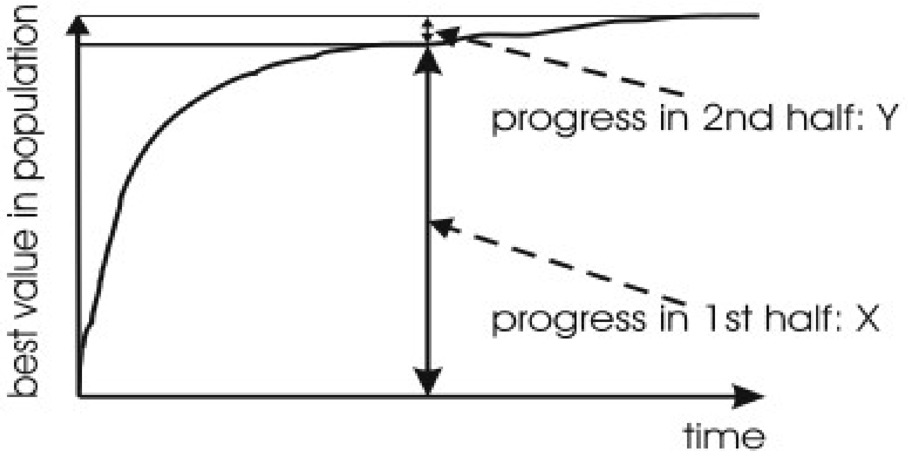
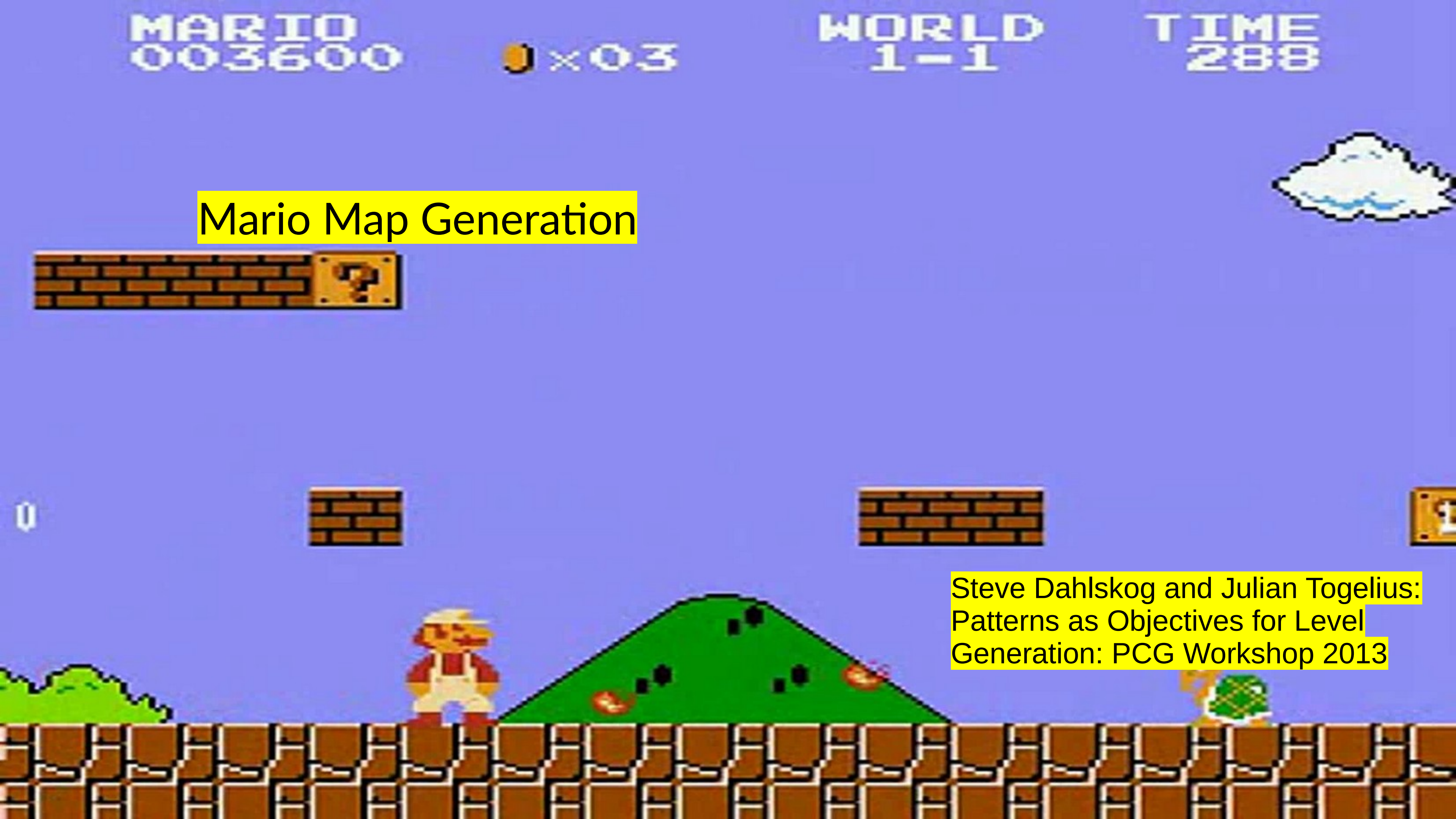


