THE PELITA CONTEST

(A BRIEF INTRODUCTION)

THE PELITA CONTEST

(A BRIEF INTRODUCTION)

Rike–Benjamin Schuppner
Institute for Theoretical Biologie | HU Berlin

rikebs@debilski.de // debilski.de // @debilski

THE PELITA CONTEST

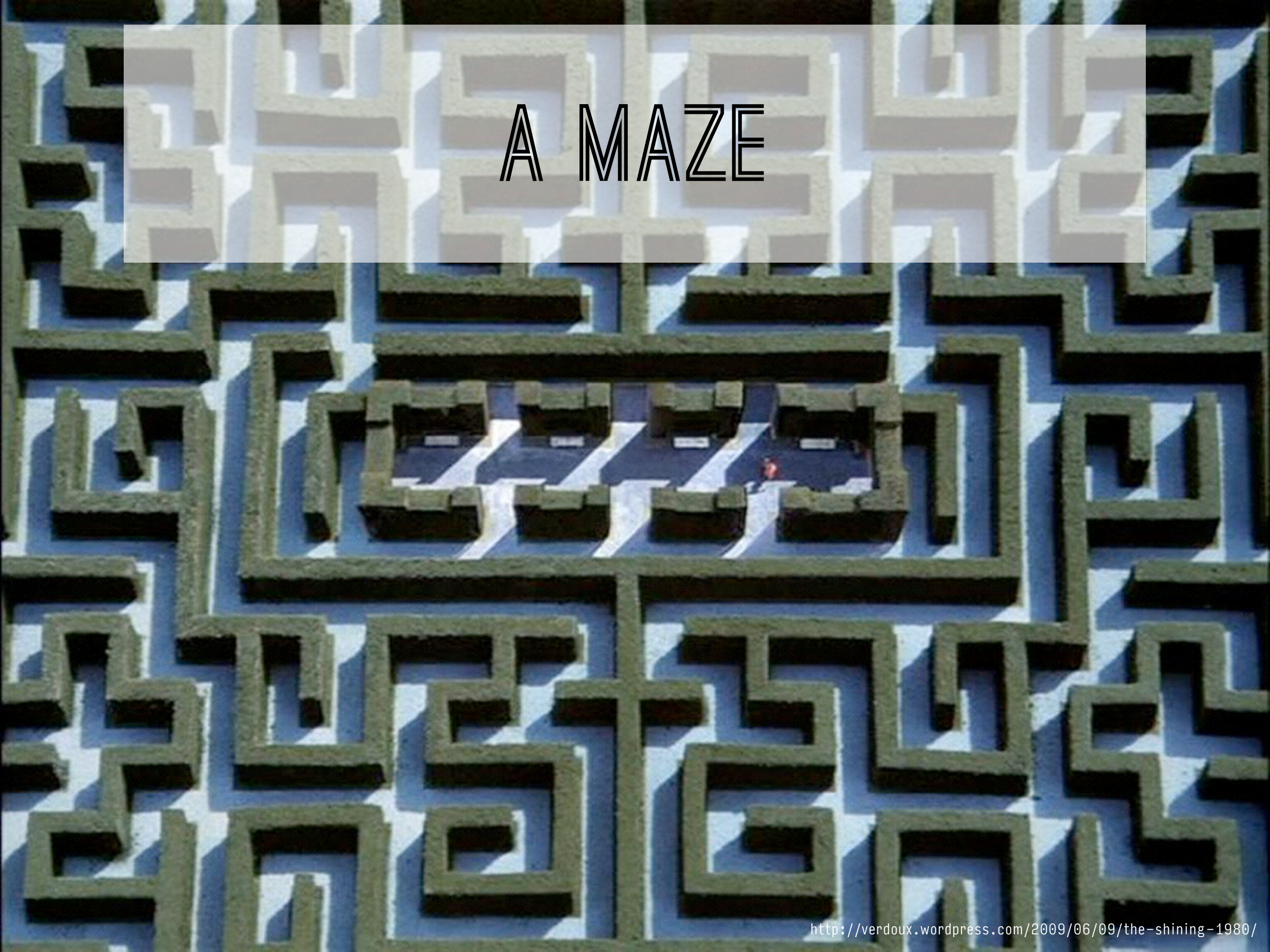## (A BRIEF INTRODUCTION)

