Hacking
Reinforcement Learning

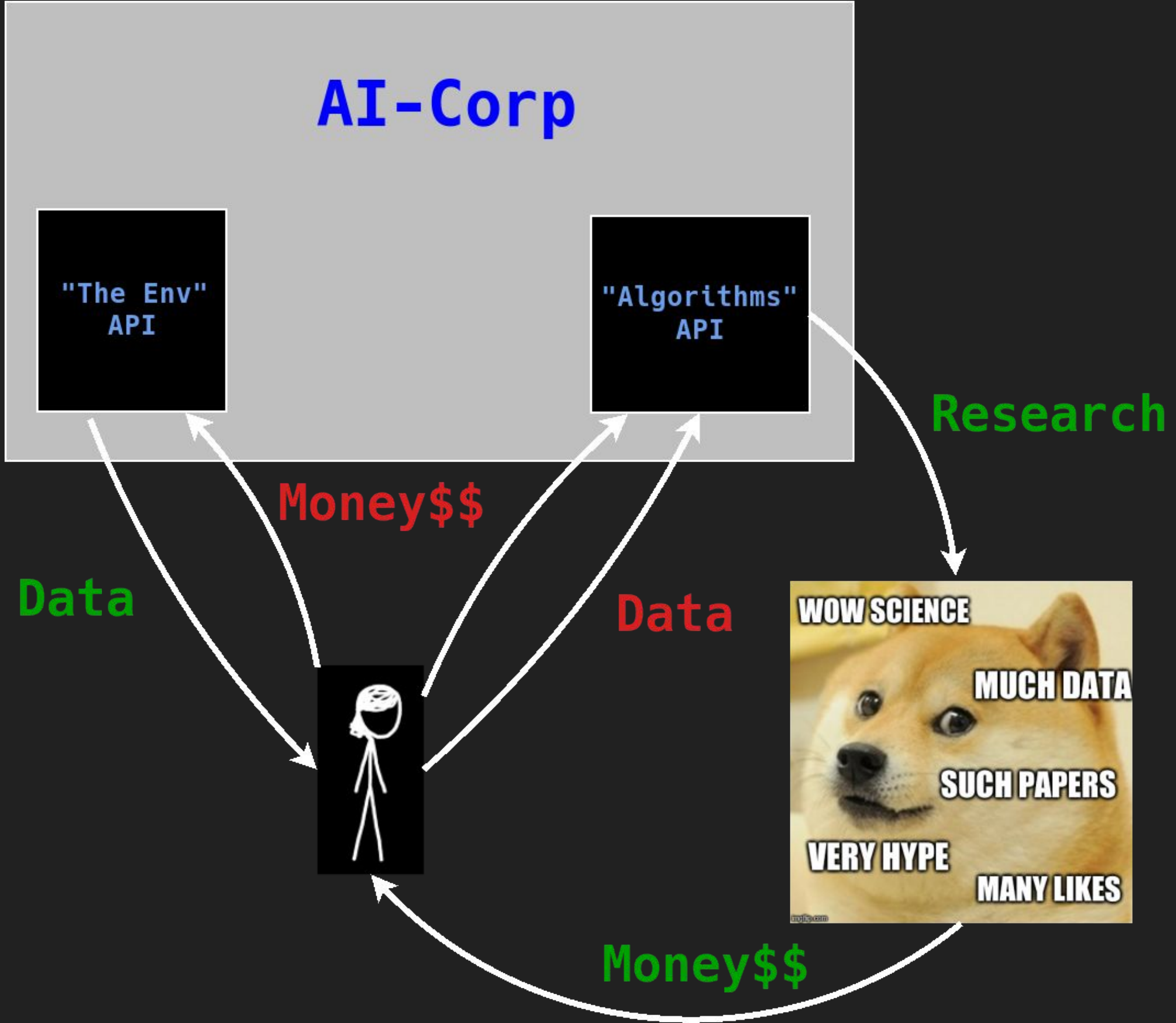
Guillem Duran Ballester

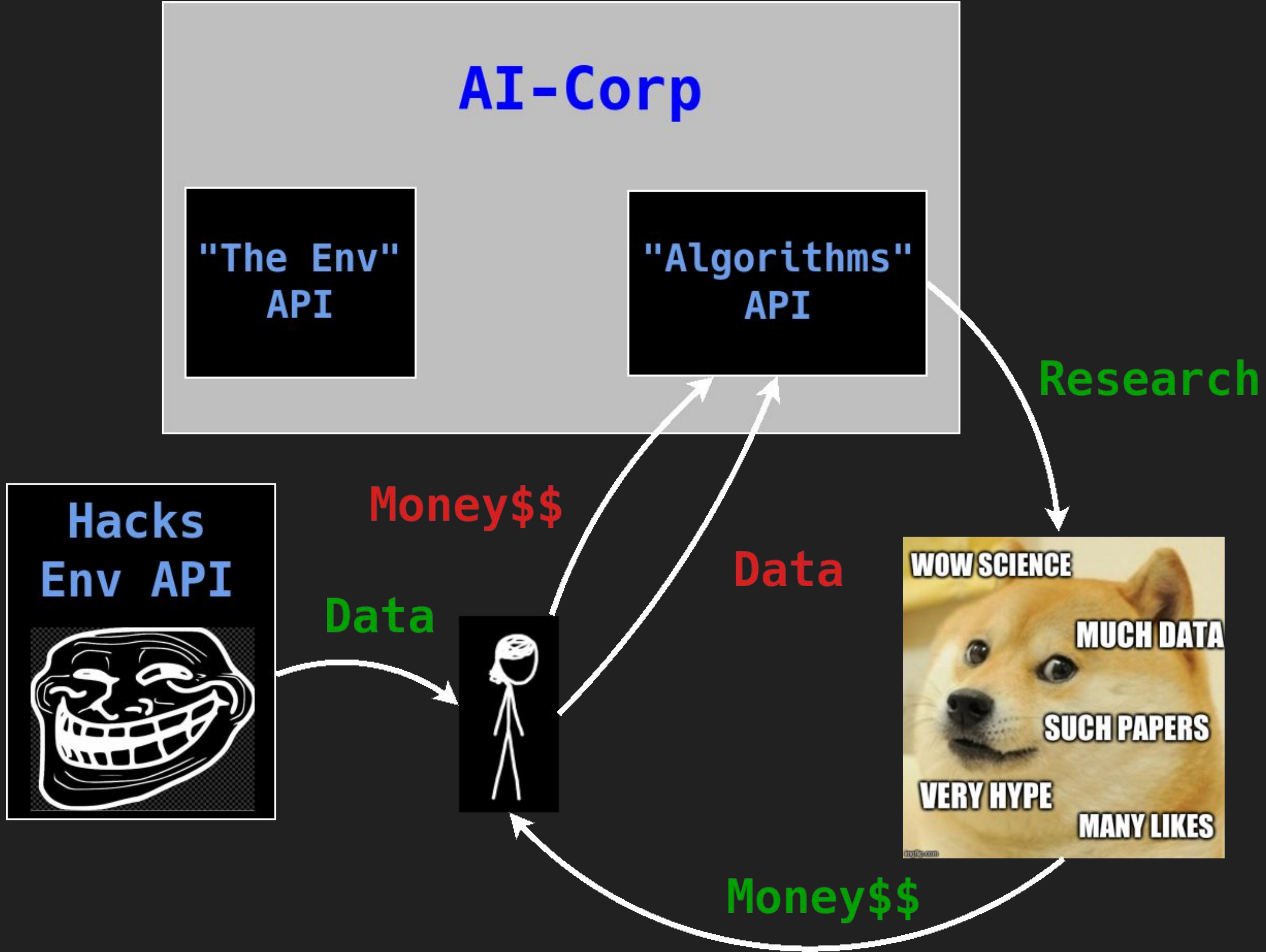
Guillemdb

@Miau_DB

europython
Edinburgh 23-29 July
2018

A tale about hacking AI-Corp

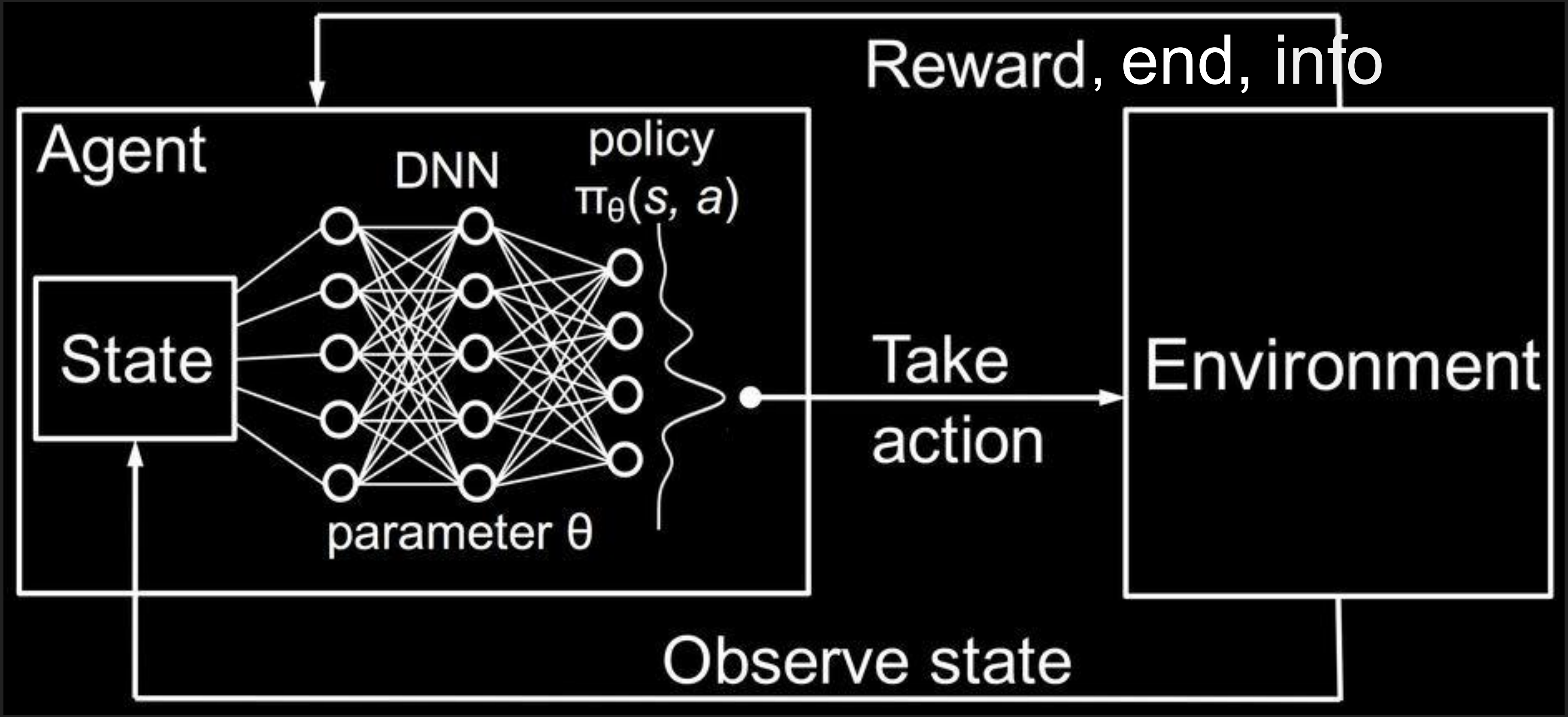

AI-Corp
"The Env"
API
"Algorithms"
API
Research
Money$$
Data
Data
WOW SCIENCE
MUCH DATA
SUCH PAPERS
VERY HYPE
MANY LIKES
Money$$


AI-Corp
"The Env"
API
"Algorithms"
API
Research
Hacks
Env API
Money$$
Data
Data
WOW SCIENCE
MUCH DATA
SUCH PAPERS
VERY HYPE
MANY LIKES
Money$$

Hacking RL

1. **Information gathering**
2. Scanning
3. Exploitation & privilege escalation
4. Maintaining access & covering tracks

What is RL?


Reward, end, info
Agent
DNN
policy
$\pi_\theta(s, a)$
State
parameter θ
Take
action
Environment
Observe state

Our Hobby:
Developing FractalAI

"Study hard what interests you the most in the most undisciplined, irreverent and original manner possible.”