Fig. 3.7. Why long runs might not be worth performing. X shows the fitness increase in the first half of the run, while Y belongs to the second half

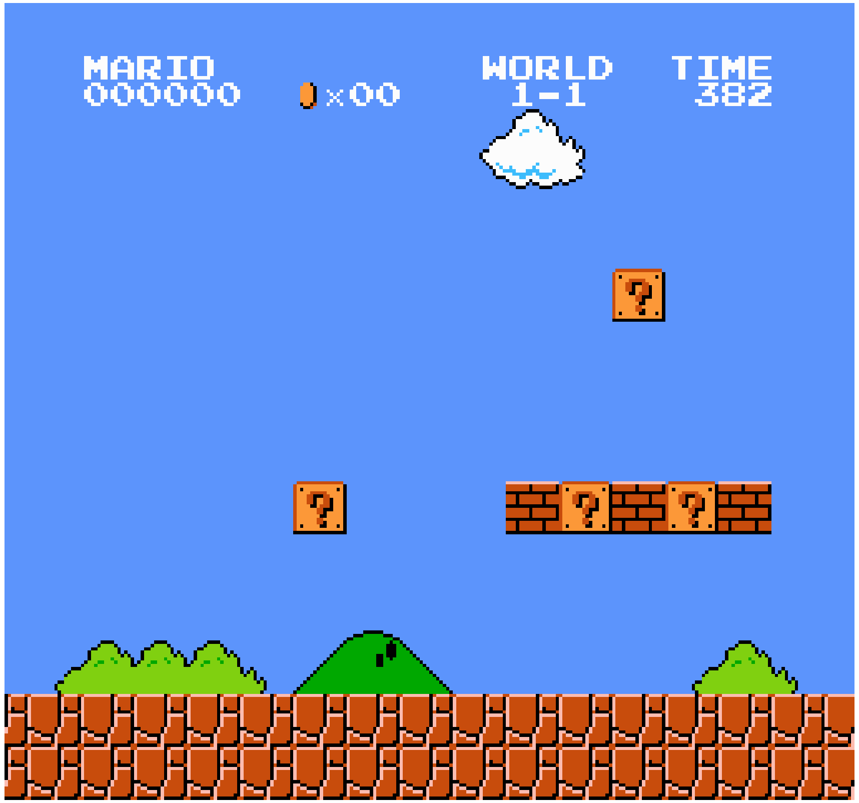
- Designing and tuning EAs is an art:
  - Population size
  - Number of generations
  - Fitness function
  - Representation
  - Mutation rate
  - Crossover operations
  - Selection procedure
  - Number of solutions to keep

