UNIVERSITY OF AGDER
FACULTY OF ENGINEERING AND SCIENCE

# CAIRL: High-Performance Reinforcement Learning Environment Suite

CAIRL

Per-Arne Andersen, PhD Candidate

# Outline

- Personal Introduction
- Motivation
- Reinforcement Learning Introduction
- State of the art
- Motivation
- CaiRL the proposal
- Results
- Interface and Tournament
- Conclusion and Future Work

CAIRL

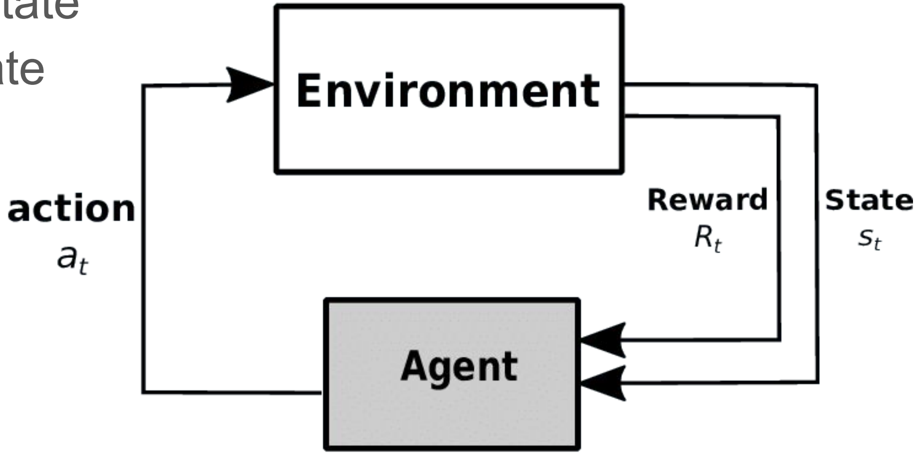
UNIVERSITY OF AGDER
FACULTY OF ENGINEERING AND SCIENCE

Per-Arne Andersen

- PhD Student
- University of Agder, Norway
- Reinforcement Learning
- Current Research:
    - (RTS Games)
    - *Exploration Methods*
    - *RL applied to Industry*
    - *Model Based RL*
    - (Climate) Efficient RL

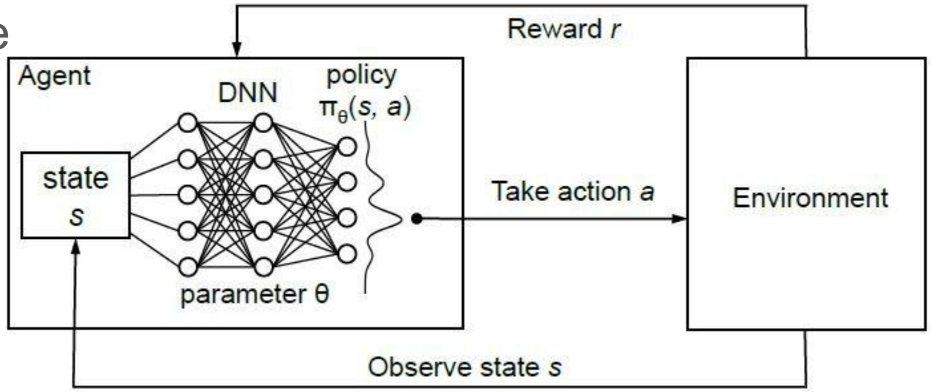
# Traditional Reinforcement Learning

- An agent (Algorithm)
- Makes decisions (actions) given a state
- Actions change the environment state
- Agent observe the change and is given a reward



