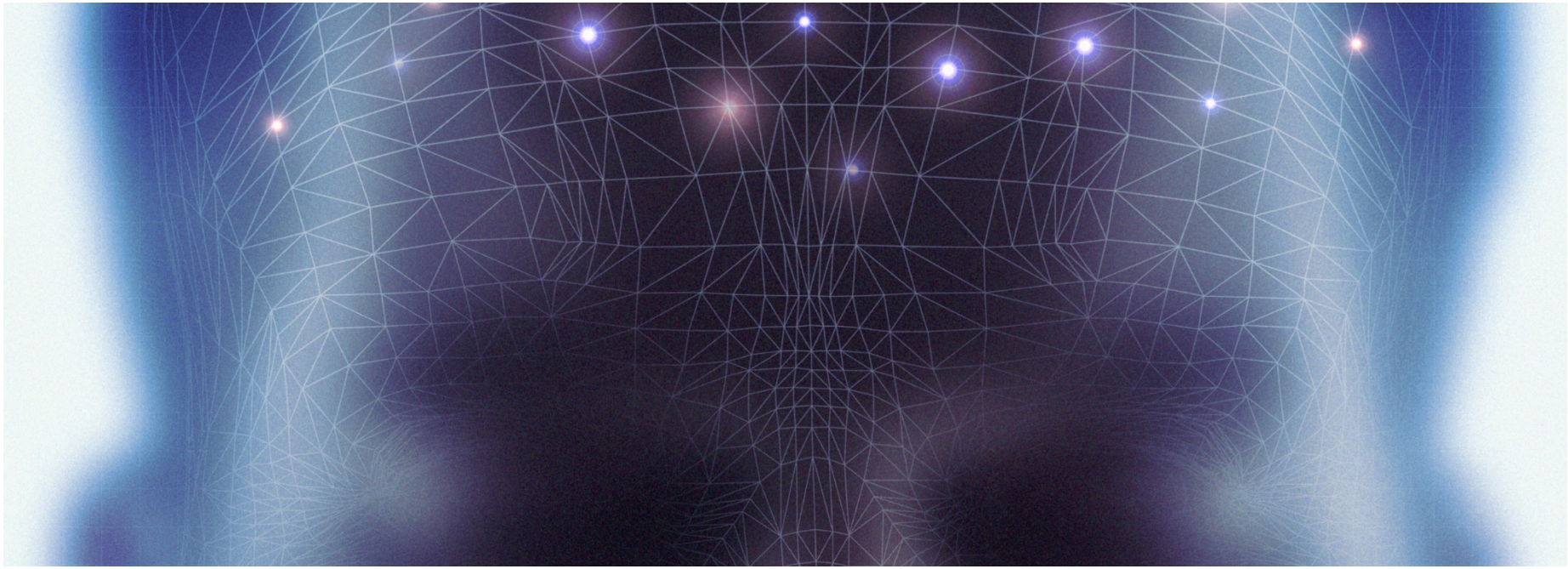
Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confedraziun svizra

Swiss Confederation

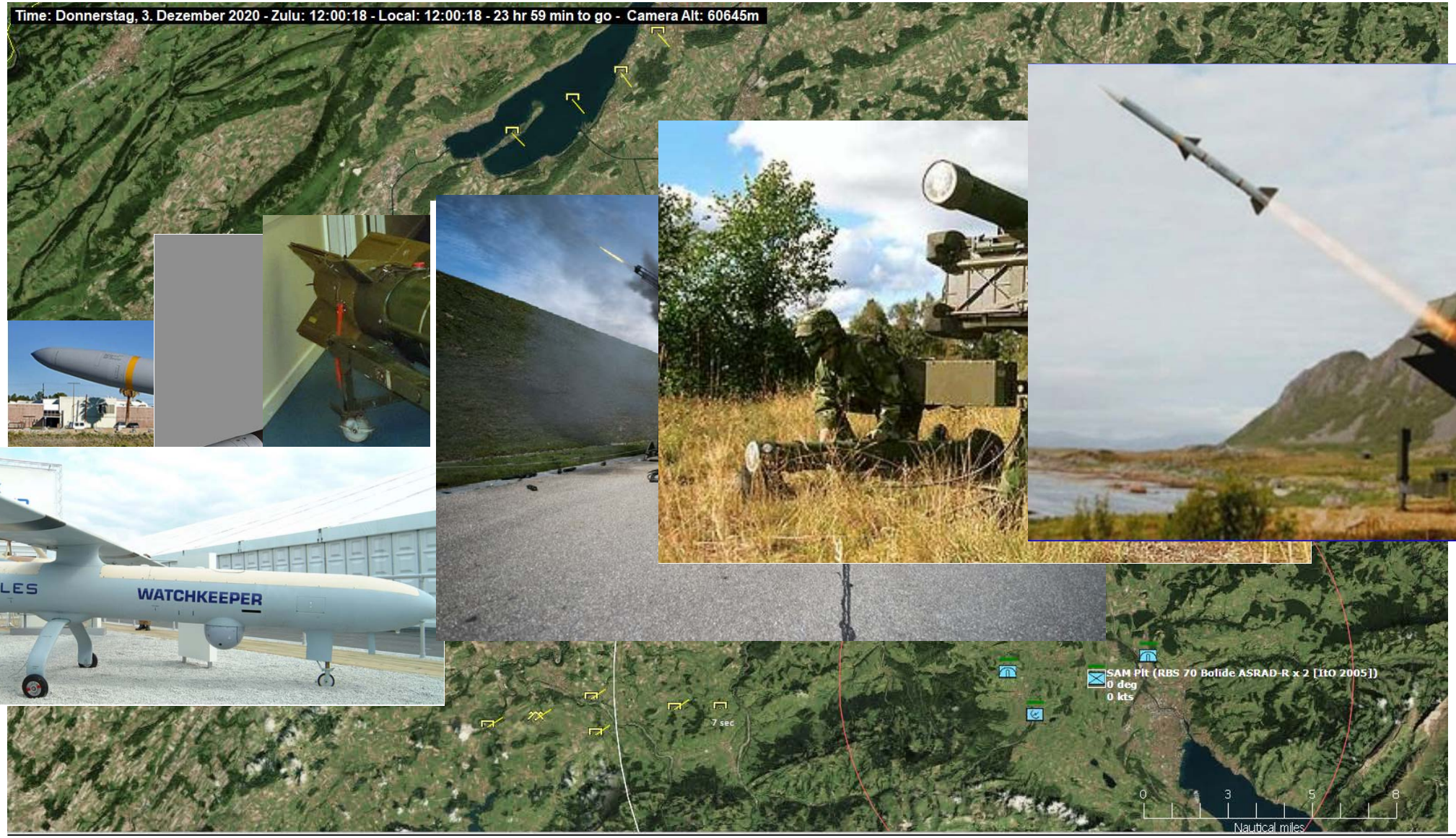
Federal Department of Defence,
Civil Protection and Sport DDPS
armasuisse
Science and Technology