R. P. Feynman

Sergio Hernández
@EntropyFarmer

Guillem Duran
@Miau_DB

Causal entropic forces

- Paper by Alex. Wissner-Gross (2013)
- Intelligence is a thermodynamic process
- No neural networks → Equations

Intelligent decision

Number of future possible outcomes

$$\mathbf{F}(\mathbf{X}_0, \tau) = T_c \nabla_{\mathbf{X}} S_c(\mathbf{X}, \tau)|_{\mathbf{X}_0}$$

Direction of maximum

Given your current state

Count all the paths that exist

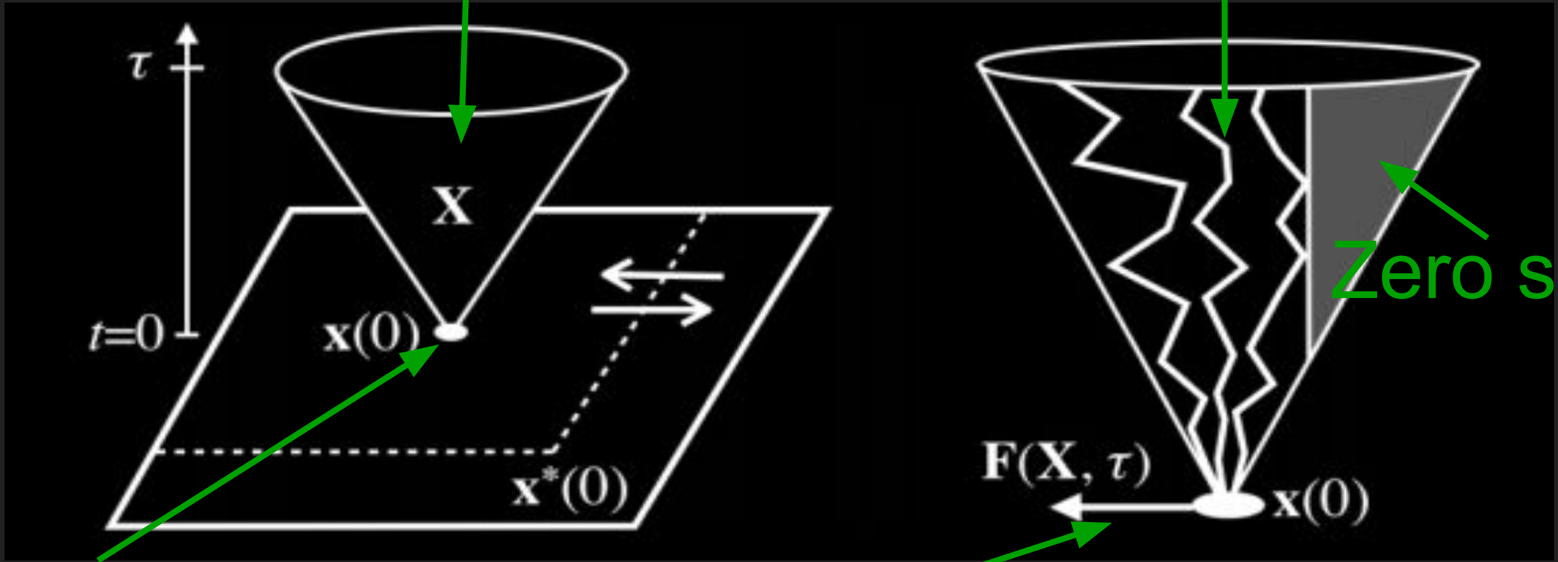
Until you reach the time horizon

$$S_c(\mathbf{X}, \tau) = -k_{\mathrm{B}} \int_{\mathbf{x}(t)} \Pr\left(\mathbf{x}(t) | \mathbf{x}(0)\right) \ln \Pr\left(\mathbf{x}(t) | \mathbf{x}(0)\right) \mathcal{D}\mathbf{x}(t)$$

Map them to a score

Cone: Space of future possible outcomes

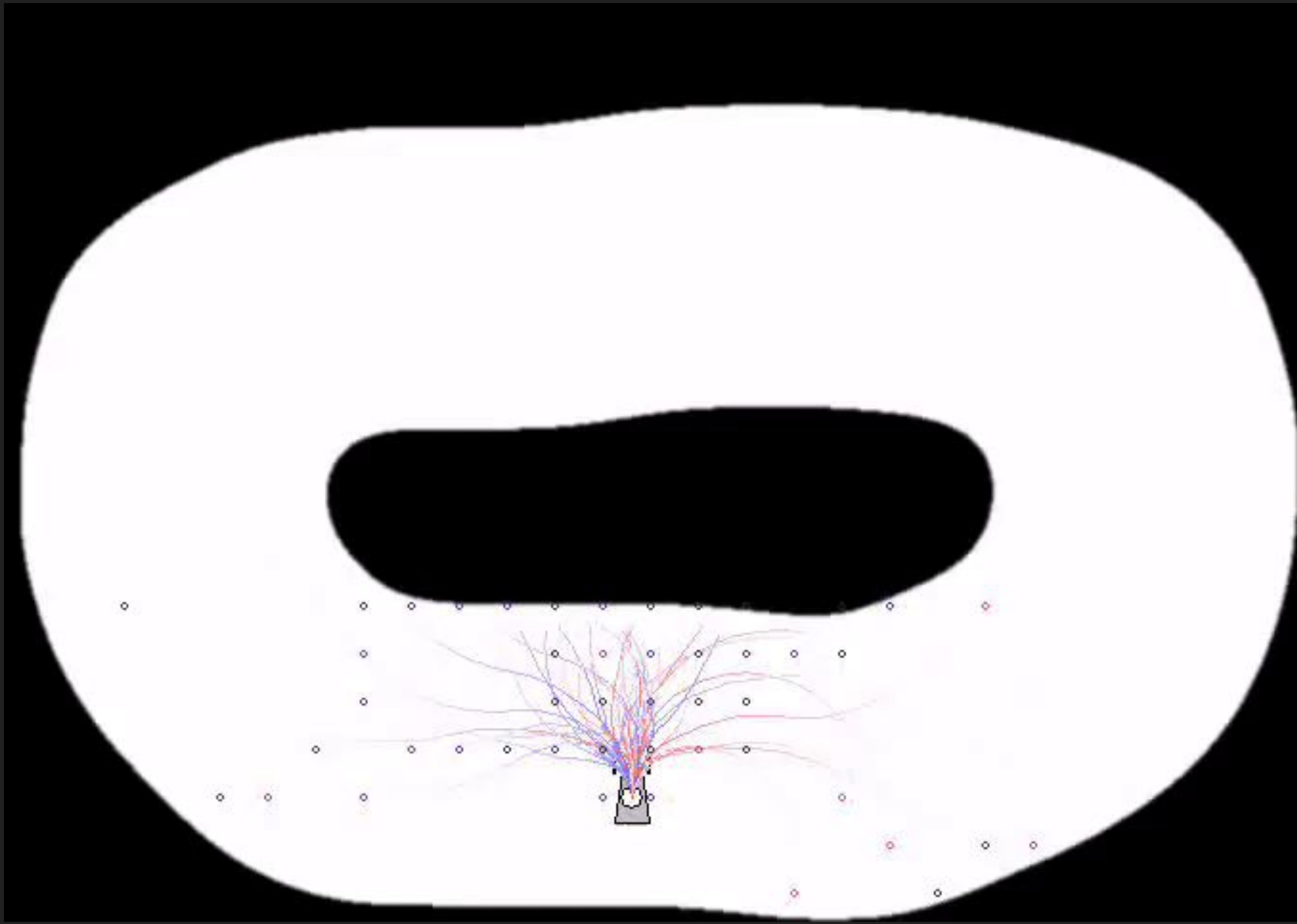
Sample random walks

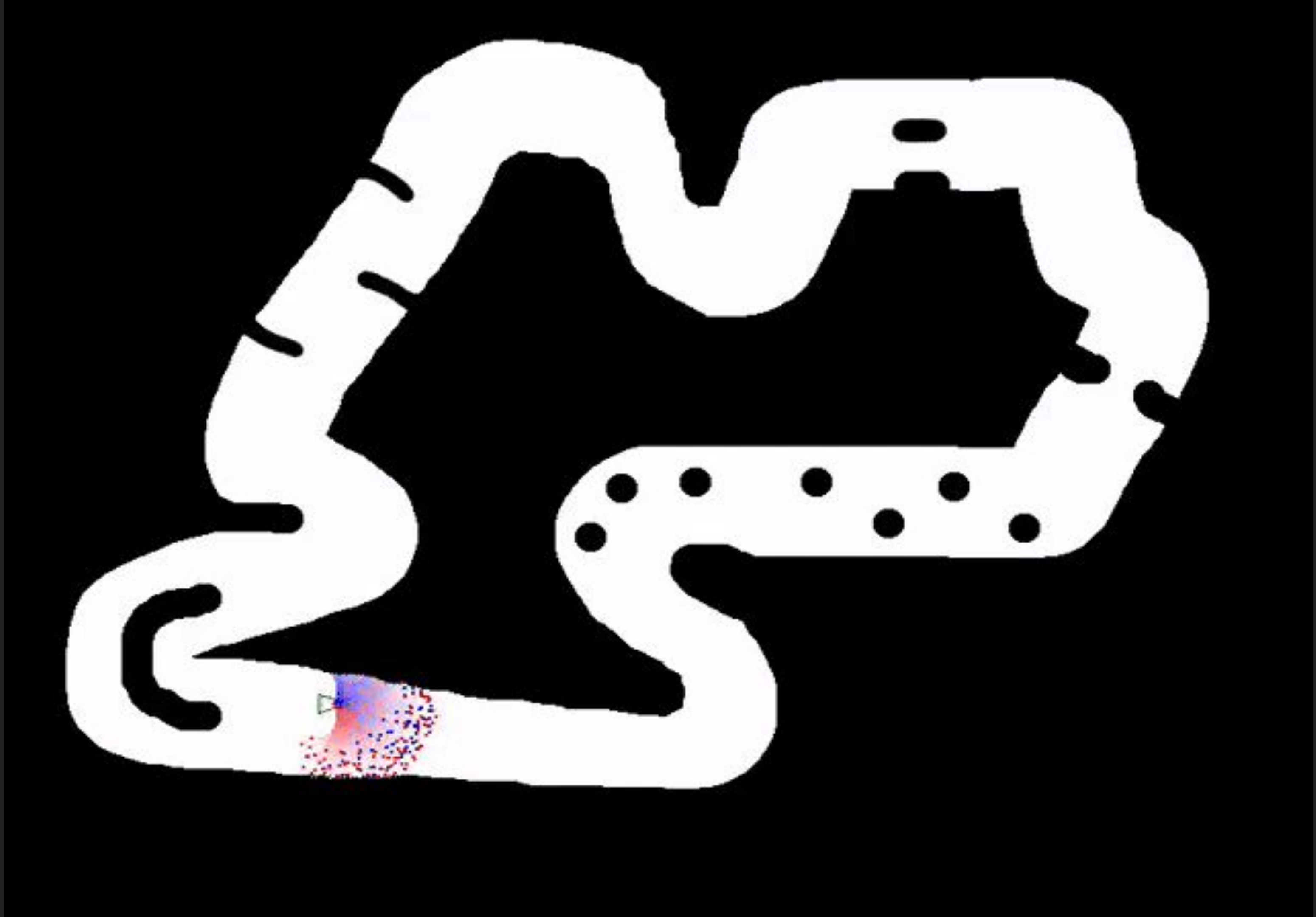


Zero score

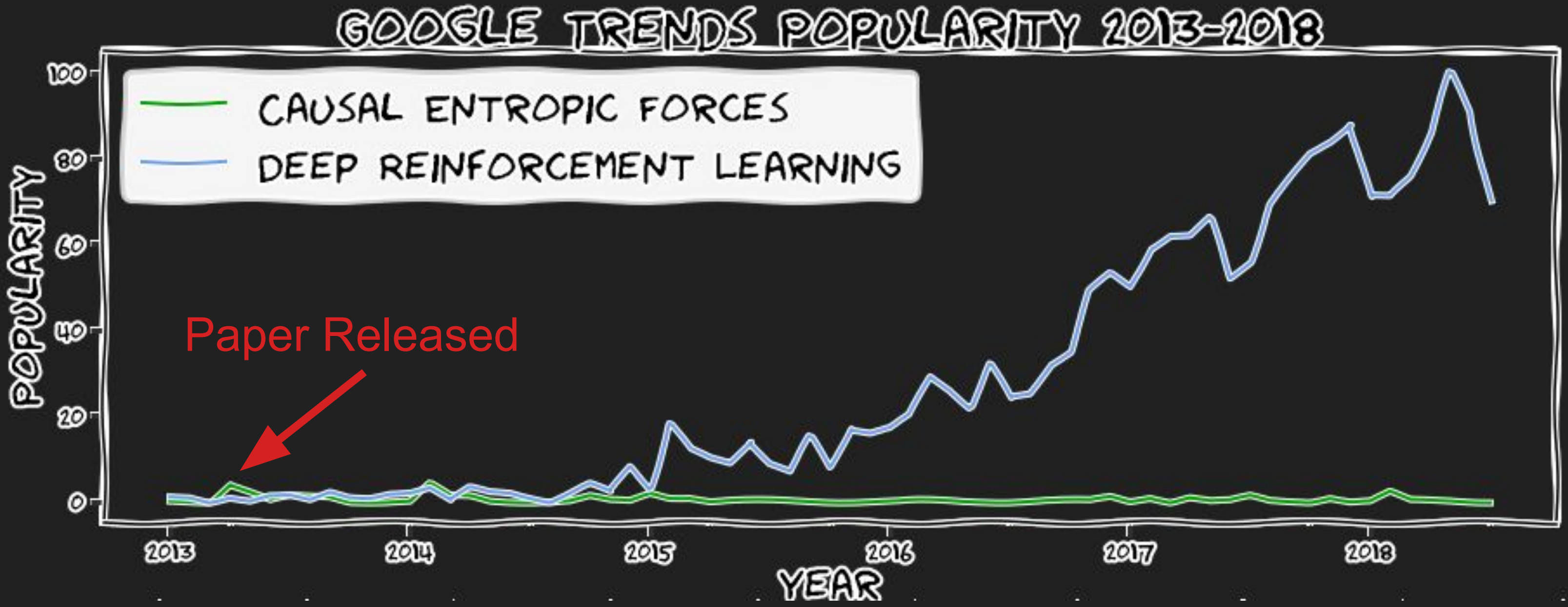
Present

Move away from the wall so fewer walks get 0 score





Nobody likes entropic forces



- All rewards equal 1
- **NP hard!**

FractalAI

- Finds low probability points and paths
- Constrained resources
- Total control of exploration process
- **Linear time**

