BULLET
PHYSICS LIBRARY

UAV Simulation using PyBullet

gym-pybullet-drones*

01

Why simulate?

Why simulate? 01

Enable sharing of work

Enable benchmarking

Accelerate developement and deployment of algorithms

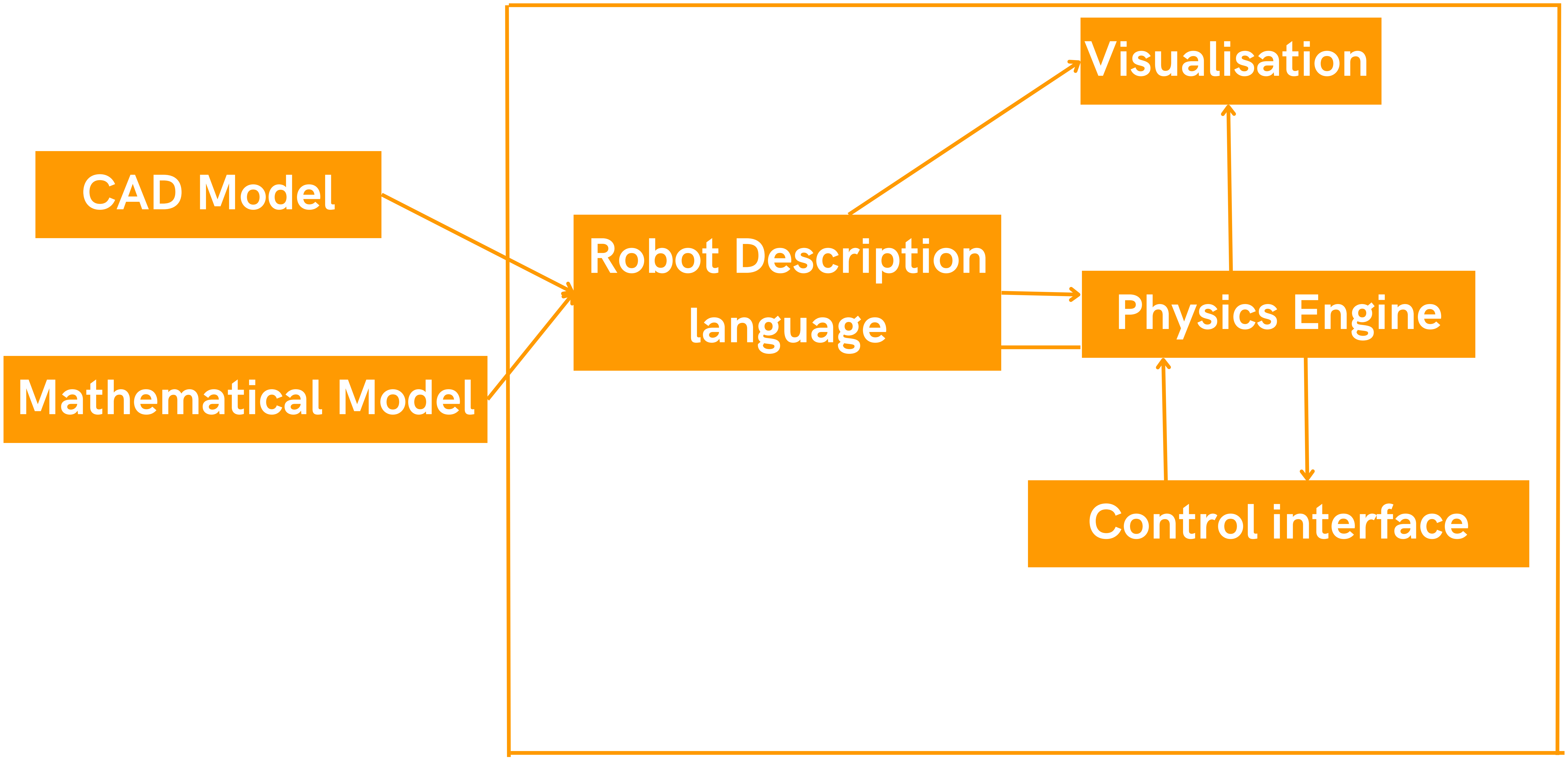
Work accross continents

Safe environment for testing

02

Robot Simulators

Robot Simulators 02



02

Choosing a Simulator

(for Reinforcement Learning in UAVs)

