ELTE EÖTVÖS LORÁND UNIVERSITY

COLLECTIVE INTELLIGENCE

Zoltán Barta

PhD student, ELTE, AI Department

✉ dguqkf@inf.elte.hu

Tamás Takács

PhD student, ELTE, AI Department

✉ tamastheactual@inf.elte.hu

🌐 getlar.github.io

Collective Intelligence

Practice 1.

Introduction

Budapest, 11th January 2025

- [ ] Course Details
- [ ] Topic Overview
- [ ] Technical Details

Staff

Practice:

- Zoltán Barta
  - dquqkf@inf.elte.hu

- Tamás Takács
  - tamastheactual@inf.elte.hu (cjrnie@inf.elte.hu)
  - https://getlar.github.io/

Lecture:

- László Gulyás (lgulyas@inf.elte.hu)

Collective Intelligence

Practice 1.

Introduction

Budapest, 11th January 2025

Course Details

- ☐ Collective Intelligence
- ☐ Technical Details

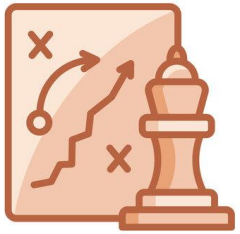
Practice Structure

Weeks 1-5:

- Collective Intelligence Overview
- Agent-based modeling (ABMs)
- NetLogo
- **1st Assignment + Defense**

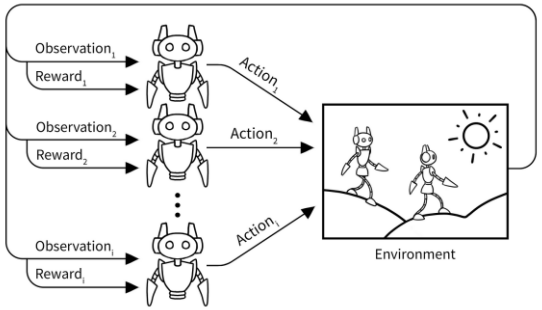
Weeks 6-7:

- Games
- Algorithmic Game Theory



Weeks 8-14:

- RL Introduction
- MARL Introduction + Algorithm Design
- **2nd Assignment + Defense**



Practice Structure

- Classes take place **every Monday from 8:00 AM to 10:00 AM** in Room 7.50 (Bosch Room).
  - The **first three practice sessions will also be available online** to accommodate travel and other difficulties with attending offline sessions.
  - Recordings of the practice sessions will not be provided.

- The **course syllabus** is available on Canvas.

- All **code and practice materials** will be shared on Canvas.

- The **assignments and their descriptions** will also be accessible on Canvas.

For any questions regarding the practice material or administrative matters, please message me or Zoltán Barta on Teams. Questions related to the lectures should be directed to Professor László Gulyás in email.

Assignment Grading

- The final grade is a **composite score** based on the three assignments from three different topics.
- The composite grade will serve as the final grade for the course. However, **to officially receive the grade from the assignments, you must also pass a small test on the lecture content at the end of the semester**. You will have two opportunities to take this test.
- The assignments are 'at-home' exercises that must be submitted on Canvas by the deadline. **No extensions will be granted!!!**
- The assignments **must be presented and defended in person** during the practice class.
- The **equation** for calculating the final grade is as follows:

Final Grade = 0.25 * First Assignment + 0.75 * Second Assignment

Evaluation

Each defense will take approximately 5-10 minutes. For the defense, you are required to:

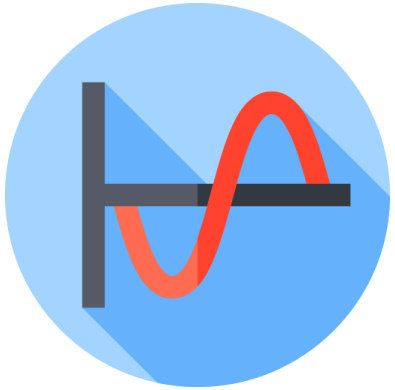
- **Prepare a mini-presentation** (3-4 slides) highlighting core implementation details, including an explanation of the main part of the code, experiments conducted, parameters used, and small GIFs showcasing the working code.
- **Be prepared to answer questions about the assignment** topic and your code.
- Submit your **solutions to Canvas** ON TIME!

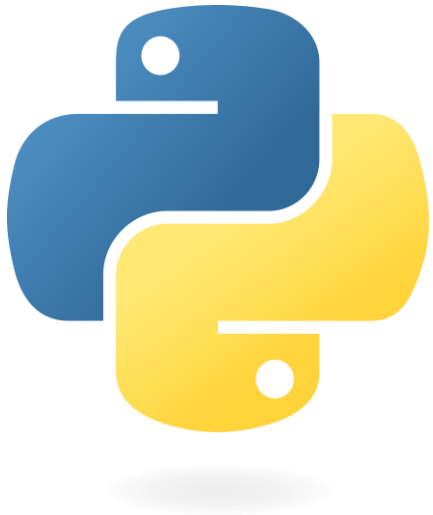
Grading will be based on the code, the presentation, and your responses to questions.

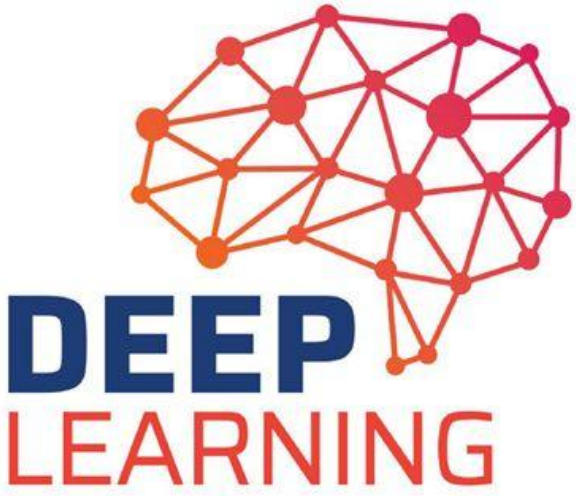
Prerequisites

Have a basic understanding of:

- Calculus
- Coding Skills (mostly Python)
- Deep Learning and DL models
- Reinforcement Learning







Description

At first glance, **"collective intelligence"** might seem like an abstract or overly niche topic with limited real-world application. Some may assume it lacks relevance to industry or the job market. However, the reality is quite the opposite.

The ability to describe and analyze collective behavior in systems where entities interact continuously is a highly transferable and valuable skill. In essence, **this is an abstract representation of our society.**

In our day-to-day lives, we operate under simple, second-nature rules. While these rules alone may not lead to complex outcomes, their combination with social interactions can result **in extraordinary emergent phenomena**. The interplay of individual actions and simple rules within a collective context often creates remarkable results.

Description

For example:

- **Driving theory** alone **is not sufficient** (e.g., KRESZ in Hungary).
- Despite its complexity, it is still functional.
- Highly **populous and dynamic** environment.
- Minor accidents are inevitable but do not disrupt the system as a whole.
- Interacting entities (drivers) follow **simple internal rules**.
- These interactions result in an intelligent system that:
  - **Adapts Gradually**: Some rules may lead to longer waits, prompting adjustments.
  - **Evolves Over Time**: More coordinated drivers navigate situations faster.
  - **Learns**: Drivers improve their ability to handle traffic jams in the future.

Description

For example:

- **Simple rules:**
  - Always move to the right.
  - Maintain a safe distance from the person in front of you.
  - Allow individuals in a rush to pass.
- These rules are **not explicitly enforced by elevator operators**, nor are there any sanctions for not following them. Yet, **most people adhere to them naturally.**
- These simple rules **lead to intelligent behavior**, resulting in an organized, safe, and rational system for everyone using the elevators.

Description

For example:

- **Simple rules:**
  - Adjust direction as needed.
  - Modify speed or acceleration.
  - Avoid collision with others.
- By following these simple rules, downhill skiing and snowboarding **becomes safe and enjoyable** for everyone. Most accidents occur off-piste, where these rules are less adhered to. (entity only interacts with the environment)
- **Decisions are made locally** (based on individual, micro-level rules), but the **collective outcome is global** (achieving the macro goal of a smooth, safe system).
- **Implicit communication** is present.

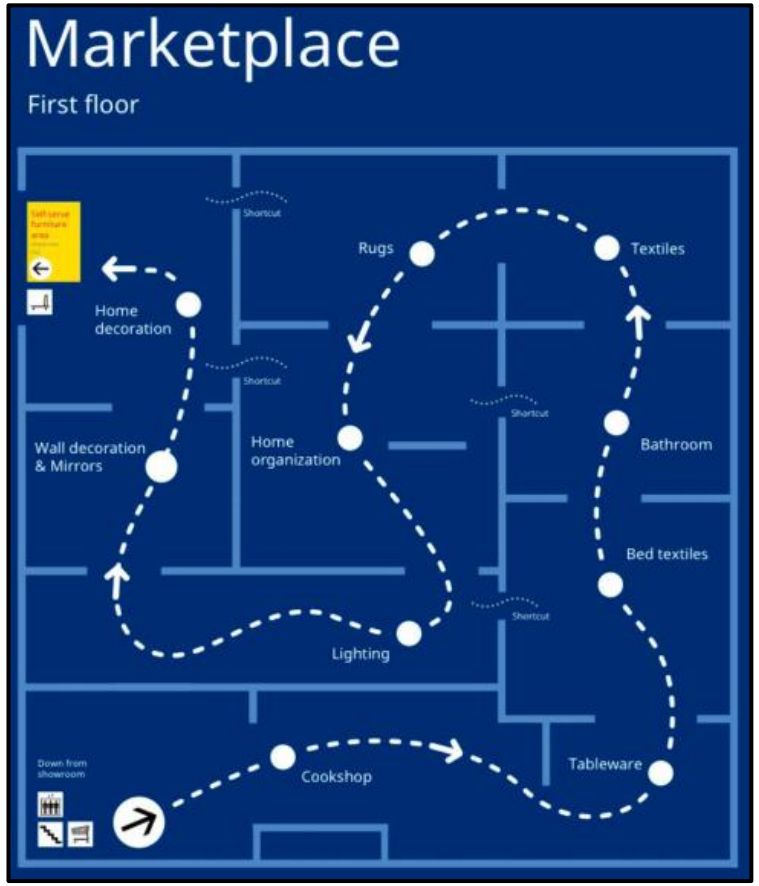
Description

Other interesting examples include:

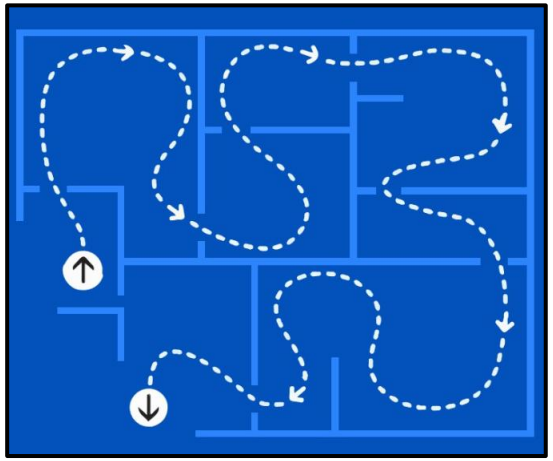
- Bird Flocking or Fish Schooling
- Crowd Movement in emergency situations
- Urban Traffic Light synchronization
- Bee Hive Communication

Description

Industry (Naïve Example)



- What **behaviors can I observe** from my customers in this layout?
- Based on this layout, **can I predict the behavior** of my customers?
- How can I **describe my customers' behavior** given this specific layout?
- **What type of behavior am I aiming** to encourage, and what behaviors am I currently observing?
- **How can I adjust the environment** to align the observed behaviors with the desired behaviors?