🎉🎉🎉🍰🍰🍰 Rike–Benjamin Schuppner 🍰🍰🍰🎉🎉🎉
Institute for Theoretical Biologie | HU Berlin

rikebs@debilski.de // debilski.de // @debilski
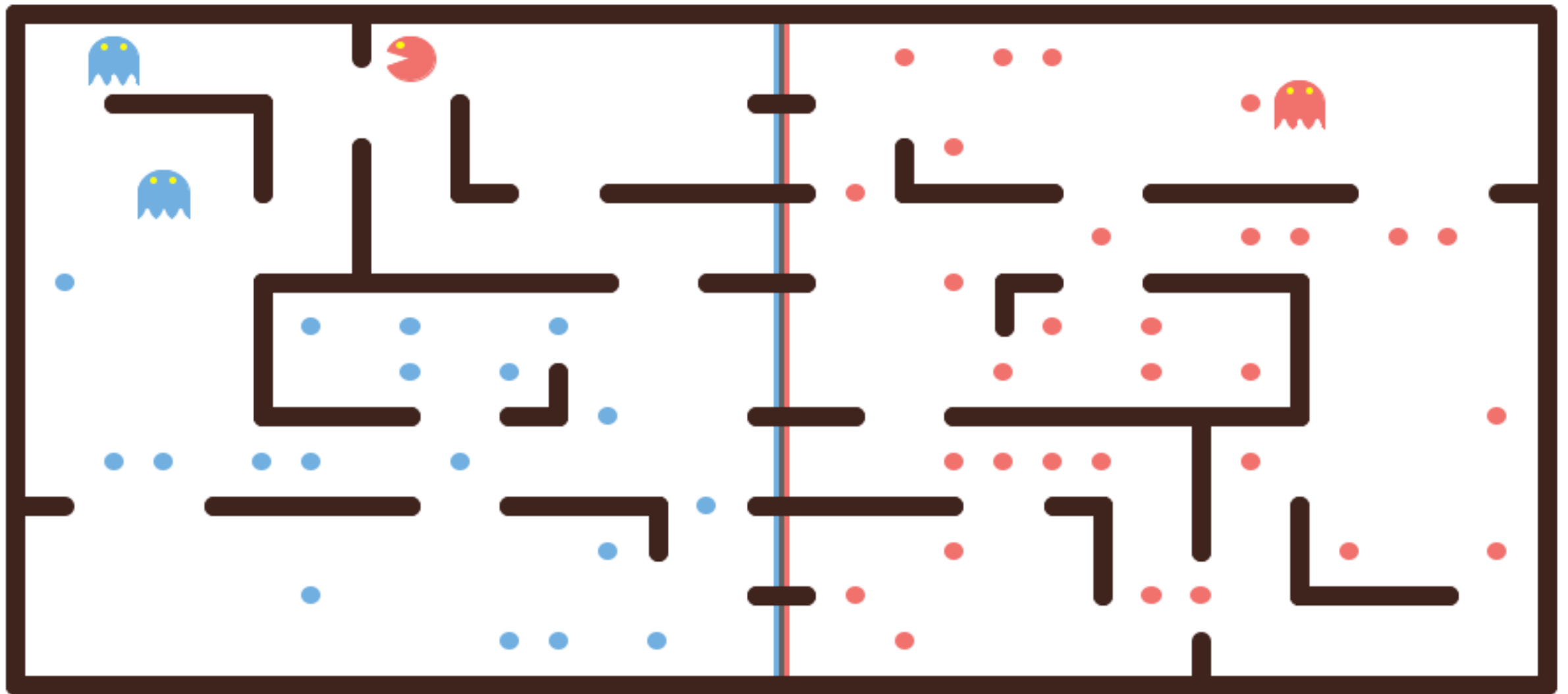
IN SHORT

A MAZE

MOVING AROUND

ENEMY BOTS

ATTACK

PELITA

BEFORE YOU ASK

- Pelita
- **A**ctor-based **T**oolkit for **I**nteractive **L**anguage **E**ducation in **P**ython
- 'Pill-eater'
- Created 2011-2012 especially for the summer school
- (Idea from John DeNero and Dan Klein, UC Berkeley[1])

[1] http://www.denero.org/content/pubs/eaai10_denero_pacman.pdf

AUTHORS

- git shortlog -sn
  Rike-Benjamin Schuppner
  Valentin Haenel
  Tiziano Zito
  Zbigniew Jędrzejewski-Szmek
  Bastian Venthur
  Pietro Berkes
  Nicola Chiapolini
  Pauli Virtanen
  abject
  Anna Chabuda
  Jakob Jordan
  Sasza Kijek
  Francesc Alted
  Christian Steigies
  Bartosz Telenczuk
  Ola Pidde