FractalAI

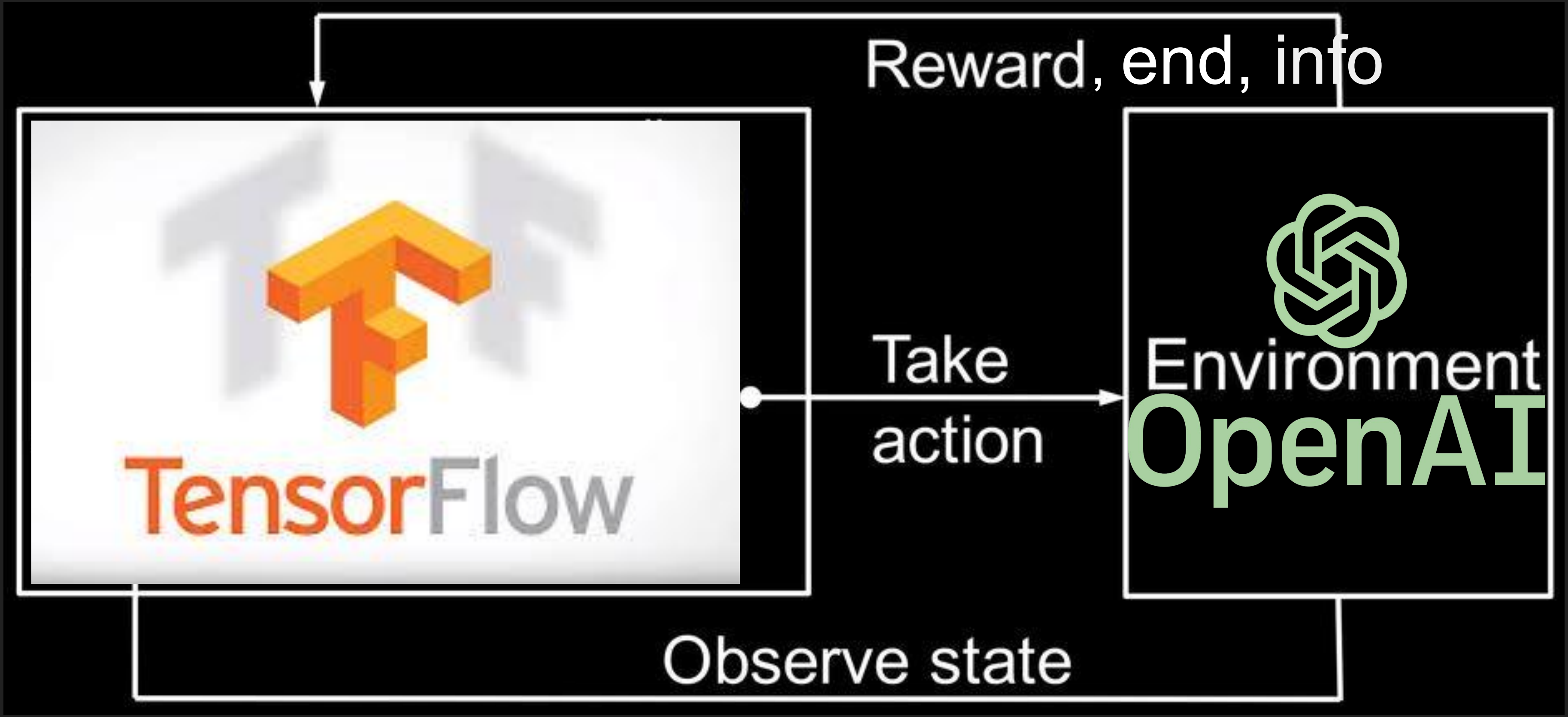
A set of rules for:

1. Defining a cloud of points (Swarm)
2. Moving a Swarm in any Cone
3. Measuring and comparing Swarms
4. Analyzing the history of a Swarm

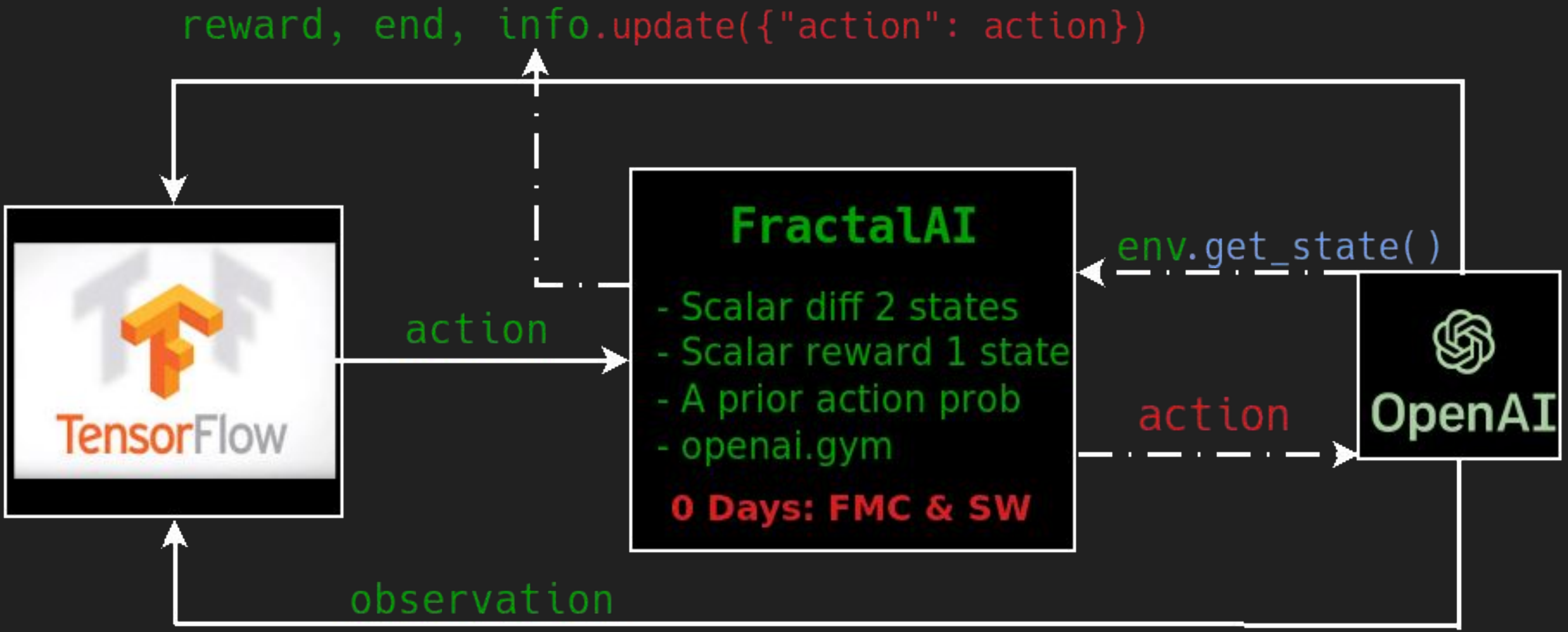
Hacking RL

1. Information gathering
2. **Finding vulnerabilities & Scanning**
3. Exploitation & privilege escalation
4. Covering tracks & Maintaining access

RL

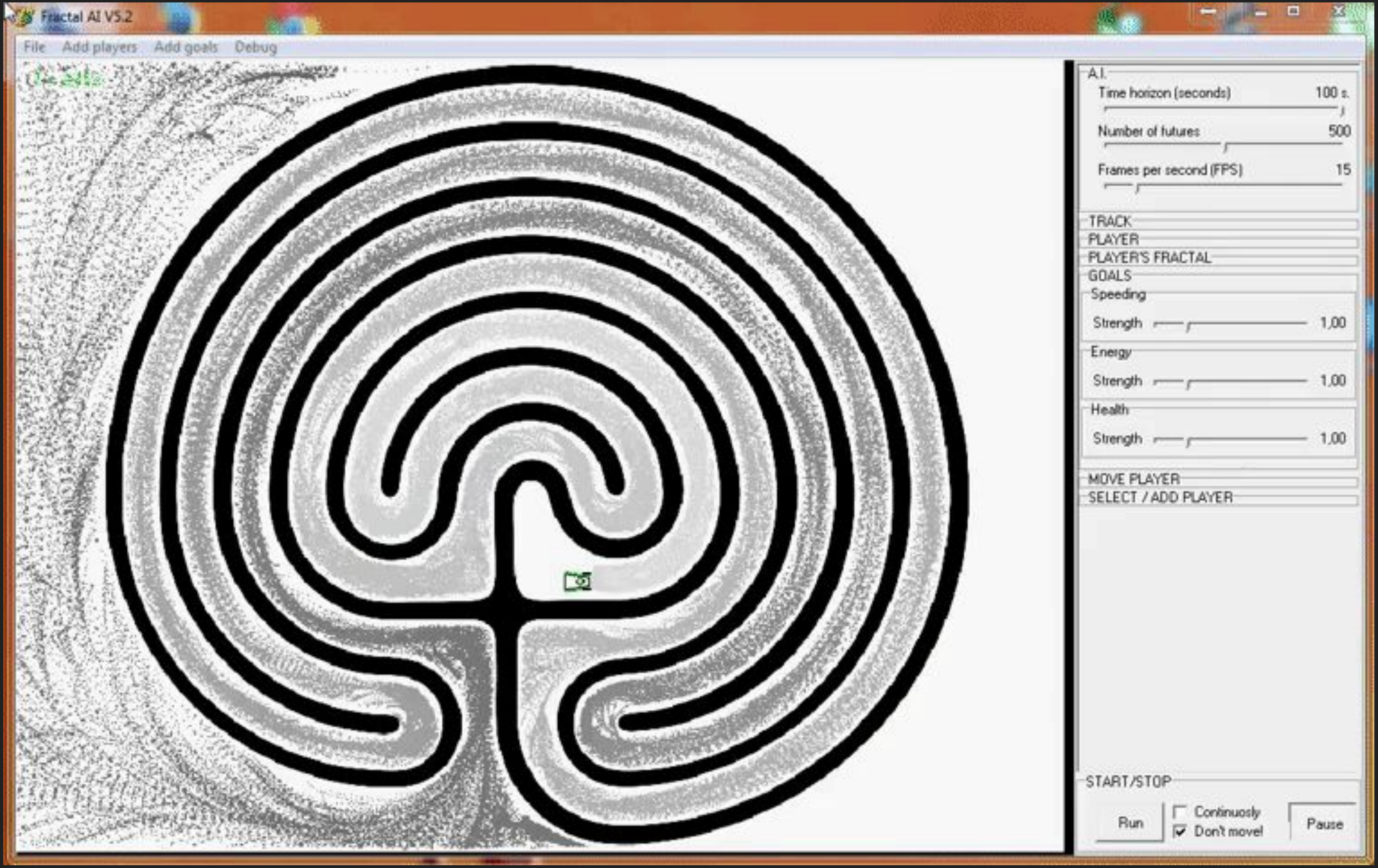

Reward, end, info
TensorFlow
Take
action
Environment
OpenAI
Observe state

Finding an attack vector



Swarms are cool

- They move in **linear time**.
- **Pixels/RAM + Reward**.
- They **guess** density **distributions**
- They follow useful paths


Fractal AI V5.2
File
Add players
Add goals
Debug
A.I.
Time horizon (seconds)
100 s.
Number of futures
500
Frames per second (FPS)
15
TRACK
PLAYER
PLAYER'S FRACTAL
GOALS
Speeding
Strength
1,00
Energy
Strength
1,00
Health
Strength
1,00
MOVE PLAYER
SELECT / ADD PLAYER
START/STOP
Run
Continuosly
Don't move!
Pause

Cunningham's Law

"The best way to *get* the right *answer* on the *Internet* is not to ask a question; it's to post the *wrong answer*."

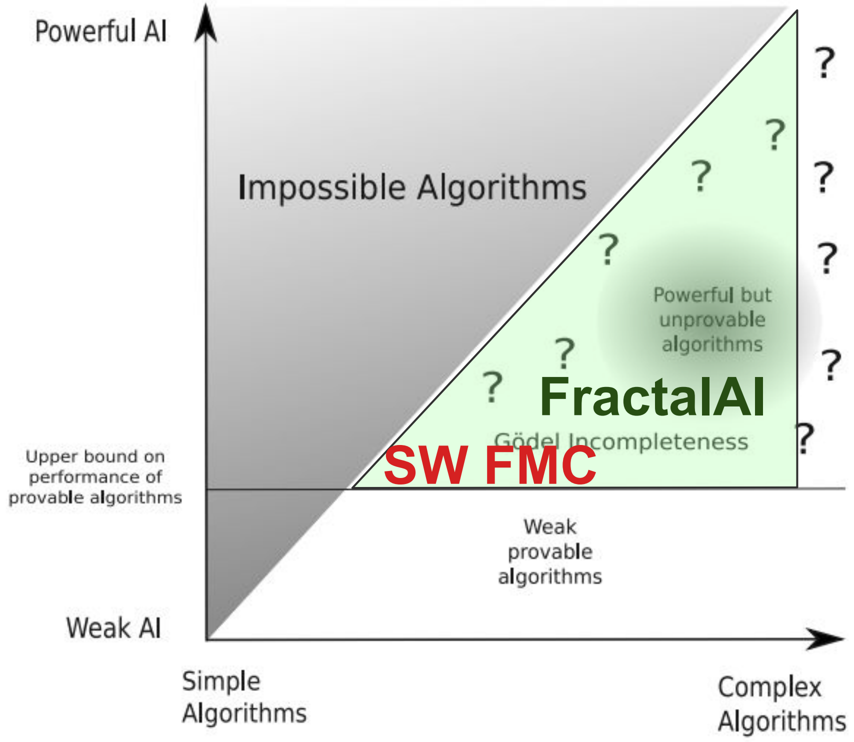


Figure 5.1.: Theorem 5.3.3 rules out simple but powerful artificial intelligence algorithms, as indicated by the greyed out region in the upper left. Theorem 5.6.1 upper bounds how powerful an algorithm can be before it can no longer be proven to be a powerful algorithm. This is indicated by the horizontal line separating the region of provable algorithms from the region of Gödel incompleteness.