- What kind of behavior **can I anticipate with this layout?**

Description

The goal of this course is to **explore the concept of intelligence** in environments populated by multiple interacting entities. We will learn to understand, describe, analyze, and predict collective behavior using a bottom-up approach.

The course focuses on the **emergence of complex behavior in systems governed by simple rules**, emphasizing agent behavior, decentralized systems, game theory, and multi-agent reinforcement learning.

By the end of the course, students will:

- Understand the principles of emergent intelligence and describe it through agent-based modeling.
- Analyze the concepts of collective intelligence and rationality within the context of games.
- Predict and learn collective behaviors using reinforcement learning techniques.

Course Projects

- Assignment 3 consists of larger-scale projects designed with **MSc theses or TDK** (Scientific Students' Associations) in mind.
- These projects are **based on existing work created by past students** of the course in various topics.
- For Assignment 3, you will need to select **one project from the available options that interests you**.
- Your task will be **to expand upon the chosen project using a provided task description**. Past code and presentations will be made available as resources.
- **Only one new project idea is allowed per semester**, and acceptance rules are very strict. To propose a new project, you must submit a detailed pitch including a project plan and research direction.

Course Projects - Formation

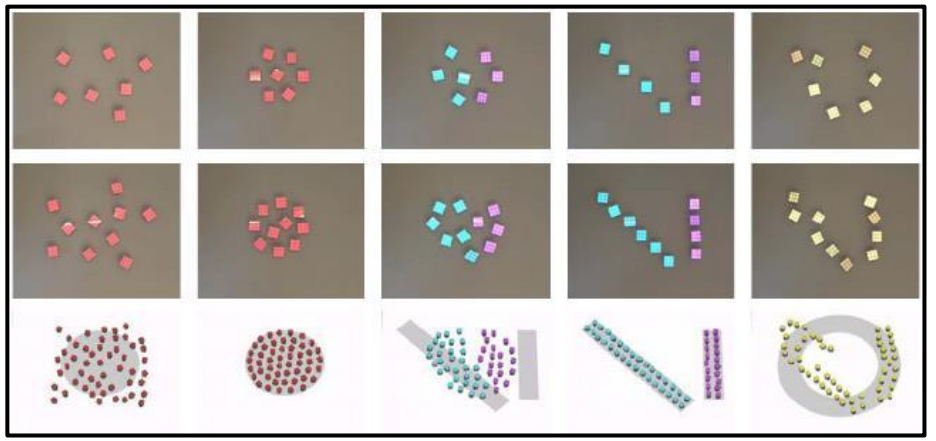
Goal:

- Create a multi-agent system in which the **agents learn to form a predefined shape.** No agent should be outside this shape, and inside they should be uniformly distributed.
- The agents will first form a simple shape (circle) then move to a different, more complex shape

Methodology:

- ANNs
- Reinforcement Learning (PPO)
- PettingZoo

```
mination, truncation, info = env.last()

tion:


TAINS = False
ircle.predict(observation, deterministic=True)[0]

TAINS = True
riangle.predict(observation, deterministic=True)[0]


iep < 5000


ctiveInt\formation.py
```



github.com/elte-collective-intelligence/student-formation

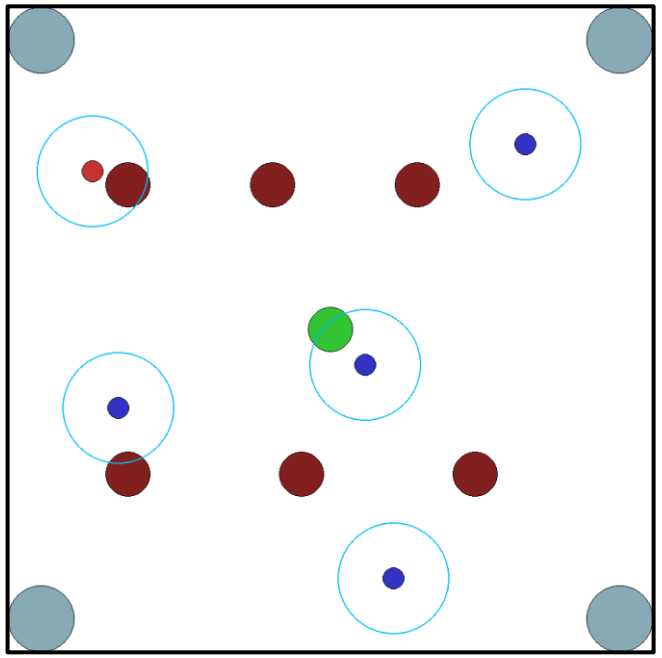
Course Projects - Patrolling

Goal:

- Develop a multi-agent system **with autonomous drones**.
- **Patroller drones:** Detect and neutralize intruders. (Seek collaboration)
- **Intruder drones:** Reach a target undetected. (Try to slip by patrollers)
- Try to achieve a win rate of 50-50%.

Methodology:

- ANNs
- Reinforcement Learning (PPO)
- PettingZoo
- Optuna

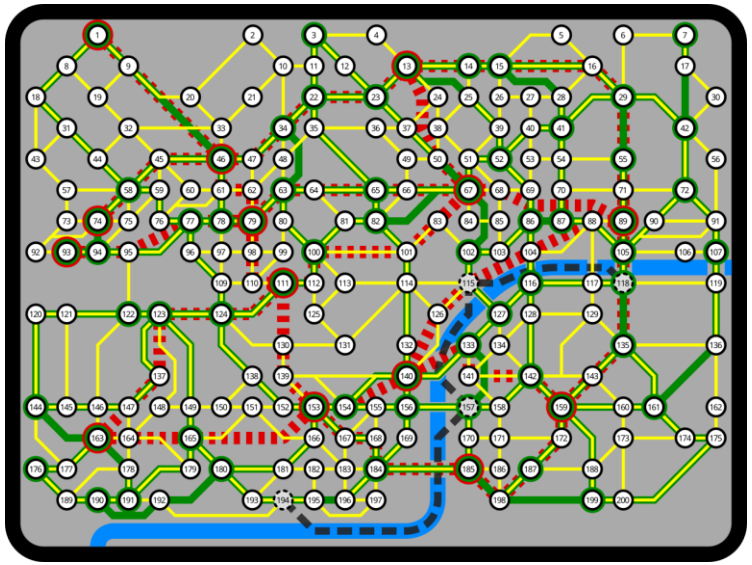


github.com/elte-collective-intelligence/student-patrolling

Course Projects – Mechanism Design (Scotland Yard)

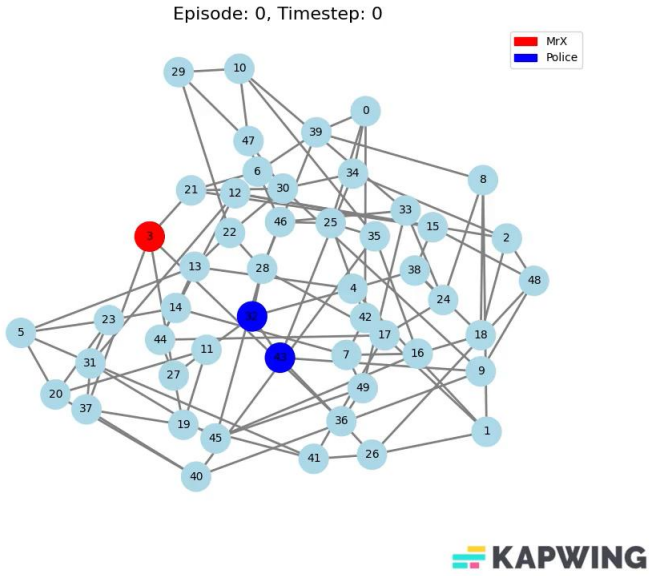
Goal:

- Detective Agents: Capture "Mr. X" within the city of London.
- Mr. X (Prisoner): Evade capture.
- Try to achieve a win rate of 50-50% utilizing meta-learning.



