THE PELITA CONTEST

(A BRIEF INTRODUCTION)

ADVANCED SCIENTIFIC PROGRAMMING IN PYTHON
#ASPPAPAC2018

THE PELITA CONTEST

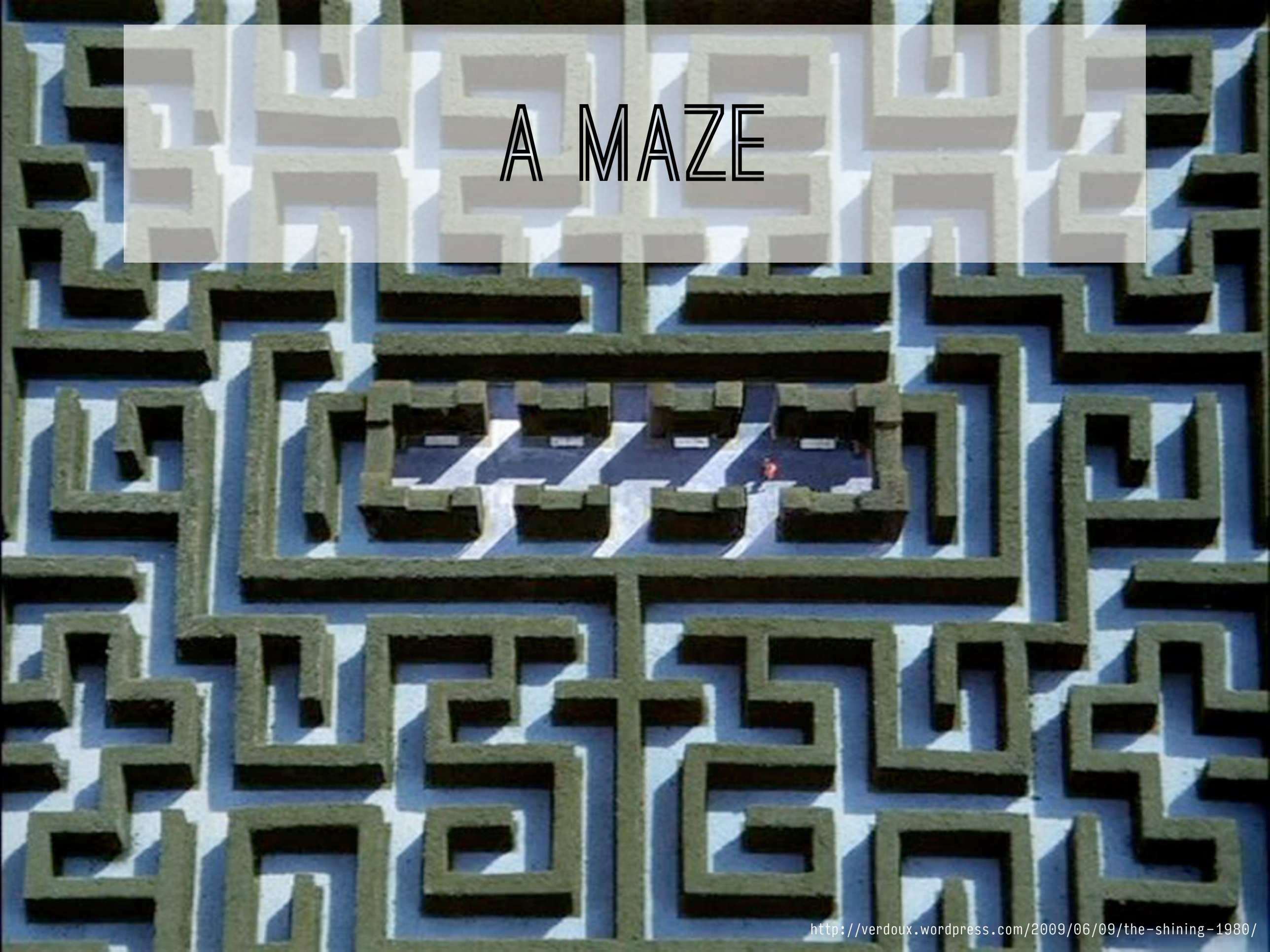
(A BRIEF INTRODUCTION)

