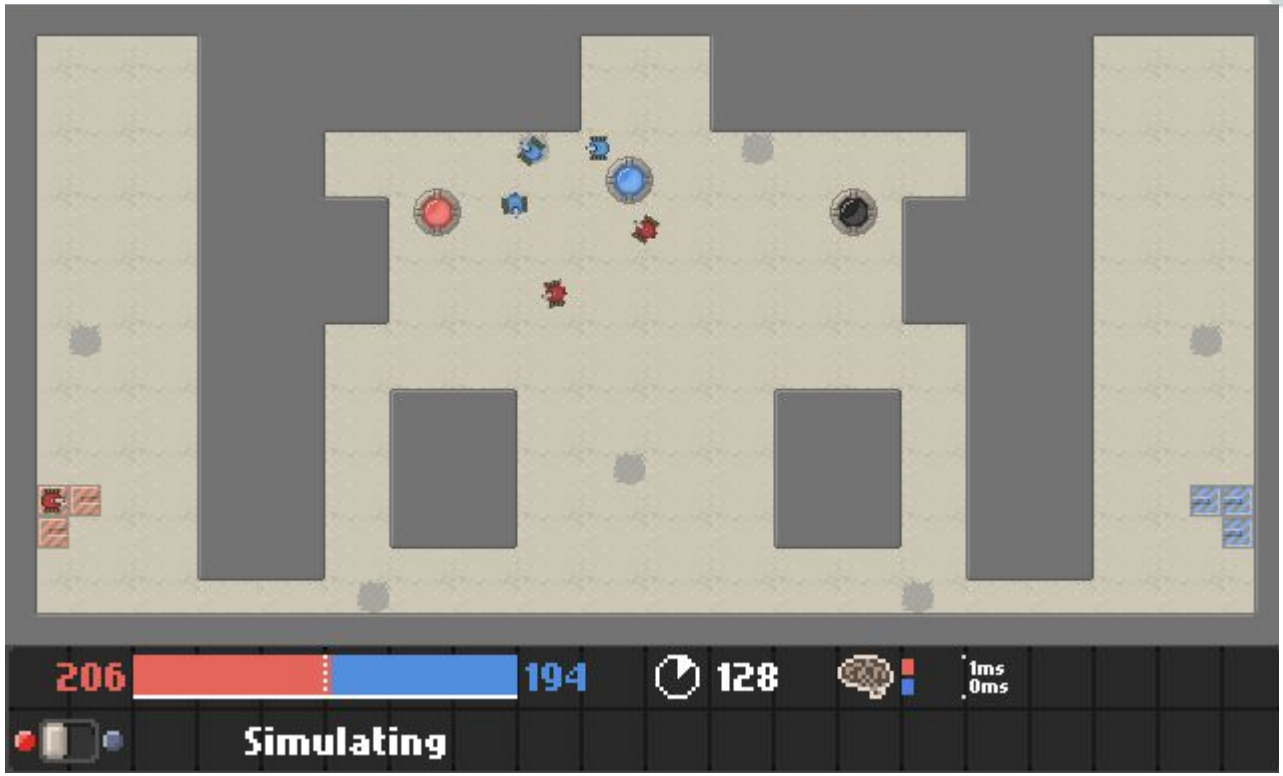
CAT 2018

Getting you started with the Domination Game

0.

Game Demo

Game Demo



1. Installation

Installation

1. Install Python 2 (Miniconda heavily recommended, see https://conda.io/miniconda.html)

2. Install pygame (`pip install pygame`) https://www.pygame.org

3. Clone or download the CAT 2018 GitHub repository (https://github.com/Nathanaelion/CAT2018)

4. Run demo.py to see if everything works (`python demo.py`)

2.

