Machine Learning Project work: “Tennis Table Tournament”

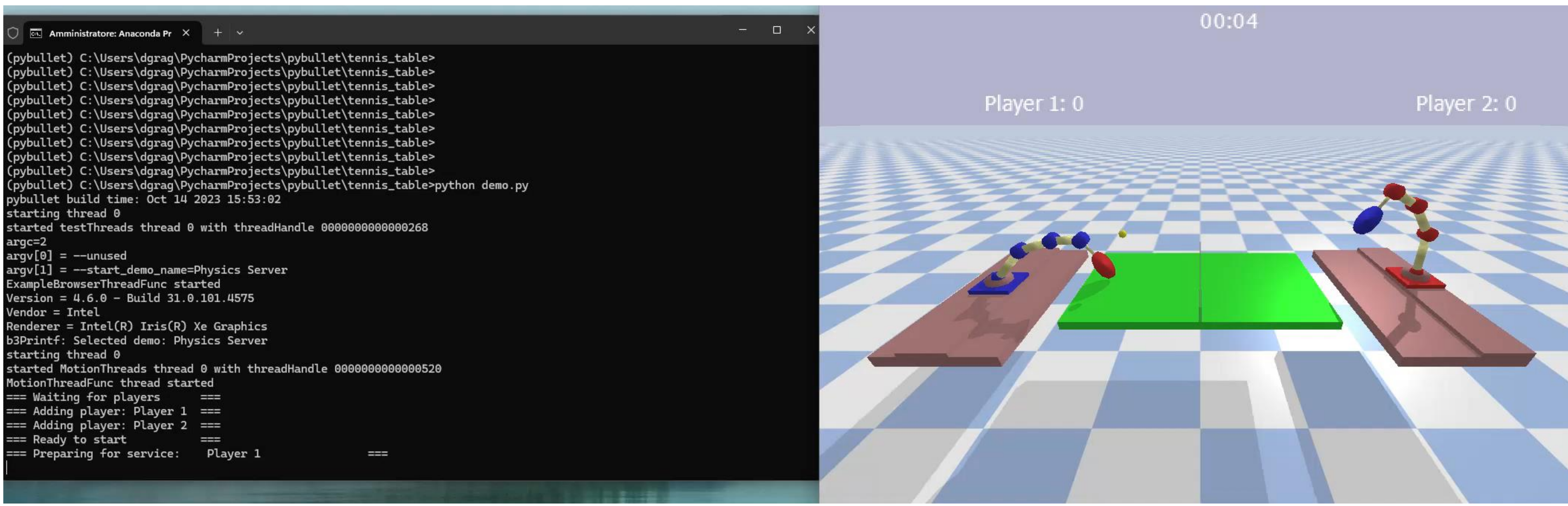
Mario Vento, Pasquale Foggia
and Diego Gragnaniello

MIVIA Tennis Table Tournament!

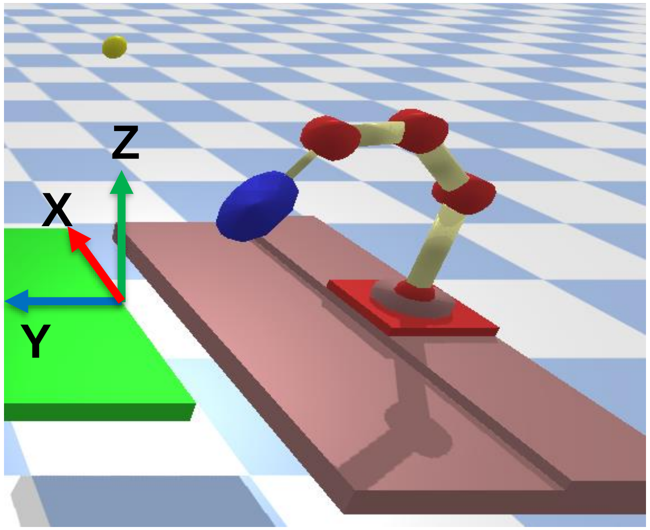
- Control a robotic arm to play tennis table, starting from a virtual environment
- Train your neural network on our GPU to respond to different environment inputs
- Win the games and rule the leaderboard! Future teams may challenge you!