OVERVIEW
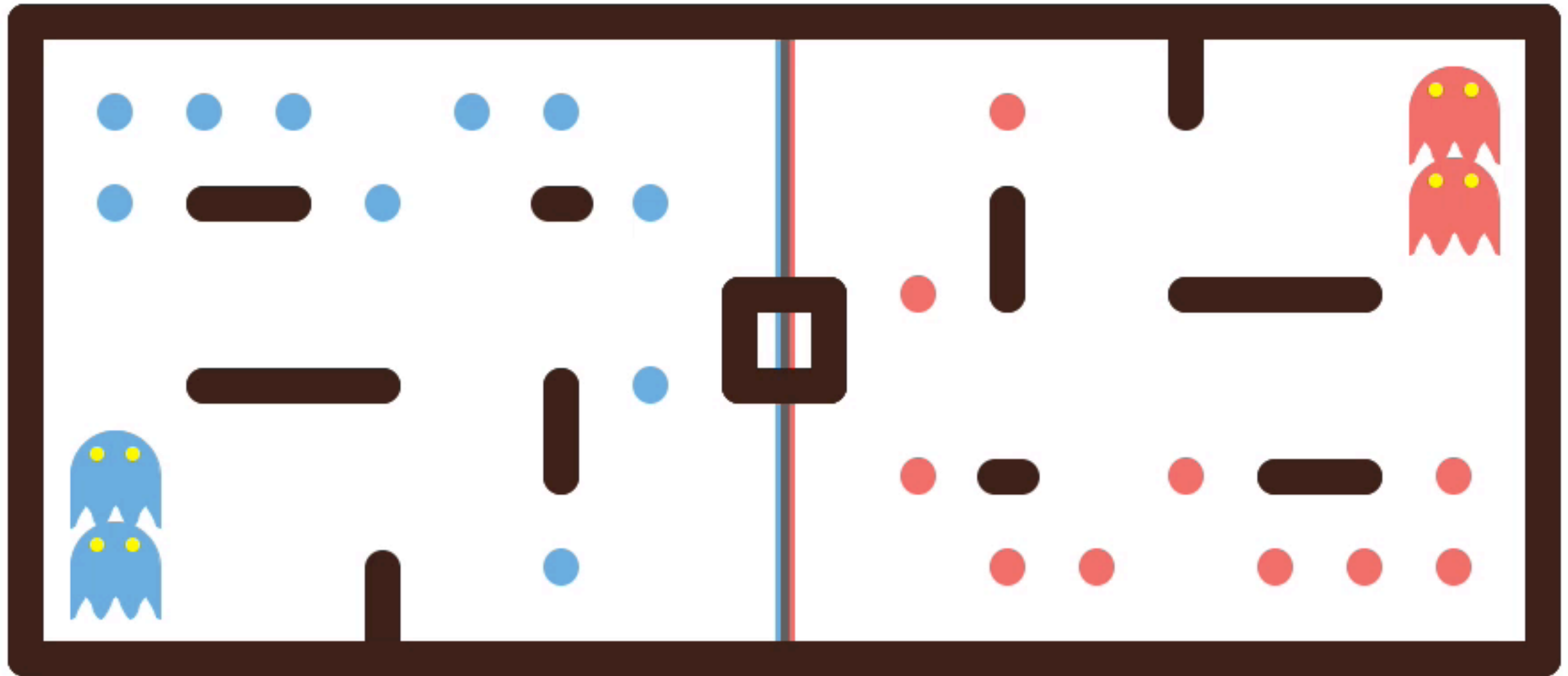
PLAY/PAUSE STEP ROUND

slower faster show grid

QUIT

OVERVIEW

The BFSPlayers 3 : 0 The RandomPlayers

Bots for team 0

Bots for team 1

QUIT

Bot 2, Round 12
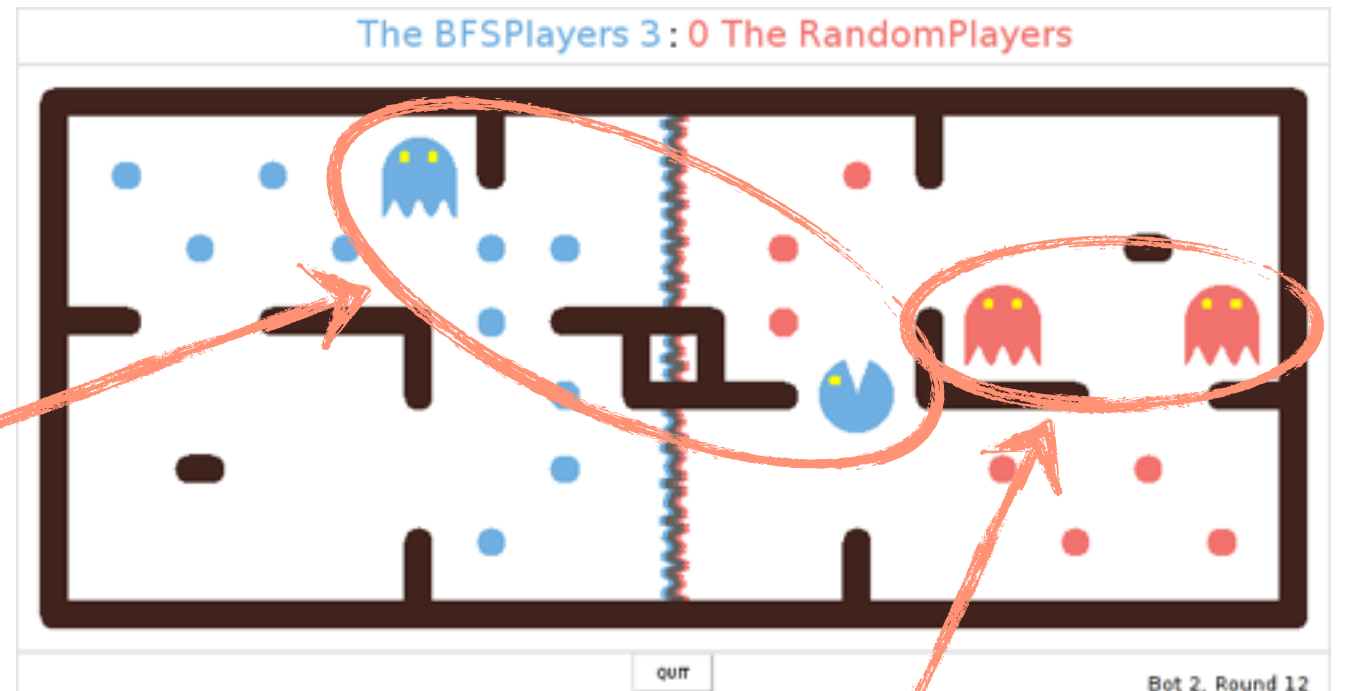
- Each Team owns two Bots