Methodology:

- GNNs
- Reinforcement Learning (PPO)
- Gymnasium



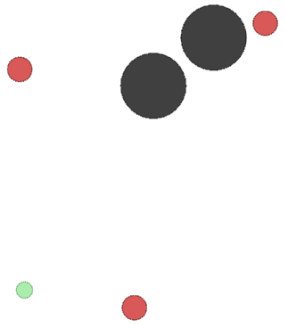
KAPWING

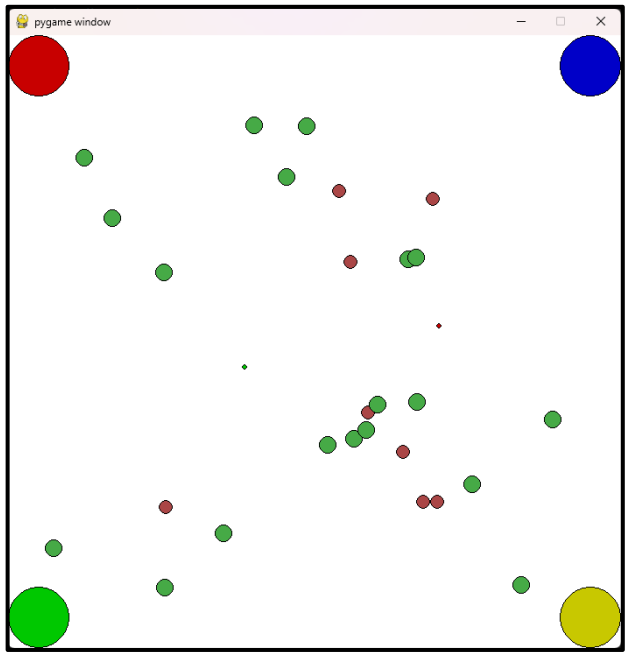
github.com/elte-collective-intelligence/student-mechanism-design

Course Projects – Search & Rescue

Goal:

- Rescue Simulation in Multi-Agent Systems using PettingZoo
- Simulate and implement a multi-agent rescue scenario where rescuers guide victims to safe zones while avoiding obstacles
- Collision Detection
- Victim Search
- Clustering and Delegation





Methodology:

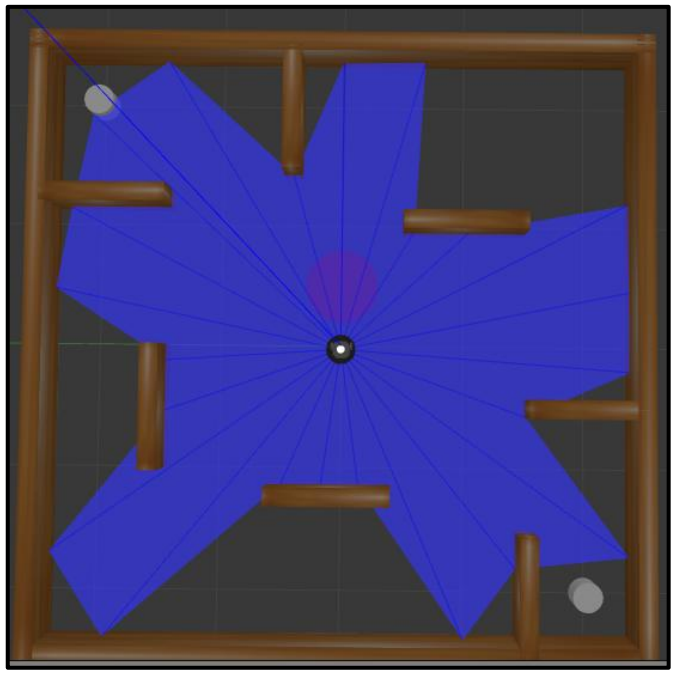
- ANNs
- Reinforcement Learning (PPO)
- PettingZoo

github.com/elte-collective-intelligence/student-search

Course Projects – TurtleBot3s in Gazebo

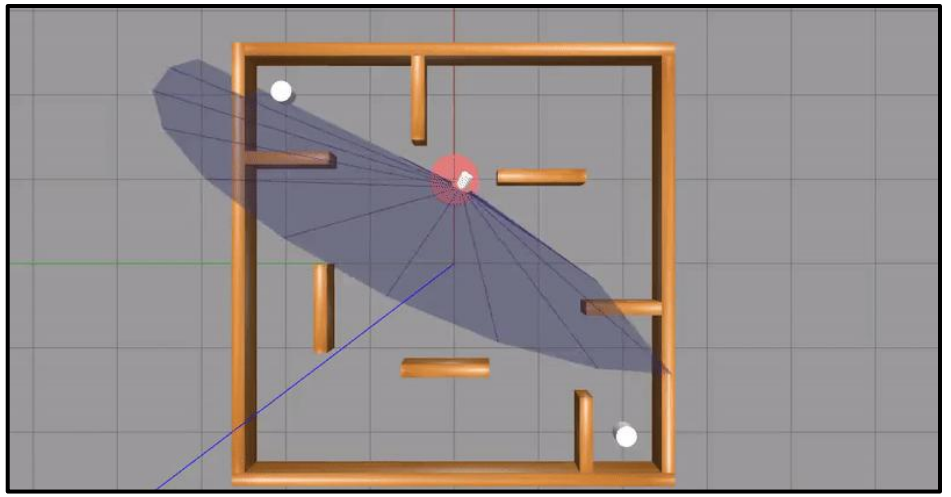
Goal:

- To evaluate the scalability of Deep Q-Networks (DQN) under different environmental conditions in a TurtleBot setting.
- Multi-stage learning.
- Single agent learning adaptability to multi-agent setting.



Methodology:

- DNNs
- Reinforcement Learning (DQN)
- Gazebo
- ROS2 Humble

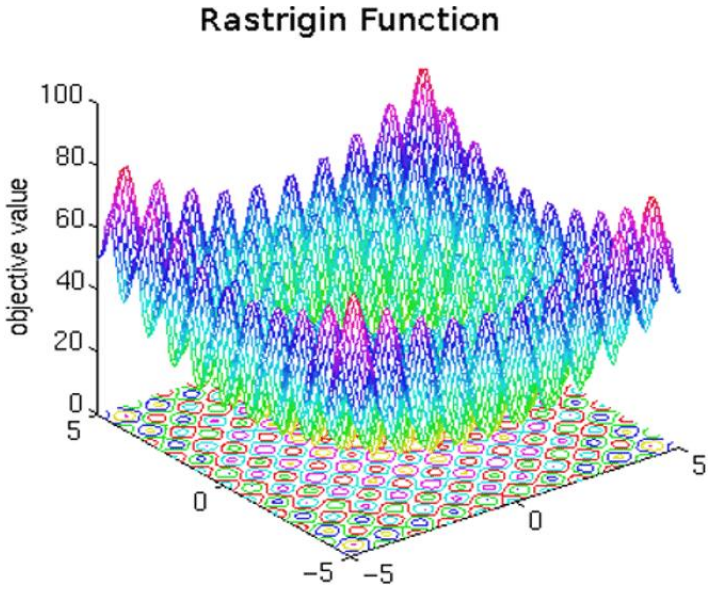


github.com/elte-collective-intelligence/student-turtlebot3s

Course Projects – Particle Swarm Optimization

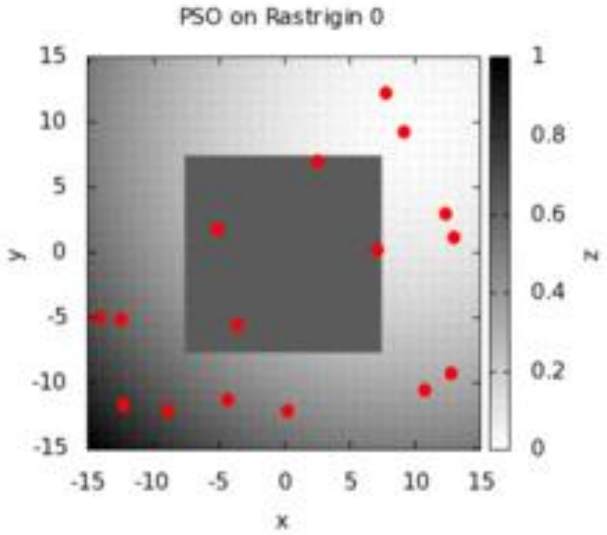
Goal:

- Given a particular function find the location with the best fitness value:
  - Can be complex
  - Can be noisy
  - Can model real world situations



Methodology:

- ANNs
- Reinforcement Learning (PPO)
- PettingZoo
- SmartSwarm

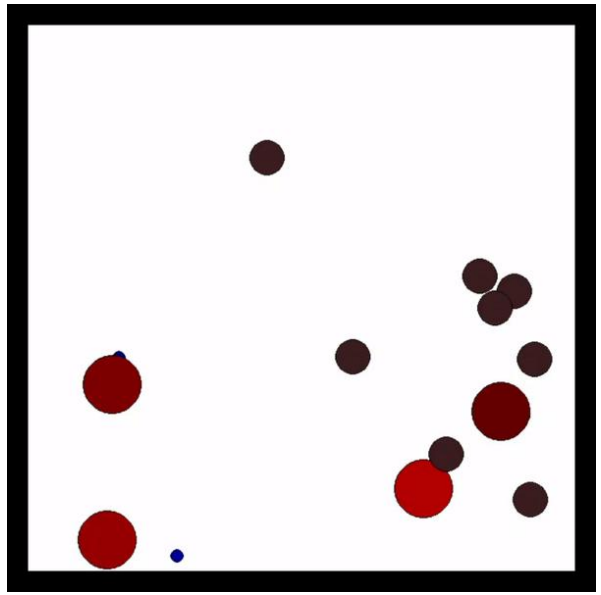


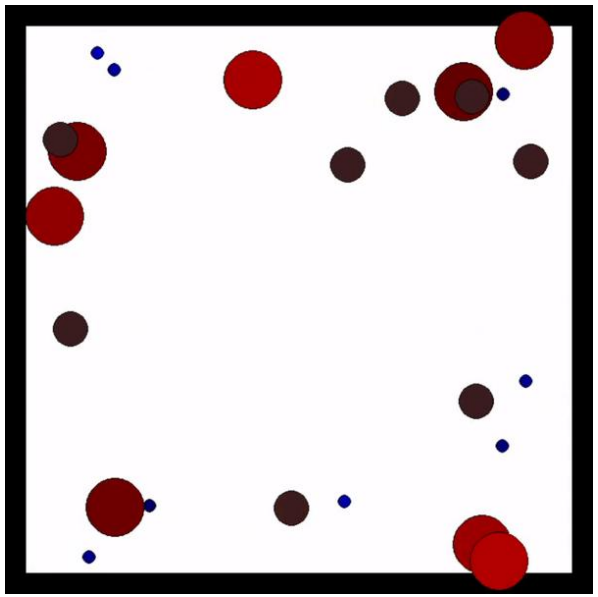
github.com/elte-collective-intelligence/student-particle-swarm-optimization

Course Projects – Pathfinding

Goal:

- Multiple agents, target locations, static obstacles
- Agents must travel to a specific point in space while avoiding collisions
  - Unique Target per Agent
- Train Agents that can:
  - Reach Target location
  - Avoid colliding with other agents and obstacles





Methodology:

- ANNs
- Reinforcement Learning (PPO)
- PettingZoo
- Gymnasium

github.com/elte-collective-intelligence/student-pathfinding

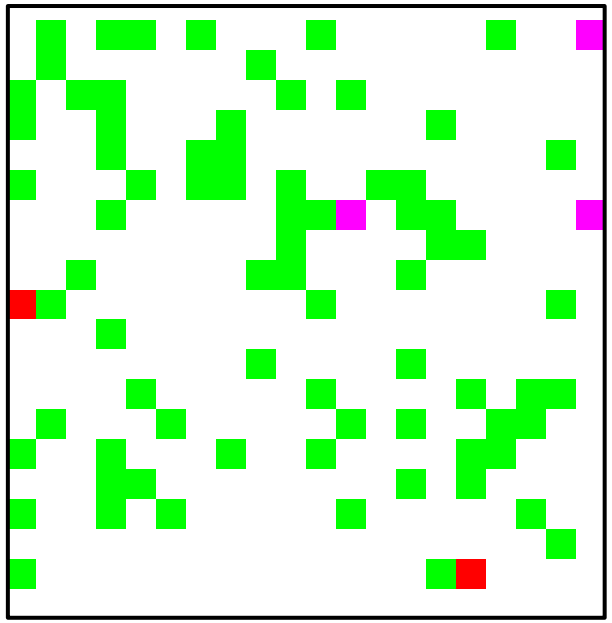
Course Projects – Sorting & Clustering

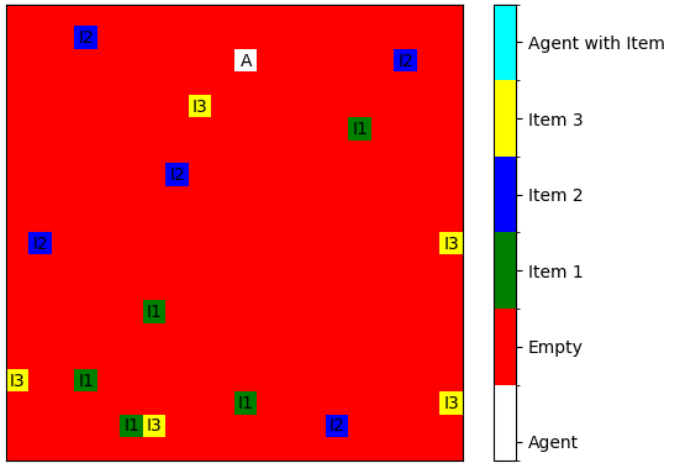
Goal:

- Goal is to successfully create clusters of items within the shortest amount of time (clustering). OR
- The agents are required to sort the food together as much as possible (sorting).

Methodology:

- ANNs
- Reinforcement Learning (PPO)
- PettingZoo
- Gymnasium





github.com/elte-collective-intelligence/student-sorting-clustering

Collective Intelligence

Practice 1.

Introduction

Budapest, 11th January 2025

☐ Course Details

Collective Intelligence

☐ Technical Details

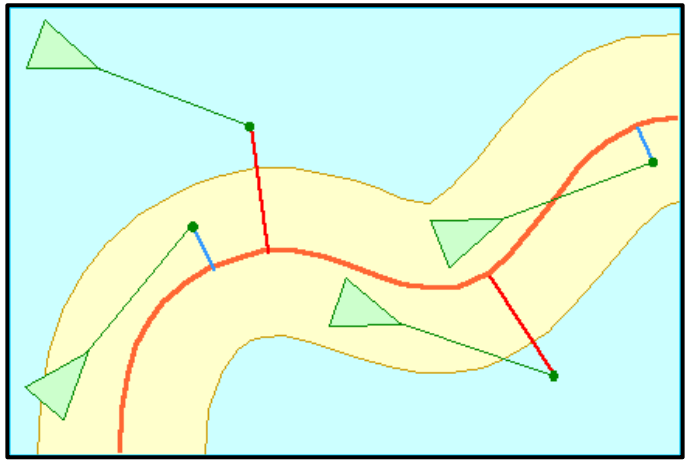
Boids – Swarm Intelligence

Boids is an artificial life simulation originally developed by **Craig Reynolds (1986)**. The aim of the simulation was to replicate the behavior of flocks of birds.

Instead of controlling the interactions of an entire flock, however, **the Boids simulation only specifies the behavior of each individual bird**.

With only a **few simple rules**, the program manages to generate a result that is complex and realistic.

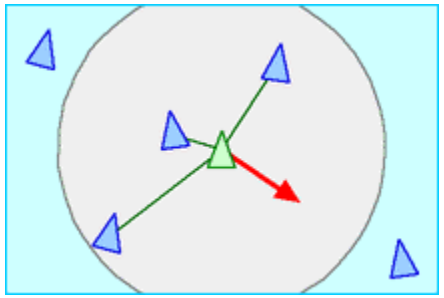
Variations of it are still in use for **simulating flocks of birds in movies** etc.



Boids

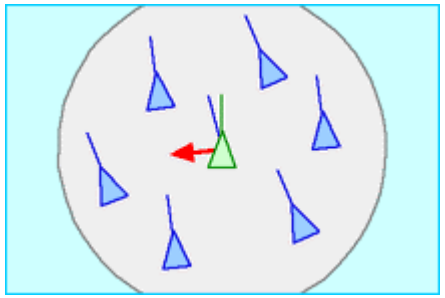
In the Boids simulation birds, each have their own position, velocity, and orientation.

There are only 3 rules which specify the behavior of each bird:



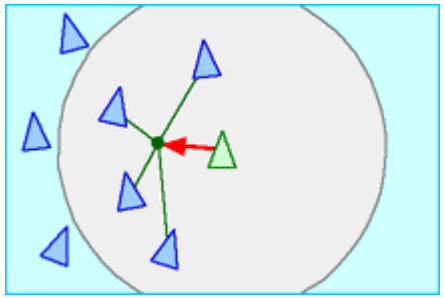
Separation

Each bird attempts to maintain a reasonable amount of distance between itself and any nearby birds, to prevent overcrowding



Alignment

Birds try to change their position so that it corresponds with the average alignment of other nearby birds.

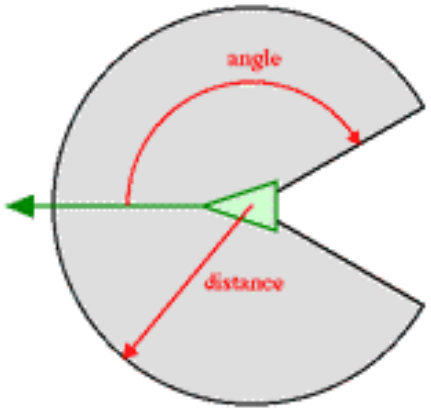


Cohesion

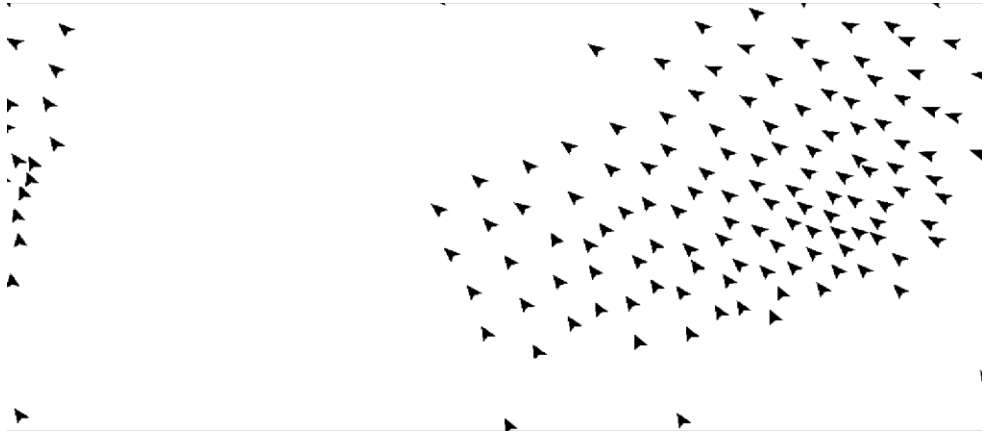
Every bird attempts to move towards the average position of other nearby birds.

Each boid has **direct access to the whole scene's geometric description**, but flocking requires that it reacts only to flock mates within a certain small neighborhood around itself.

Boids



Flock mates **outside this local neighborhood are ignored**. The neighborhood could be considered a model of limited perception (as by fish in murky water) but it is probably more correct to think of it as defining the region in which flock mates influence a boids steering.



Swarm Intelligence

A key aspect of swarm intelligence systems is the **lack of a centralized control agent**--instead, each individual unit in the swarm follows its own defined rules, sometimes resulting in surprising overall behavior for the group.

Swarm intelligence is also exemplified in ant colonies during food collection, a behavior modeled in Ant Colony Optimization (**ACO**). ACO is a powerful problem-solving strategy used to find optimal paths through complex structures.

Swarm optimization techniques are applicable to a wide range of challenges, such as solving the Traveling Salesman Problem or enabling swarm robotics for tasks like search-and-rescue operations, patrolling, and more.