	Physics Engine	Rendering Engine	Language	Synchro./Steppable Physics & Rendering	RGB, Depth, and Segmentation Views	Multiple Vehicles	*Gym* API	Multi-agent *Gym*-like API
This work	PyBullet	OpenGL3†	Python	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Flightmare [7]	*Ad hoc*	Unity	C++	**Yes**	**Yes**	**Yes**	W/o Vision	No
AirSim [8]	PhysX¶	UE4	C++	No	**Yes**	**Yes**	No	No
CrazyS [9]	Gazebo§	OGRE	C++	**Yes**	No Segmentation	No	No	No

† or TinyRenderer ¶ or FastPhysicsEngine § ODE, Bullet, DART, or Simbody

ref: https://arxiv.org/abs/2103.02142

Choosing a Simulator 02

	`gym-pybullet-drones`	AirSim	Flightmare
Physics	PyBullet	FastPhysicsEngine/PhysX	*Ad hoc*/Gazebo
Rendering	PyBullet	Unreal Engine 4	Unity
Language	Python	C++/C#	C++/Python
RGB/Depth/Segm. views	**Yes**	**Yes**	**Yes**
Multi-agent control	**Yes**	**Yes**	**Yes**
ROS interface	ROS2/Python	ROS/C++	ROS/C++
Hardware-In-The-Loop	No	**Yes**	No
Fully steppable physics	**Yes**	No	**Yes**
Aerodynamic effects	Drag, downwash, ground	Drag	Drag
OpenAI `Gym` *interface*	**Yes**	**Yes**	**Yes**
RLlib `MultiAgentEnv` *interface*	**Yes**	No	No

ref: https://github.com/utiasDSL/gym-pybullet-drones

03

PyBullet

Initial Setup

Loading the physics client: Either direct or GUI mode

Direct: no GUI window to display the world

GUI: displays a GUI window

Setting Gravity: PyBullet doesn't setup gravity, and needs to be set manually using their setGravity function

setGravity(x,y,z) sets in the accelerations for the x,y, and z axis

Loading Models

- **URDF:** Universal Robot Description File

    Describes the positioning of robot links and joints

    Describes the dynamics of joints (eg. Do links rotate about that joint?)

    Describes collision boxes and the visual asepect of the robot

- **SDF:** Simulation Description Format

    Describes the simulation world, not restricted to a specific entity

```xml
<!-- Link Definitions -->
<link name="link1">
  <!-- Visual representation of the link -->
  <visual>
    <!-- Geometry of the link -->
    <!-- You can use different shapes (box, cylinder, sphere, mesh, etc.) -->
    <geometry>
      <!-- Example: Box -->
      <box size="0.1 0.1 0.2" />
    </geometry>
    <!-- Material of the link (color, texture, etc.) -->
    <material>
      <color rgba="0.5 0.5 0.5 1" />
    </material>
  </visual>
```

Loading Models

- **Loading URDF Files**

loadURDF('file_name.urdf', initialPos, initialOrientation)

Returns a unique model id and spawns the model

Model id is useful to get and manipulate robot information like:

getBaseAndPositionOrientation(id)

resetBaseAndPositionOrientation(id, newPos, newOrient)