Running a Basic Game

Running a Basic Game

```
from domination import core

# create Game object with default settings
game = core.Game()

# run game
game.run()

# access stats
print game.stats.score_red
```

The GameStats Object

Property	Description
`score_red / score_blue`	scores of respective teams
`score`	score as float (red / total)
`steps`	number of steps the game lasted
`ammo_red / ammo_blue`	number of collected ammo packs
`deaths_red / deaths_blue`	number of deaths
`think_time_red / think_time_blue`	total computation time in seconds

3.

Customizing Settings

Customizing Settings

```
# create Settings object
settings = core.Settings(
    max_steps = 200,
    max_score = 400,
    # ...
    think_time = 0.1
)

# pass settings when creating Game object
game = core.Game(settings = settings)
```

Tournament Settings*

Setting	Value
max_steps	150
max_score	400
field_known	True
ammo_amount	3
spawn_time	10
think_time	0.05**

** all settings except those in red have default values (see doc)*
*** think time can differ on each machine - if problematic talk to us*

4.

Customizing the Field

Customizing the Field - Generator

```
# create FieldGenerator with parameters
generator = core.FieldGenerator(
    width = 41,
    height = 24,
    # ...
    num_points = 3
)

# FieldGenerator.generate() return Field instance
field = generator.generate()
```

Customizing the Field - From String

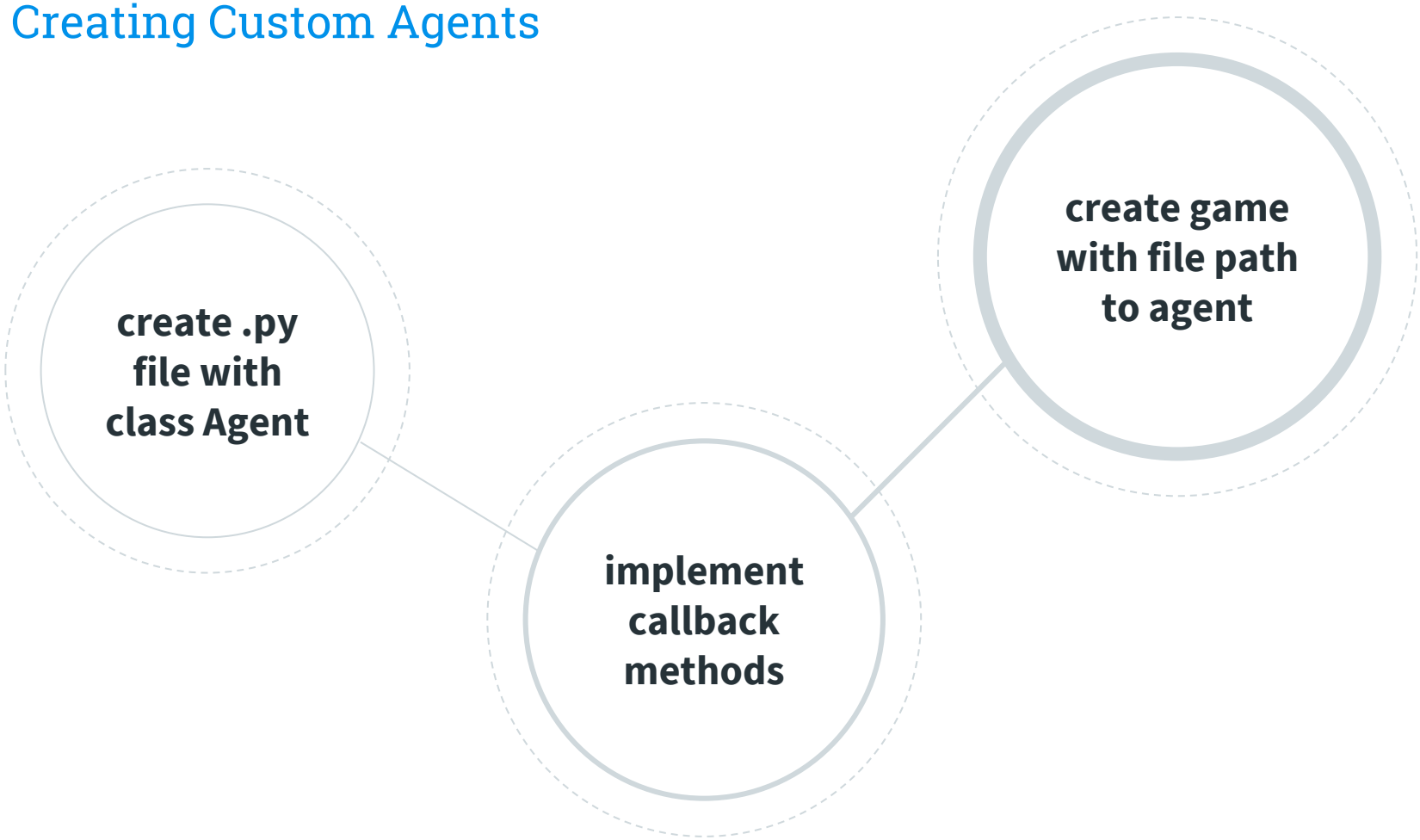
```
template = """wwwwwwwwwwwwwwwww
              w______C______w
              w_R___www___B_w
              w__A__www__A__w
              wwwwwwwwwwwwwwwww"""

# create field from string
field = core.Field.from_string(template)

# create game with specified field
game = core.Game(field = field)
```