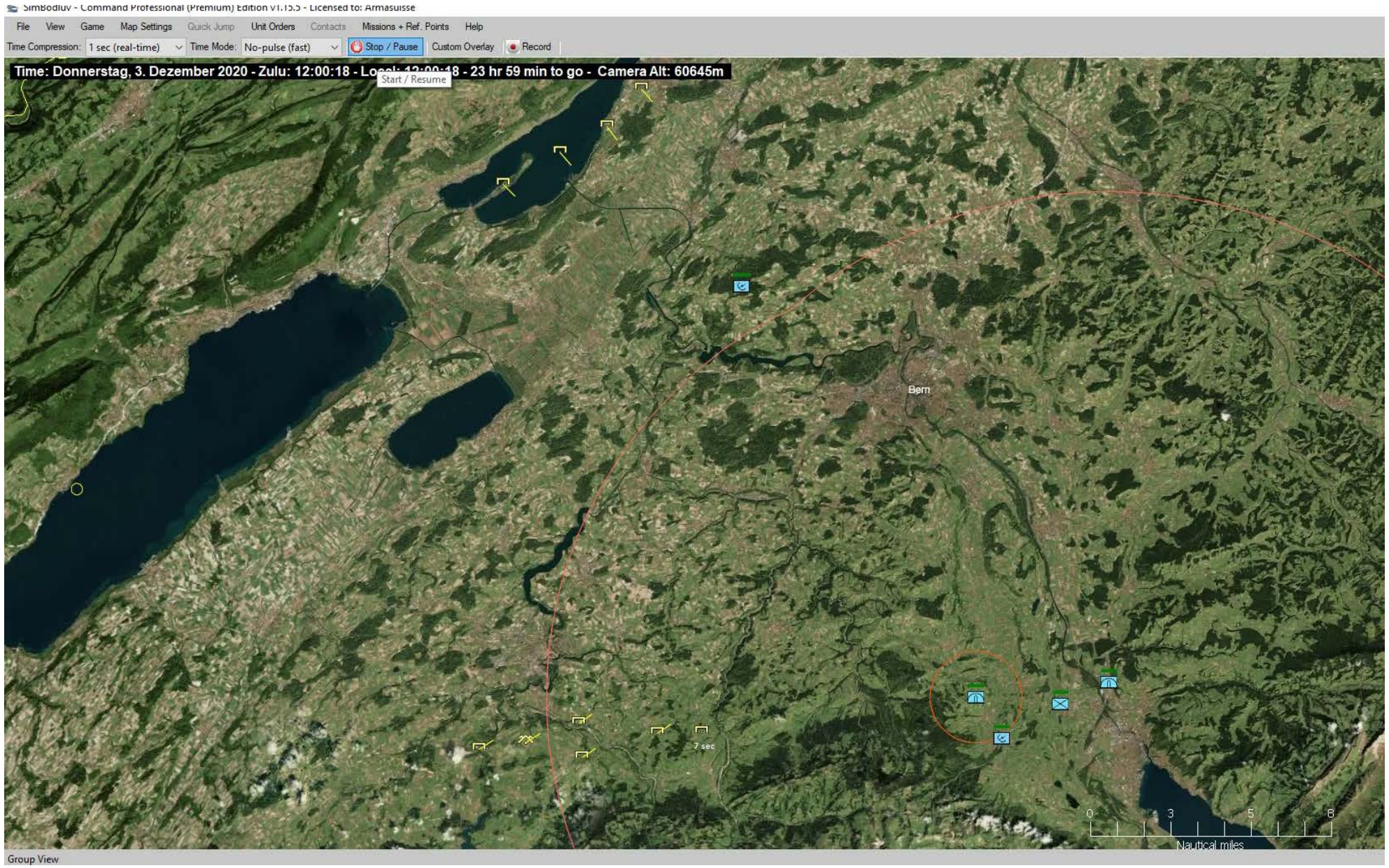


Deep Self-optimizing Artificial Intelligence for Tactical Analysis, Training and Optimization