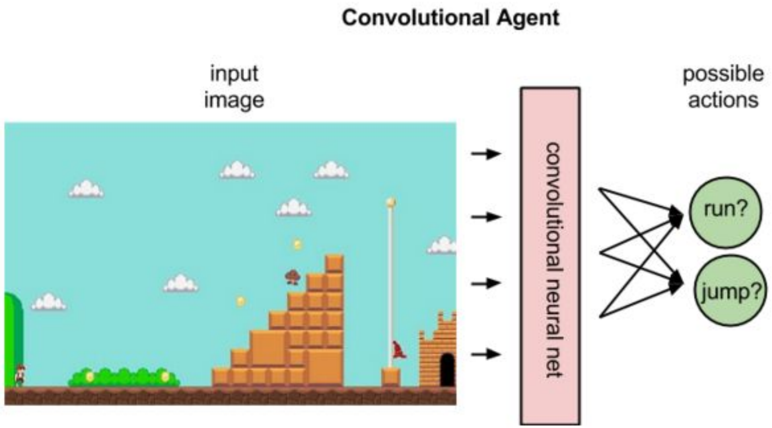
# Deep Reinforcement Learning

- An agent (Algorithm)
- Makes decisions (actions) given a state
- Actions change the environment state
- Agent observe the change and is given a reward



# Deep Reinforcement Learning

- Sequential Decision Making fits well with nature
    - Industry applications
    - Medical applications
    - Autonomous applications
    - Games
- But why focus on Games?
- Because:
    - They are cheap to test on compared to many industry applications
    - It is possible to optimize for efficiency
    - It is trivial to adapt the game problem/objective



# OpenAI Five


# OpenAI Five

OpenAI Five

https://openai.com/blog/openai-five/

OpenAI Five plays 180 years worth of games against itself every day, learning via self-play. It trains using a scaled-up version of Proximal Policy Optimization running on 256 GPUs and 128,000 CPU cores — a larger-scale version of the system we built to play the much-simpler solo variant of the game last year. Using a separate LSTM for each hero and no human data, it learns recognizable strategies. This indicates that reinforcement learning can yield long-term planning with large but achievable scale — without fundamental advances, contrary to our own expectations upon starting the project.

# OpenAI Five

- Costs ~25 000$ per day
- Not solved yet!

**Comparison chart**

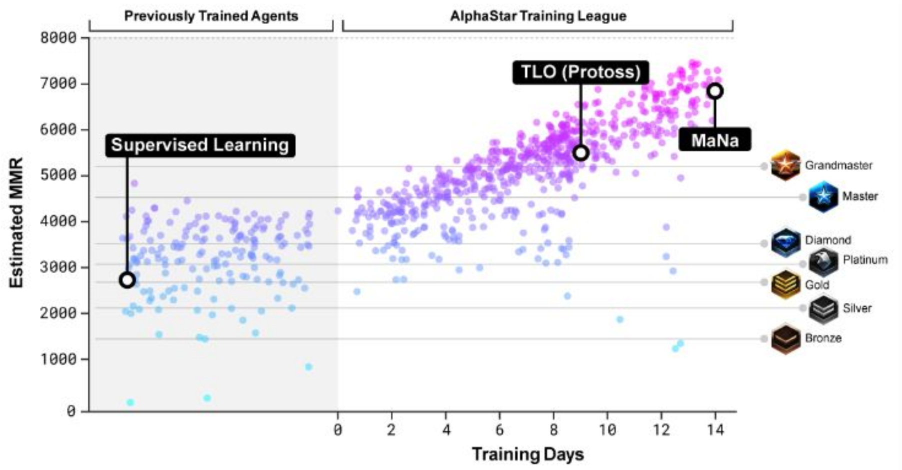
| | OpenAI 1v1 bot (2017) | OpenAI Five (2018) |
|---|---|---|
| CPUs | 60,000 CPU cores on Microsoft Azure | 128,000 pre-emptible CPU cores on the Google Cloud Platform (GCP) |
| GPUs | 256 K80 GPUs on Azure | 256 P100 GPUs on the GCP |
| Experience collected | ~300 years per day | ~180 years per day |
| Size of observation | ~3.3kB | ~36.8kB |
| Observations per second of gameplay | 10 | 7.5 |
| Batch size | 8,388,608 observations | 1,048,576 observations |
| Batches per minute | ~20 | ~60 |

# Alpha Star - Starcraft II


# Alpha Star

- 800 GPU’s????
- $12,976,128 to train
- https://medium.com/swlh/deepmin
t-at-what-cost-32891dd990e4

The models are then further trained using IMPALA and population-based training, plus some other tricks I’ll get to later. This is called the AlphaStar League. Within the population, each agent is given a slightly different reward function, some of which include rewards for exploiting other specific agents in the league. Each agent in the population is trained with 16 TPUv3s, which are estimated to be equivalent to about 50 GPUs each. The population-based training was run for 14 days.