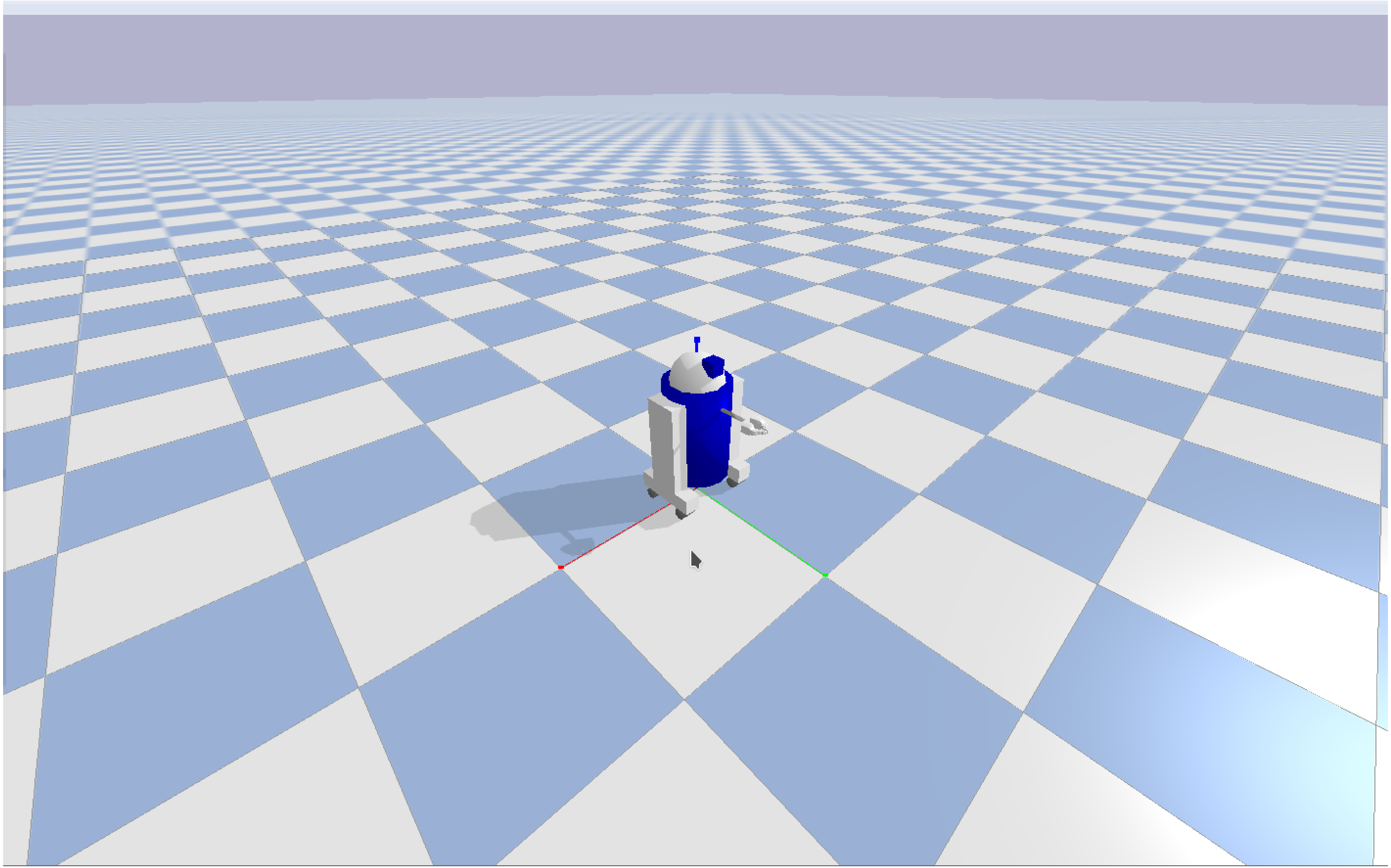
Position and Orientation

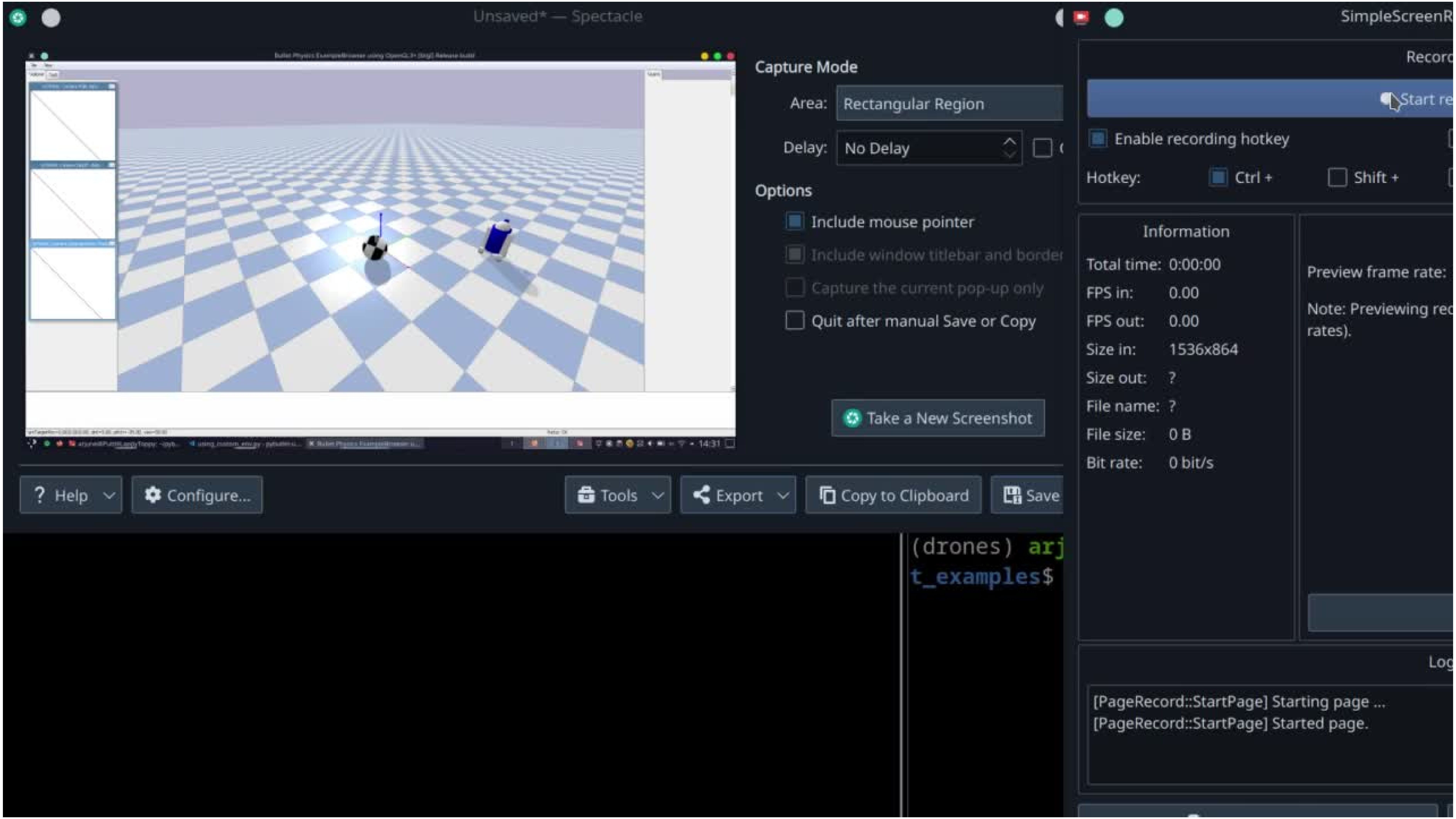
- **Position:** Cartesian [x,y,z] coordinate system
- **Orientation:** Quaternion [x,y,z,w] system

If you are using the Eulerian system, must convert to Quaternion

getQuaternionFromEuler(eulerAngles)

getEulerFromQuaternion(quatAngles)