Matthias Sommer, Oleg Szehr

Air Defense Scenario




SimBoddluv - Command Professional (Premium) Edition v1.15.5 - Licensed to: Armasuisse
File View Game Map Settings Quick Jump Unit Orders Contacts Missions + Ref. Points Help
Time Compression: 1 sec (real-time) Time Mode: No-pulse (fast) Stop / Pause Custom Overlay Record
Time: Donnerstag, 3. Dezember 2020 - Zulu: 12:00:18 - Local: 12:00:18 - 23 hr 59 min to go - Camera Alt: 60645m
Start / Resume
Bern
7 sec
0 3 5 8
Nautical miles
Group View

Wargaming: Human - Human

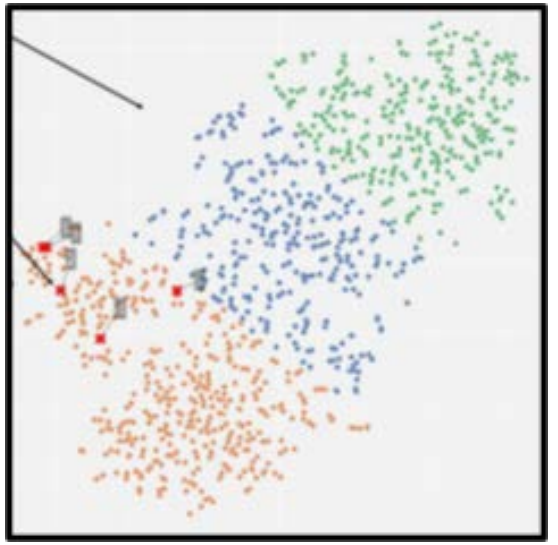
- Training tactical-strategical thinking
- Conflict analysis
- Mission planning
- Validation of known Courses of Action

- Think inside the box
- Slow
- Restricted reproducibility

Red Teaming: Human - AI

- Training tactical-strategical thinking
- Conflict analysis
- Mission planning and optimization

- Diverse and new Courses of Action (Think outside of the box)
- Slow



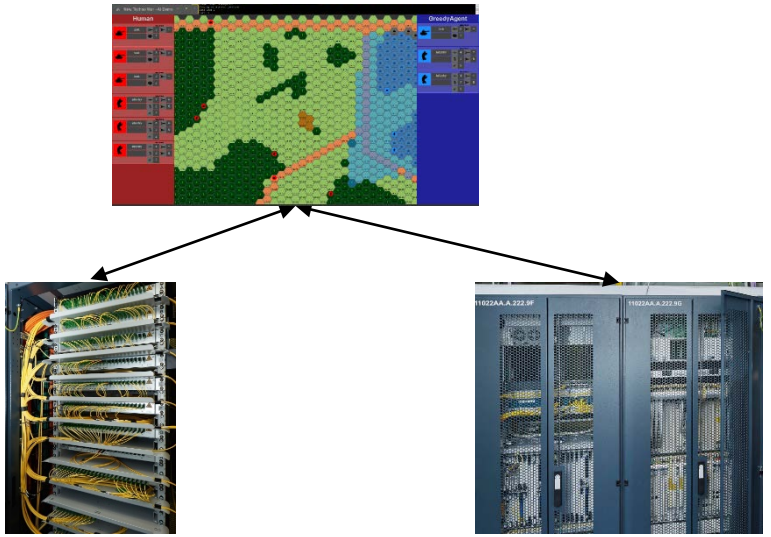
Data Farming: AI - AI

- Concept development and experimentation (CD&E)
- Armed Forces development
- Conflict analysis
- Procurement

- Diverse and new Courses of Action (Think outside of the box)
- Fast
- Solution space can be scanned extensively

Requirements for autonomous wargaming: Why AI?