Rike-Benjamin Schuppner
Institute for Theoretical Biologie | HU Berlin

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

IN SHORT

A MAZE

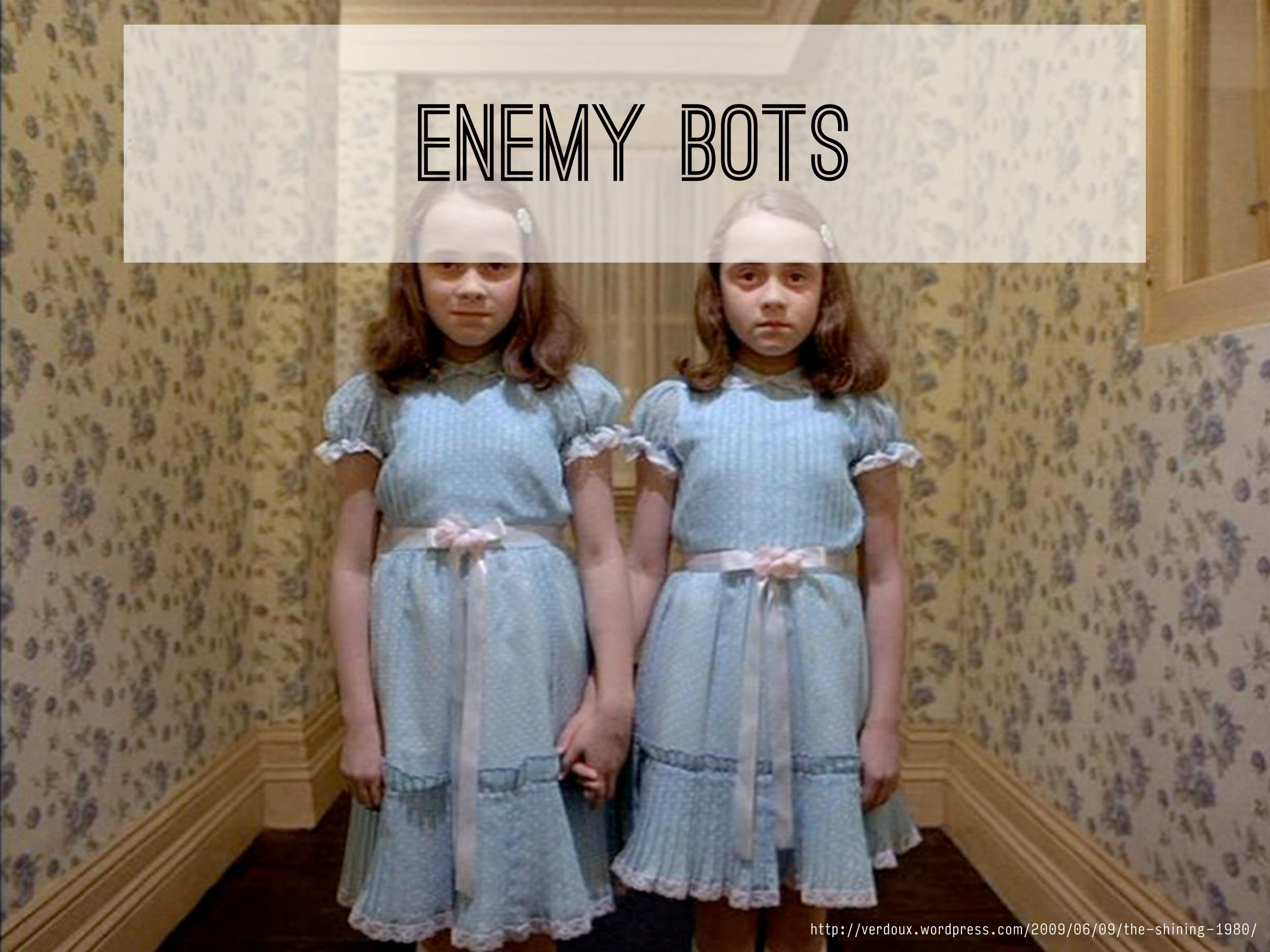


http://verdoux.wordpress.com/2009/06/09/the-shining-1980/

MOVING AROUND

http://verdoux.wordpress.com/2009/06/09/the-shining-1980/

ENEMY BOTS

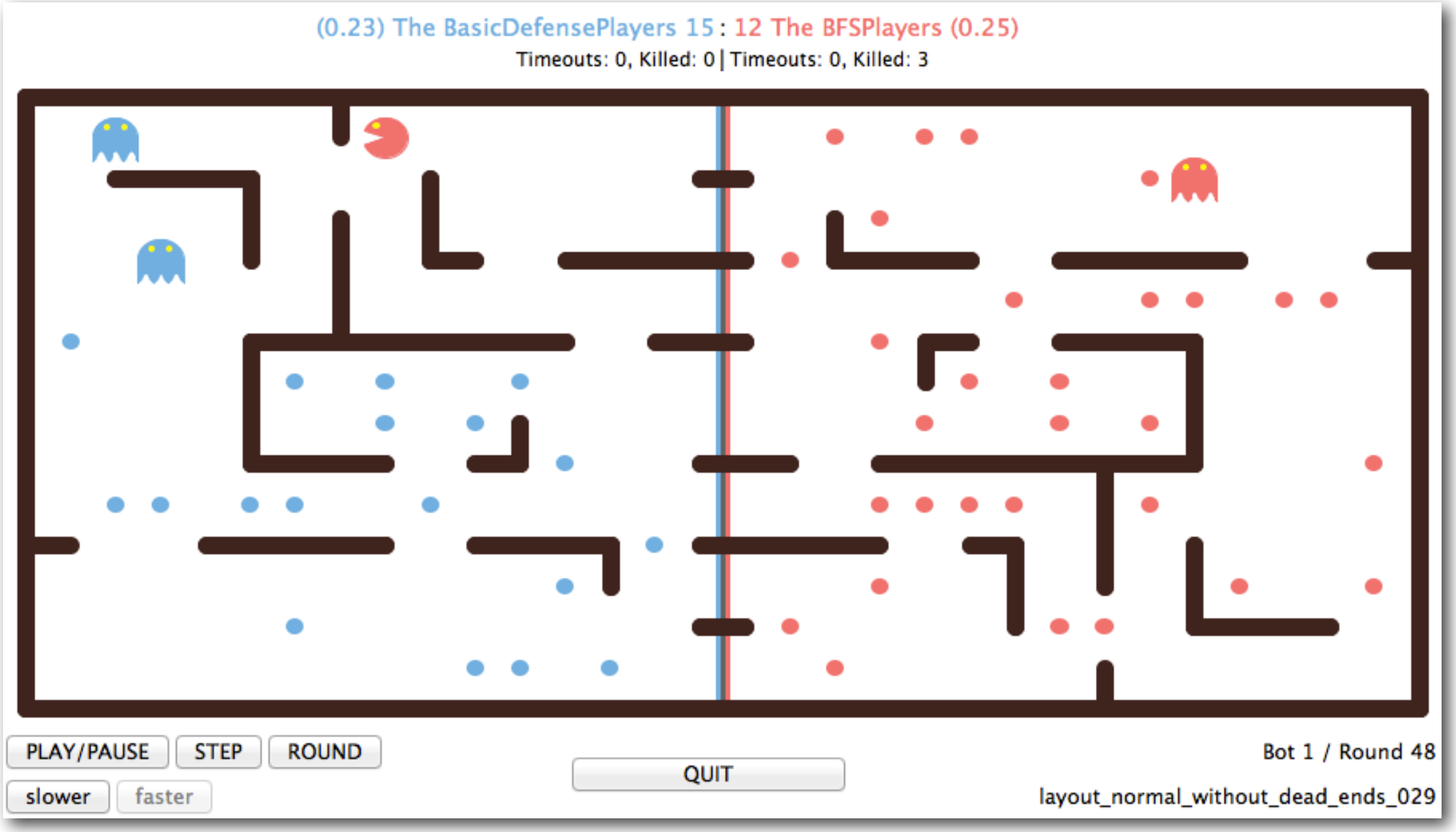


http://verdoux.wordpress.com/2009/06/09/the-shining-1980/

ATTACK

http://img.timeinc.net/time/photoessays/2008/top10_1950s/top10_1950s_them.jpg

PELITA


(0.23) The BasicDefensePlayers 15 : 12 The BFSPlayers (0.25)
Timeouts: 0, Killed: 0 | Timeouts: 0, Killed: 3
PLAY/PAUSE
STEP
ROUND
slower
faster
QUIT
Bot 1 / Round 48
layout_normal_without_dead_ends_029

BEFORE YOU ASK

- Pelita
- Actor-based Toolkit for Interactive Language Education in Python
- 'Pill-eater'
- Created 2011–2012 especially for the summer school
- (Idea from John DeNero and Dan Klein, UC Berkeley[1])