OVERVIEW

The BFSPlayers 3 : 0 The RandomPlayers

Bots for team 0

Bots for team 1

QUIT

Bot 2, Round 12
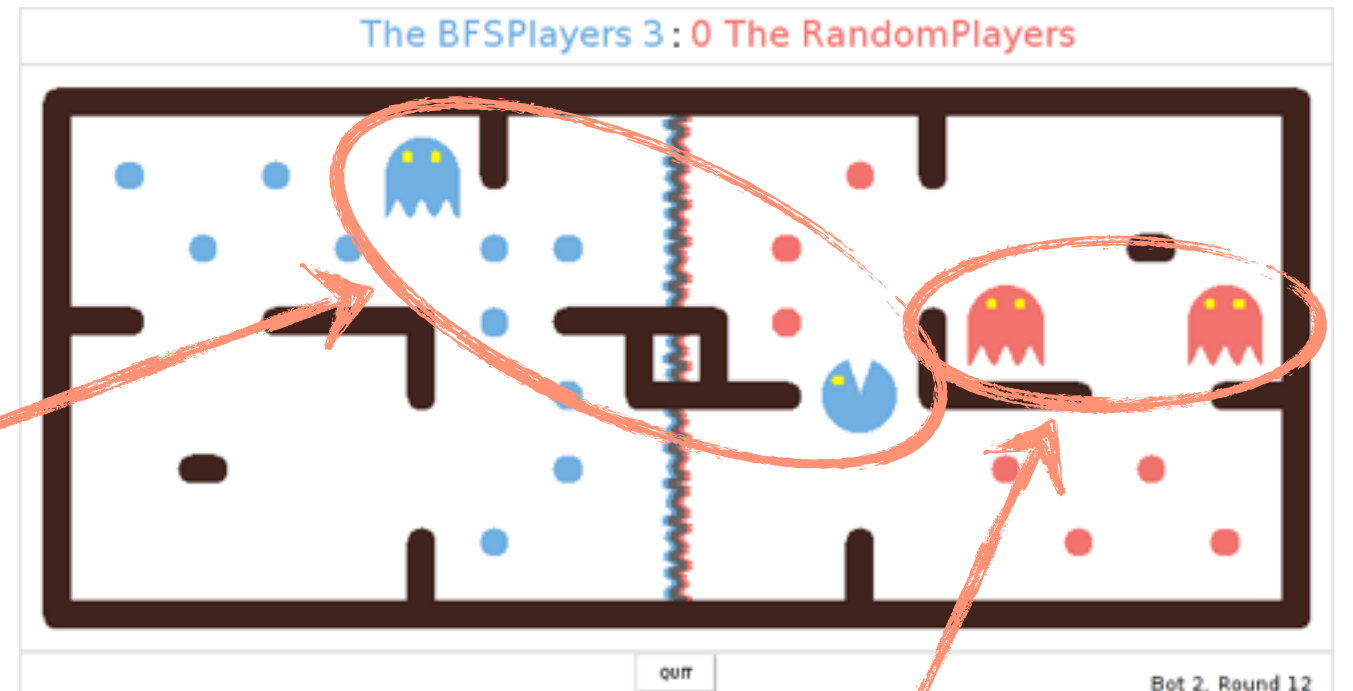
- Each Team owns two Bots
- Each Bot is controlled by a Player

OVERVIEW

- Each Team owns two Bots
- Each Bot is controlled by a Player
- Harvester or Destroyer Bots

OVERVIEW

- Each Team owns two Bots
- Each Bot is controlled by a Player
- Harvester or Destroyer Bots
- Bots are Destroyers in homezone
- Harvesters in enemy's homezone
- Game ends when all food pellets are eaten

THE RULES

THE RULES

- **Eating**: When a Bot eats a food pellet, the food is permanently removed and one point is scored for that Bot's team.

- **Timeout**: Each Player only has 3 seconds to return a valid move. If it doesn't, a random move is executed. (All later return values are discarded.)
5 timeouts and you're out!

- **Eating another Bot**: When a Bot is eaten by an opposing destroyer, it returns to its starting position (as a harvester). 5 points are awarded for eating an opponent.

- **Winning**: A game ends when either one team eats all of the opponents' food pellets, or the team with more points after 300 rounds.