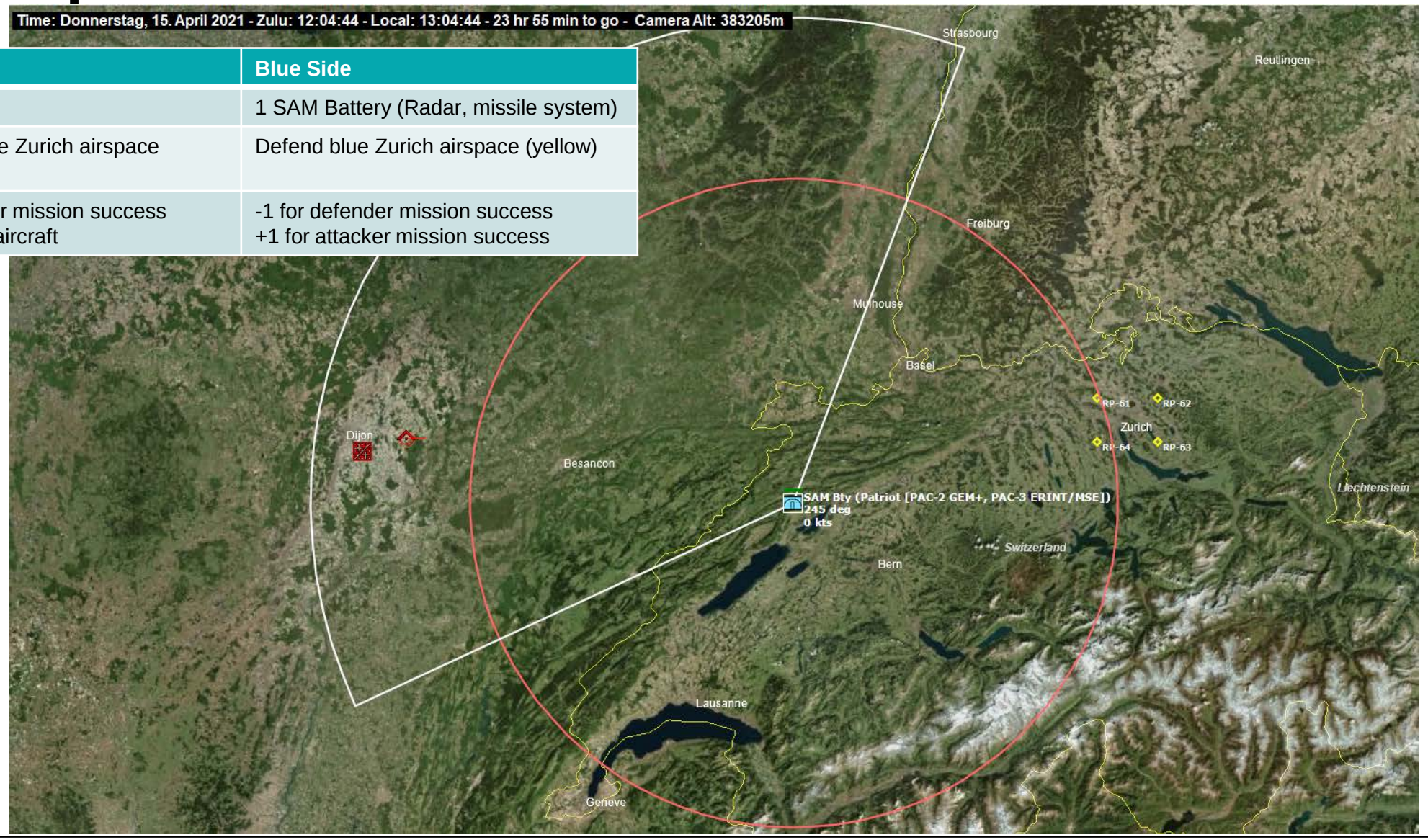
	OR / Scripting / Search	AI
"Think outside the box"	Concepts from programmers	Potential for "creativity"
Strong game play	Exhaustive search not possible	Focus on relevant CoA
Generalization ability	Case-specific	Inherently transferable
Autonomous learning, self optimizing	Requires human experience	Learns tabula rasa