5.

Creating Custom Agents

check out `default_agent.py` for a working example

Creating Custom Agents



Creating Custom Agents - init and observe

```
class Agent(object):
    NAME = “some fancy name”

    def __init__(self, id, team, settings = None,
        field_rects = None, field_grid = None,
        nav_mesh = None, **kwargs):

        # init code goes here

    def observe(self, observation):

        # observation code goes here
```

The Observation Object

Property	Description
`loc`	Agents current location `(x, y)`
`angle`	Agents viewing angle in radians
`friends / foes`	Visible or all friends / foes
`cps`	All control points `(x, y, owner)`
`hit`	Previously hit target
`...`	many more, see documentation

Creating Custom Agents - action

```
class Agent(object):
    # ...

    def action(self):

        # some fancy AI magic to compute best action

        return (turn, speed, shoot)
```

Creating Custom Agents - debug and finalize

```
class Agent(object):
    # ...

    def debug(self, surface):

        # use this to draw on the game surface
        # NOT available during actual tournament

    def finalize(self, interrupted = False):

        # called after the game has finished
```

Creating Custom Agents - Using them in a Game

```
game = core.Game(
    red  = “path/to/red_agent.py”,
    blue = “path/to/blue_agent.py”
)

game.run()
```

6. Advanced Techniques

Advanced Techniques - Communication

Agents are instances of the same class ⇒ class variables are shared

```
class Agent(object):
    current_objective = “def”  # this is shared

    def action(self):
        # accessing class variable within a method
        if self.__class__.current_objective == “def”:
            # ...

        self.__class__.current_objective = “att”

        # ...
```

Advanced Techniques - Blob

Agents can have additional (binary) data (NN weights etc.)

```
class Agent(object):
    # receive blob as keyword argument
    def __init__(self, blob = None, **kwargs):

        # read from blob (file object opened with ‘rb’)
        self.blobcontent = blob.read()

        # reset blob file for other agents
        blob.seek(0)
```

Utilities

```
# normally already imported by the game engine
from domination import utilities

# basic useful tools:
utilities.mean([1, 2, 3, 4])                    # 2.5

# game-related tools:
utilities.point_dist((0, 0), (1, 1))            # √2

utilities.find_path(start, end, mesh, grid)     # path
```

That’s it!

If you need help,
consult the docs
or talk to us!

Resources

https://domination-game.readthedocs.io/en/latest/

https://github.com/Nathanaelion/CAT2018

Many thanks to Thomas van der Berg and Tim Doolan for creating and letting us use this awesome environment!

mpoemsl@uos.de

sselbach@uos.de