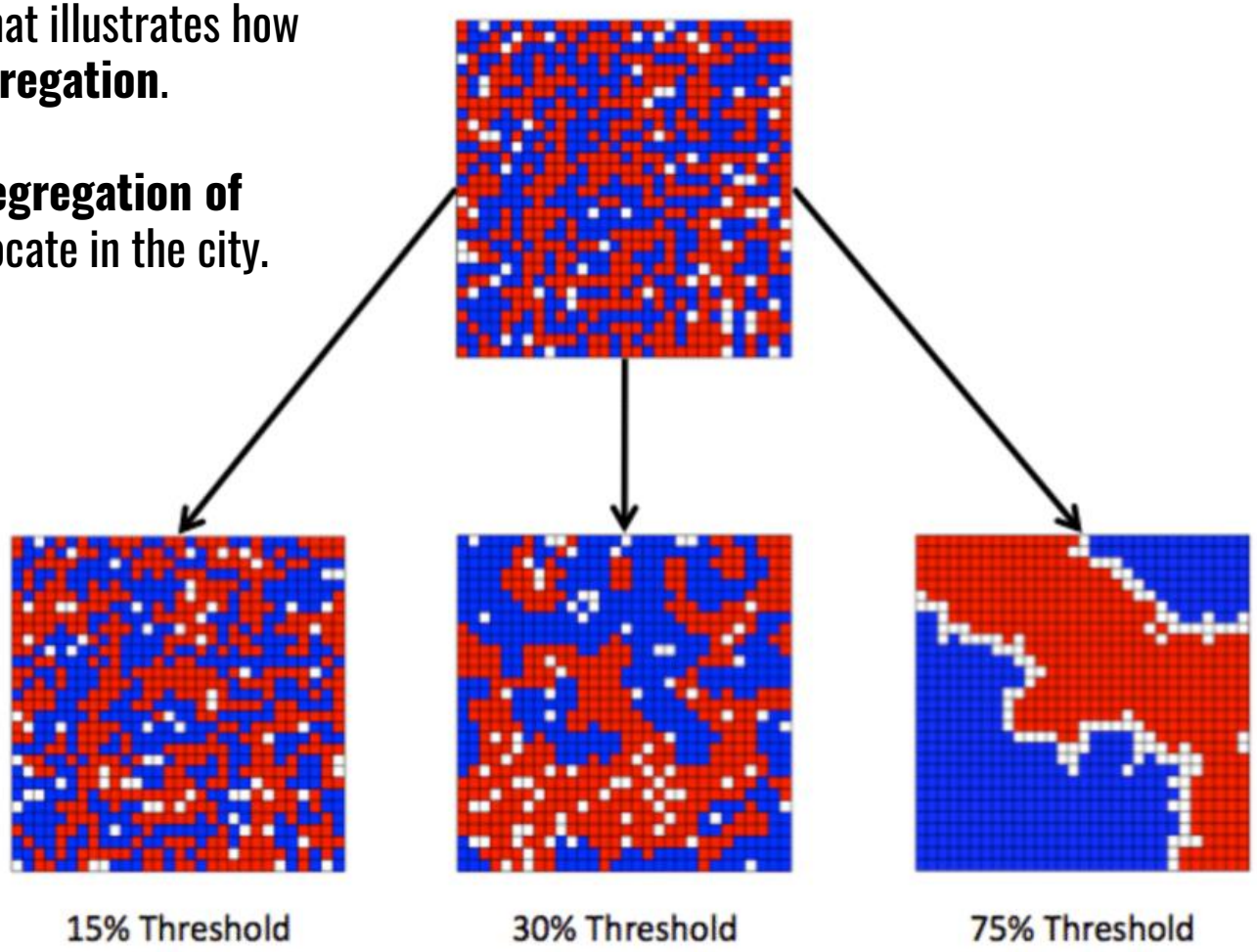
Schelling's Model of Segregation – Social Science

The Schelling model of segregation is an agent-based model that illustrates how **individual tendencies** regarding neighbors **can lead to segregation**.

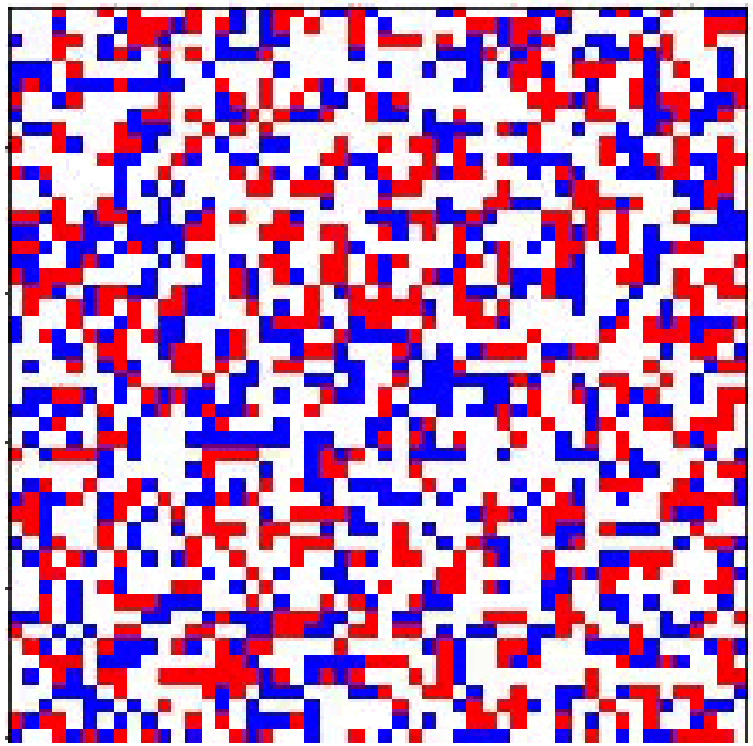
The model is especially useful for the **study of residential segregation of ethnic groups** where agents represent householders who relocate in the city.

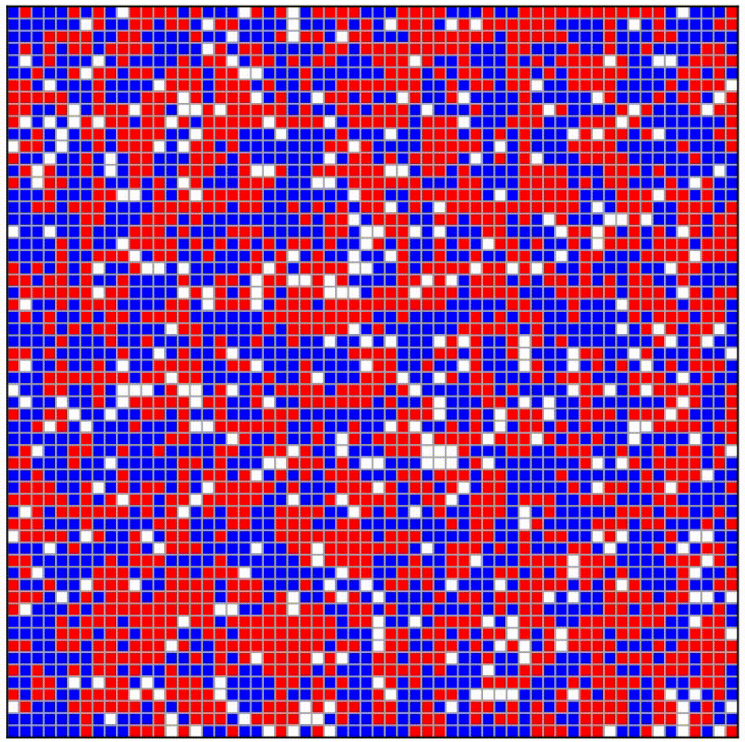
The model was introduced by **Thomas Schelling in 1978** to illustrate how individual incentives and individual perceptions of difference can lead collectively to segregation.

The canonical Schelling model **does not consider variables which may affect the agent's ability to relocate positions** in the grid.



Schelling's Model of Segregation

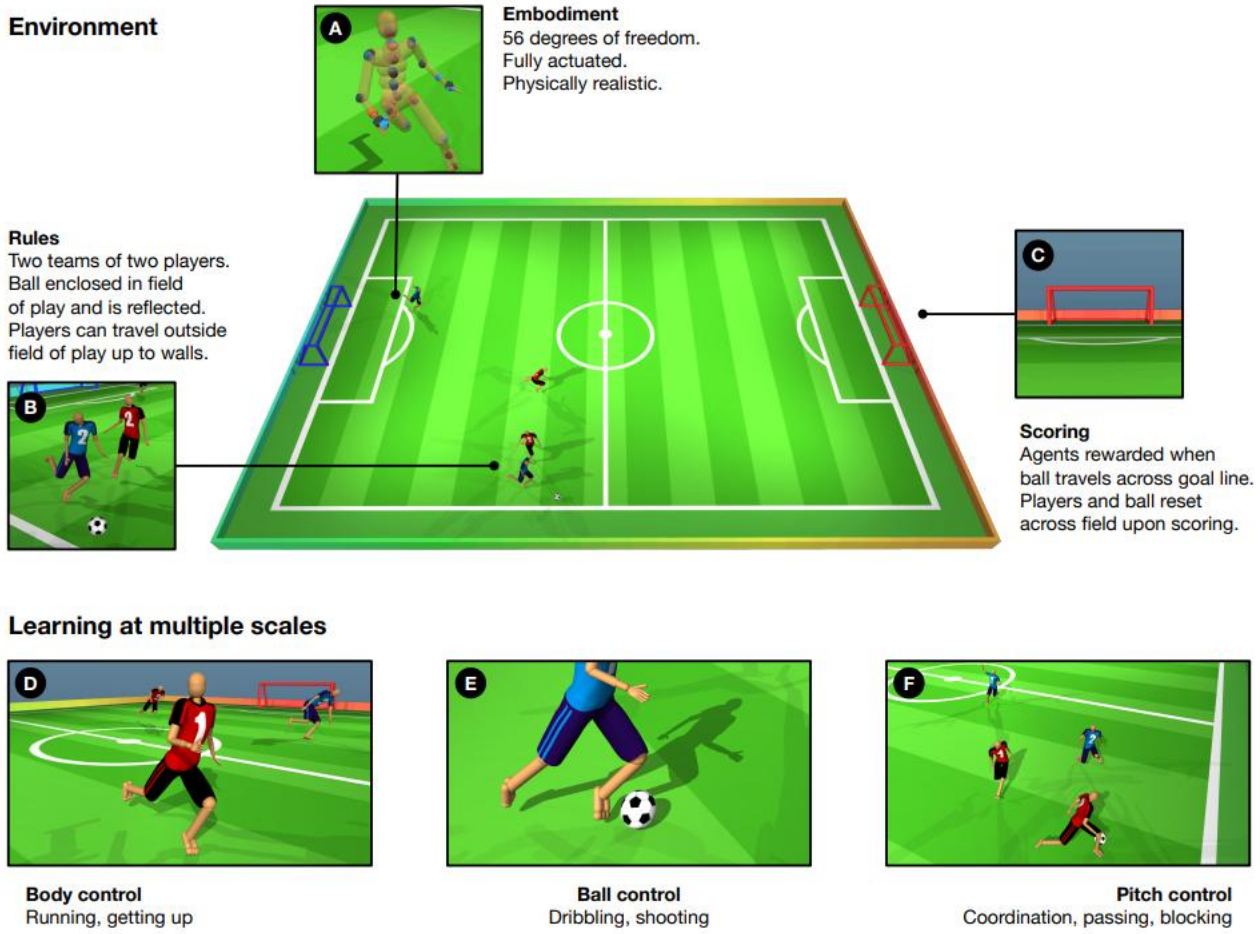




Local decision-making can result in unintended global patterns.