The task to address

- The aim of this project is to train a neural network to play Tennis Table!
- We use a custom virtual environment to do it



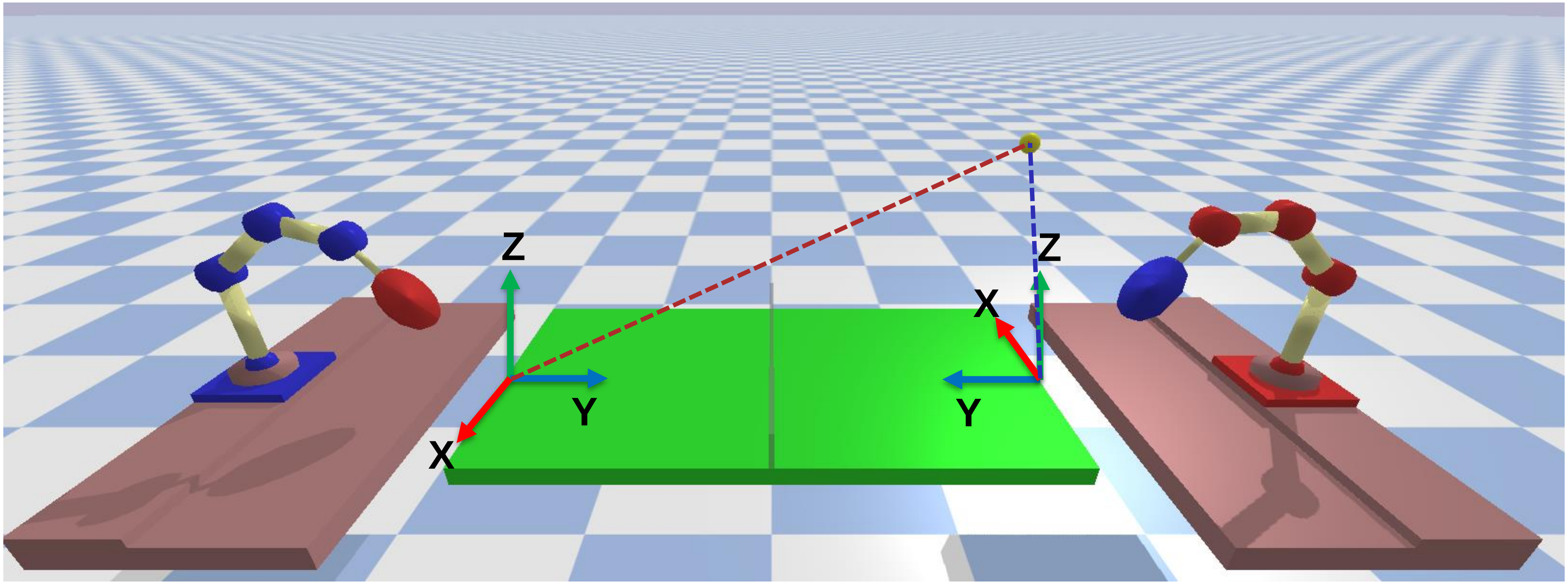
The 37 state variables



- 0-10 joints position
- 11-13 pad center position (x,y,z)
- 14-16 pad normal versor (x,y,z)
- 17-19 Current ball position (x,y,z)
- 20-22 Current ball velocity (x,y,z)
- 23-25 Opponent pad center position (x,y,z)
- 26 Game waiting, cannot move (0=no, 1=yes)
- 27 Game waiting for opponent service (0=no, 1=yes)
- 28 Game playing (i.e., not waiting) (0=no, 1=yes)
- 29 Ball in your half-field (0=no, 1=yes)
- 30 Ball already touched your court (0=no, 1=yes)
- 31 Ball already touched your robot (0=no, 1=yes)
- 32 Ball in opponent half-field (0=no, 1=yes)
- 33 Ball already touched opponent's court (0=no, 1=yes)
- 34-35 Your score, Opponent score
- 36 Simulation time