Mario Map Generation



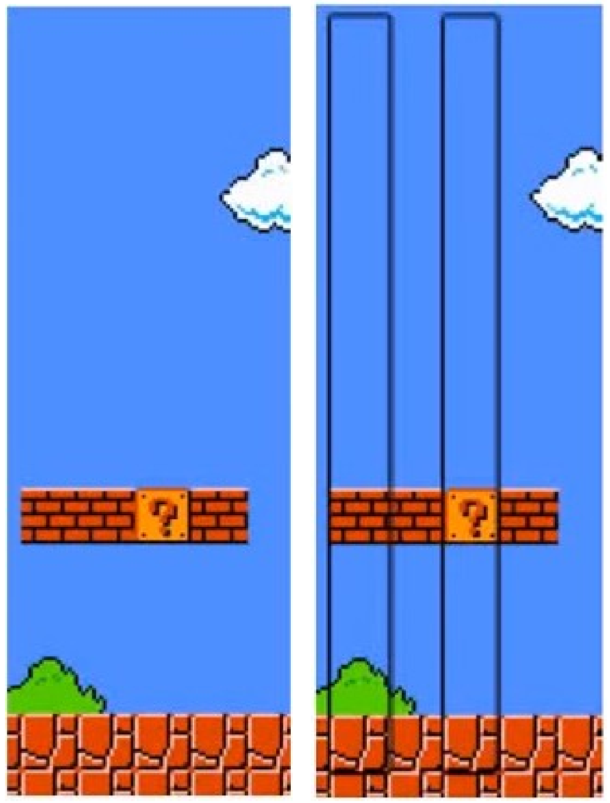
Steve Dahlskog and Julian Togelius: Patterns as Objectives for Level Generation: PCG Workshop 2013

- Representation attempt 1
  - 2D grid of integers storing value for each grid cell
    - Direct mapping to phenotype
    - But “good” space is tiny proportion of search space



- Representation attempt 2
  - A range of integers, each representing the height of an obstacle
    - 0 = hole in the ground
    - 10 = maximum height of obstacle
  - But most obstacles in SMB are close to the ground.

- Representation attempt 3
  - Represent mario level as string of symbols, each symbol corresponds to a column of blocks that existed in the original SMB



Mario map generation

- Fitness function
  - Look for level design patterns in the output that match patterns in the original SMB

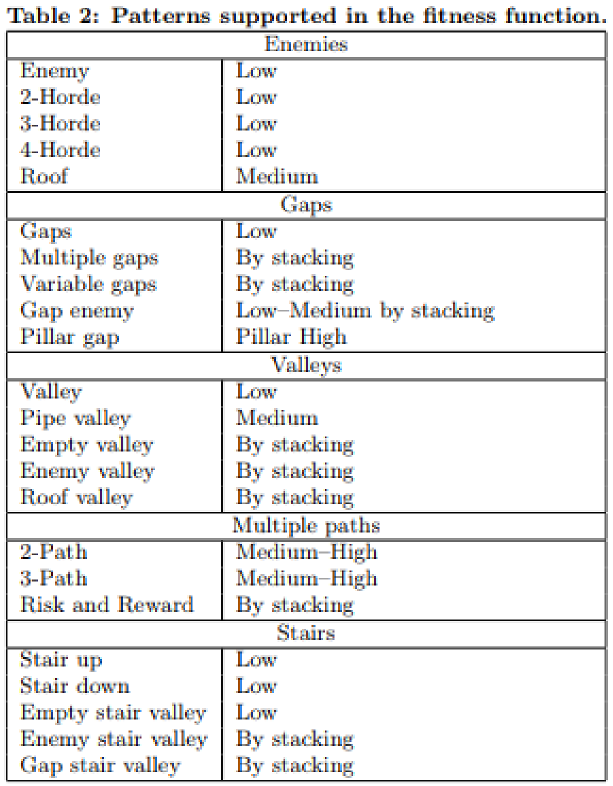
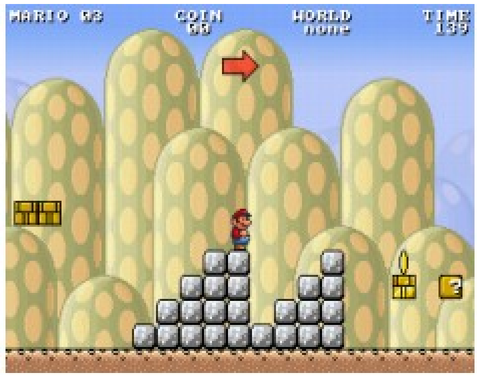