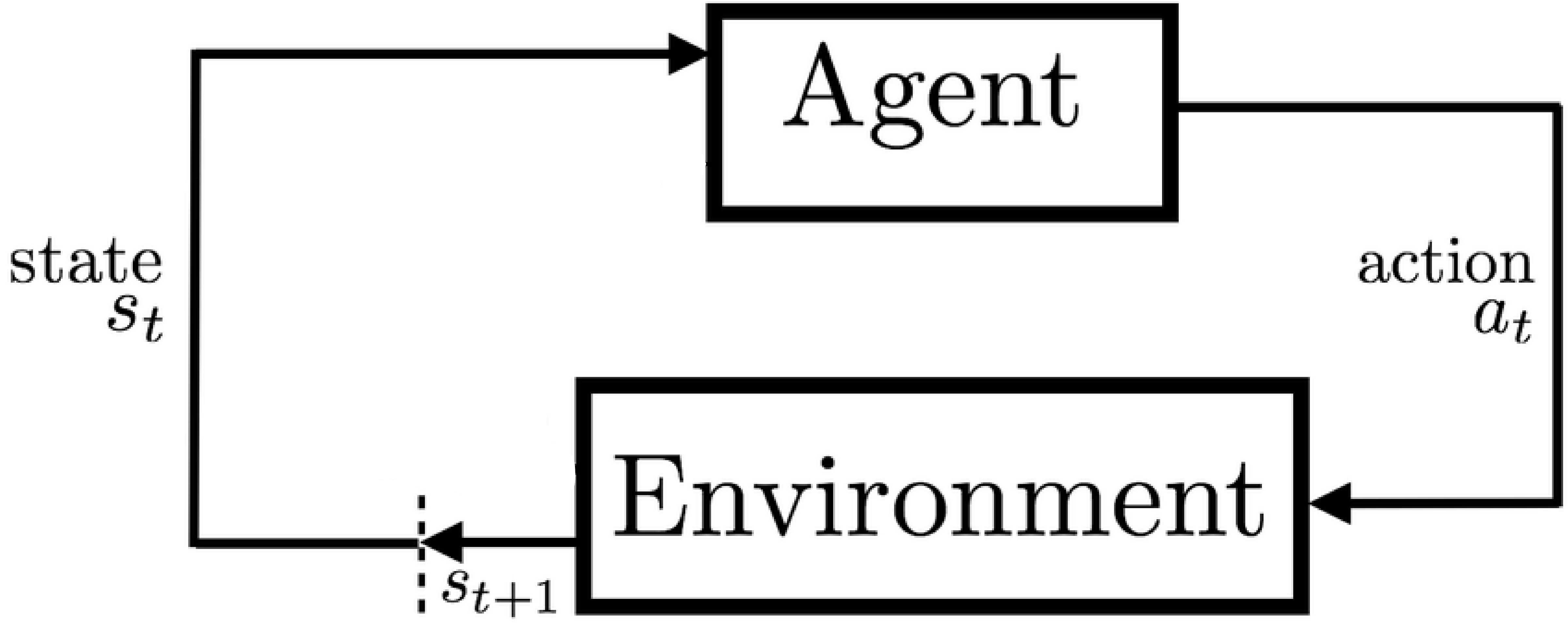



Unsaved* — Spectacle
Capture Mode
Area:
Rectangular Region
Delay:
No Delay
Options
Include mouse pointer
Include window titlebar and border
Capture the current pop-up only
Quit after manual Save or Copy
Take a New Screenshot
Help
Configure...
Tools
Export
Copy to Clipboard
Save
SimpleScreenR
Recor
Start re
Enable recording hotkey
Hotkey:
Ctrl +
Shift +
Information
Total time: 0:00:00
FPS in: 0.00
FPS out: 0.00
Size in: 1536x864
Size out: ?
File name: ?
File size: 0 B
Bit rate: 0 bit/s
Preview frame rate:
Note: Previewing rec
rates).
Log
[PageRecord::StartPage] Starting page ...
[PageRecord::StartPage] Started page.
(drones) arj
t_examples$

Demonstration

04

Agent-Environment


Agent
state
s_t
action
a_t
Environment
s_{t+1}

Examples of Agent-Environment 04

EXAMPLES	AGENT	ENVIRONMENT
Chess Playing AI	The AI algorithm or program	The chessboard, chess pieces, the rules of the game, the other player
Robotic Arm	Robotic Arm	The physical workspace in which the robotic arm operates, including objects to be manipulated and any obstacles present
Training/making program to control UAV		

Examples of Agent-Environment 04

EXAMPLES	AGENT	ENVIRONMENT
Chess Playing AI	The AI algorithm or program	The chessboard, chess pieces, the rules of the game, the other player
Robotic Arm	Robotic Arm	The physical workspace in which the robotic arm operates, including objects to be manipulated and any obstacles present
Training/making program to control UAV	The algorithm/program	the UAV, along with other elements from the workspace(obstacles)