Using a Swarm to generate data

- **S**warm **W**ave (SW)
  - Move a **Swarm** → Sample state space
  - **Cone** → Tree of visited states
  - Efficient → Only one tree

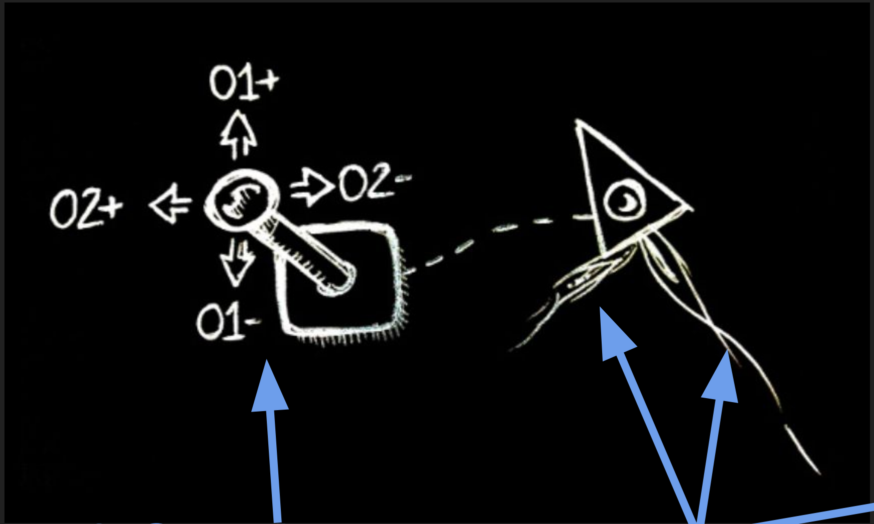

T = 0.00 s.

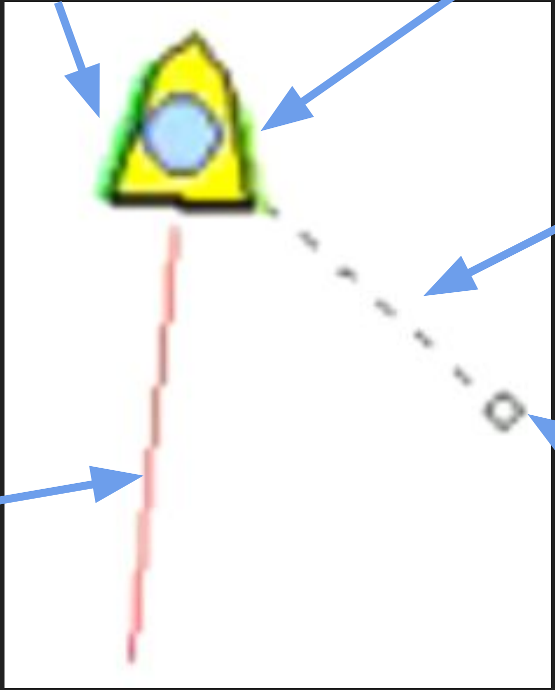
Using a Swarm to generate data

- **S**warm **W**ave (SW)
  - Move a **Swarm** → Sample state space
  - **Cone** → Tree of visited states
  - Efficient → Only one tree

- **F**ractal **M**onte **C**arlo (FMC)
  - 1 Cone per action
  - Robust → Stochastic/difficult envs
  - Distribution of action utility

Hardcore Lunar Lander



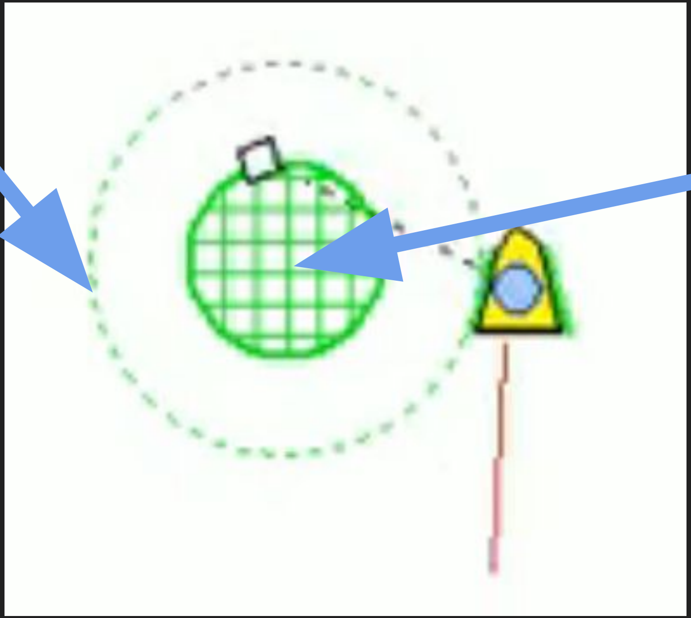


The Gameplay

Reward

- Health + Fuel level
- Closer to target → +0.2
- Reach target → +100

Catch rock outside this circle

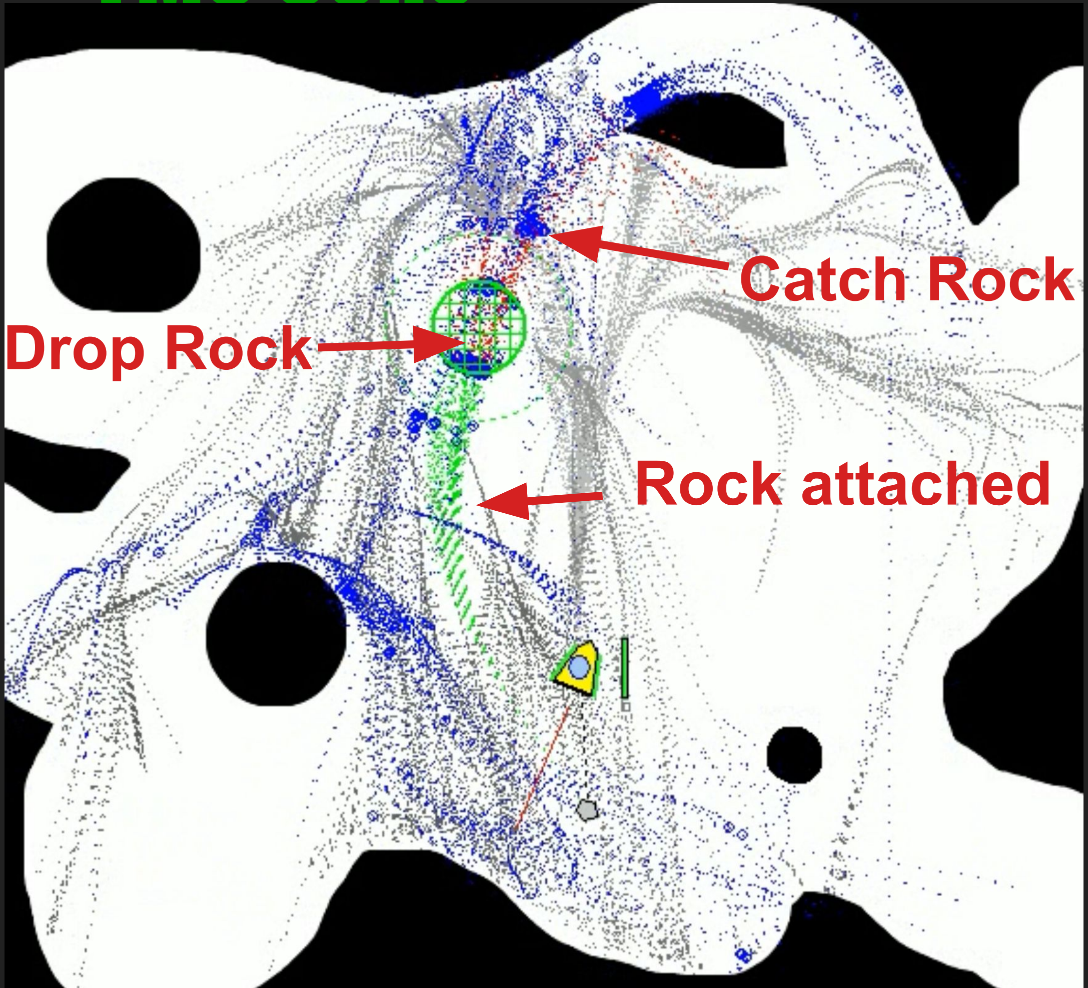


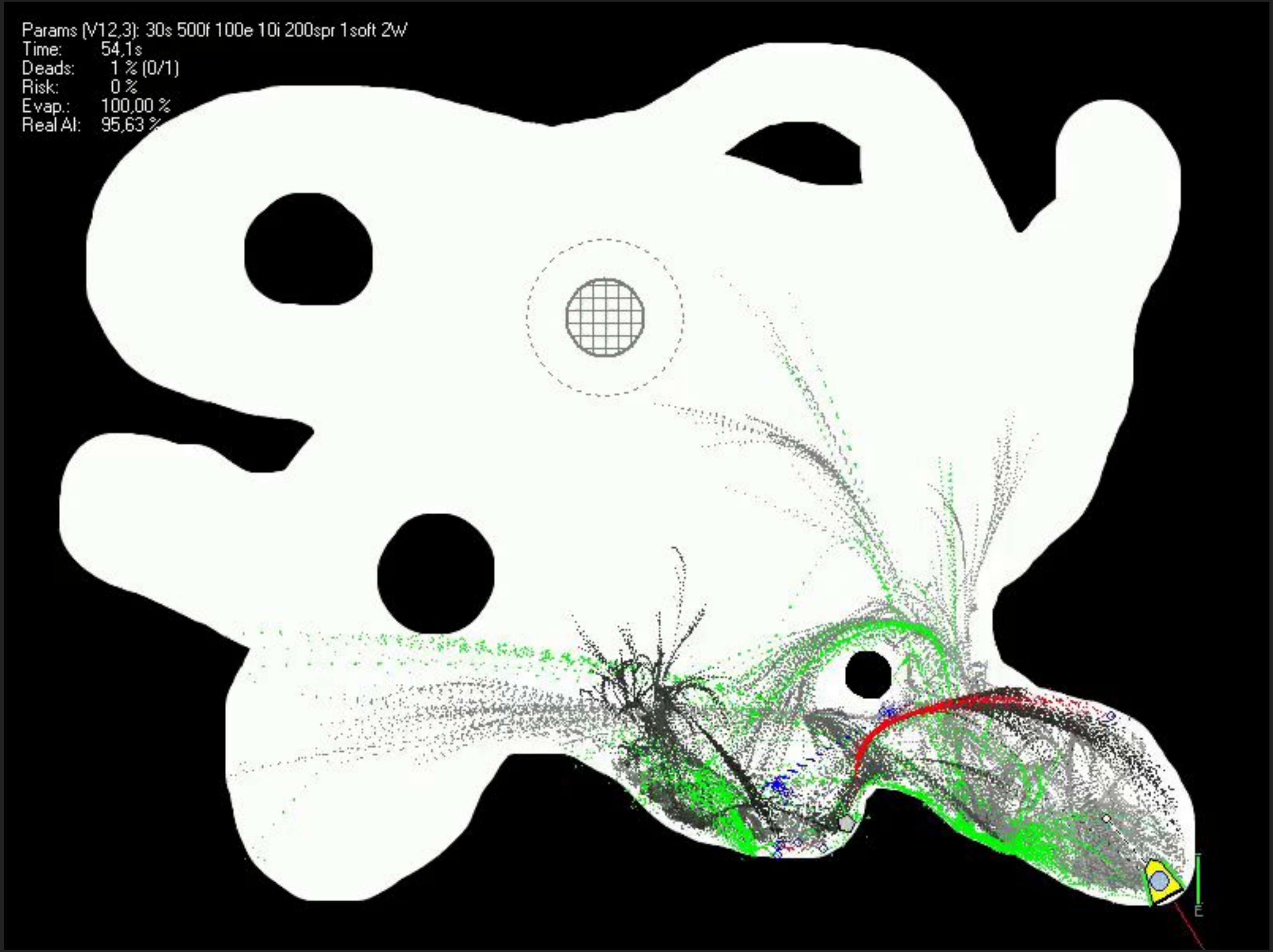
Bring rock here

Don’t crash!

FMC Cone

- Grey lines: Rocket Paths
- Colored lines: Hook's Path
- Colored change: New target (Pick up/drop rock)




Params (V12,3): 30s 500f 100e 10i 200spr 1soft 2W
Time: 54,1s
Deads: 1 % (0/1)
Risk: 0 %
Evap.: 100,00 %
Real AI: 95,63 %

Hacking RL

1. Information gathering
2. Scanning
3. **Exploitation & privilege escalation**
4. Maintaining access & covering tracks

Demo time!