- **Observations**: Bots can only observe an opponent's exact position, if they or their teammate are within 5 squares of the opponent bot. If they are further away, the opponent's positions are noised.

CONTROLLING THE BOTS
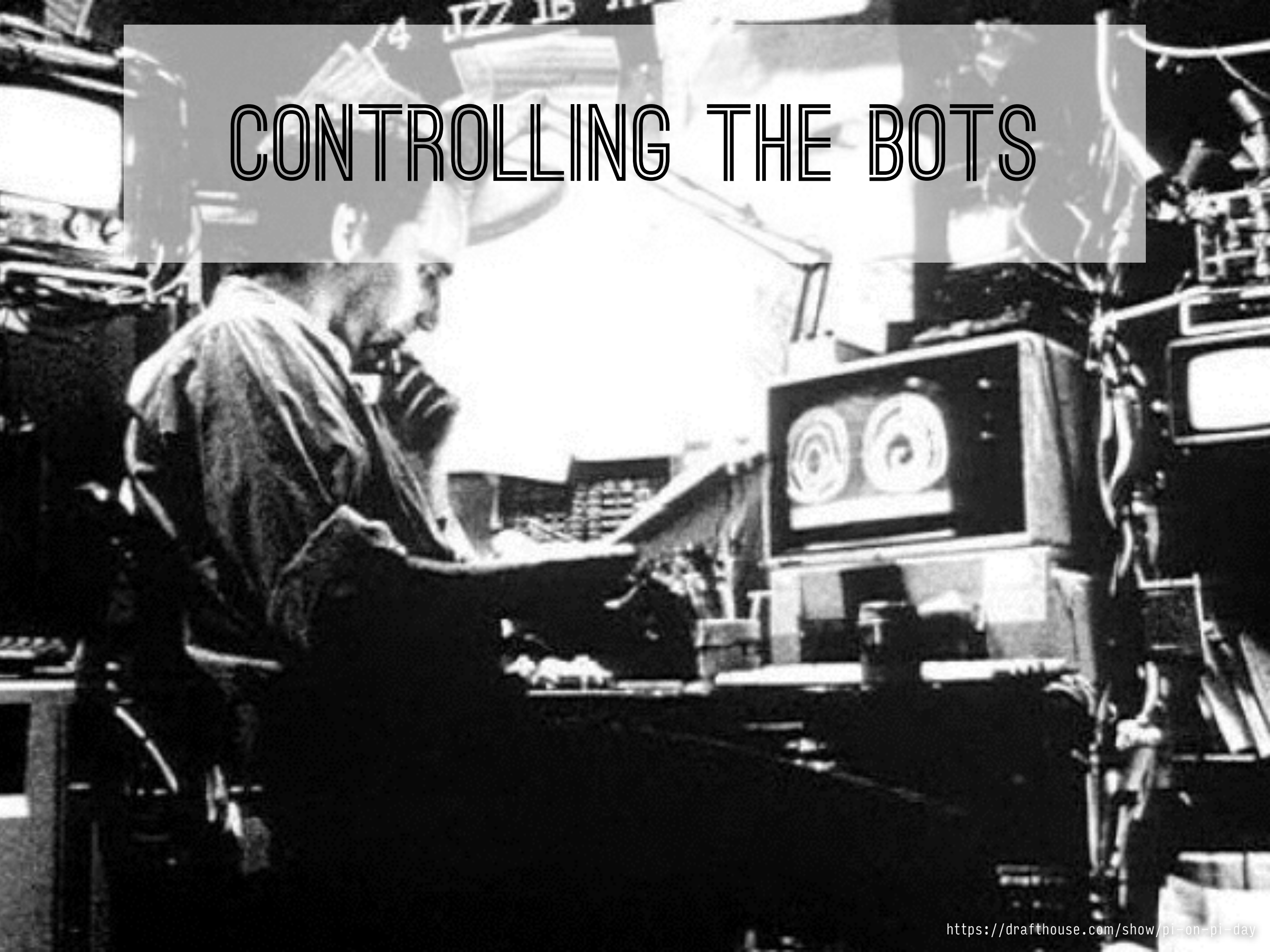
MY FIRST PLAYERS

```python
from pelita.datamodel import east
from pelita.player import AbstractPlayer

class UnidirectionalPlayer(AbstractPlayer):
    def get_move(self):
        return east

class DrunkPlayer(AbstractPlayer):
    def get_move(self):
        directions = self.legal_moves
        random_dir = self.rnd.choice(directions)
        return random_dir
```

- Careful: Invalid return values of get_move result in a random move.

API EXAMPLES

- In your get_move method, information about the current universe and food situation is available. See the documentation for more details.

- self.current_pos
    Where am I?

- self.me
    Which bot am I controlling?

- self.enemy_bots
    Who and where are the other bots?

- self.enemy_food
    Which are the positions of the food pellets?

- self.current_uni
  Retrieve the universe you live in.

- self.current_uni.maze
  How does my world look like?

- self.legal_moves
  Where can I go?

- self.me.is_destroyer
  Am I dangerous?