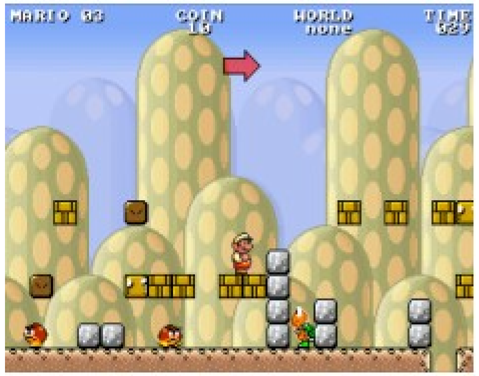
Table 2: Patterns supported in the fitness function.

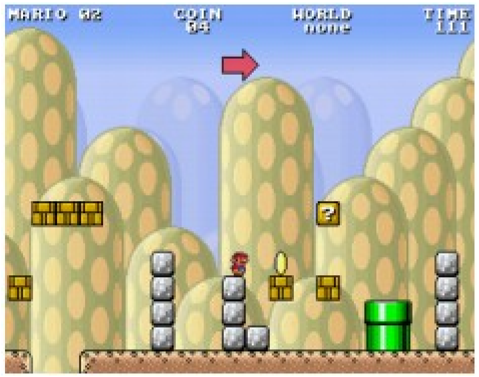
Enemies	
Enemy	Low
2-Horde	Low
3-Horde	Low
4-Horde	Low
Roof	Medium
Gaps	
Gaps	Low
Multiple gaps	By stacking
Variable gaps	By stacking
Gap enemy	Low–Medium by stacking
Pillar gap	Pillar High
Valleys	
Valley	Low
Pipe valley	Medium
Empty valley	By stacking
Enemy valley	By stacking
Roof valley	By stacking
Multiple paths	
2-Path	Medium–High
3-Path	Medium–High
Risk and Reward	By stacking
Stairs	
Stair up	Low
Stair down	Low
Empty stair valley	Low
Enemy stair valley	By stacking
Gap stair valley	By stacking

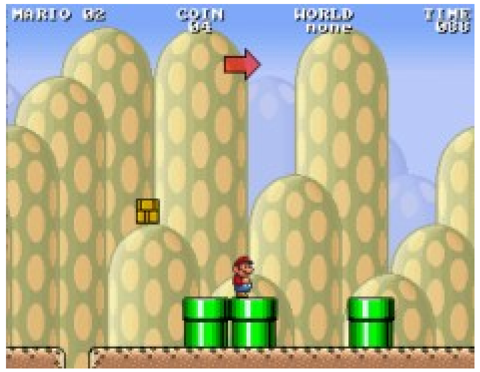
Mario map generation

- Population of 200 (randomly initialised)
- Rank by fitness, top 50% kept
- One point crossover between pairs in rank order (best breeds with second best)
- Mutation: inject a random character in a random position
- 10,000 generations