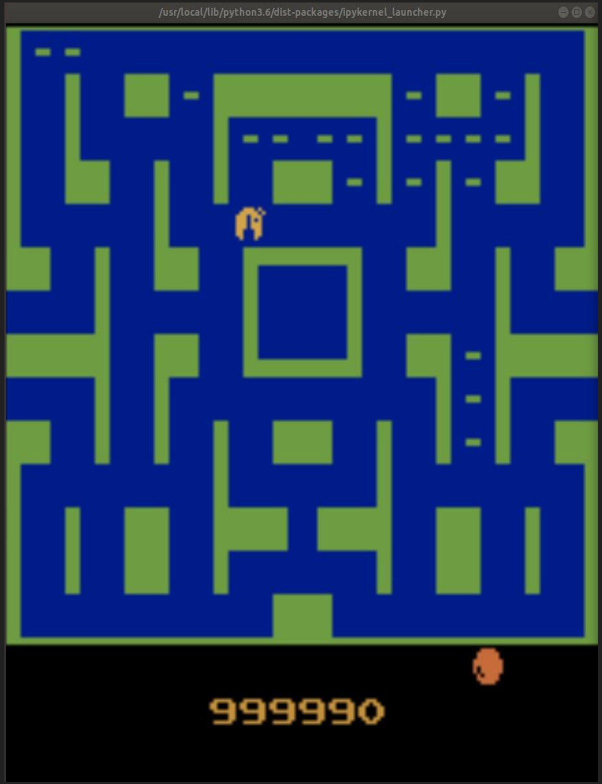
Hacking RL

1. Information gathering
2. Scanning
3. Exploitation & privilege escalation
4. **Maintaining access & managing tracks**

Performance of the Swarm Wave

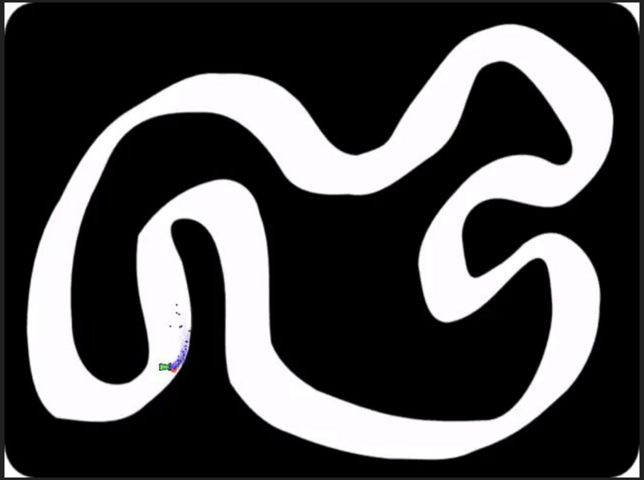
Robust to sparse rewards

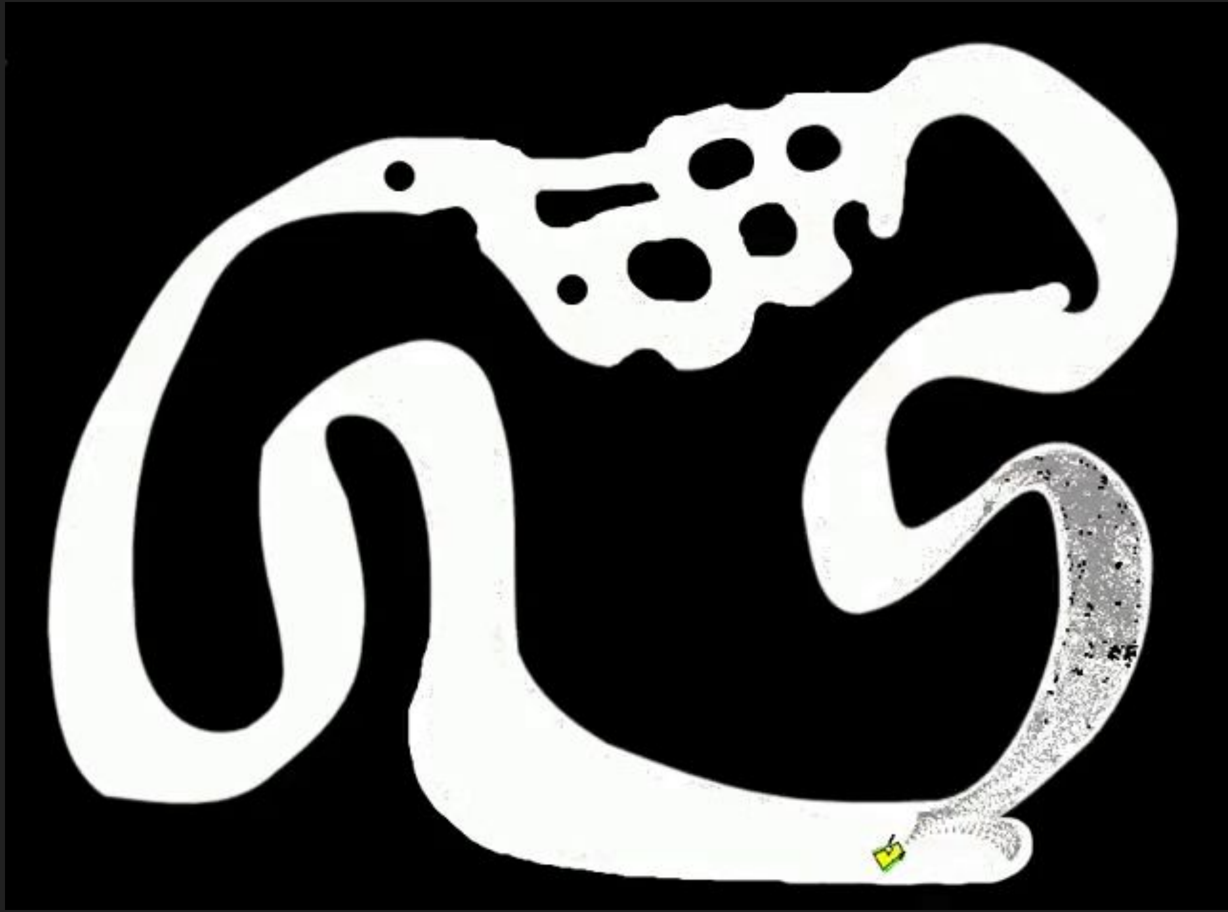
Solving Atari games is easy

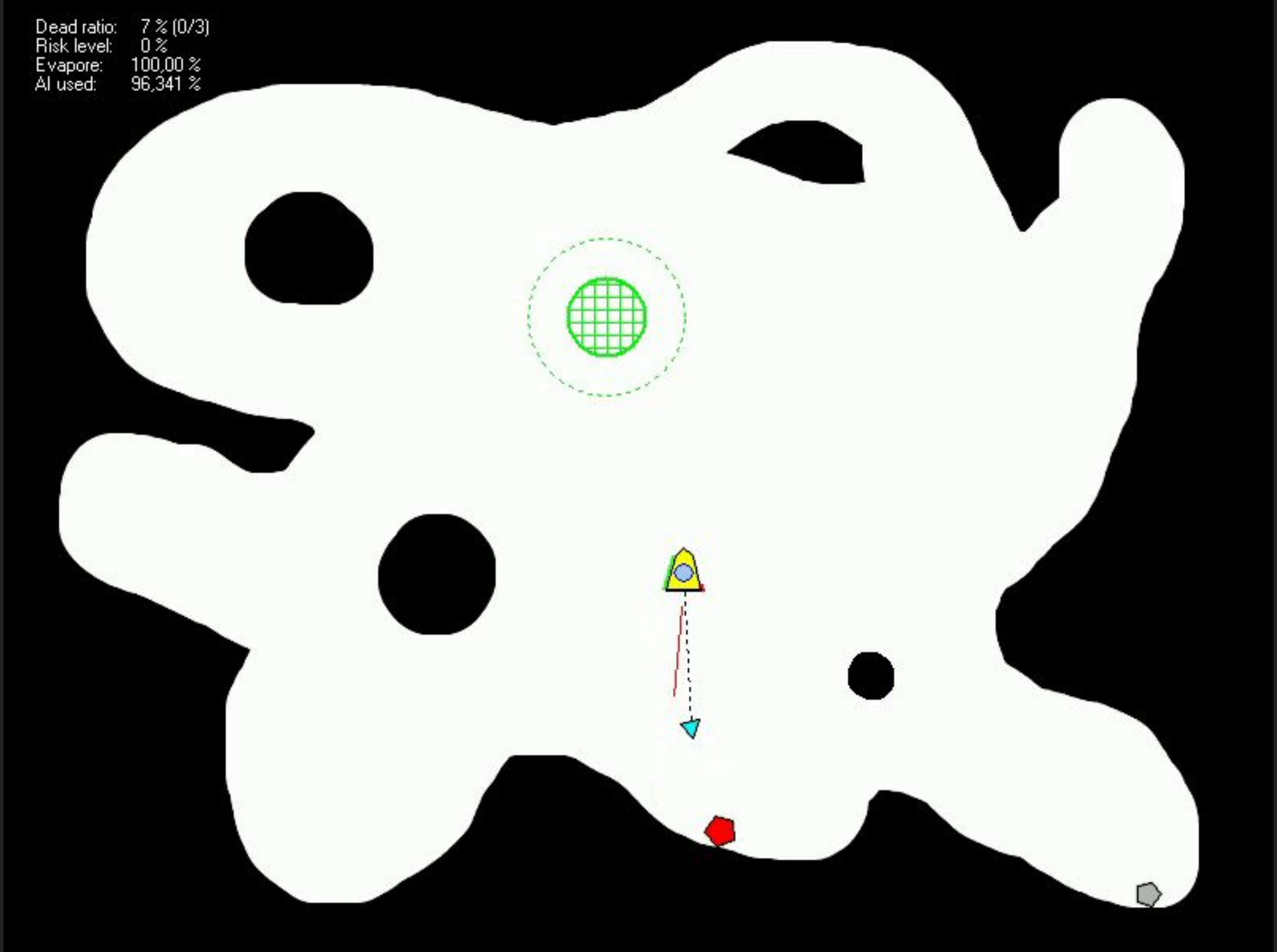


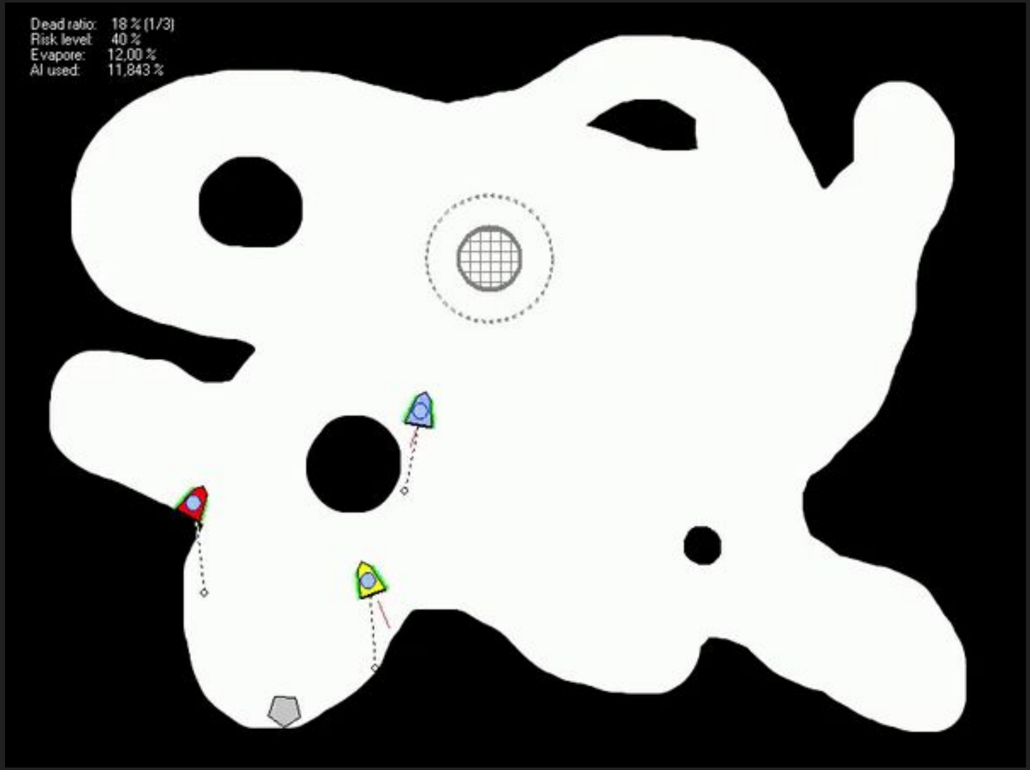
SW is useful in virtually all environments