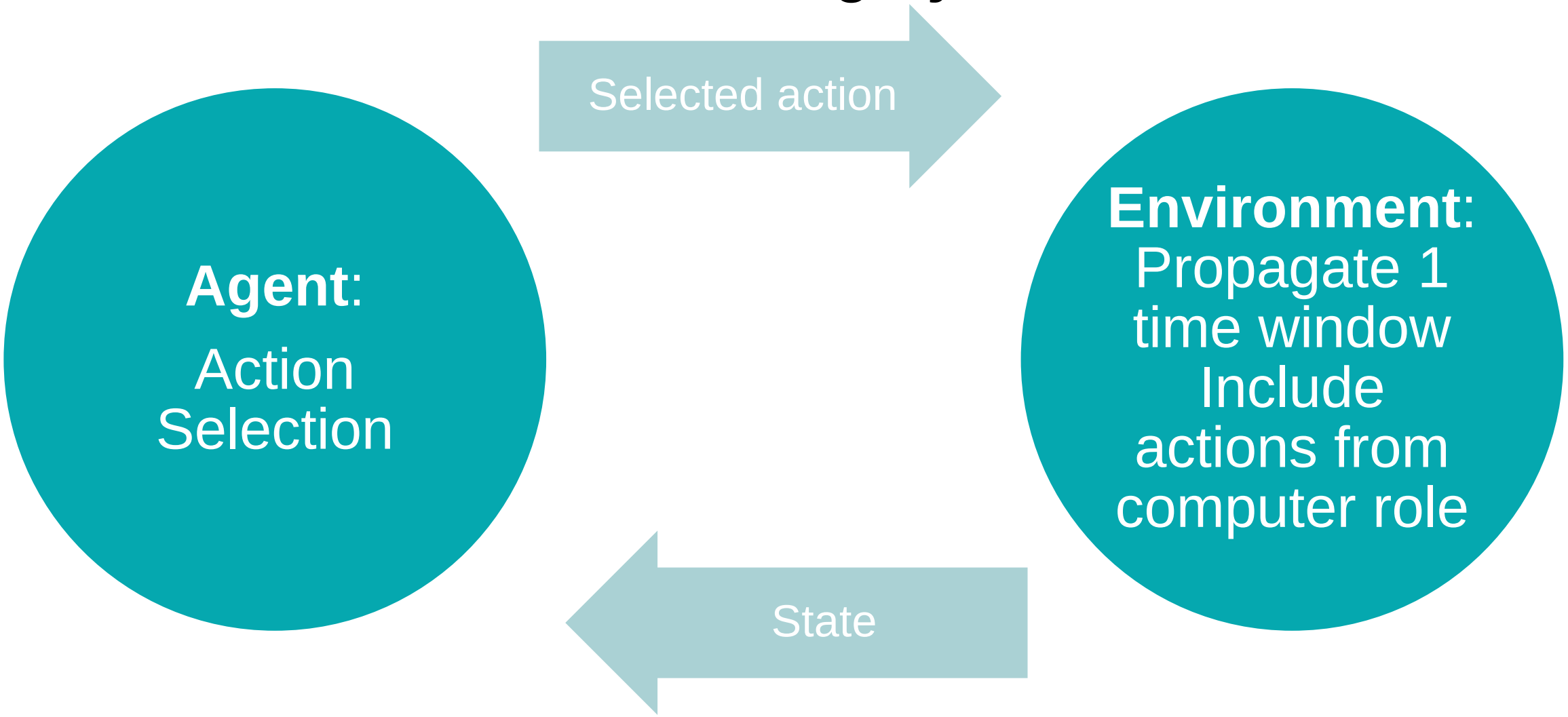
```
class MCTS():
    #Monte Carlo Tree Search

    def __init__(self, game, nnet, args):
        self.game = game
        self.nnet = nnet
        self.args = args
        self.Qsa = {}
        self.Nsa = {}
        self.Ns = {}
        self.Ps = {}

        self.Es = {}
        self.Vs = {}

    def getActionProb(self, canonicalBoard, temp=1):
        #print('canonical ' + str(canonicalBoard))
        for i in range(self.args['numMCTSSims']):
            self.search(canonicalBoard)
        #print(self.Nsa)
```

A Simplistic Air Defense Scenario

	Red Side	Blue Side
Assets	1 Fighter Jet	1 SAM Battery (Radar, missile system)
Mission	Penetrate blue Zurich airspace (yellow)	Defend blue Zurich airspace (yellow)
Red Scoring	+1 for attacker mission success -1 for loss of aircraft	-1 for defender mission success +1 for attacker mission success

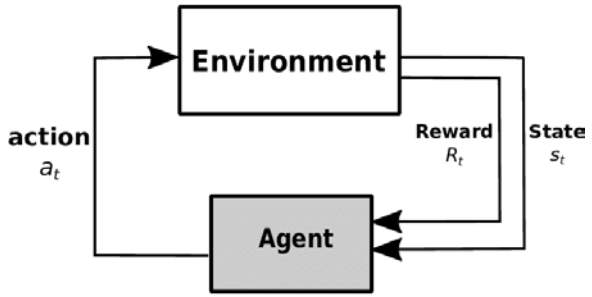


Discrete Action Scheduling Cycle



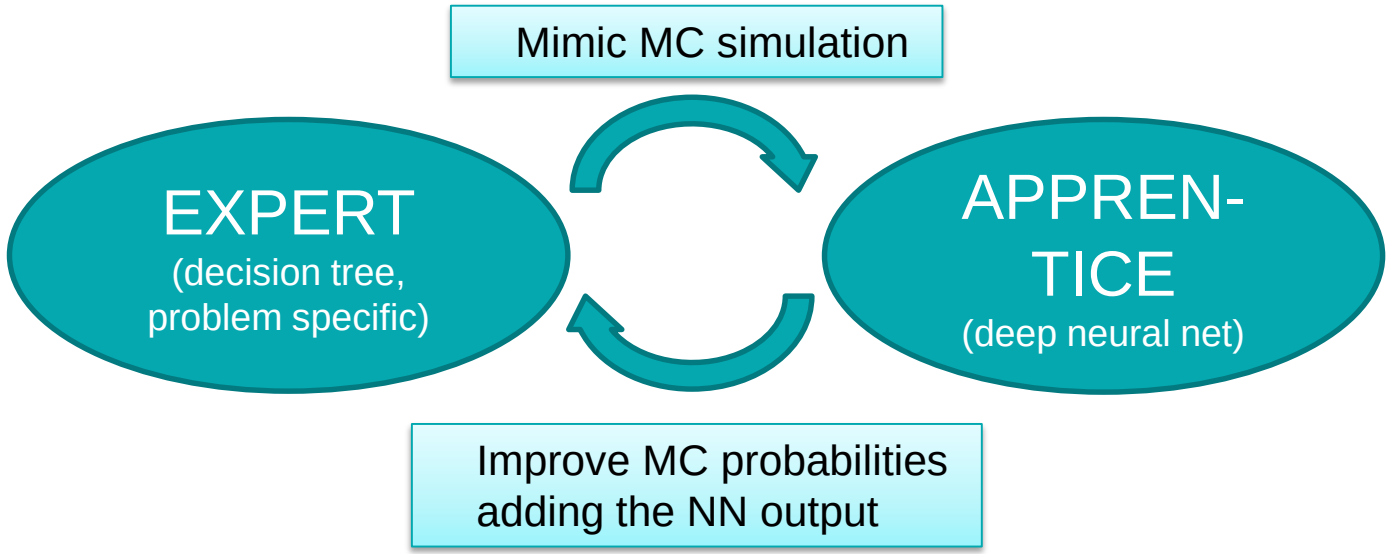
AI: Neural Monte Carlo Tree Search (nMCTS)