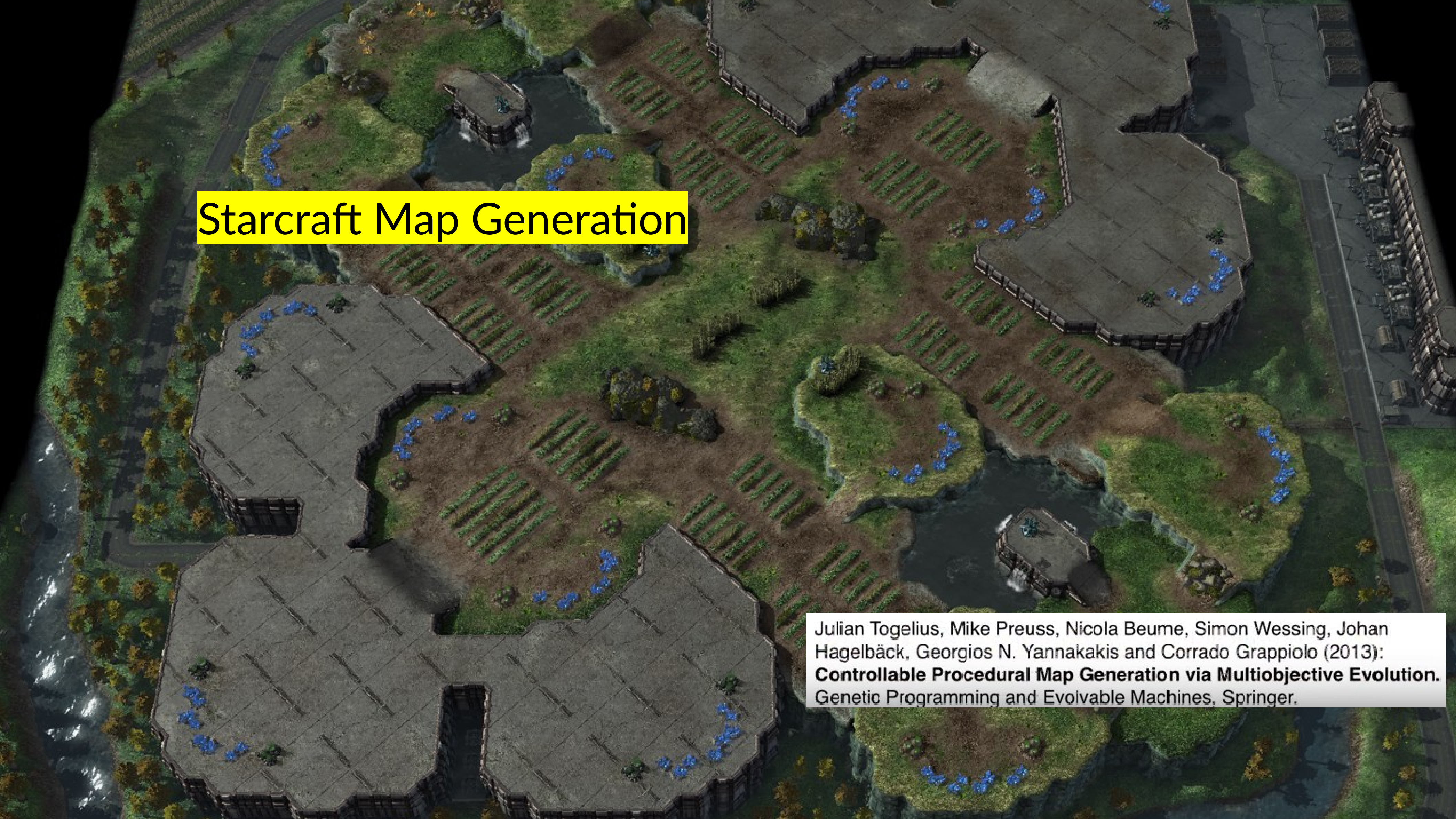








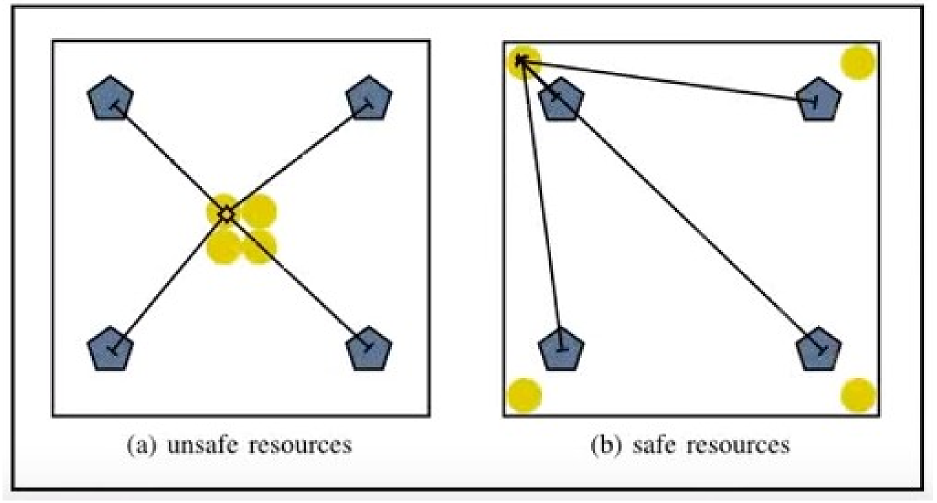
Starcraft Map Generation



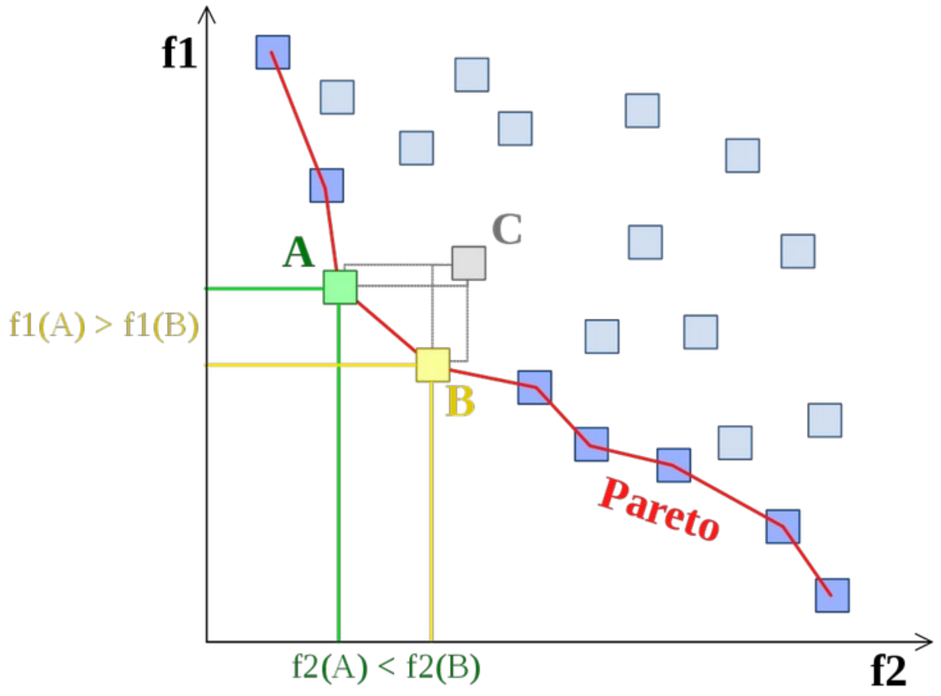
Julian Togelius, Mike Preuss, Nicola Beume, Simon Wessing, Johan Hagelbäck, Georgios N. Yannakakis and Corrado Grappiolo (2013): **Controllable Procedural Map Generation via Multiobjective Evolution.** Genetic Programming and Evolvable Machines, Springer.