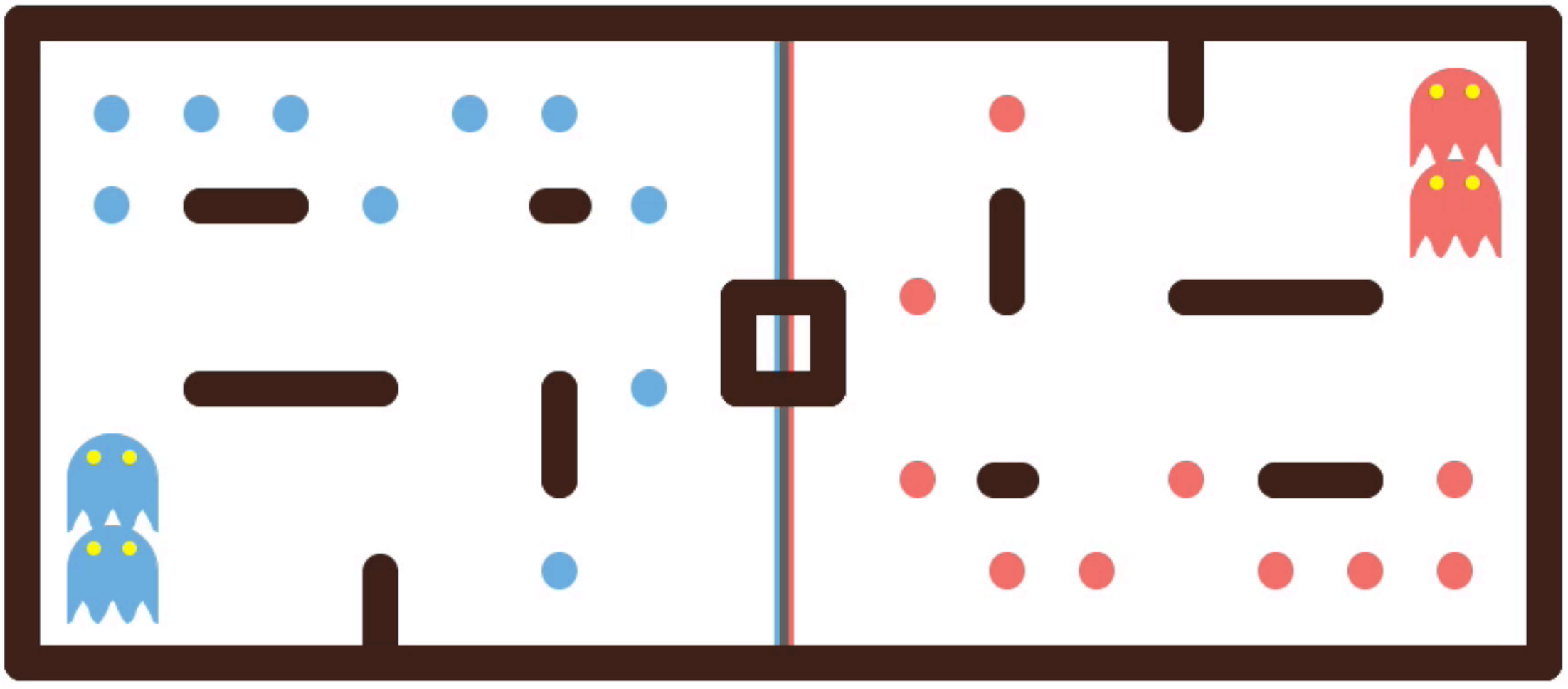
[1] http://www.denero.org/content/pubs/eaai10_denero_pacman.pdf

AUTHORS

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

OVERVIEW

(0.00) The FoodEatingPlayers 0 : 0 The RandomExplorerPlayers (0.00)