BUILDING A TEAM

- A team consists of two players (and a name)
- Create it using the SimpleTeam class
  - SimpleTeam("Magnificent Team", GoodPlayer(), RemarkablePlayer())
- Export your team using the factory function

  - def factory():
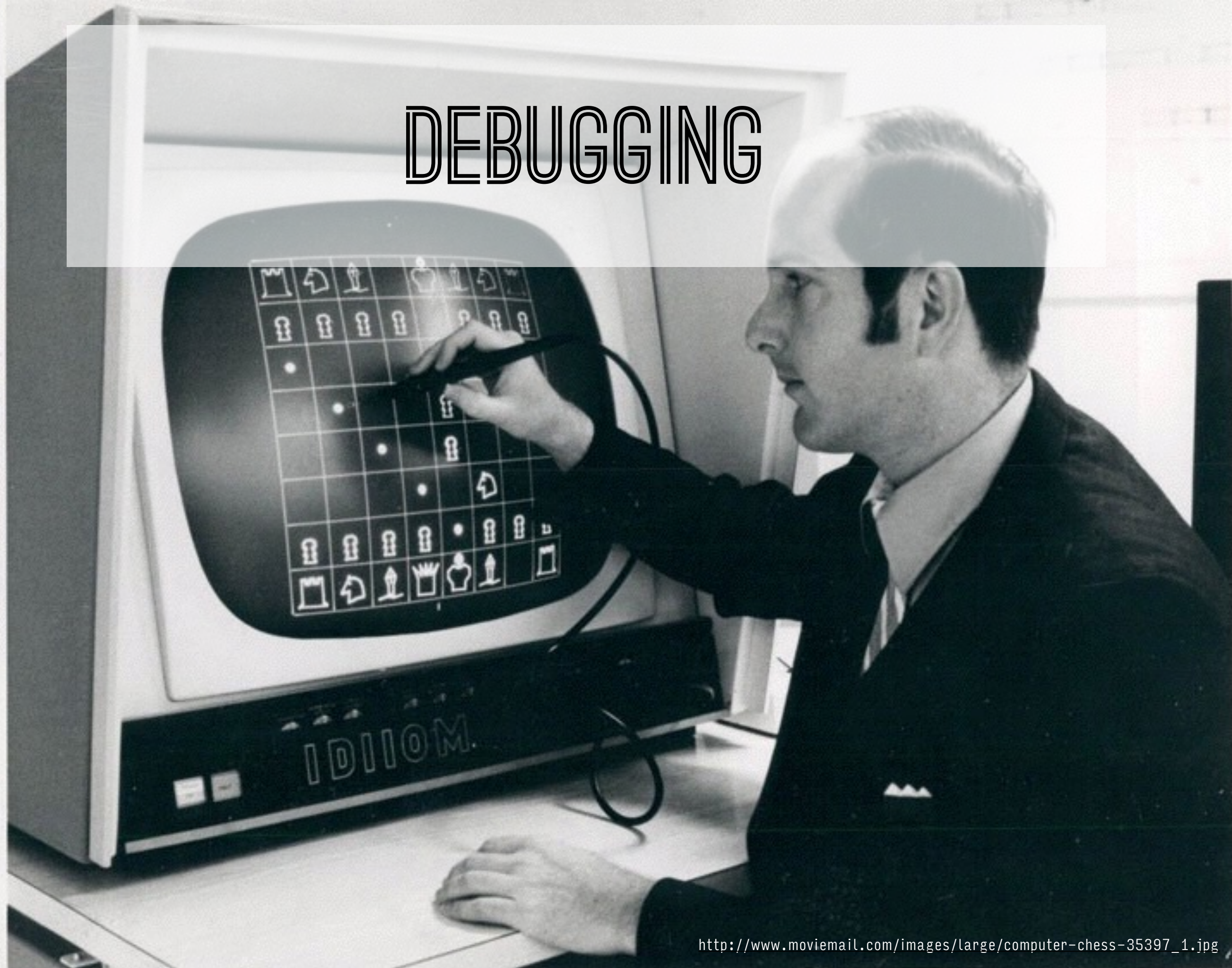   return SimpleTeam(...)

DEMO BOTS

- In ./players directory
- There are hidden bots on our servers
  - We tell you how to use them when it's time