Other terminologies Agent-Environment 04

Episode

Timestep

Action

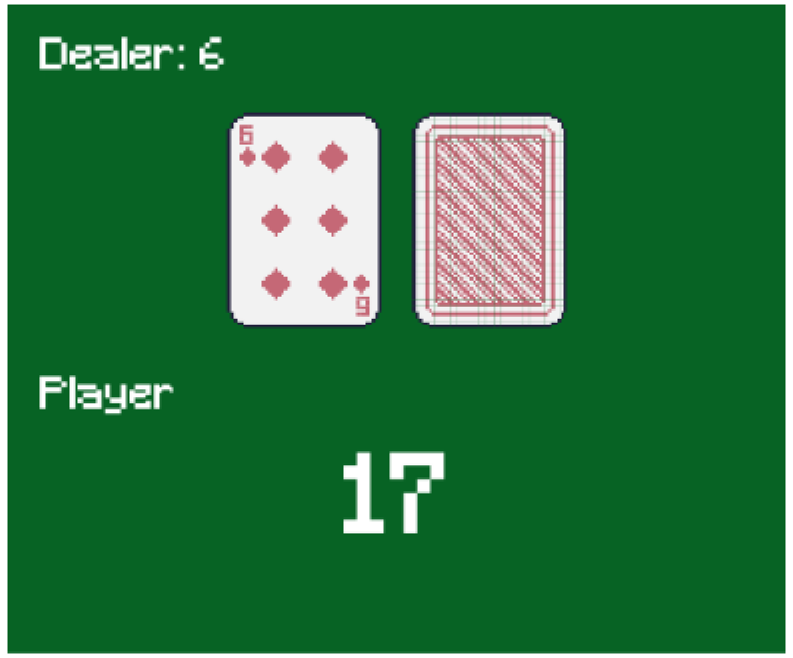
Observation

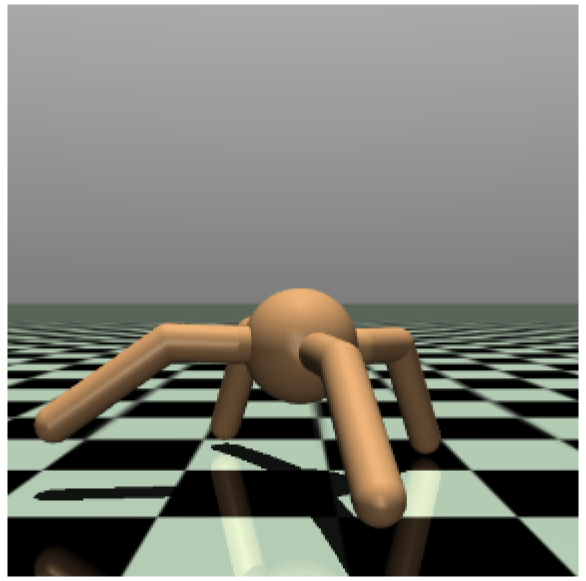
Terminal State

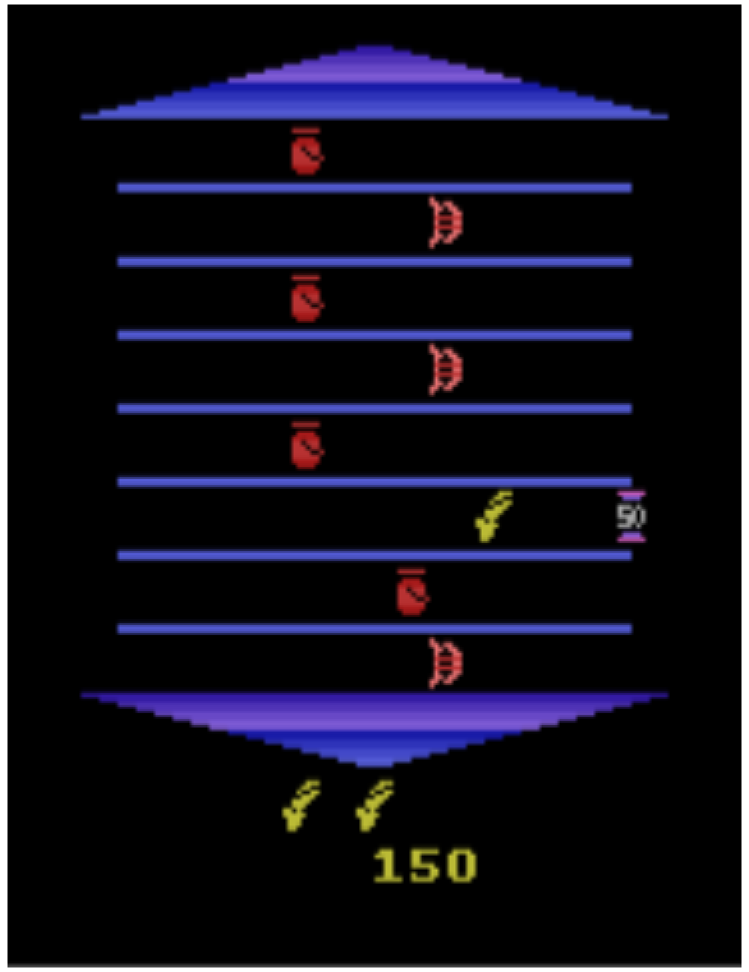
05

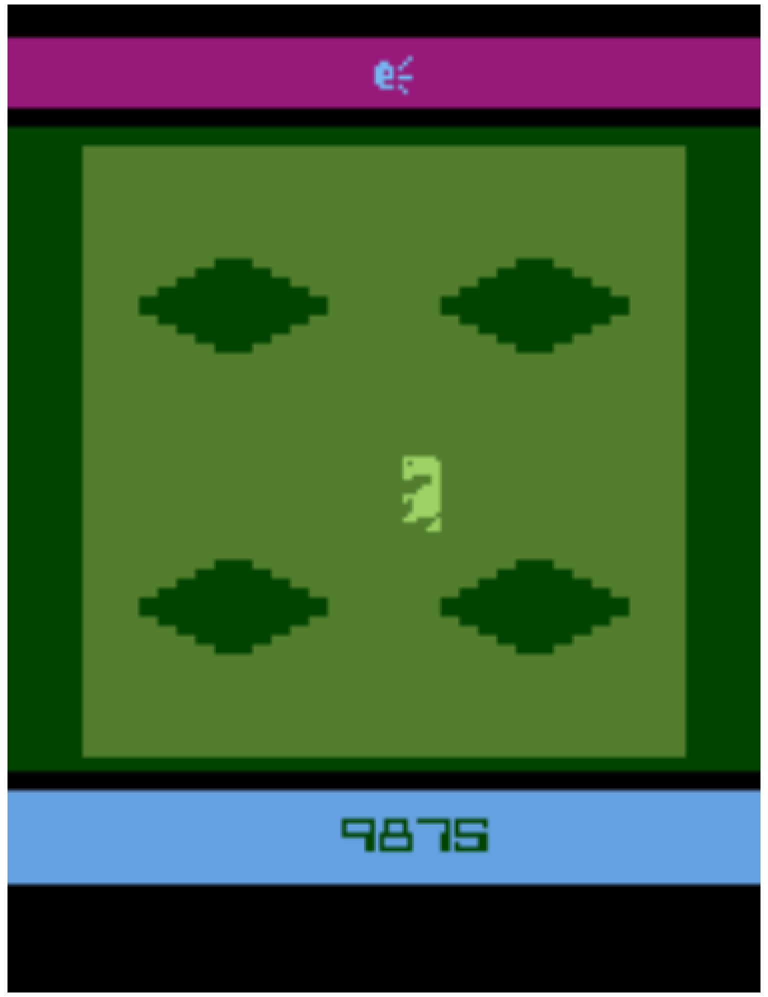
Open AI Gym

Why Open AI Gym 05









API for Open AI Gym 05

```python
class env(gym.env):
    def reset():
        pass

    def step(action):
        ...
        return observation, reward, done, info

    def render():
        pass
```

representational*

```python
class agent():
    def action(observation):
        ...
        return action
```