From Motor Control to Team Play in Simulated Humanoid Football [1]

- **Learning-based approach** for complex movements and coordination.
- Combination of **imitation learning, multi-agent reinforcement learning** and **population-based training**.
- **Curriculum Learning:**
  - Learn to control the body
  - Running and turning
  - Dribbling, shooting
  - Awareness
  - Goal-Oriented footballers



[1] Liu, S., Lever, G., Wang, Z., Merel, J., Eslami, S. M. A., Hennes, D., ... Heess, N. (2021). From Motor Control to Team Play in Simulated Humanoid Football. arXiv [Cs.AI]. Retrieved from http://arxiv.org/abs/2105.12196

From Motor Control to Team Play in Simulated Humanoid Football

youtube.com/watch?v=KHMwq9pv7mg

Emergent tool use from multi-agent interaction [2]

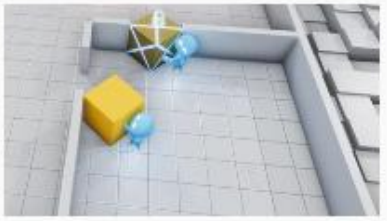
- Goal was to observe agents discovering progressively more complex tool use while playing a simple game of hide-and-seek.
- Hiders (blue) are tasked with avoiding line-of-sight from the seekers (red), and seekers are tasked with keeping vision of the hiders.
- There are objects scattered throughout the environment that hiders and seekers can grab and lock in place, as well as randomly generated immovable rooms and walls that agents must learn to navigate.

The agents can **move** by setting a force on themselves in the x and y directions as well as rotate along the z-axis.

The agents can **see** objects in their line of sight and within a frontal cone.

The agents can **sense** distance to objects, walls, and other agents around them using a lidar-like sensor.

The agents can **grab and move** objects in front of them.

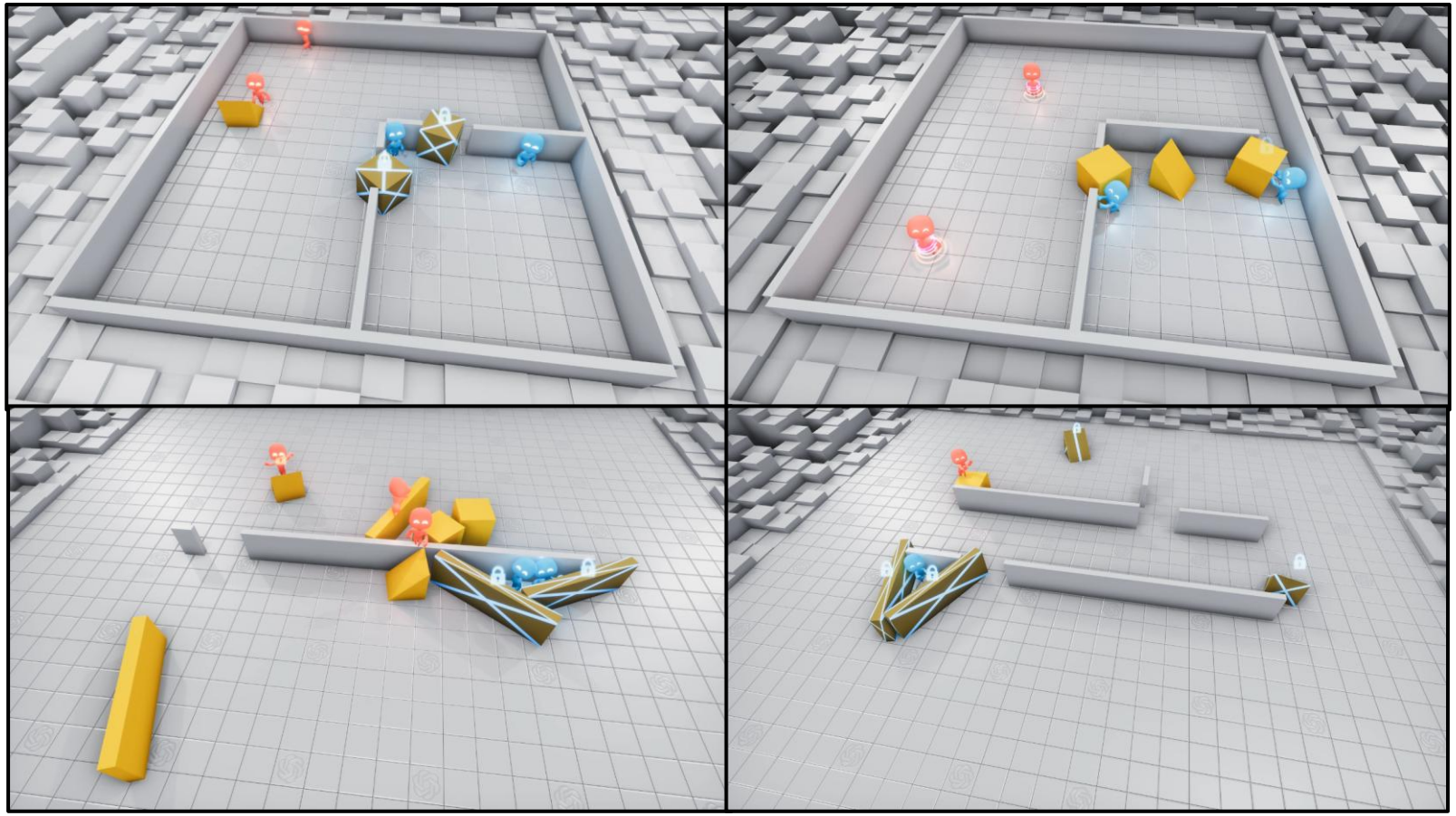


The agents can **lock** objects in place. Only the team that locked an object can unlock it.

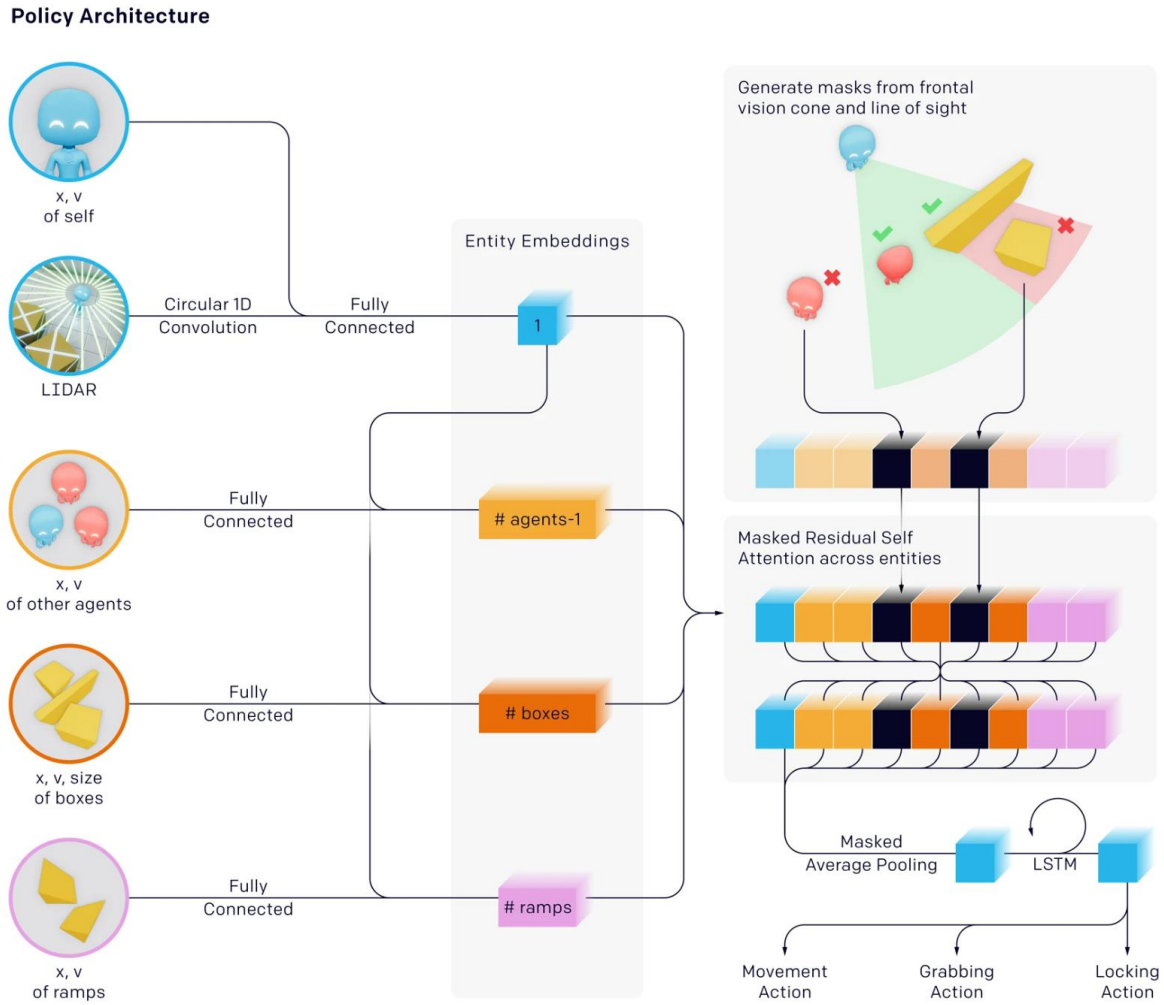
[2] Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B., & Mordatch, I. (2020). Emergent Tool Use From Multi-Agent Autocurricula. arXiv [Cs.LG]. Retrieved from http://arxiv.org/abs/1909.07528

Emergent tool use from multi-agent interaction

- There are no explicit incentives for agents to interact with objects in the environment, however as many as six distinct strategies emerged:
  - Chasing
  - Door Blocking
  - Ramp Use
  - Ramp Defense
  - Shelter Construction
  - Box Surfing



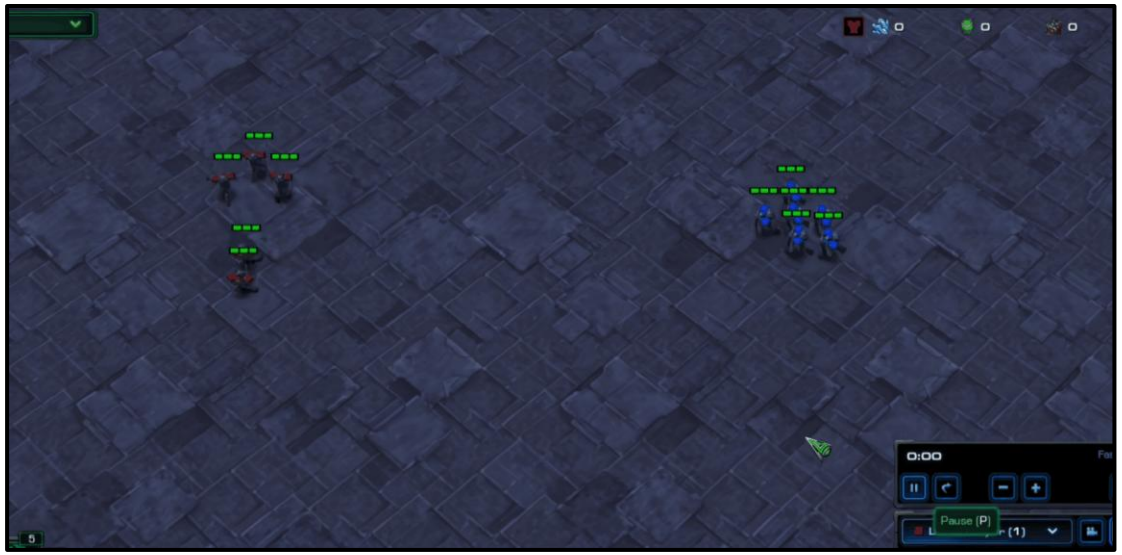
Emergent tool use from multi-agent interaction



.youtube.com/watch?v=kopoLzvh5jY

The StarCraft Multi-Agent Challenge [3]

- SMAC was introduced as a Multi Agent Reinforcement Learning benchmark.
- SMAC is based on the popular real-time strategy game StarCraft II and focuses on **micromanagement challenges** where each unit is controlled by an independent agent that must act based **on local observations**.
- SMAC consists of a set of StarCraft II micro scenarios which aim to evaluate how well independent agents are able to learn coordination to solve complex tasks.