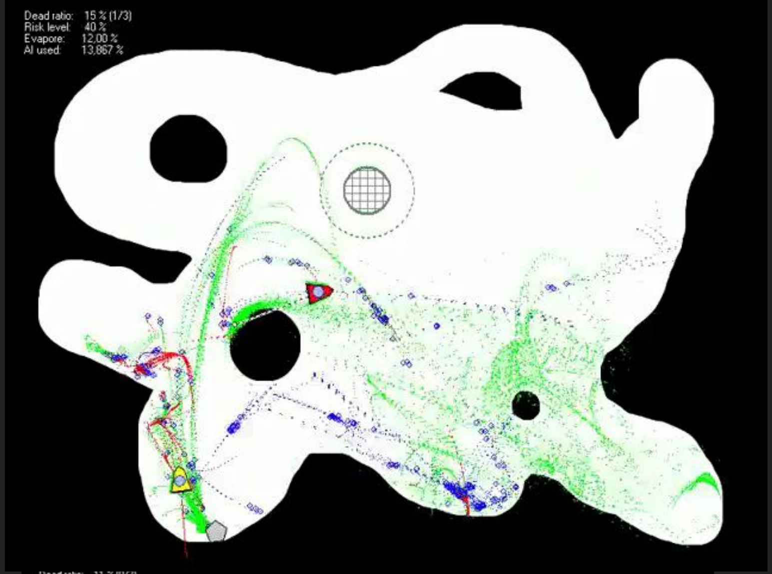
Fractal Monte Carlo

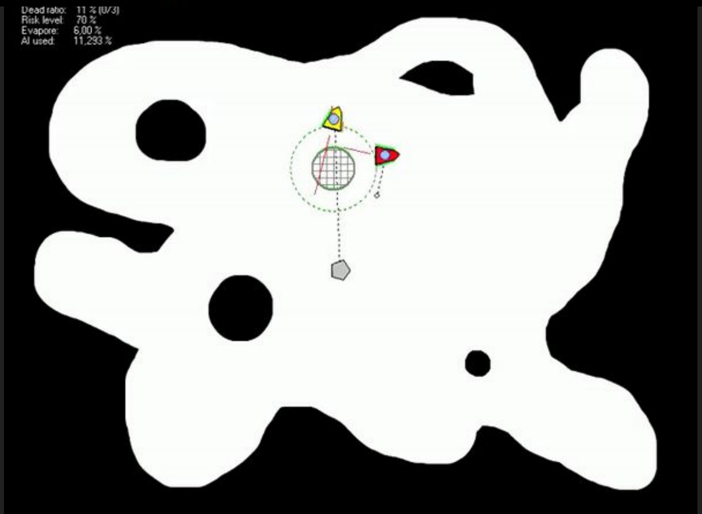




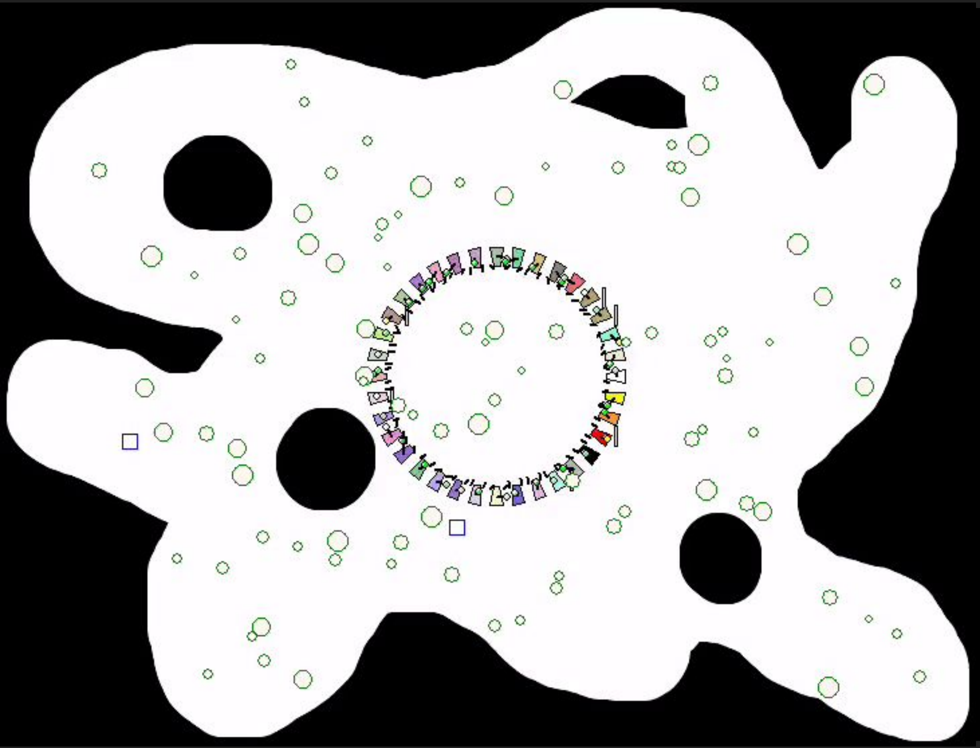

Dead ratio: 7 % (0/3)
Risk level: 0 %
Evapore: 100,00 %
AI used: 96,341 %

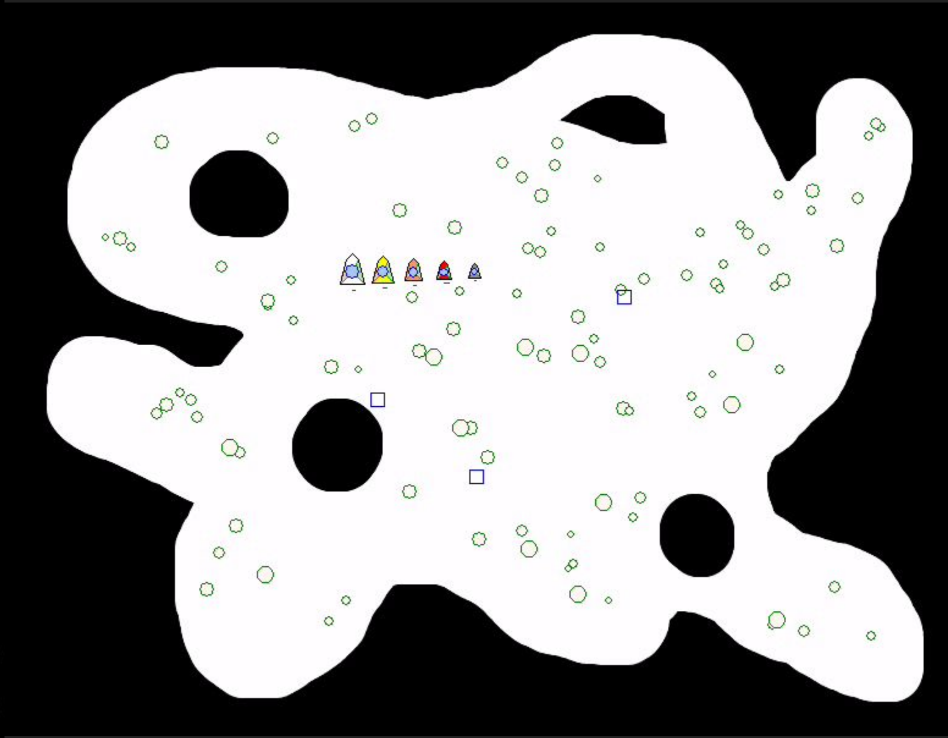

Dead ratio: 18 % (1/3)
Risk level: 40 %
Evapore: 12,00 %
AI used: 11,843 %


Dead ratio: 15 % (1/3)
Risk level: 40 %
Evapore: 12,00 %
AI used: 13,867 %

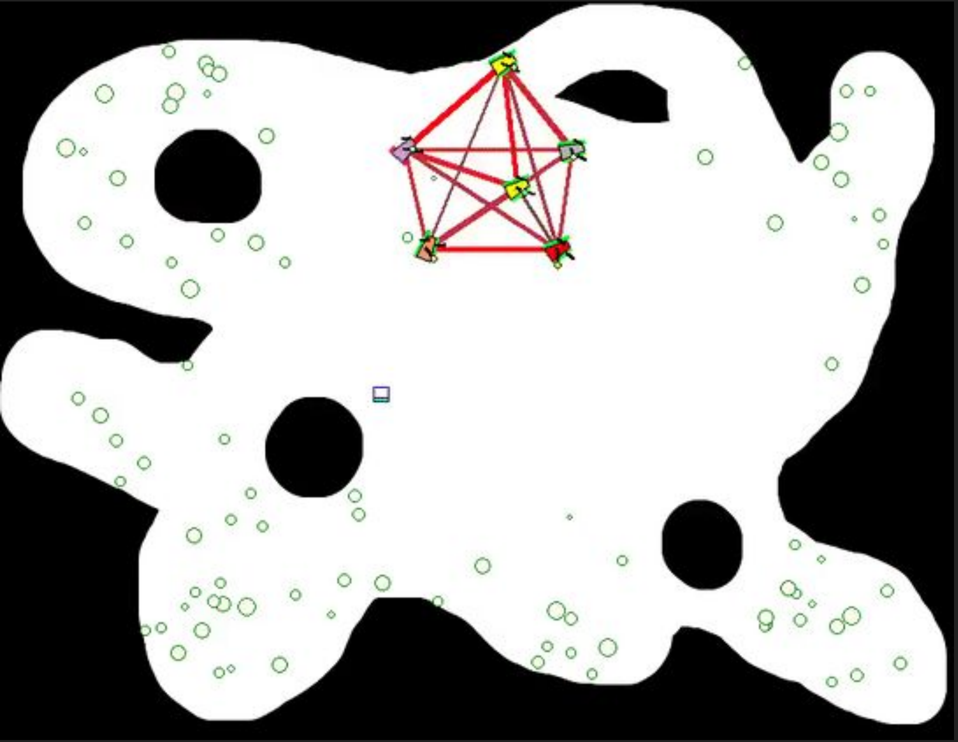

Dead ratio: 11 % (0/3)
Risk level: 70 %
Evapore: 6,00 %
AI used: 11,293 %

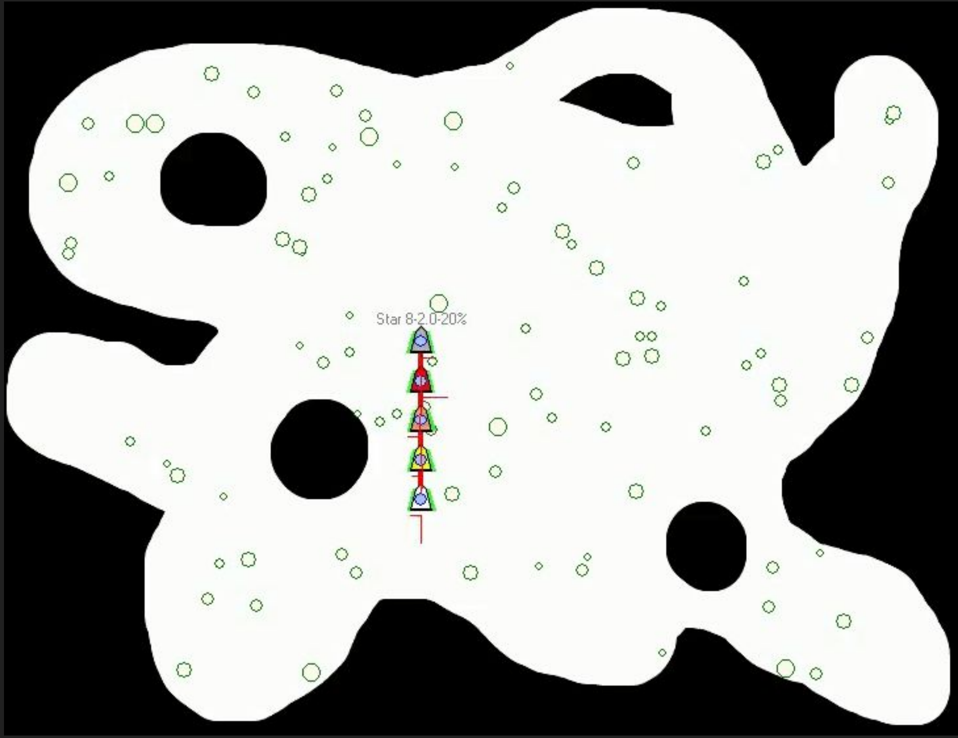
Control swarms of agents





Multi objective environments





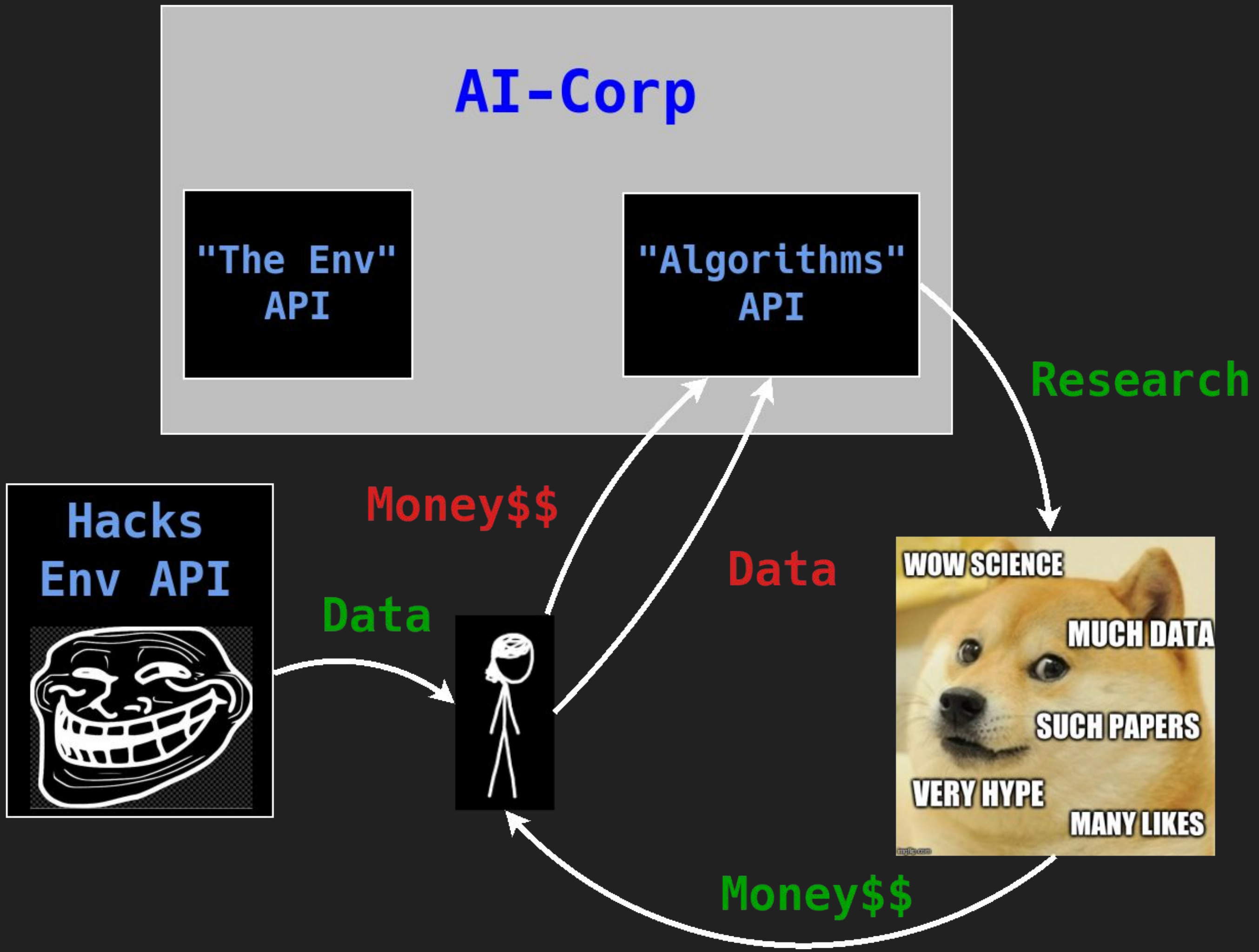
Hacking OpenAI Baselines

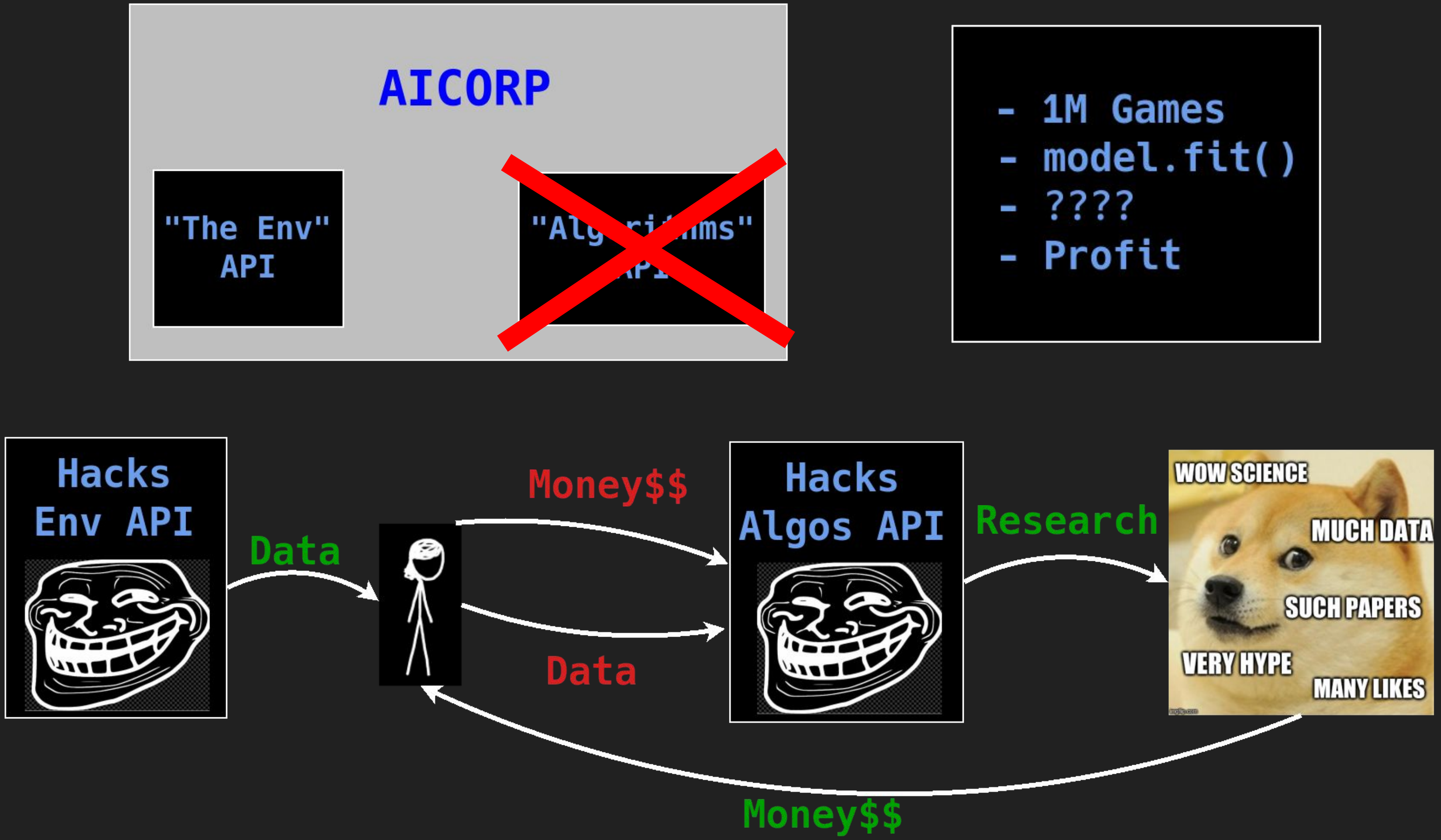
Run_atary.py → inject hacked env.

```
from baselines.common.cmd_util import atari_arg_parser#, make_atari_env
from fractalai.datasets.baselines import make_atari_env
import ray
ray.init()
```

A2c.py → recover action

```
obs, rewards, dones, infos = self.env.step(actions)
actions = [inf["action"] for inf in infos]
mb_actions.append(actions)
```


AI-Corp
"The Env"
API
"Algorithms"
API
Research
Hacks
Env API
Data
Money$$
Data
WOW SCIENCE
MUCH DATA
SUCH PAPERS
VERY HYPE
MANY LIKES
Money$$