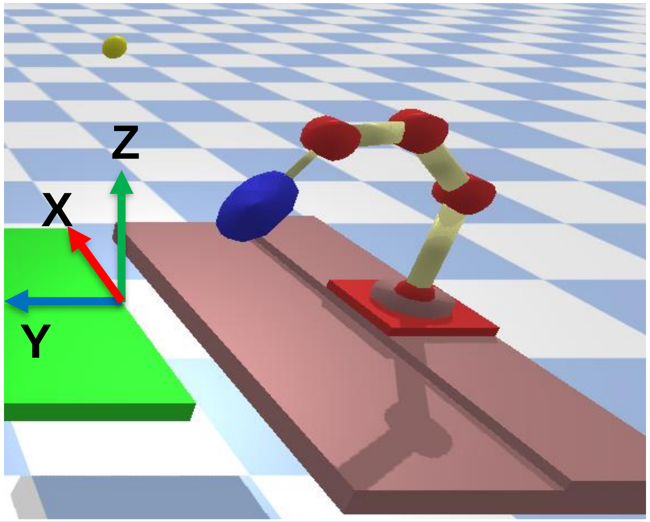
The 37 state variables

- Each player has its own coordinate system
- State variables are referred to this system



The 11 joint variables

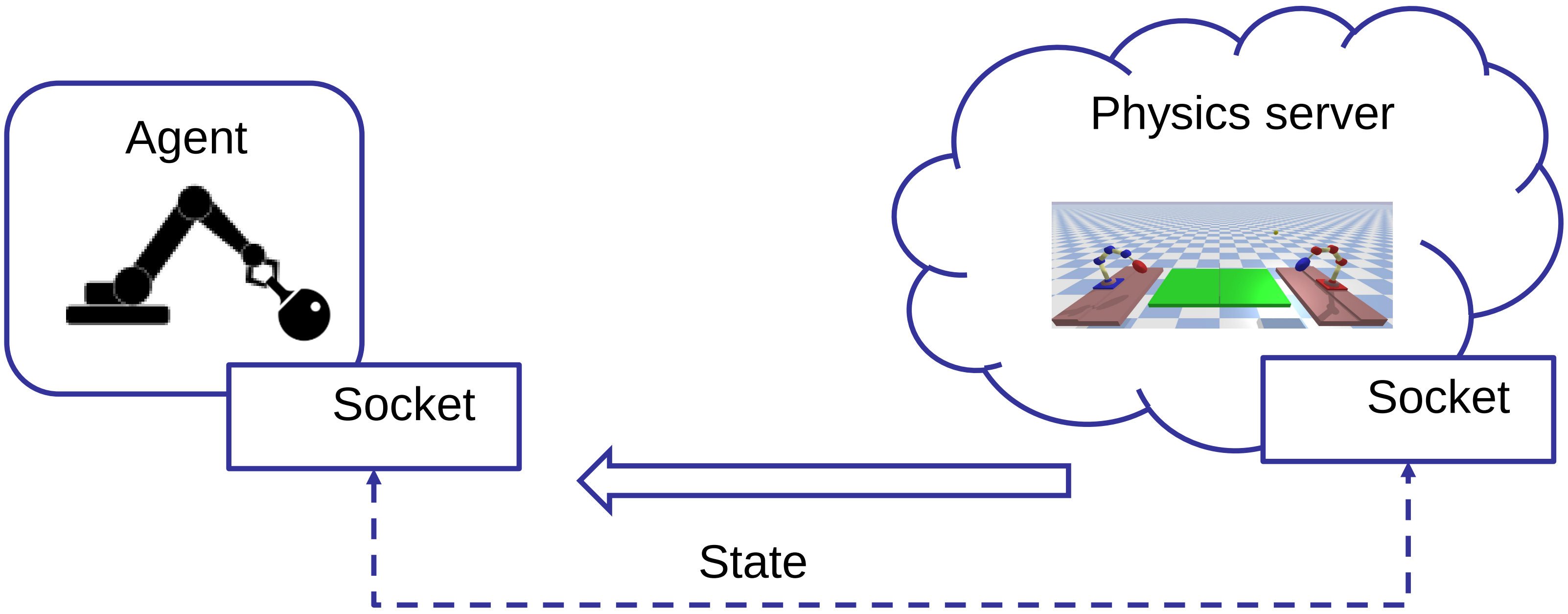
To move the robotic arm, you have to set the joints variables