DEMO TIME

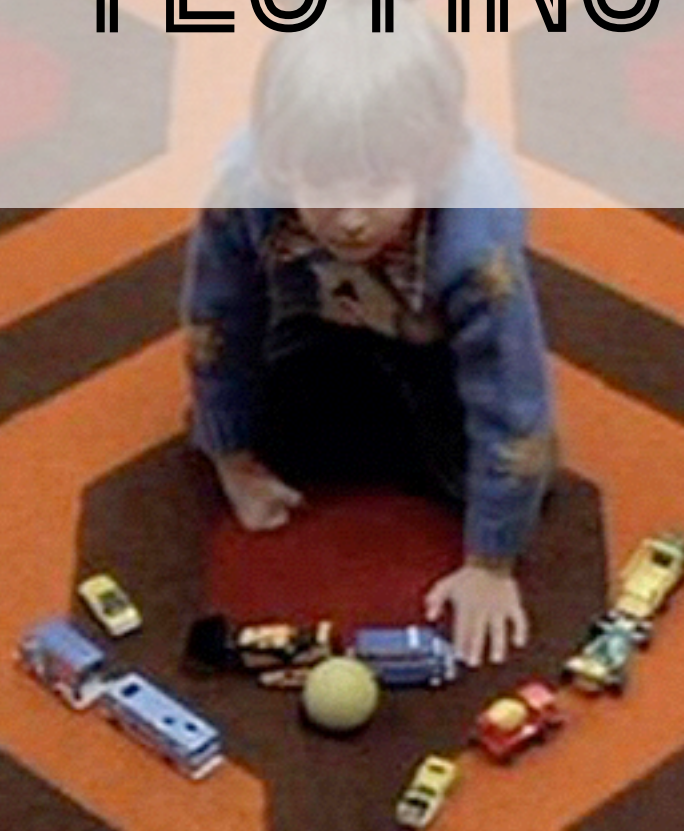- Now, let's build an example player

DEBUGGING

DEBUGGING

- Use a pre-defined DebuggablePlayer to explore the API

- class DebuggablePlayer(AbstractPlayer):
    ```
    def get_move(self):
        direction = datamodel.stop
        pdb.set_trace()
        return direction
    ```

- ./pelitagame --no-timeout DebuggablePlayer
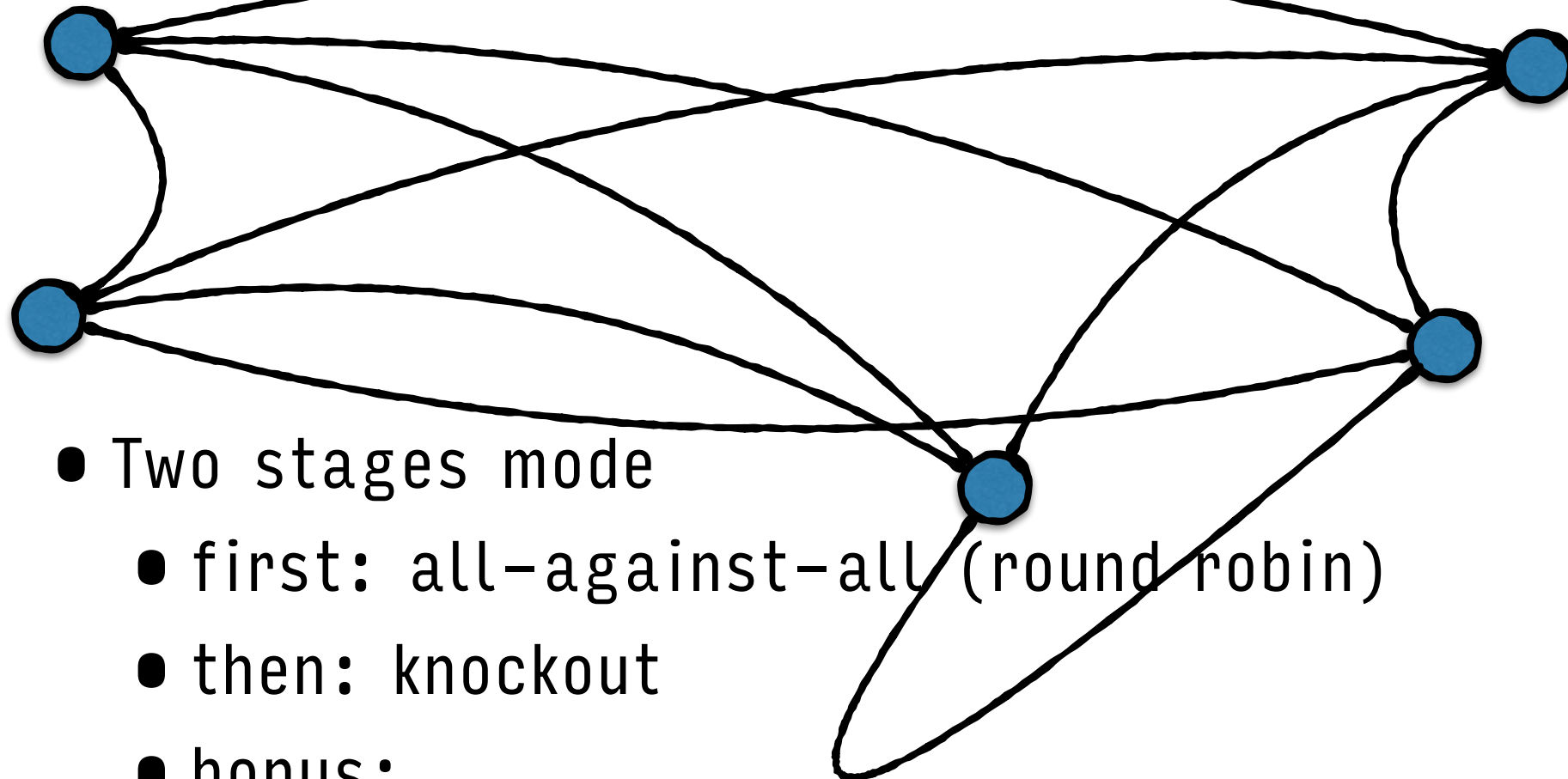
- (Pdb) p self.me