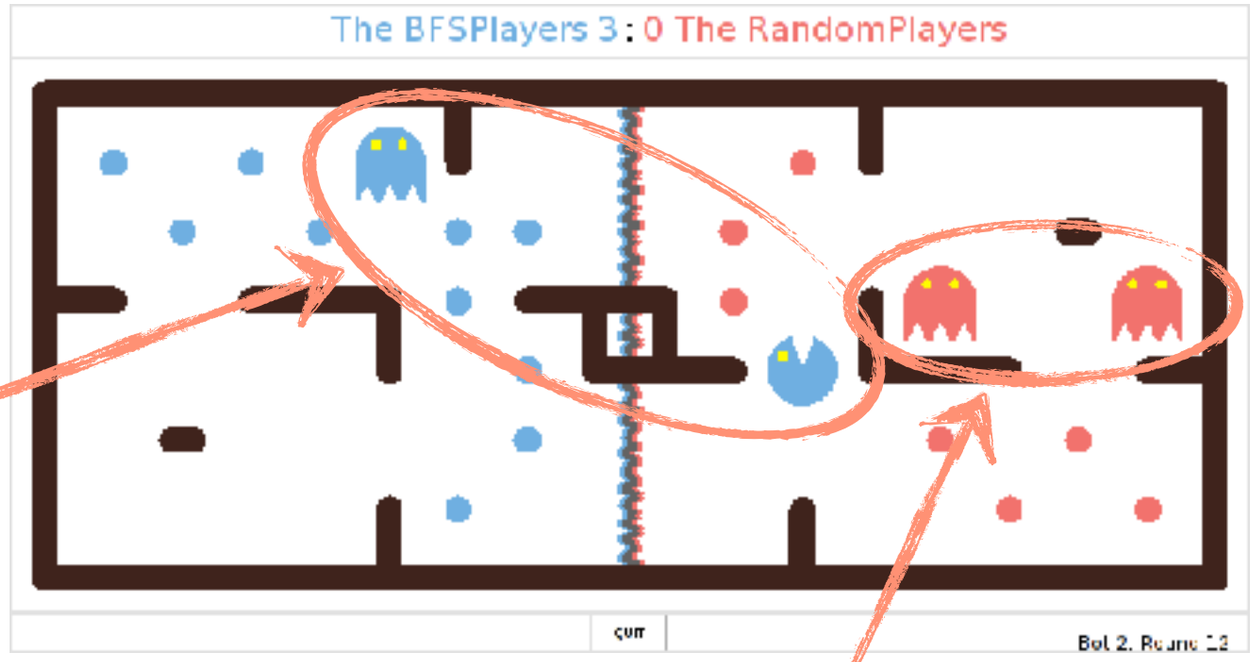


PLAY/PAUSE | STEP | ROUND

slower | faster | show grid

QUIT

OVERVIEW

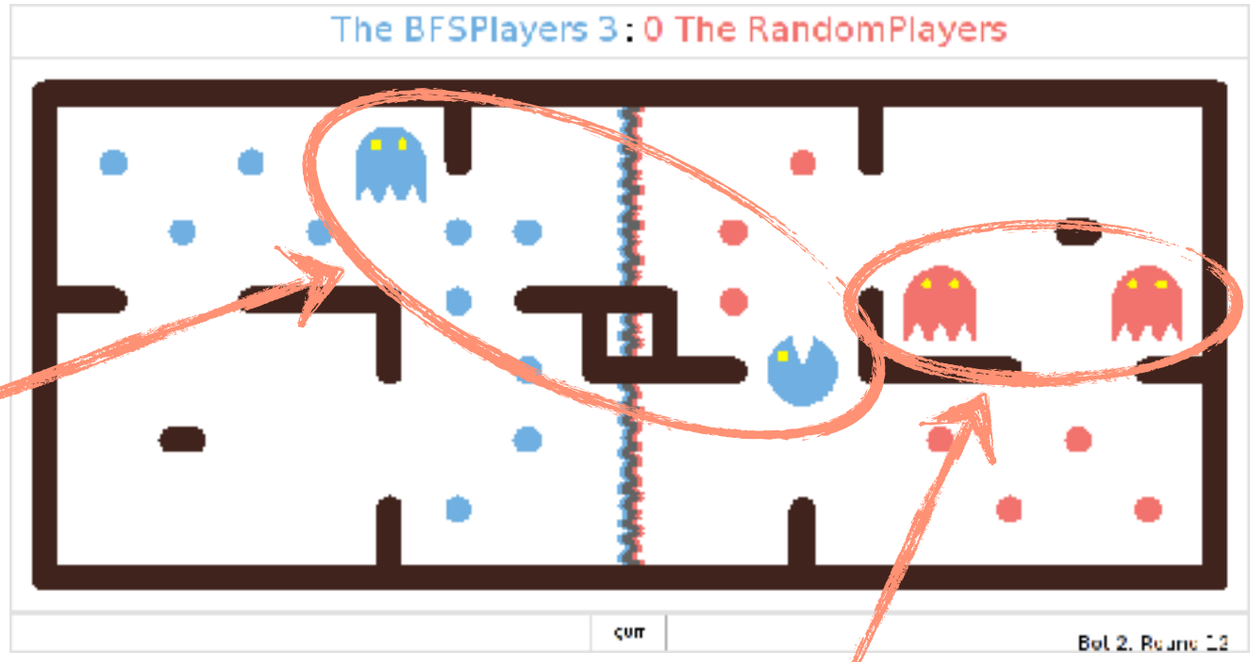


Bots for team 0

Bots for team 1

- Each Team owns two Bots

OVERVIEW



Bots for team 0

Bots for team 1

- 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

CONTROLLING THE BOTS

MY FIRST PLAYERS

```
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.
  (You could wrap get_move in a decorator, if you are unsure.)

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 `team` function
  - ```
    def team():
        return SimpleTeam(...)
    ```
    ```

THE RULES

http://filmscoreclicktrack.com/2013/01/the-ten-commandments-of-film-music/

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.

DEMO TIME

- Now, let’s build an example player

DEMO BOTS

- In pelita/player directory
- There are hidden bots on our servers
  - We tell you how to use them when it’s time