Index	Type	Values	Description
0	Translation	-0.3 ... 0.3	Forward-Backward Slider. Positive Values are forward.
1	Translation	-0.8 ... 0.8	Left-Right Slider. Positive Values are to the right.
2	Rotation	Any	Rotation around the vertical axis (Z).
3	Rotation	$-\pi/2$ ... $\pi/2$	Pitch of the first arm link.
4	Rotation	Any	Roll of the first arm link.
5	Rotation	$-\pi$*3/4 ... π*3/4	Pitch of the second arm link.
6	Rotation	Any	Roll of the second arm link.
7	Rotation	$-\pi$*3/4 ... π*3/4	Pitch of the third arm link.
8	Rotation	Any	Roll of the third arm link.
9	Rotation	$-\pi$*3/4 ... π*3/4	Pitch of the pad.
10	Rotation	Any	Roll of the pad.

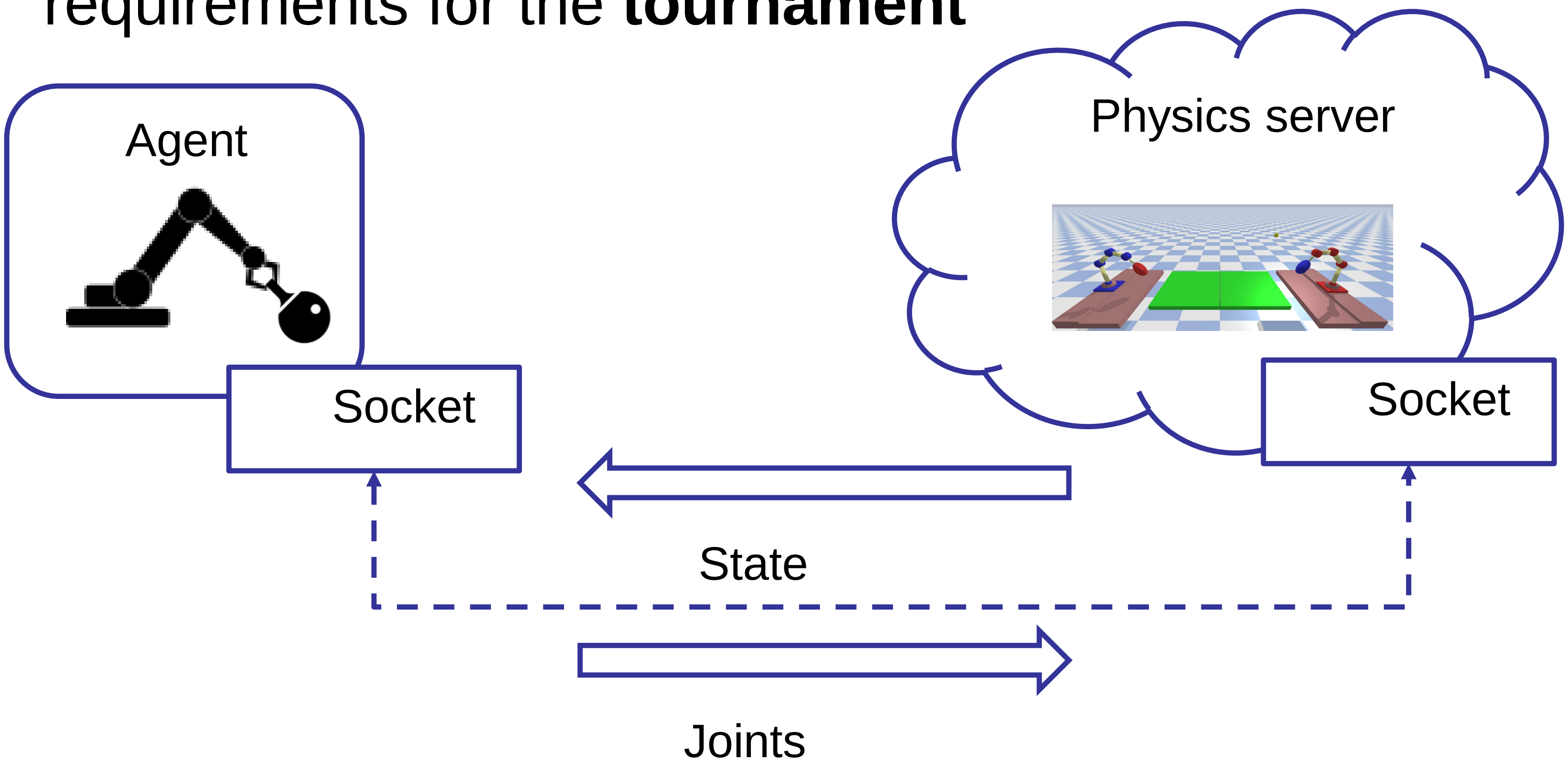
The environment

- Registers the player in the game
- Allows connection to a specific IP:PORT
- Provides the state variables through get_state()
- Execute the physics engine (PyBullet)