# Running RL Experiments

- OpenAI Gym is the dominant toolkit for running RL experiments
  - https://gym.openai.com/
- Written in Python, and has a strict interface that all environments must adhere to
  - Inherit the class Env
  - Define a variable **observation_space** and **action_space**
  - Override the following functions (minimum)
    - step(action)
    - reset()
    - render()
- So why do we care to challenge this already dominant toolkit?

# Motivation

- OpenAI Gym is written in Python. Python is slow!
- **Example**: For Loops
- Imagine the performance difference of function calls!
- Lets calculate PI!

# Motivation - Calculating PI

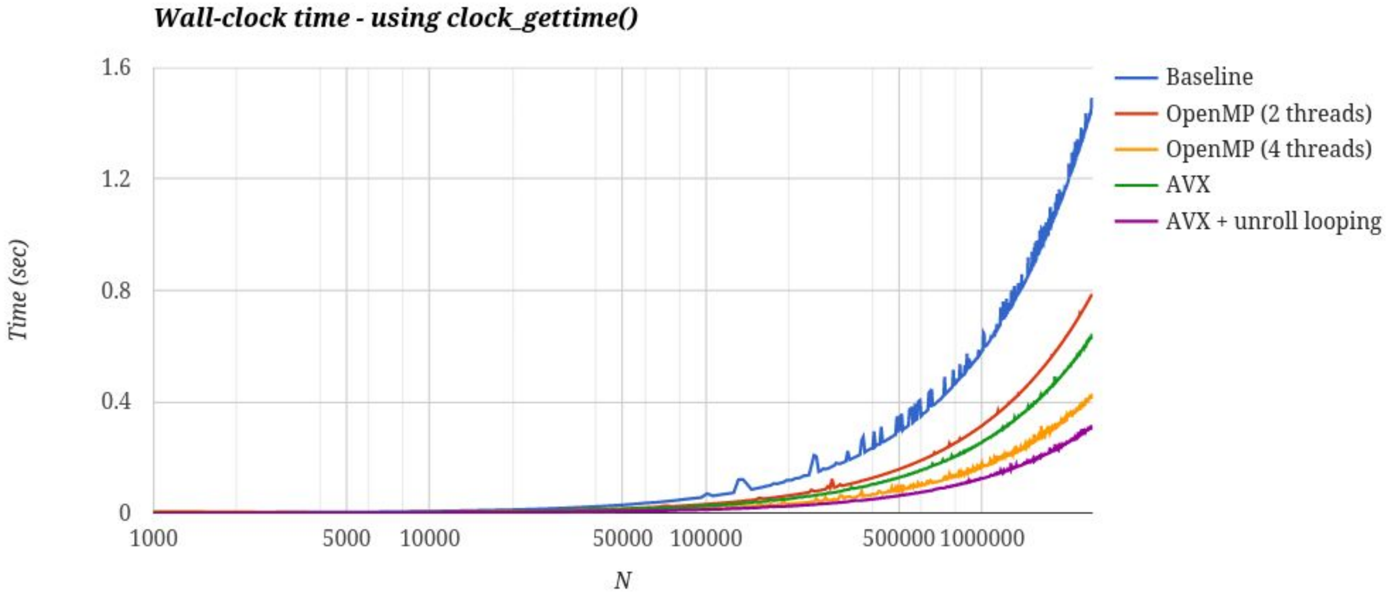
python version: https://gist.github.com/komasaru/290022bcc86f380d771e687ddc3ea5f7

c version:

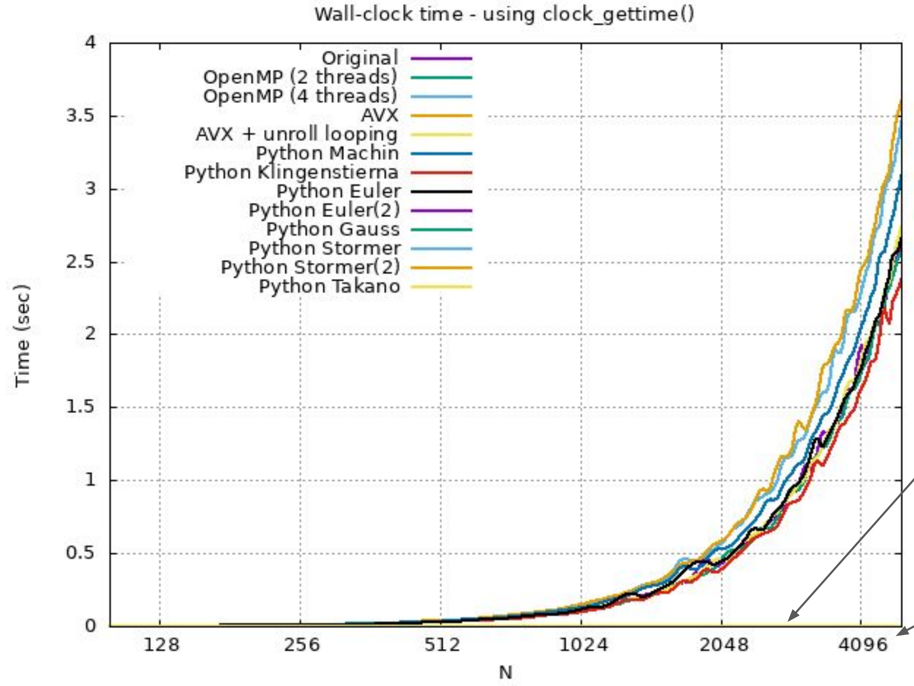
https://github.com/sysprog21/compute-pi

# Motivation - Calculating PI - C with SIMD

- 1 Million decimals takes less than a second
- Single Instruction Multiple Data



# Motivation - Calculating PI - Python vs C




C implementations are here

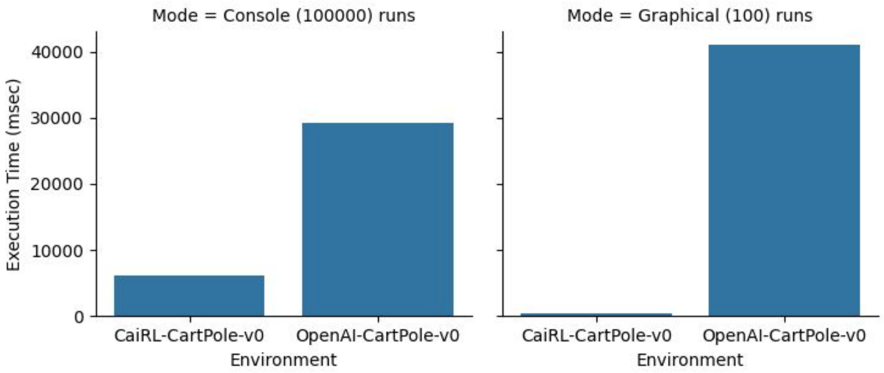
Note: only 4096 digits :(

# CaiRL: The proposal

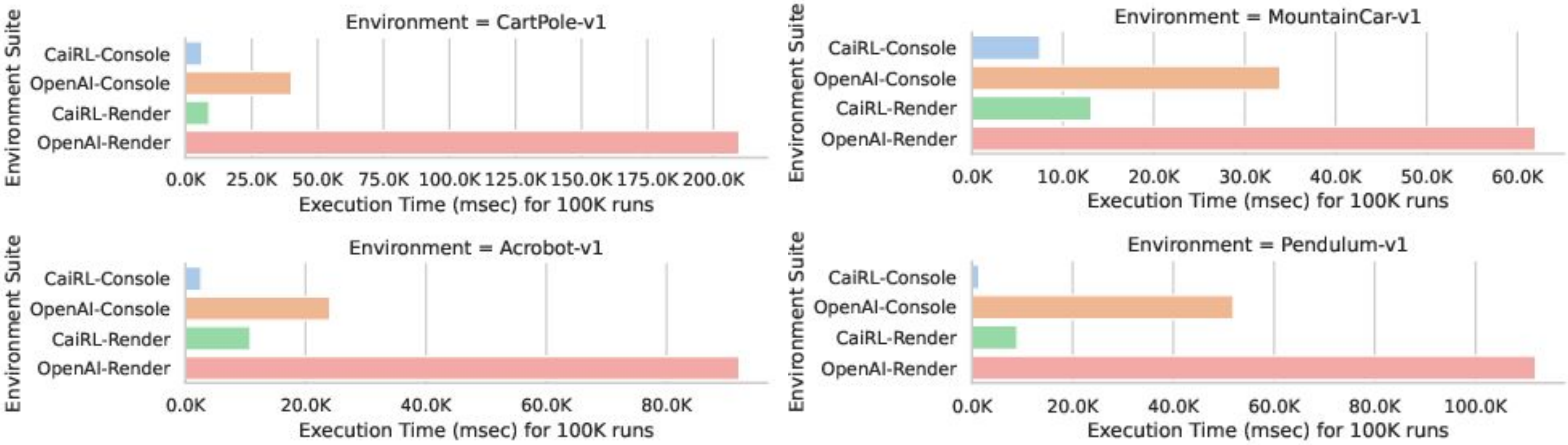
- We propose CaiRL for running games in **reinforcement learning experiments** in a **efficient manner**.
- We use **C++** as the primary backend for running experiments which enables to write:
  - **SIMD code**
  - Closer to the hardware
  - **Fewer CPU instructions** per function (on-average)
- CaiRL aims to reduce the **environmental footprint** of reinforcement learning.
  - Some focus on improving the environmental footprint of algorithms
  - NO literature on improving the experiment (TTBOOK)
- Add novel problems (games) to RL research

# CaiRL: Empirical Results

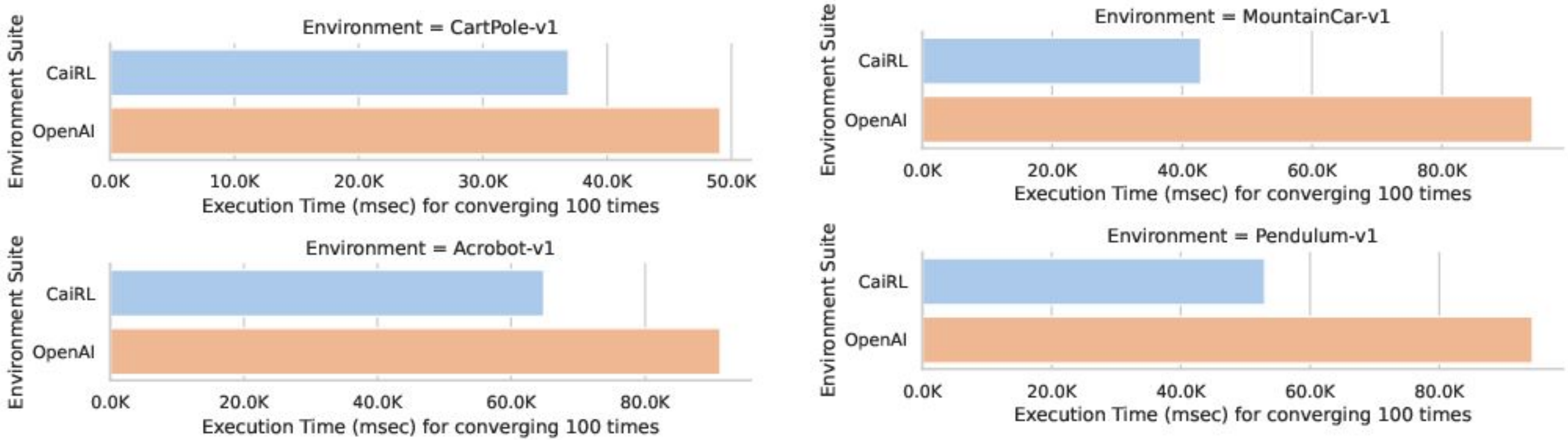
- Significantly faster in console and graphics
- *Recent work has pushed graphics even more with [https://blend2d.com](https://blend2d.com)*
- Focus on **Software Rendering**