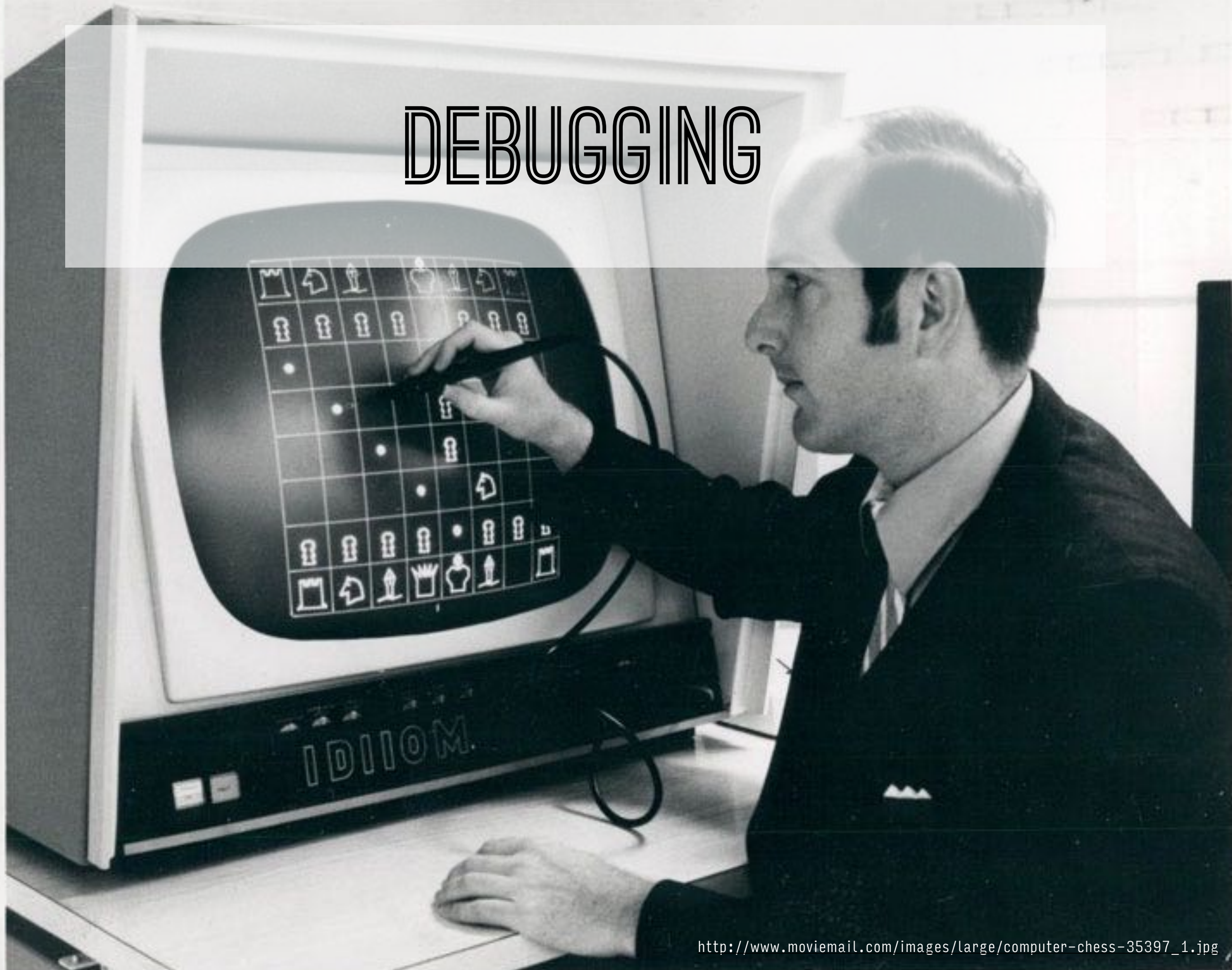
TESTING

http://verdoux.wordpress.com/2009/06/09/the-shining-1980/

TESTING

- Two ways to test your Players
- first: Simply run the game and test by watching
  - $ pelita MyTeam EnemyTeam
- second: Write proper tests and test by testing
  - Example in the template

DEBUGGING



http://www.moviemail.com/images/large/computer-chess-35397_1.jpg

DEBUGGING

- Use a pre-defined DebuggablePlayer to explore the API
- class DebuggablePlayer(AbstractPlayer):

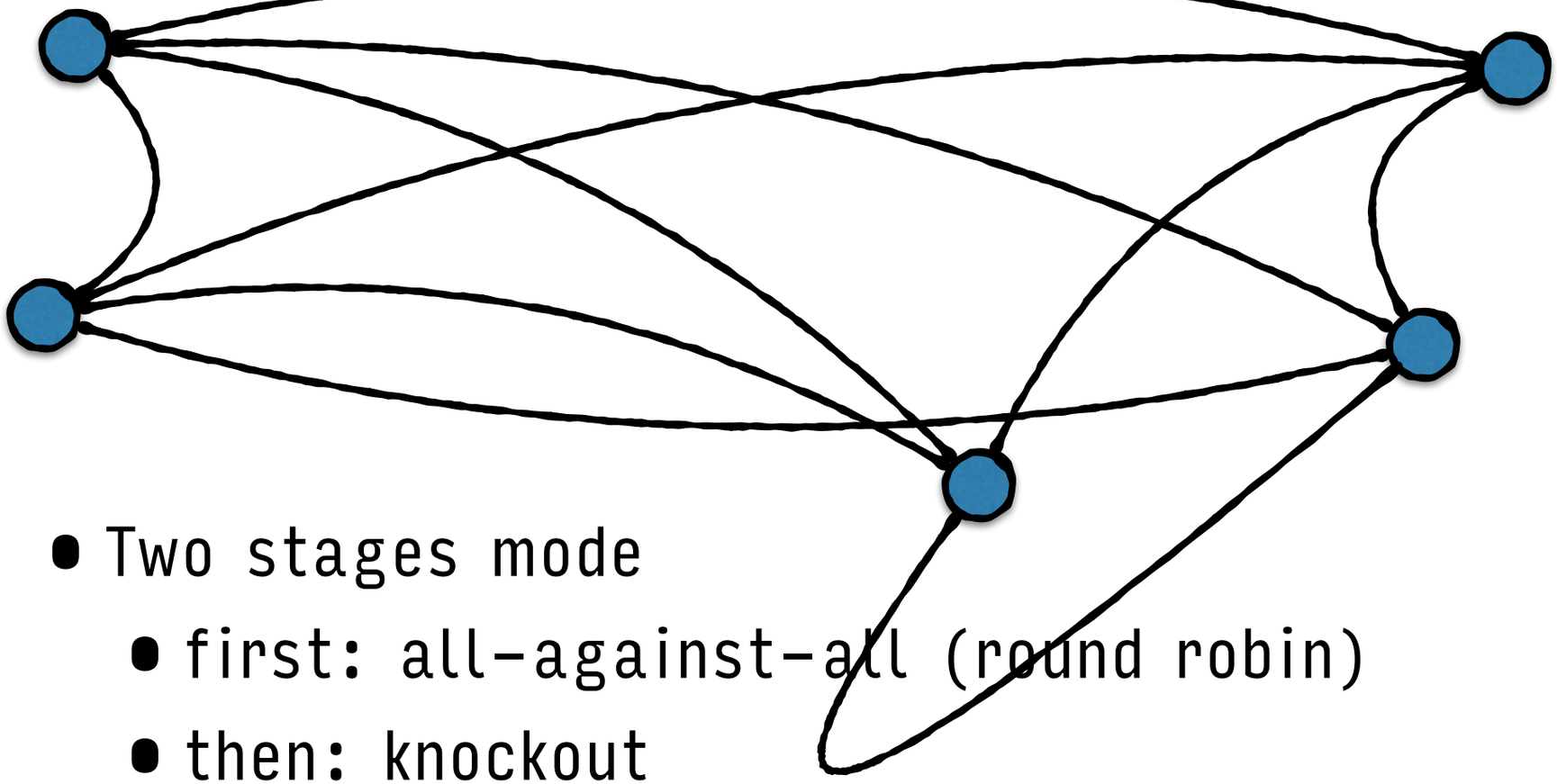
```
    def get_move(self):
        direction = datamodel.stop
        pdb.set_trace()
        return direction
```