AICORP
"The Env" API
~~"Algorithms" API~~
- 1M Games
- model.fit()
- ????
- Profit
Hacks Env API
Data
Money$$
Data
Hacks Algos API
Research
WOW SCIENCE
MUCH DATA
SUCH PAPERS
VERY HYPE
MANY LIKES
Money$$

Guillem Duran Ballester

Guillemdb

Let's coauthor papers or hire me!

- PyData Mallorca co organizer
- Telecomm. Engineer
- My hobby: hacking AI stuff
- RL Researcher Wannabe

- Save tons of money!
- SW & FMC are simple
- I learn stuff super fast
- I like teaching & sharing

Thank You!

Please Hack us:

1. Talk repo: **Guillemdb/hacking-rl**
2. Code: **FragileTheory/FractalAI**
3. More than 100 videos
4. PDFs on arXiv.org

@Miau_DB

@Entropyfarmer

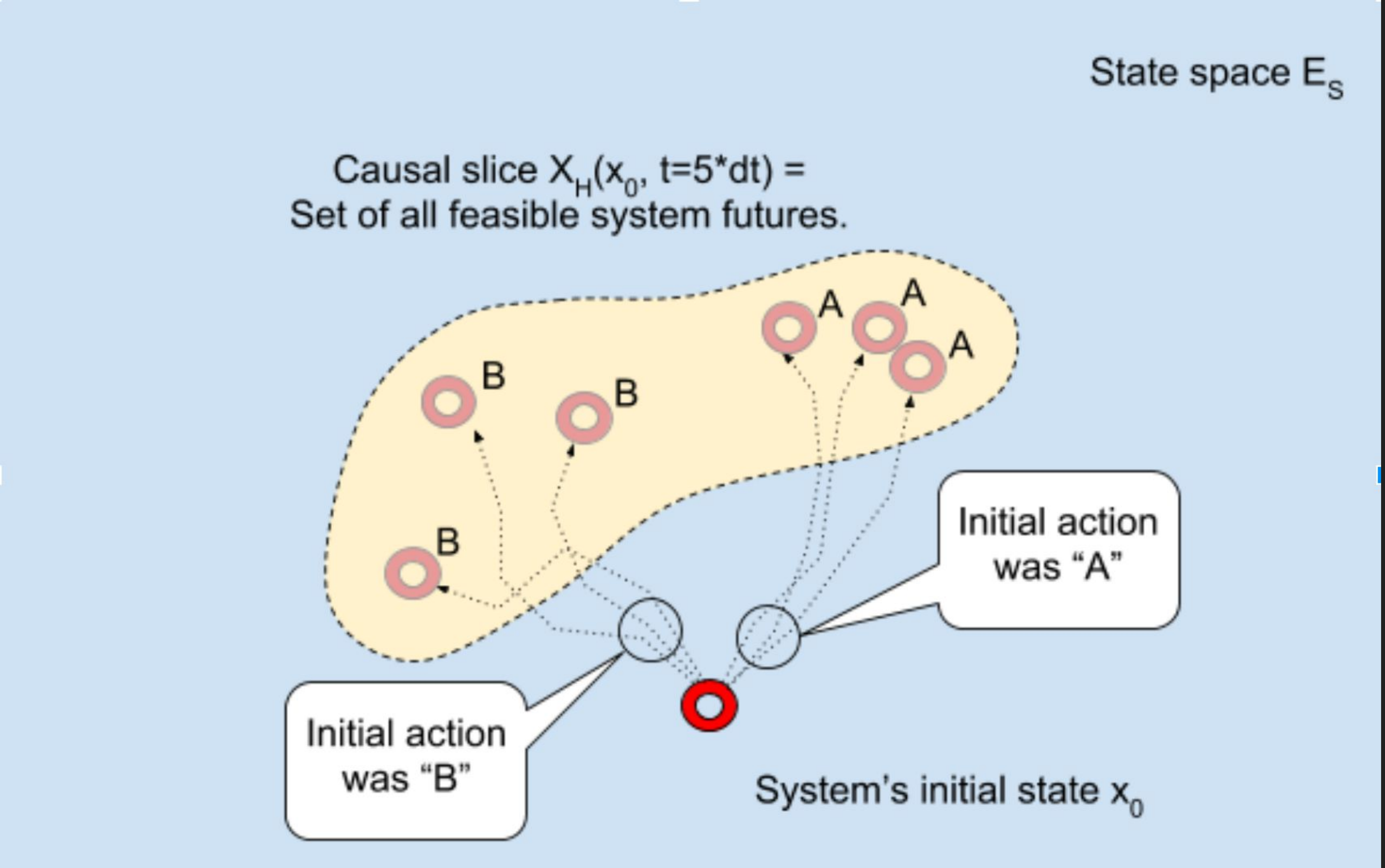
Additional Material

- How the algorithm works
- An overview of the FractalAI repository
- Reinforcement Learning as a supervised problem
- Hacking OpenAI baselines
- Papers that need some love
- Improving AlphaZero
- Combining FractalAI with neural networks

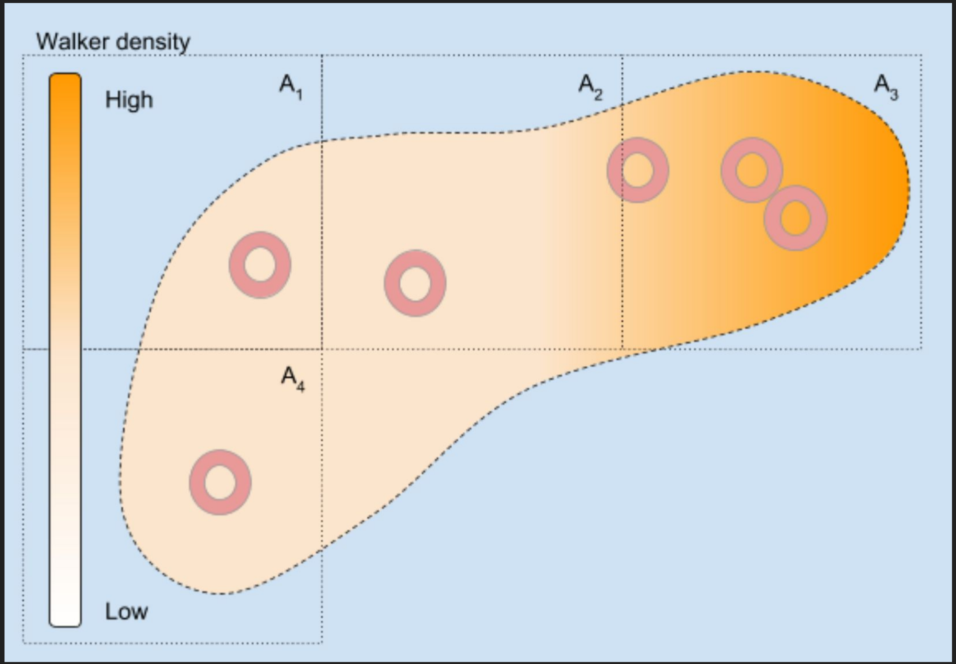
The Algorithm

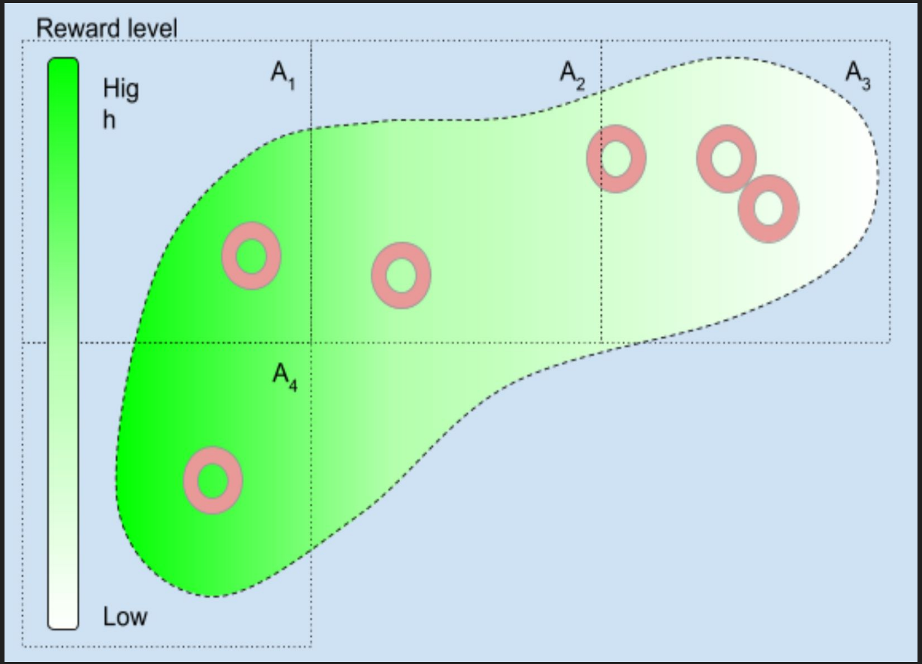
1. Random perturbation of the walkers
2. Calculate the virtual reward of each walker
   a. Distance to 1 random walker
   b. Reward of current state
3. Clone the walkers → Balance the Swarm