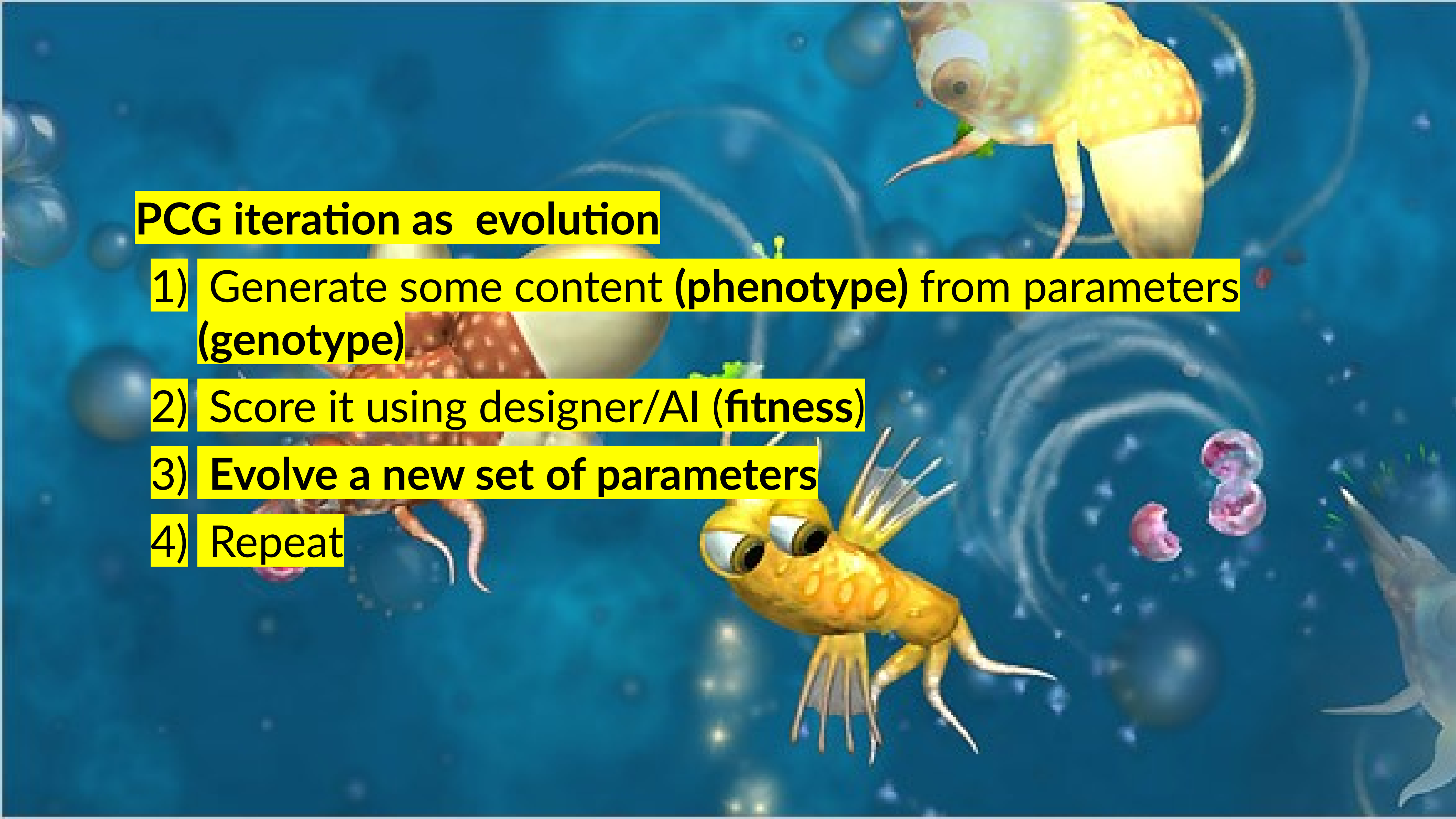
- Evolution of RTS maps
  - Desirable
    - Playability
    - Fairness
    - Skill differentiation
    - Interestingness

- Fairness fitness function
  - Distance from base to closest resource
  - Resource ownership
  - Resource safety
  - Resource fairness



- Multiobjective EAs have multiple objectives
  - **Pareto front** between them
  - The point at which increasing one objective will decrease another
  - Not always desirable – sometimes you want to look for solutions that are good for **any** of the objectives





PCG iteration as evolution

1) Generate some content **(phenotype)** from parameters **(genotype)**
2) Score it using designer/AI (**fitness**)
3) **Evolve a new set of parameters**
4) Repeat

Further reading

- Eiben & Smith, 2015. *Introduction to Evolutionary Computing* (Second Edition) Springer
- Yannakakis and Togelius, 2018. *Artificial Intelligence and Games*. Springer
- ...and the links throughout the slides