# Environment Execution Times



# Algorithm Training Time



# CaiRL: Climate Footprint

- Console has ~**21** times less CO2/kg
- GUI has ~**147 570** times less CO2/kg
- Console use ~**21** times less power
- GUI use ~**148 006** times less power

https://arxiv.org/abs/2002.05651

TABLE II
THE TABLE DESCRIPTS THE TOTAL CARBON EMISSION VALUES AND POWER CONSUMPTION USED DURING THE EXPERIMENTS. THE CARBON EMISSION IS MEASURED IN CO2/KG AND POWER DRAW IS MEASURED IN MILLIWATT-HOUR (MWH).

| Measurement | Environment | CaiRL | Gym | Ratio |
|---|---|---|---|---|
| CO2/kg | Console | **0.000014** | 0.000067 | 20.8955 |
| CO2/kg | Graphical | **0.000051** | 0.075265 | 147578.431373 |
| Power (mWh) | Console | **0.000319** | 0.001483 | 21.5104 |
| Power (mWh) | Graphical | **0.001131** | 1.673959 | 148006.9849 |

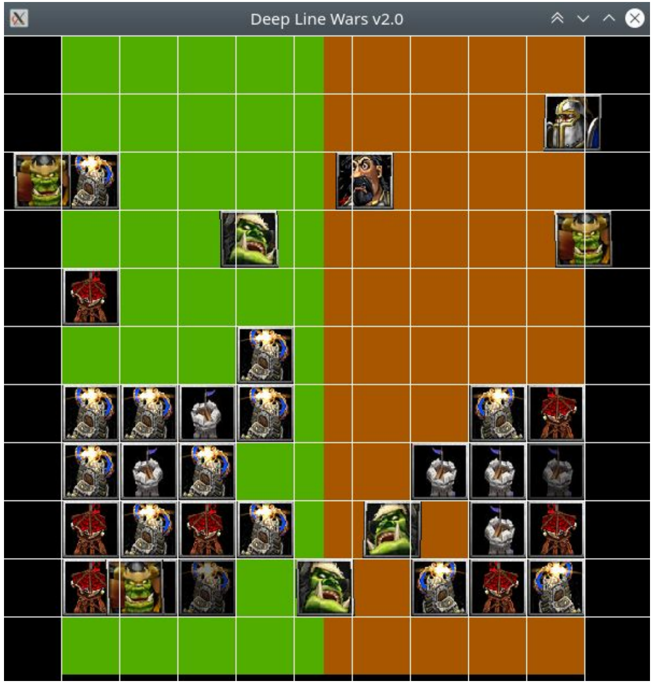
# CaiRL: The interface

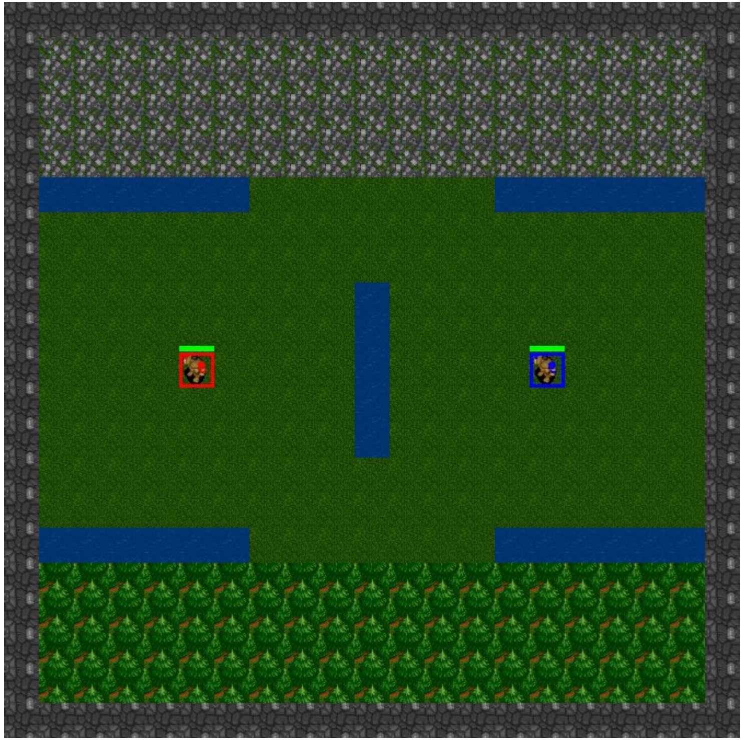
```
 # Running OpenAI Gym
import gym
env = gym.make("CartPole-v0")
for episode in range(10):
    env.reset()
    terminal = False
    while not terminal:
        state, reward, terminal, info = env.step(env.action_space.sample())
        # Do Reinforcement Learning Stuff here ...

# Running CaiRL Environment Suite
import cairl.gym
env = cairl.gym.make("CartPole-v0")
for episode in range(10):
    env.reset()
    terminal = False
    while not terminal:
        state, reward, terminal, info = env.step(env.action_space.sample())
        # Do Reinforcement Learning Stuff here ...
```