The participants

- Define their own reward function
- Set the robot joints through set_joints()
- To do it, they train a ML model using one of the approaches studied during the course (also using the MIVIA lab GPUs)
- Verify that the trained model is compliant with the requirements for the **tournament**



Match rules

1. Alternate serves every point
2. The player can hit the ball before or after it hits the player’s court
3. No “let” on serve
4. Also, a point is awarded if the ball gets stuck, hits twice a robot, or goes too far from the field
5. The game ends:
    1. when one player reaches 11 points;
    2. after a certain duration (e.g., 5 minutes);
    3. **for knockout matches only**: tie breaks after the time expires, i.e., whoever scores two consecutive points wins.

Tournament rules

32 teams (31 + 1 baseline)

Two stages tournament:

1. **Group stage**: selects 16 out of 32 teams
2. **Knockout stage**: 1 vs. 1 direct elimination matches

Tournament rules: group stage

1. 4 randomly drawn teams in each group (for a total of 8 groups)

2. Each team pair plays one against (3 matches per group)

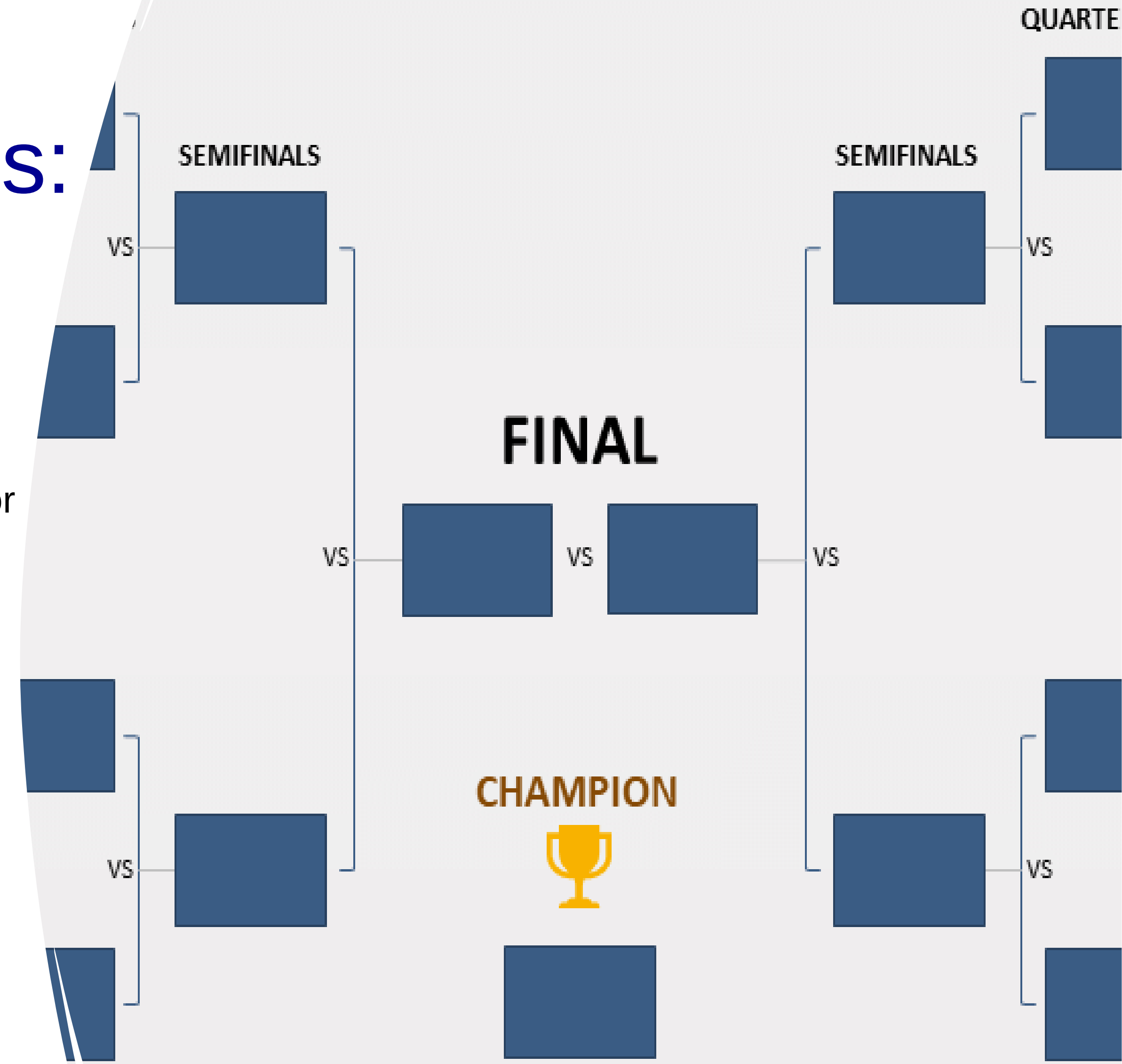
3. Group leaderboard:
   a) Match winner earns 3 points
   b) If the game is drawn, each team receives 1 point
   c) Ties (i.e., two or more teams achieve the same final score) are solved considering (in order):
      1. the greater Point difference;
      2. then the greater Points scored;
      3. wins in the face-off match;
      4. running a knockout playoff with tiebreak.