Reinforcement Learning (RL):
through trial, error and successive self-optimization



- Neural RL achieves super-human performance in Go, Chess, Hex …
- Compared to other RL algorithms, *nMCTS* learns stronger policies faster
- *nMCTS* combines techniques from
  - Monte Carlo (MC) simulation
  - game tree search
  - deep learning

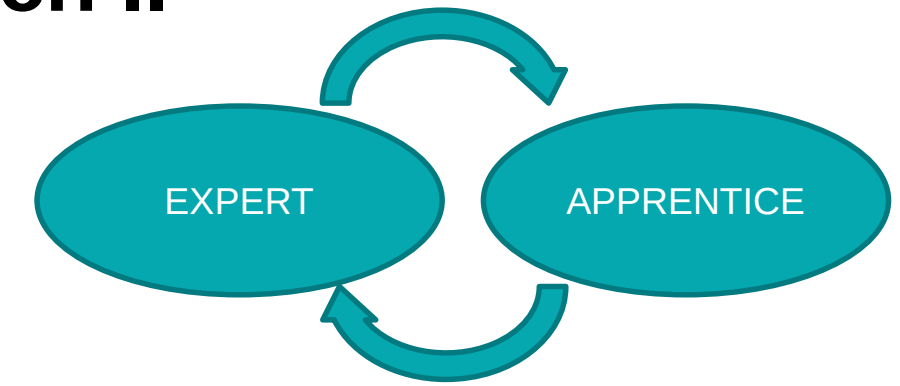
nMCTS : Two components, improve one another iteratively



EXPERT:
- Run MC simulation
- Record outcomes
  - update quality values of state-action pairs
  - Update MC probabilities
- Perform simulation again, update, …, ...

APPRENTICE:
- Train neural network to imitate the behavior of MC
  - Mimic action probabilities (train on tree policy targets)
  - Mimic values of states (train on rewards)

AI: Neural Monte Carlo Tree Search II



EXPERT (MCTS):

+ Provides accurate actions and valuation.
- Requires significant computational effort. Is slow.
- Evaluation of similar states requires full simulation ‘from scratch’

APPRENTICE (Neural Network):

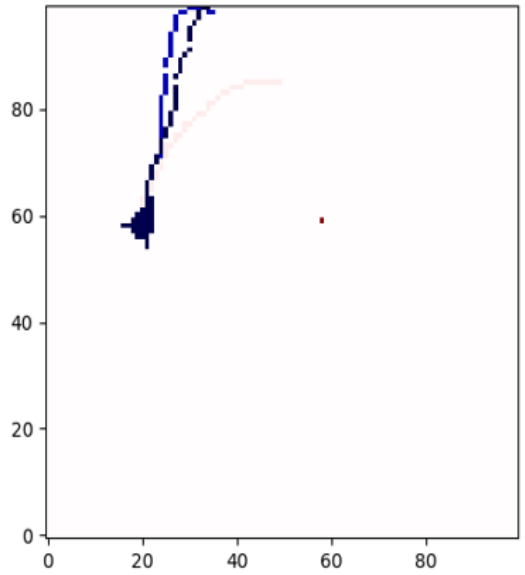
- Cannot learn actions/valuation by gradient descent in complex scenarios.
+ Once trained to imitate EXPERT: Fast access to EXPERT advice
+ Generalizes EXPERT advice to similar states.

Each training iteration consists of:

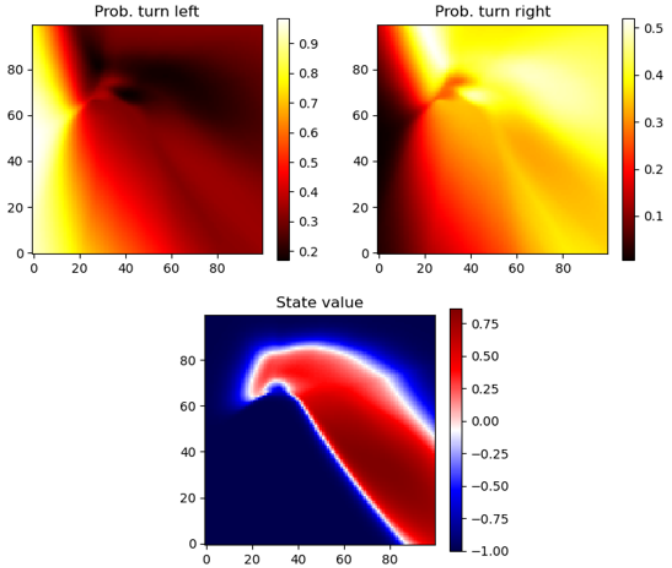
- Multiple neural network-guided MC simulations
- Re-training of the neural network and update of MC probabilities
- Validation to ensure improvement