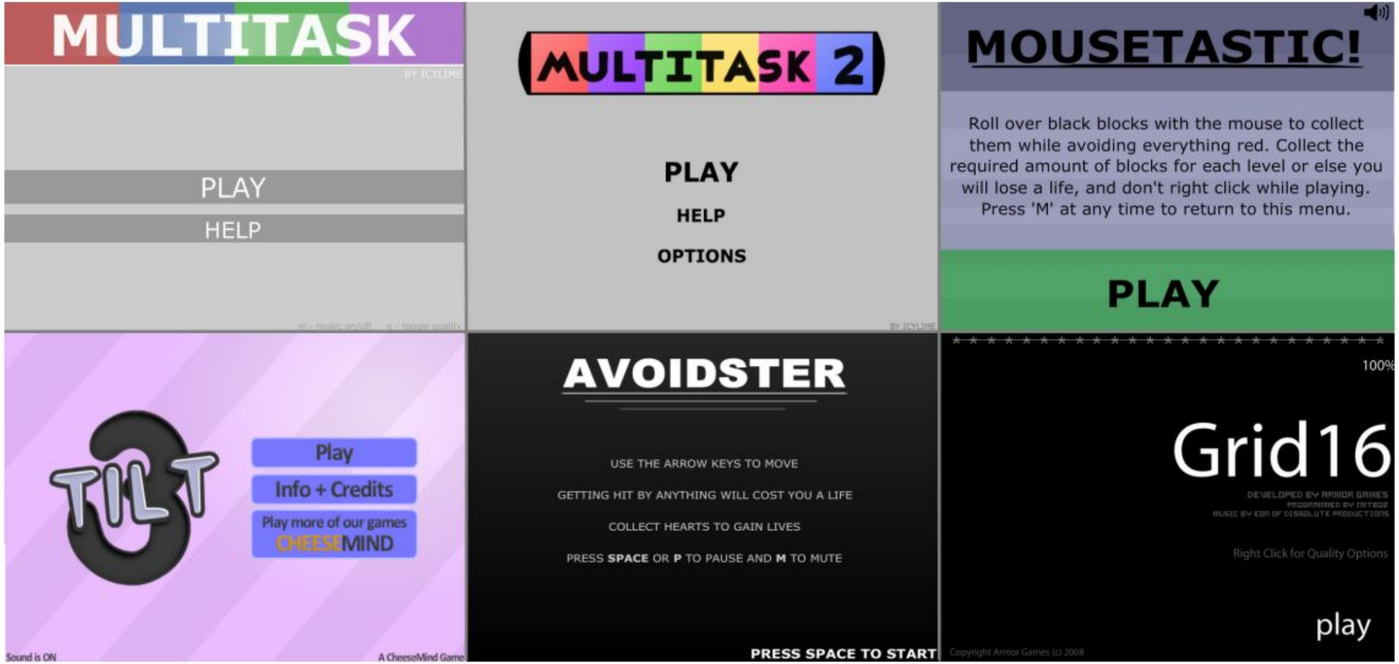
The only difference

# CaiRL: New Games



# CaiRL: New Games

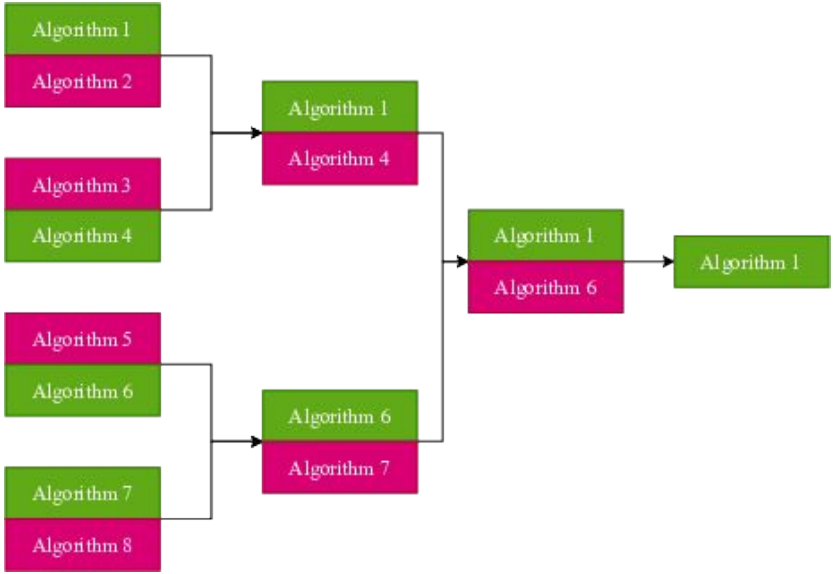


CAIRL

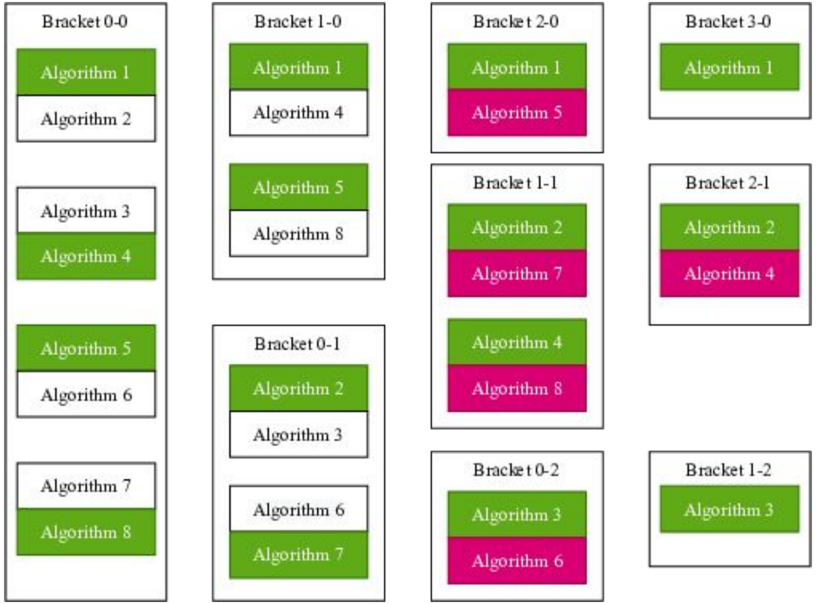
UNIVERSITY OF AGDER
FACULTY OF ENGINEERING AND SCIENCE

# CaiRL: Tournaments

Single Elimination Tournament




Swiss Tournament




Per-Arne Andersen, PhD Candidate

# CaiRL: Conclusion

**Cairl**:

- Improves CPU cycle efficiency in RL experiments
  - Reduces CO2 footprint
  - Reduce wall clock time
- Similar to OpenAI Gym for compatibility
- Still in early stages. Needs adoption and feedback
- https://github.com/cair/rl

**Environment:**

- Only use high-level programming where really needed

# Future Work

- Built-in emission counters/estimation
- Add novel problems to the toolkit
- Build competition platform similar to OpenAI Gym Webpage

Thanks for Listening!