TESTING

TESTING

- Two ways to test your Players
- first: Simply run the game and test by watching
  - $ ./pelitagame MyTeam EnemyTeam
- second: Write unittests and test by testing
  - Example in the template
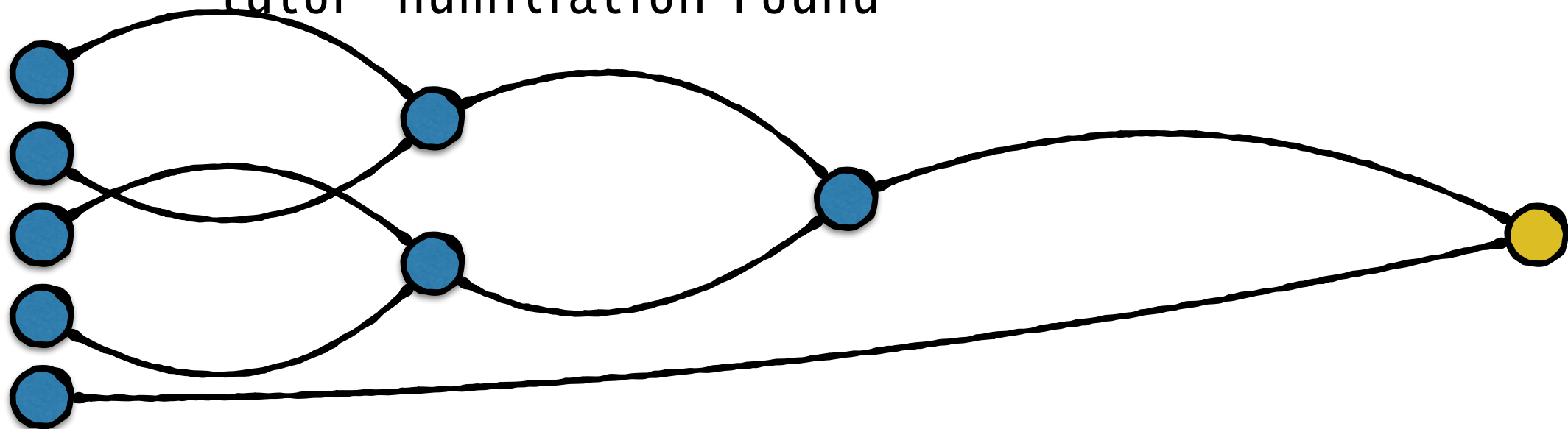
TOURNAMENT

TOURNAMENT

- Two stages mode
  - first: all-against-all (round robin)
  - then: knockout
  - bonus:
    tutor-humiliation round

TOURNAMENT

- Clone the group repository
- It contains a module in team/. (Uses __init__.py)
- Exports a 'factory' method:

```
def factory():
    return SimpleTeam("The Winners", MyPlayer(),
MyPlayer())
```

- Run it as
    ./pelitagame ../groupN/team
- Additionally contains util and testing repositories

NOTES ON WRITING

NOTES ON WRITING

- Mazes don't have dead-ends
- Hard to catch another bot which outruns you
- We'd like to see bots which combine their powers and attack from two sides

NOTES ON WRITING

- Think about shortest-path algorithms
- Keep track of opponents
- Investigate communication between the Players
- Re-use your code
- Think about working in a team

NOTES ON WRITING

- Use the internal random number generator:
- instead of
  - random.choice
- you use
  - self.rnd.choice
- (more stable)

NOTES ON WRITING

- The match environment:
  - numpy is installed
  - also: pylint (just so you know)
  - additional packages may or may not be negotiable

GETTING READY

- Clone the pelita and group repos:
  git clone https://github.com/ASPP/pelita.git
  git clone https://github.com/ASPP/groupN.git

- Run a simple demo game:
  ~/pelita/pelitagame groupN/team

- For help:
  ~/pelita/pelitagame --help

- See the Pelita documentation:
  https://ASPP.github.io/pelita

- Questions? Ask us.

- Vent your frustration: #aspp2016

- Or on Facebook: Kung-Fu Python

REPO CLOSES

REPO CLOSES

SATURDAY, 5PM.

MOVIE STILLS

- 'Them' (1954, dir. Gordon Douglas)
- 'The Ten Commandments' (1956, dir. Cecil B. DeMille)
- 'Det sjunde inseglet' (1957, dir. Ingmar Bergman)
- 'Smultronstället' (1957, dir. Ingmar Bergman)
- 'The Shining' (1980, dir. Stanley Kubrick)
- 'Pi' (1998, dir. Darren Aronofsky)

- 'Computer Chess' (2013, dir. Andrew Bujalski)