Illustration of training process of AI components:

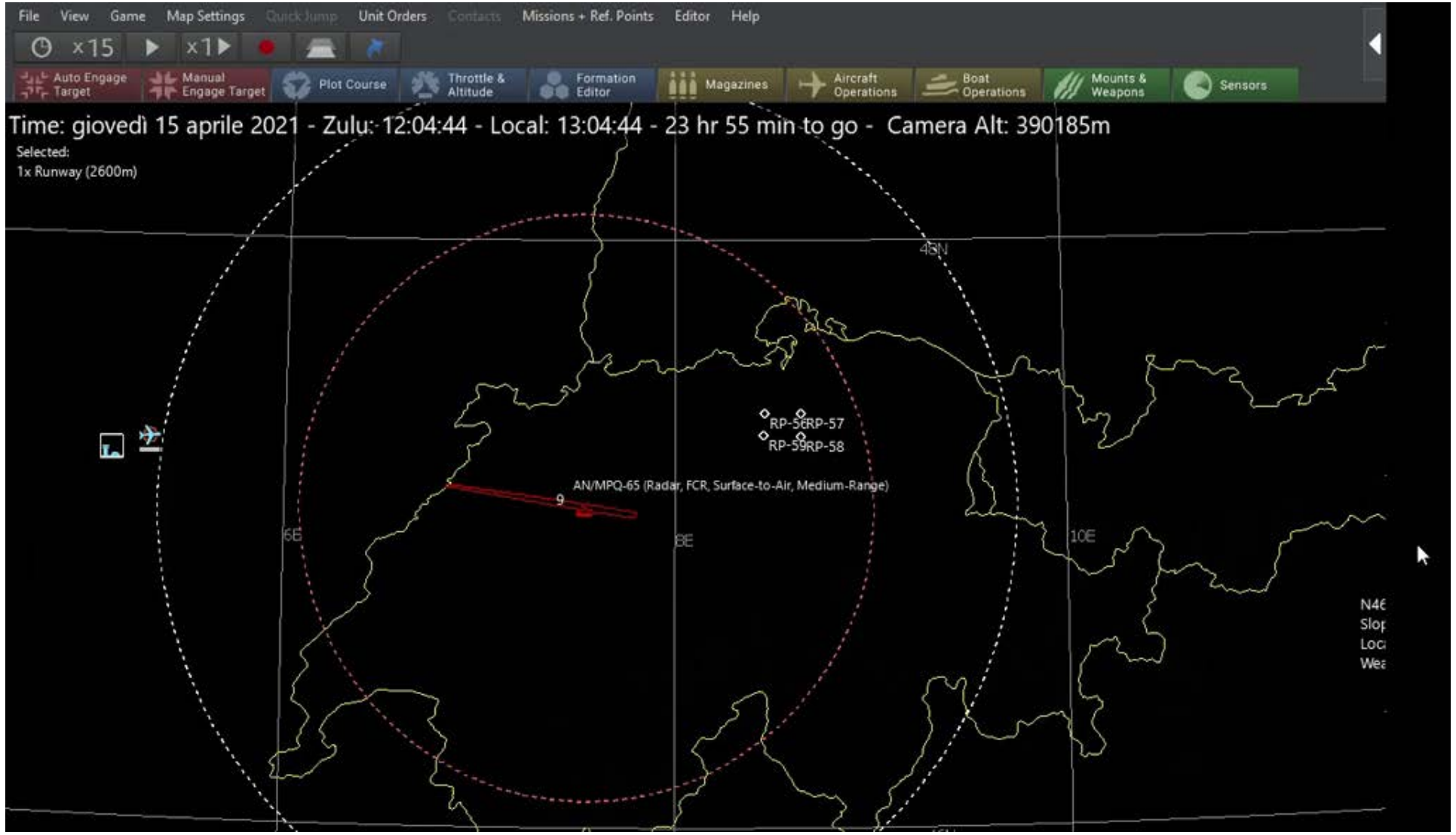
Expert



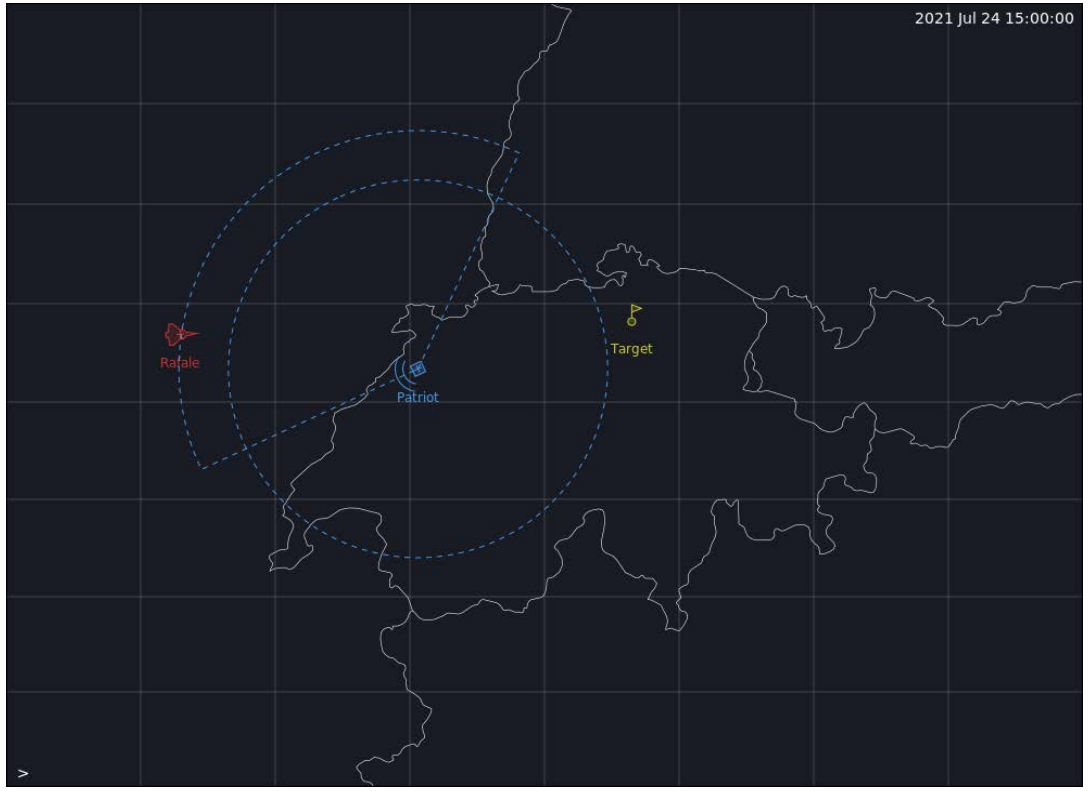
Apprentice

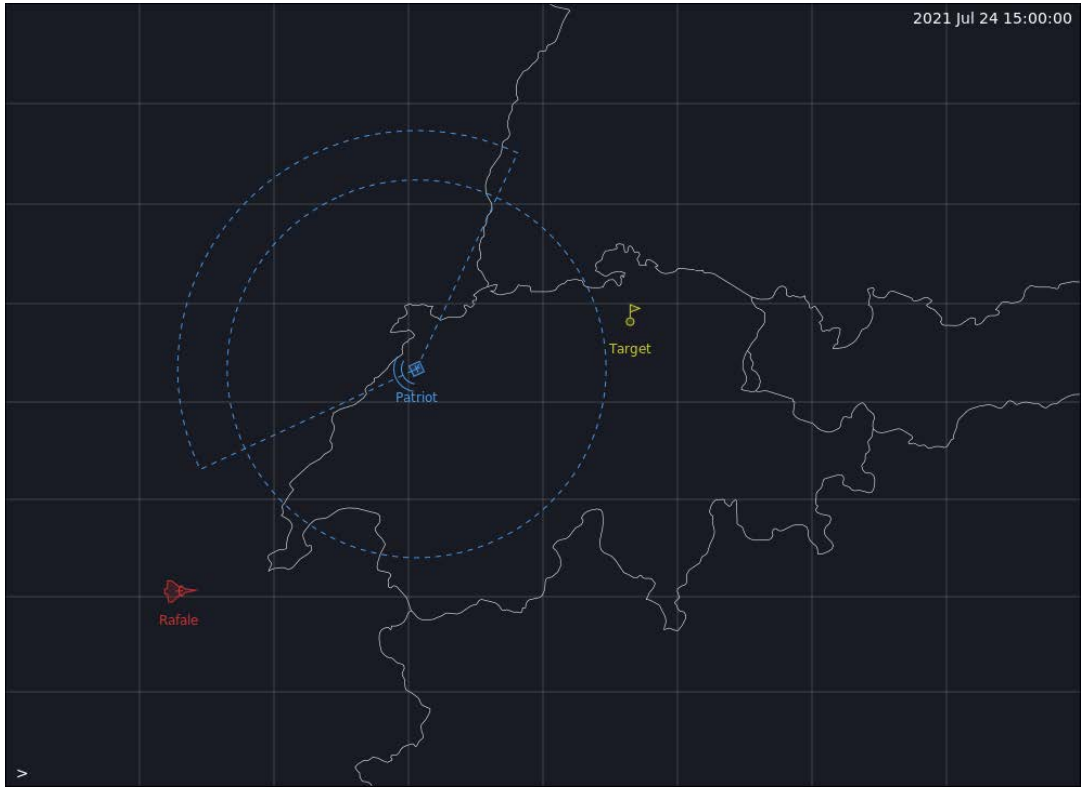


Result: Routing of trained agent



Generalization ability





Conclusions

Proof of Concept

- Successful integration of COTS game with PyTorch
- MCTS able to learn "intelligent behavior"

Bottleneck

- Training performance (API access)
- Workaround: Training with surrogate model

Next step

- More complex scenarios, e.g. invading aircraft can attack air defense battery

Take-Away

- AI has the potential to autonomously learn challenging winning strategies for wargames. What operation purpose can you think of?

Military Relevance: So What?

AI-based Simulation

- Autonomous decision-making
- Robust Analysis through Data Farming
- Tactical-strategic simulation engine