Random perturbation

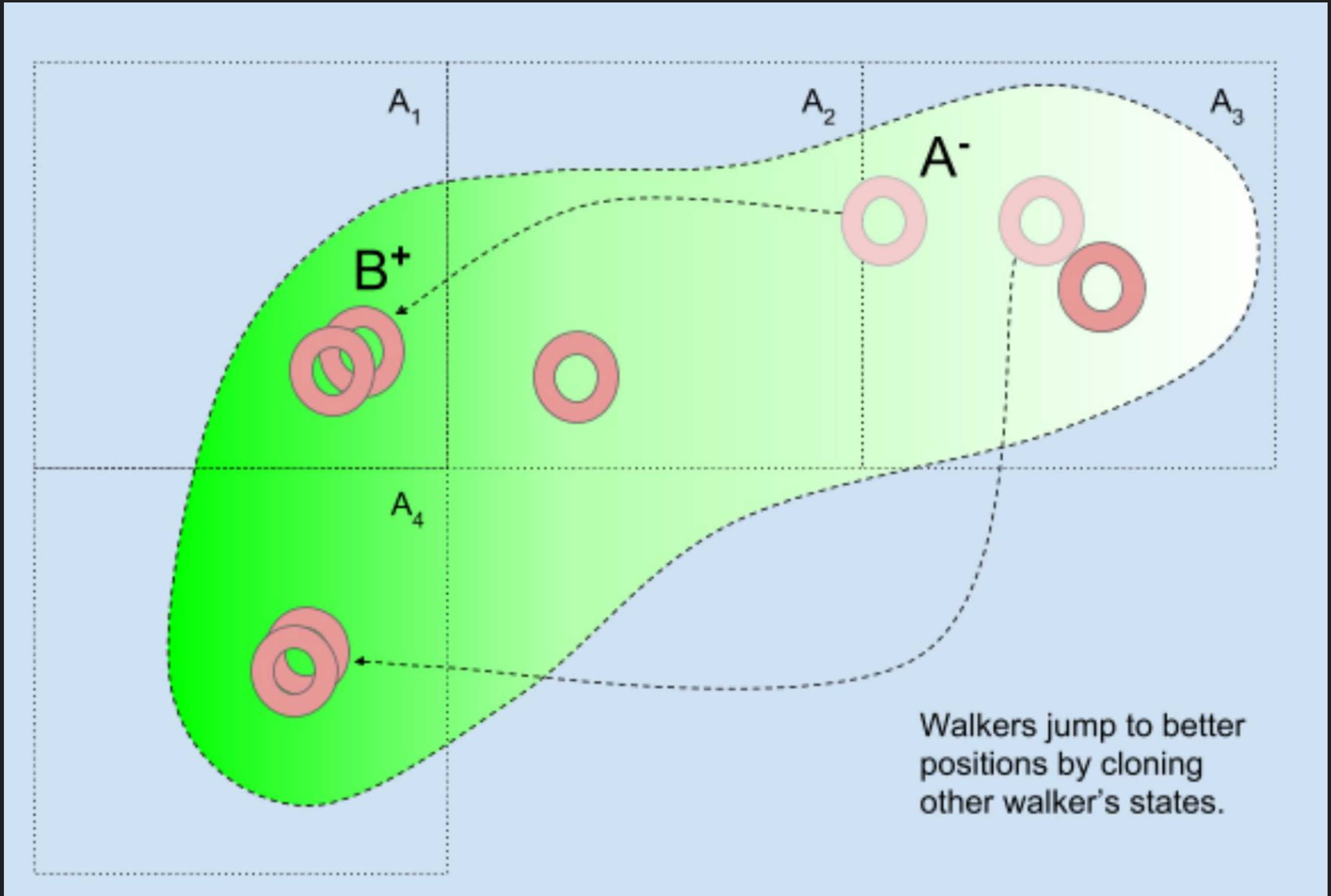


Walkers & Reward density

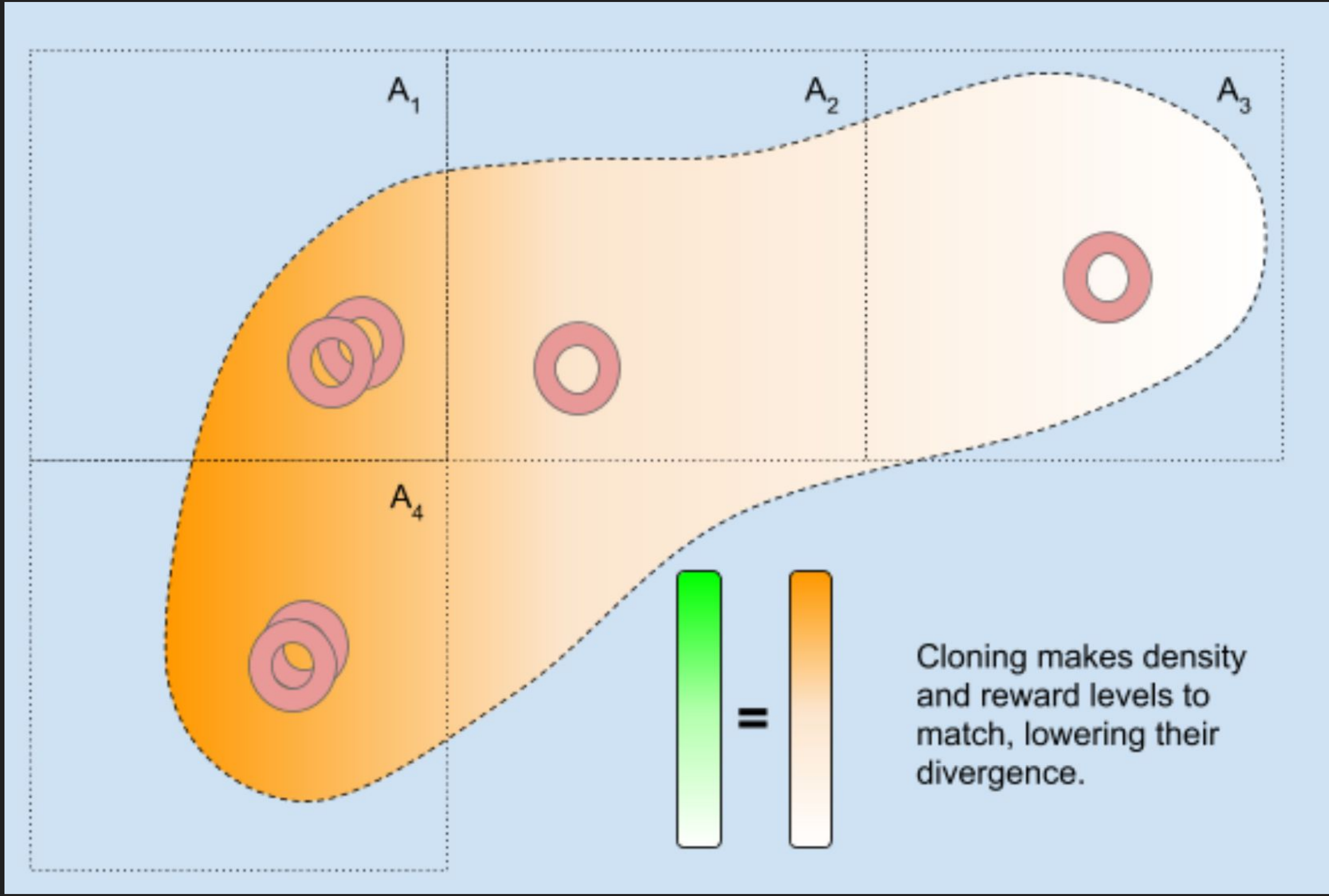




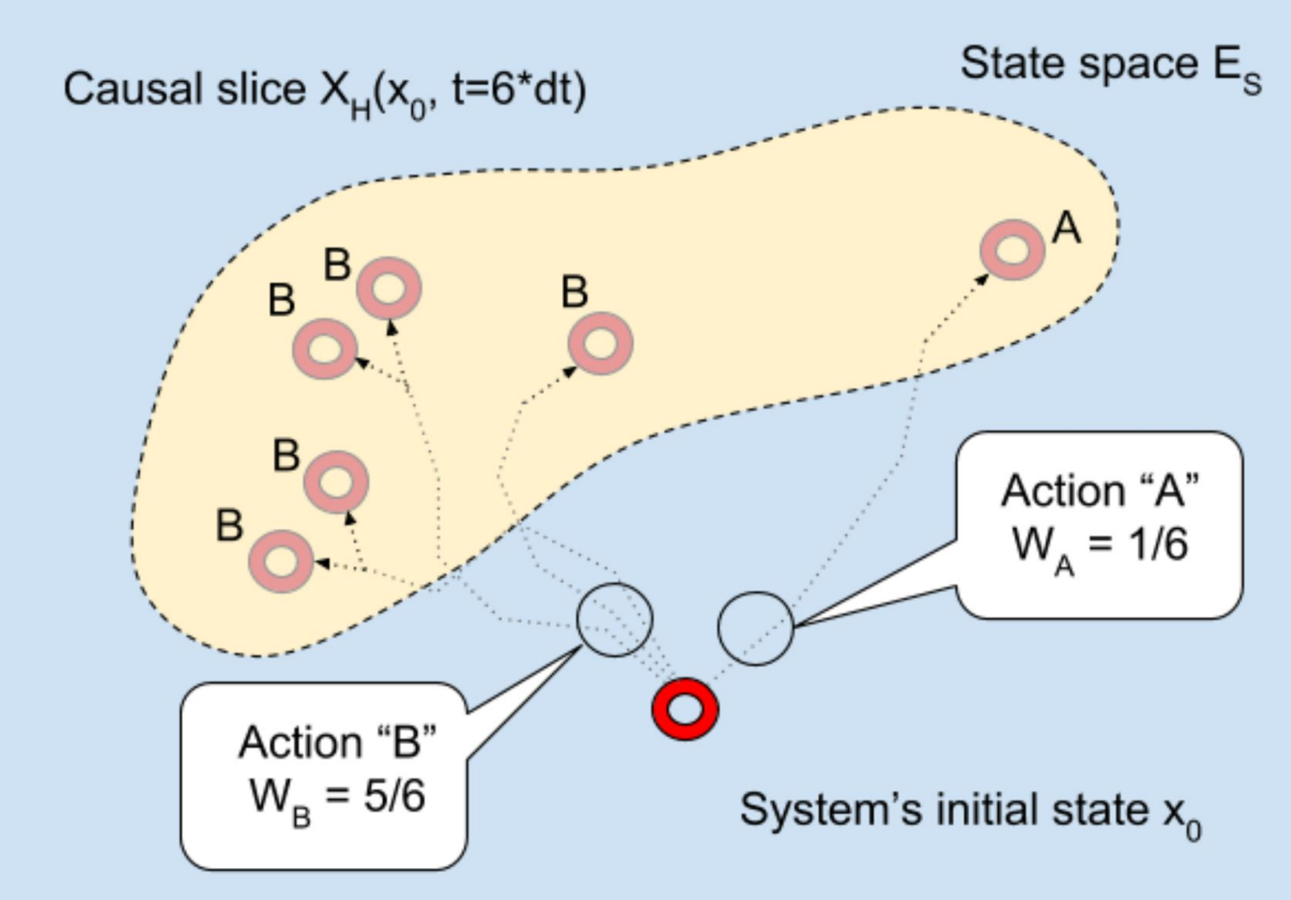
Cloning Process



Cloning balances both densities



Choose the action that most walkers share



RL is training a DNN model

- ML without labels → Environment
- Sample the environment
- Dataset of games → Map states to scores
- Predict good actions

Which Envs are compromised?

- Atari games → Solved 32 Games!
- Sega games → Good performance
- dm_control → x1000+ with tricks
- I hope soon in DoTA 2 & challenging environments

If you run it on your laptop in 50 games

- Pwns planning SoTA
- Cheaper than a human (No Pitfall)
- 17+ games with max scores (1M Bug)
- Beats human record → 56.36% games

RL as a supervised task

- Train autoencoder with a SW
- Generate 1M Games and overfit on them
- Use a GAN to mimic a fractal
- Use FMC to calculate Q-vals/Advantages
- Trained model as a prior

Give love to papers!

- Reproducing world models
- Playing Atari from demonstrations (OpenAI)
- Playing Atari from YouTube Videos (Deepmind)
- RUDDER

Efficiency on MsPacman

An example run:

- 128 walkers
- 14.20 samples / action
- Scored 27971 points
- Game len 6892
- 97894 samples
- 1min 38s. Runtime
- 70.34 fps

SW vs. UCT & p-IW (Assuming 2 x M4.16xlarge)

	UCT 150k	p-IW 150k	p-IW 0.5s	p-IW 32s
Score	**x1.25**	**x0.91**	**x1.85**	**x1.21**
Sampling Efficiency	**x1260**	**x1260**	**x1848**	**x29581**

When UCT(AlphaZero) finishes ⅔ of its first step,

SW has already beaten by 25% its final score

Improving Alphazero

- Change UTC for SW → sample x1000 + faster
- Stones as reward → SW jumps local optima
- Embedding of conv. layers for distance
- Use FMC to get better Q-values
- Heuristics only valid in Go

SW: Presenting an unfair benchmark

- A fair benchmark requires sampling 1M score at 150k samples / step

- 10 min play: 12000 steps - One step: 400 µs
- 1 core game: 4.8s x 150k x 50 rounds -> 416 days
- Ideal M4.16xlarge: $3.20 / Hour →

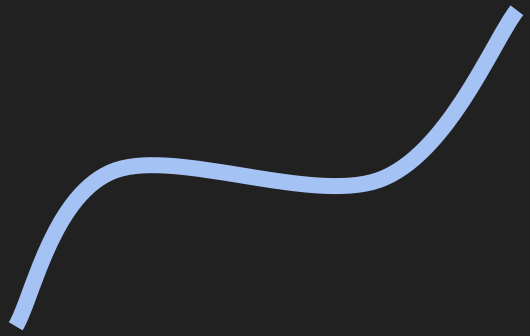
  500$ per game running 1 instance for 6.5 days

- **26,500$ on 53 games → Sponsors are welcome**

Counting Paths vs. Trees

- Samples / step: confusing → Tree of games

Traditional Planning



Swarm Wave