Tournament rules: knockout stage

1. The **group winner** and the **runner-up** of each group (for a total of 16 teams) access the final stage

2. Each **group winner** plays against the runner-up of another group (group #1 versus group #2, #3 versus #4, and so on)

3. Knockout matches with tiebreaks after the time limit expires.



IMPORTANT: Model training and validation is not a feed-forward process

- Alternate trainings and validations:
  - After each validation, try to understand when the model fails and why
  - Change your model and/or training algorithm to improve the performance

IMPORTANT: Model training and validation is not a feed-forward process

- Alternate trainings and validations:
  - If the performance seems very good, be sure that the test is challenging enough
  - Keep track of your countermeasures/improvements for the final project presentation

What you can use

- You can train your model against anything

- Any algorithm (whatever kind of classifier, preprocessing, training strategies like discrete or continuous reinforcement learning, validation)
  - E.g., you can
    - train a network with supervised learning to address the inverse kinematics and another one to play using reinforcement learning
    - train two different networks when serving or receiving.
  - But you must be able to explain what you have used
  - The whole system must be **runnable inside the provided environment in the MIVIA server**

- You can use local computing power

What you must submit

1. Code and trained model:
   - **Training code**: the code used for training your system
   - **Test code**: the code needed to test your system (test script; see next slide)
   - **Trained model**: the file containing the trained weights
   - The training and test code must be in the form of a Python Notebook

2. A **8 minutes** presentation (pptx/pdf) and a report (pdf), both in English
   - Please **don't restate** the problem, the application contest, or any other basic ML concept in your presentation! **Go straight to your contribution**
   - Put your names and team number at the beginning of both the presentation and the report

How to must submit

1. Create a directory:
   - **Named** "{:02d}".format(team_number)
   - **Containing the files**:
     - train.ipynb
     - test.ipynb
     - model.pth (or model0.pth, model1.pth, etc…)
     - slide.pptx or slide.pdf
     - report.pdf

2. Share the directory with professors via Google Drive
3. Upload the directory in the provided MIVIA computing server

When to submit

At the first exam date we:

- Discuss all the project works (8 minute presentation per team)
- Do the tournament

We will ask you to submit your solution a couple of days before that date.

Share the load

- Each member of the team will be requested to submit an estimate of the individual effort contributed by all members
  - To prevent "free riders"
  - Submissions will be "blind" (each member will not see the submissions of other members)

IMPORTANT: Don’t forget to

- Write the **names of all the team members** in the files
- Ensure that the **link** you submit is **readable to anyone** (no authorization must be requested)
- Make sure that the **test script** is **compliant with the specification** (if you have doubts about the specification, **ask**)