- pelita --no-timeout DebuggablePlayer
- (Pdb) p self.me

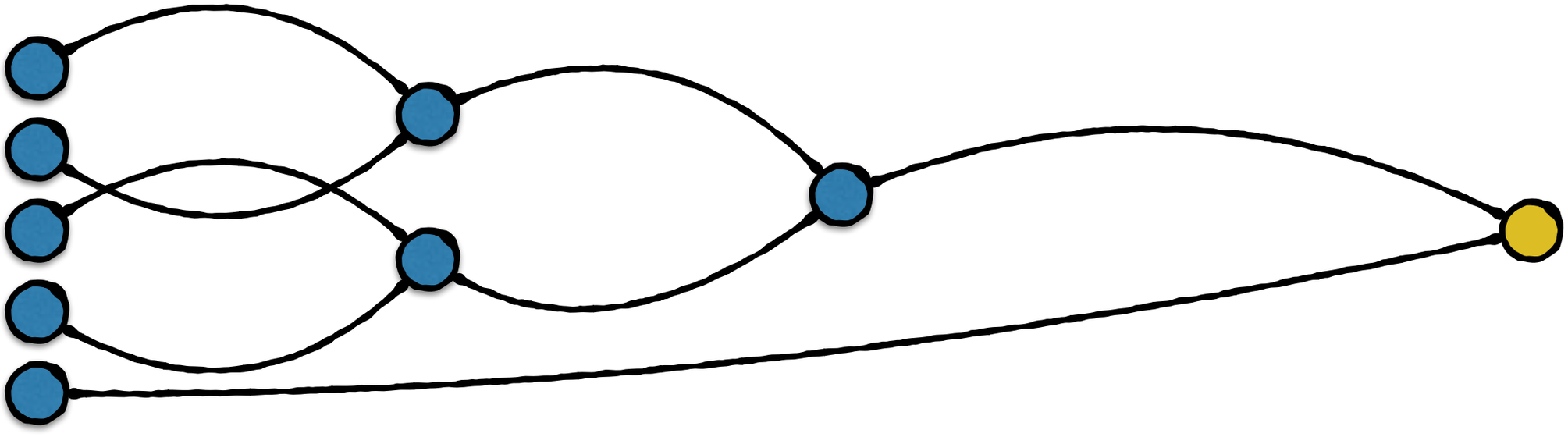
TOURNAMENT

http://magiaeimagem.files.wordpress.com/2010/02/ingmar-bergman-the-seventh-seal.jpg