representational*

API for Open AI Gym 05

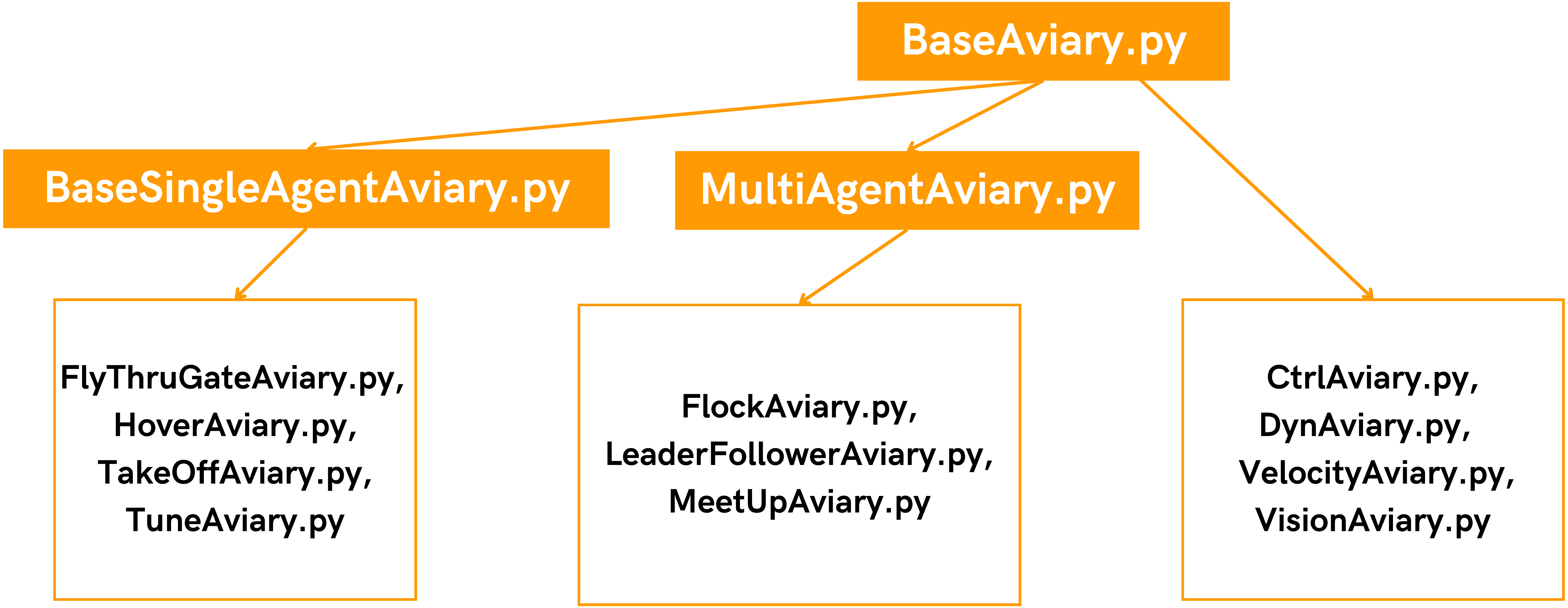
```
env = gym.make('name_of_environment')

for episodes in range(n):
    env.reset()

    while not done:
        observation, reward, done, info = env.step(action)
        action = agent.action(observation)
        env.render()
```

representational*

06

gym-pybullet-drones

Environments in gym-pybullet-drones 06



Custom environment in gym-pybullet-drones 06

Deriving from

BaseAviary.py

```
class env(BaseAviary):
    def __init__(num_drones):
        super().__init__(num_drones)

    def _actionSpace():
        # define your action space using gym.spaces
        return actionSpace

    def _observationSpace():
        # define your observation space using gym.spaces
        return observationSpace

    def _computeDone():
        # set done to true when simultion ends (e.g. crash)
        return done

    def _preprocessAction(action):
        # convert input dictionary into rpm array
        return action

    def _computeReward():
        # If applicable, define reward function
        return None

    def _computeDone():
        # If applicable, set done to true when episode ends
        return None

    def _computeInfo():
        # If applicable return information about environment
        return None
```

representational*

Custom environment in gym-pybullet-drones 06

Deriving from

BaseAgentAviary.py

BaseMultiAgentAviary.py

```
class env(BaseSingleAgentAviary):

    def __init__(observationType, actionType):
        super().__init__(observationType, actionType)

    def _computeReward():
        # define reward function
        return reward
    def _computeDone():
        # set done to true when episode ends
        return done

    def _computeInfo():
        # information about environment
        return info

    def _clipState():
        # normalise observation to observation space
        return state
```

representational*

Custom environment in gym-pybullet-drones 06

ActionType while deriving from

BaseAgentAviary.py

BaseMultiAgentAviary.py

```
ActionType.RPM                # RPMS
ActionType.DYN                # Desired thrust and torques
ActionType.PID                # PID control
ActionType.VEL                # Velocity input (using PID control)
ActionType.TUN                # Tune the coefficients of a PID controller
ActionType.ONE_D_RPM          # 1D (identical input to all motors) with RPMs
ActionType.ONE_D_DYN          # 1D (identical input to all motors) with desired thrust and torques
ActionType.ONE_D_PID          # 1D (identical input to all motors) with PID control
```

Custom environment in gym-pybullet-drones 06

ObservationType while deriving from

BaseAgentAviary.py **BaseMultiAgentAviary.py**

```
ObservationType.KIN     # Kinematic information (pose, linear and angular velocities)
ObservationType.RGB     # RGB camera capture
```

Demonstration

PyBullet Getting Started Documentation

Installation

gym-pybullet-drones basics

basic environment examples