[3] Samvelyan, M., Rashid, T., de Witt, C. S., Farquhar, G., Nardelli, N., Rudner, T. G. J., ... Whiteson, S. (2019). The StarCraft Multi-Agent Challenge. arXiv [Cs.LG]. Retrieved from http://arxiv.org/abs/1902.04043

Collective Intelligence

Practice 1.

Introduction

Budapest, 11th January 2025

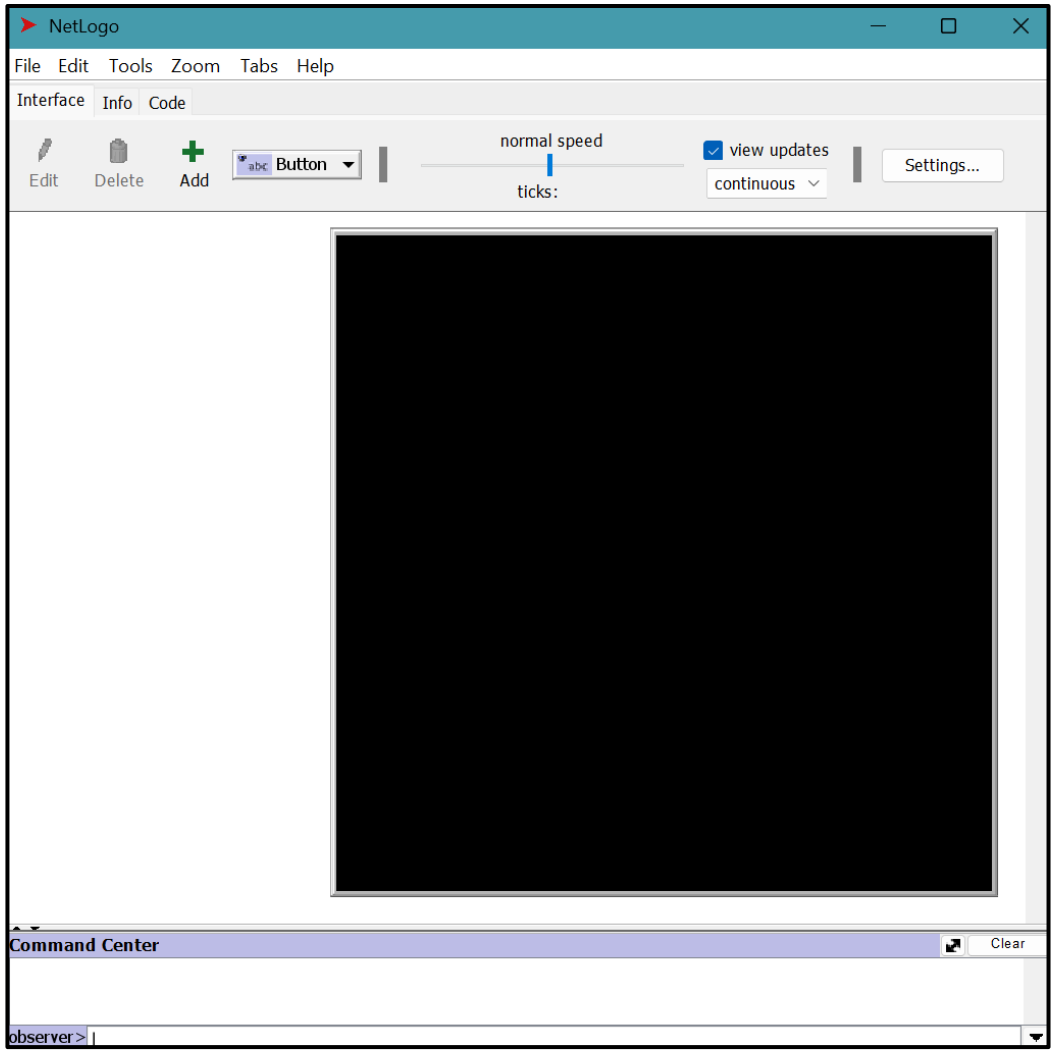
- [ ] Course Details
- [ ] Collective Intelligence
- [x] Technical Details

What tools will we be using?

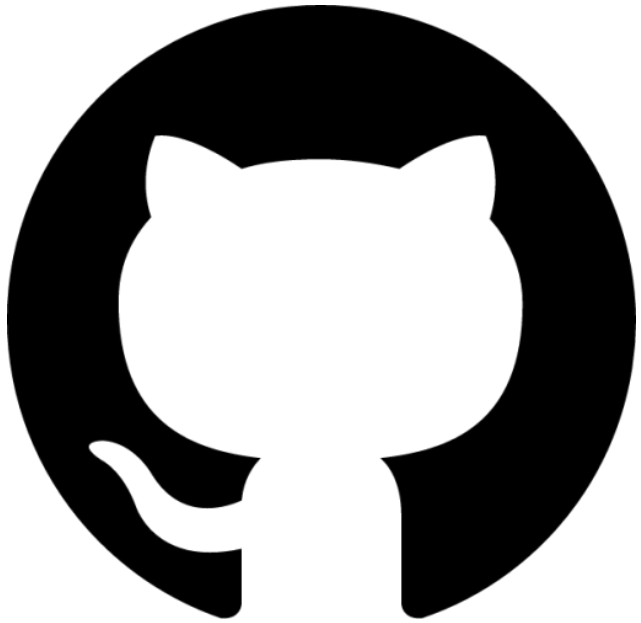
NetLogo

- Multi-agent programmable modeling environment. **Most used agent modeling software in research**. Well-renowned.
- Newest stable release (with GUI) works on Mac, Windows and Linux as well.
- Comes with a privately installed Java 17. (this will not affect any other software running Java on your computer)
- Downloads: ccl.northwestern.edu/netlogo/6.4.0/
- User Manual: ccl.northwestern.edu/netlogo/docs/NetLogo%20User%20Manual.pdf
- NetLogo Book: intro-to-abm.com/
- **Installation is straightforward**. (no technical skills required)

What tools will we be using?



What tools will we be using?



GitHub

- The **second assignment will require Git** Version Control Software installed on your system.
- **GitHub Organizations** will be used to track project progress.
- Each student will be assigned a dedicated repository within the organization.
- Git Download Link: git-scm.com/downloads (straightforward as well)
- Organization Link: github.com/elte-collective-intelligence
- The organization's README contains detailed information about the repository structure, access management across projects, and student responsibilities.

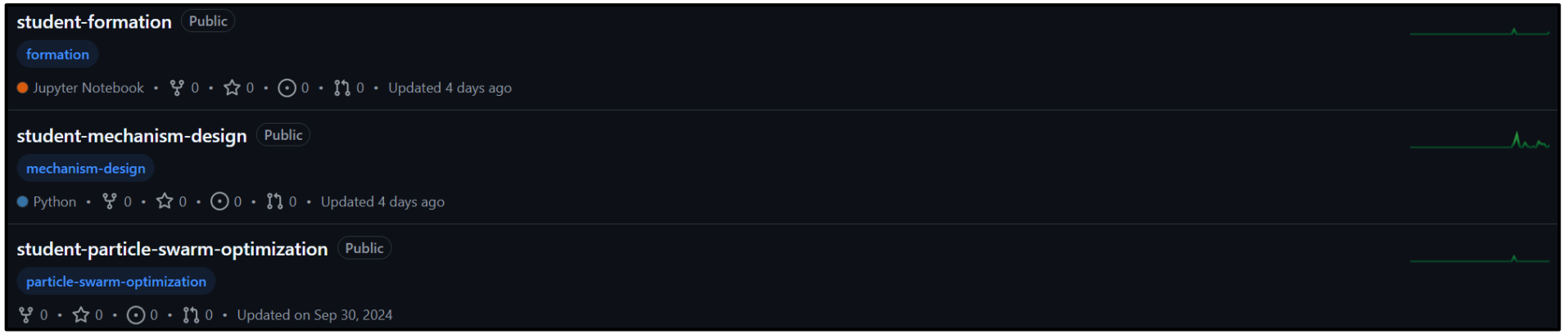
What tools will we be using?

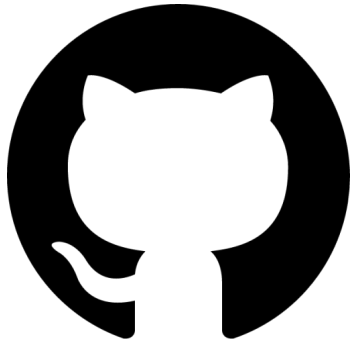
To authenticate with GitHub using Git, ensure that you:

- Use the GitHub CLI by running the following command in a Git terminal and follow the provided instructions.