TOURNAMENT



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



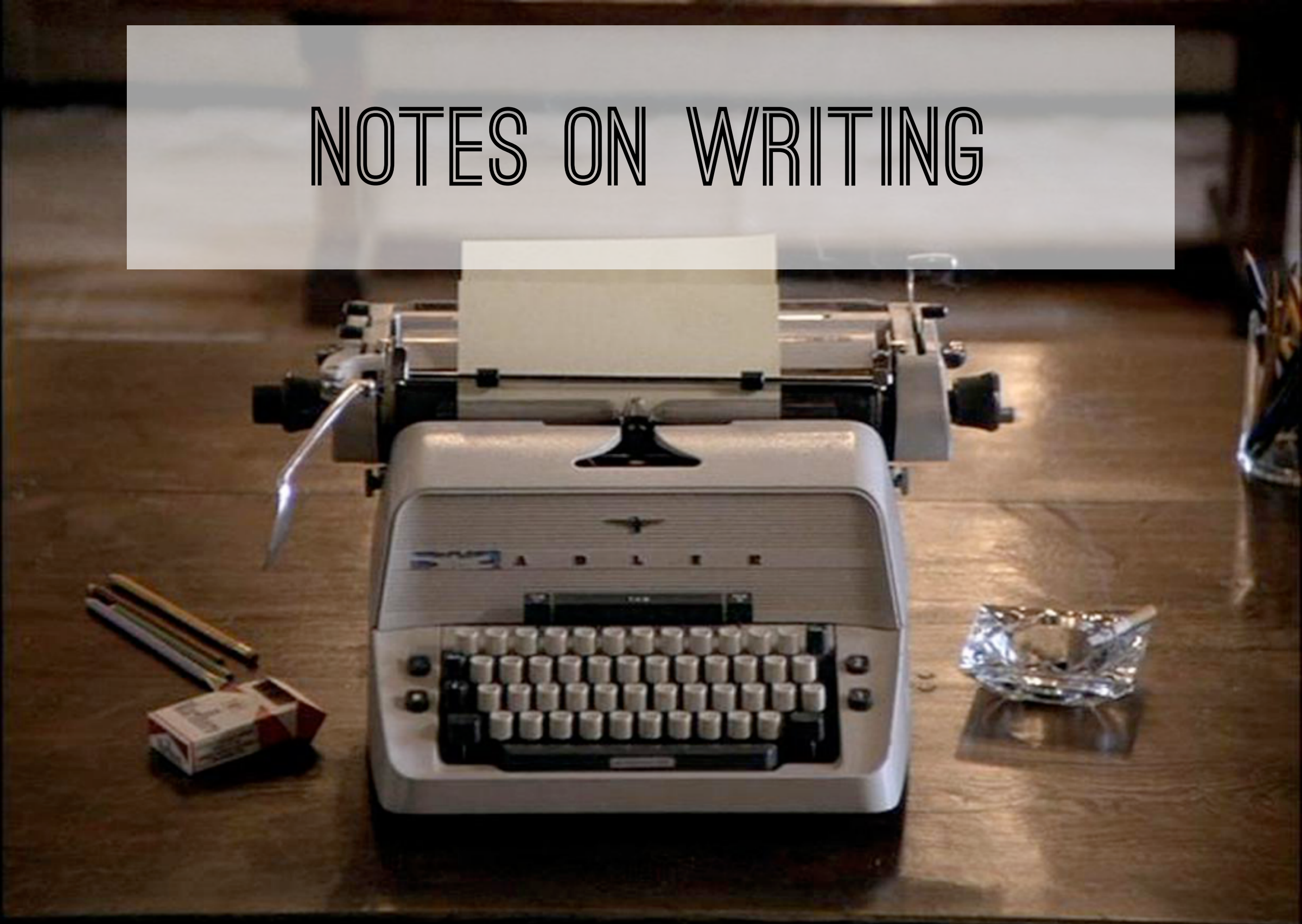
TOURNAMENT

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

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

- Run it as
  pelita groupN/team
- Additionally contains util and testing repositories
- Test with py.test or simply run ‘make test’
- Team can run on either side of the map. Quick test with ‘make left’, ‘make right’.

NOTES ON WRITING



http://verdoux.wordpress.com/2009/06/09/the-shining-1980/

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
- Layouts are fixed and you can check them all before
- (There are pathological layouts in layouts/bad. Useful to find edge cases.)

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:
  - standard anaconda installation
  - also: pylint (just so you know)
  - additional packages may or may not be negotiable

TIMELINE

- Thursday 6pm–8pm: Group work
- Friday 2pm–6pm: Group work
- Friday 6pm: First report
- Saturday 8.30am–17pm: Group work
- Saturday 17.30pm: Final report and Tournament
- Thereafter: alcohol-fuelled discussions about that one bug that was responsible for it all

GETTING READY

- Clone the pelita and group repos:
  `git clone https://github.com/ASPP/pelita.git`
  `git clone https://github.com/ASPP/groupN.git`
- Install pelita:
  `pip install git+https://github.com/ASPP/pelita.git`
- Run a simple demo game:
  `cd groupN`
  `pelita groupN/team`
- For help:
  `pelita --help`
- See the Pelita documentation:
  https://ASPP.github.io/pelita
- Questions? Ask the tutors.

REPO CLOSES

http://25.media.tumblr.com/tumblr_m3vmn8RTTH1qathv6o1_500.png

REPO CLOSES

SATURDAY, 5PM.

http://25.media.tumblr.com/tumblr_m3vmn8RTTH1qathv6o1_500.png

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)