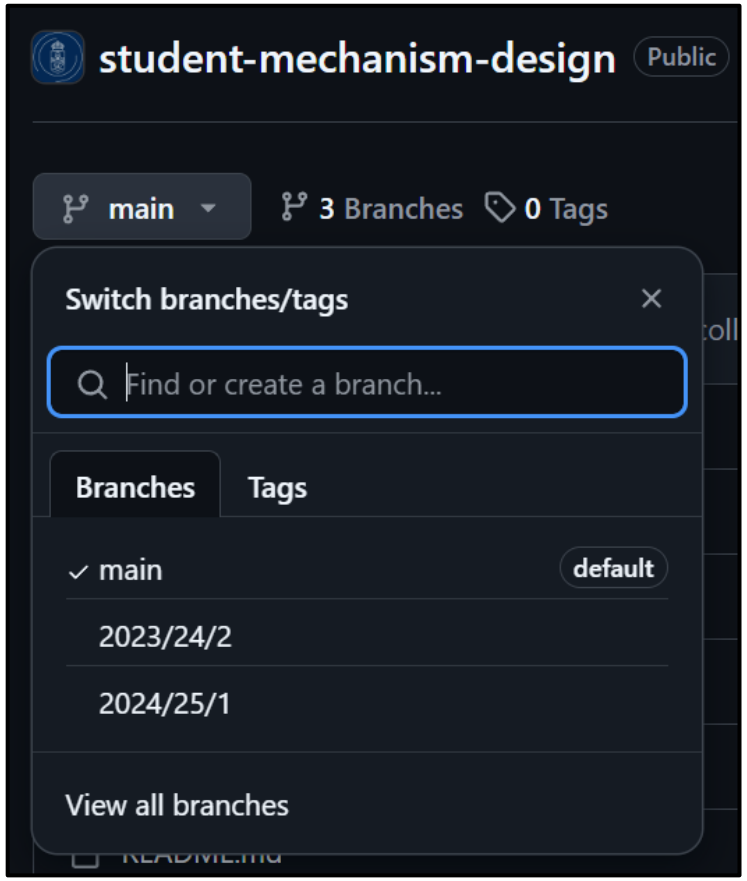
```
$gh auth login
```

Repository Structure:



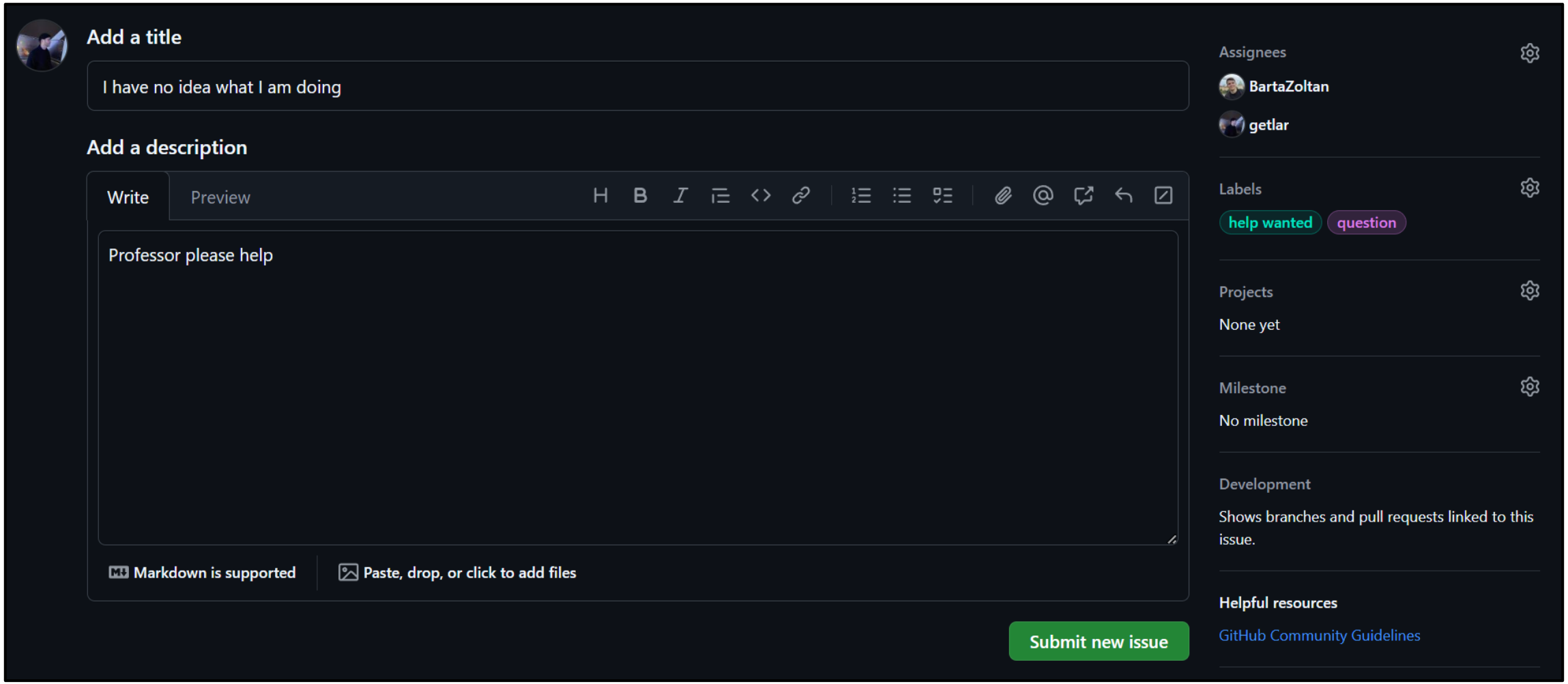


What tools will we be using?

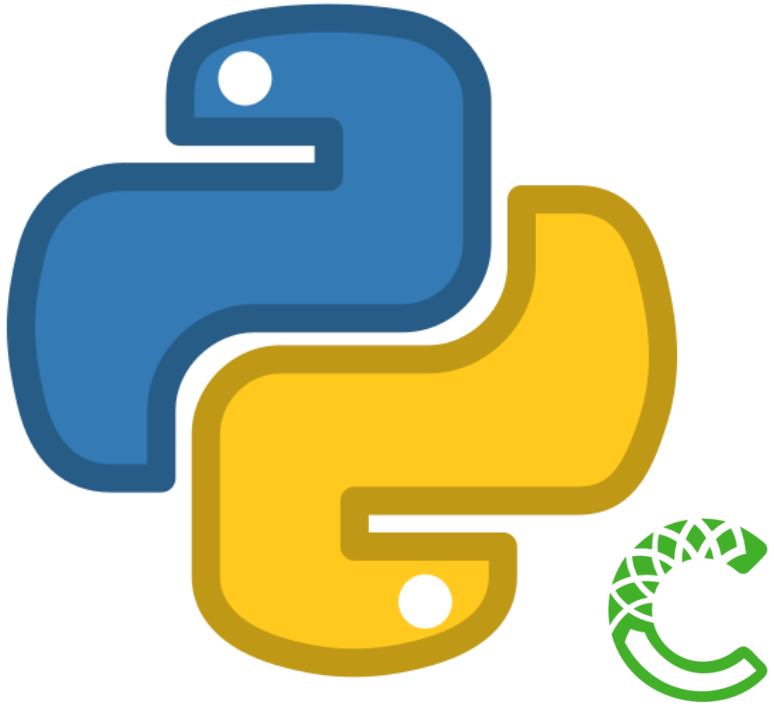


- Each project is **organized into semesters**, with separate branches for each semester.
- **All students have read access** to code across all repositories and semesters.
- Past semester **branches are protected**, ensuring no modifications can be made to them.
- **Only assigned students have write access to their project repository**, allowing them to branch off from the current version.
- Modification rules can be adjusted slightly upon request.
- **Students must adhere to the branch naming conventions** and, at the end of the semester, **merge their modifications into the main branch**.
- Students are encouraged to **maintain thorough documentation and keep a research diary** to ensure that future students can build on the existing code effectively.

Issues with the assignment?



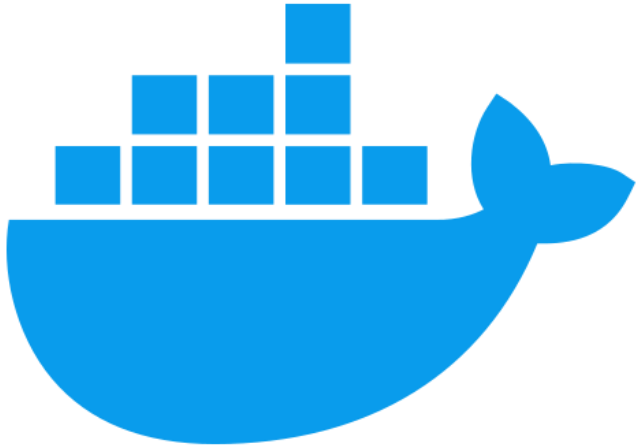
What tools will we be using?



Python + Python Package Manager

- We will be using multiple Python packages, so a **package manager will be essential**.
- Practice code will be distributed **through Colab Notebooks and raw Python scripts**, along with the required dependencies to reproduce the results.
- We recommend using **Miniconda** because it is lightweight and easy to use across all systems.
- However, you are **free to use any package manager** or environment you are familiar with.
- **Version pinning** is crucial in this domain, as package updates can quickly break the code.
- Cheat Sheet: Conda Cheat Sheet

What tools will we be using?



Docker

- Works seamlessly with Miniconda, which helps with package versioning and cross-platform compatibility.
- For raw Python code, we will provide a simple Dockerfile that contains everything needed to reproduce the code demonstrated in class.
- Docker will also be required for the second assignment. (Apptainer is also a possibility)
- Cheat Sheet: Docker Cheat Sheet
- (I also recommend using VS Code Dev Containers tool for a more seamless experience)

Project Registration

Editable Excel Sheet

Topic List 2025 - Assignment 2.xlsx

Only 2 Students/Project are allowed!

Please write your name next to your chosen project by the end of next week. If your name is not listed, a project will be assigned to you randomly.

Summary

Key things for the semester:

- Make sure to **choose a project** before next week
- Make sure you **have access to the GitHub** organization
- **2 assignments**, with 0.25 and 0.75 weightings respectively
- Both **assignment must be submitted to Canvas** (for administrative reasons, .zip, .ipynb, .py, .nlogo)
- The **practice material** will also be submitted to **Canvas**
- If you have **a project idea**, feel free to reach out for further discussion
- To receive your final grade for the assignments, **you must pass a test** based on the lecture content

Further Links + Resources

- If you are not familiar with Git: https://learngitbranching.js.org/
- If you are not familiar with Docker: https://docker-curriculum.com/

Resources

Books:

- Albrecht, Christianos, Schäfer: **Multi-Agent Reinforcement Learning: Foundations and Modern Approaches**
  Freely available: MARL Book
- Wilensky, Rand: **An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with NetLogo**

  Freely available: Introduction to Agent-Based Modeling

Courses:

- Huggingface
- Introduction to MARL

That’s all for today!