

Skal prøve så godt som jeg kan å dokumentere simulasjon miljøet og hvor fysikken ligger i denne doccen

## Viktige filer og roller

fil	hva filen gjør
table.xml	Definerer bordets, puckens og paddle friksjon, demping og materialer.
position_control_wrapper.py	Kontrollerer bevegelsen til padler og robot (brukes i challenge prosjekt, men kan være relevant for oss) og angir "force".
kinematics.py	Håndterer beregninger av padle og leddbevegelser. (muligens viktig når vi lager manualle Aler)
challenge_core.py	Styrer simulation step og fysikkoppdateringer.
air_hockey_challenge_wrapper.py	Setter opp simuleringsmiljøet.
main.py	Utfører MuJoCo fysikkoppdateringer for hver frame.

Hvordan mujoco og fysikken fungerer

### 1. Initialisering

main.py laster table.xml:

```
model = mujoco.MjModel.from_xml_path(xml_path)
data = mujoco.MjData(model)
```

## Oppdatering av puckbevegelse

Hvert simuleringssteg (main.py):

```
mujoco.mj_step(model, data)
```

Oppdaterer puckens posisjon, hastighet og krefter.

Bruker table.xml-innstillinger (table friction="0.08" og puck damping="0.001").

## Padle bevegelse

position\_control\_wrapper.py anvender kontrollinndata:

(linje 98 til 129 i filen)

```
control_action = controller._controller(desired_pos, desired_vel, desired_acc, cur_pos, cur_vel)
```

Bruker PID-kontroller for å beregne kraft:

```
torque = p_gain * error + d_gain * (clipped_vel - current_vel) + i_error
```

Anvender kraft til padler via `mjoco.mj_mulM()`.

Interaksjoner og kollisjoner

`table.xml` definerer puck-bord-interaksjoner:

```
<geom name="table_surface" type="box" friction="0.08 0.08 0.0"/>
```

Kollisjonsdeteksjon skjer automatisk i MuJoCo via `mj_step()` i `main.py`, men vi kan bruk `air_hockey_challenge_wrapper.py` senere

## Viktige metoder i hver fil

fil	funksjon	hva filen gjør
table.xml	<code>&lt;geom friction="..." /&gt;</code>	Definerer friksjon og demping.
table.xml	<code>&lt;joint damping="..." /&gt;</code>	Setter motstand i puckens bevegelser.
main.py	<code>mj_step(model, data)</code>	Utfører simuleringssteg.
position_control_wrapper.py	<code>_controller()</code>	Beregner padlekrefter.
position_control_wrapper.py	<code>_compute_action()</code>	Sender padlebevegelse til MuJoCo.
kinematics.py	<code>forward_kinematics()</code>	Beregner padle og leddposisjon.
challenge_core.py	<code>_step()</code>	Oppdaterer simuleringsstatus.
air_hockey_challenge_wrapper.py	<code>step(action)</code>	Behandler handlinger fra AI eller manuell styring.

## Hvordan endre fysikken for diverse ting

### Endre puckens friksjon

Rediger `table.xml`:

```
<geom name="surface" type="box" friction="0.08 0.08 0.0"/>
```

Høyere friksjon betyr at pucken stopper raskere.

Lavere friksjon betyr at pucken glir lengre.

### Endre padlekraft

Endre `position_control_wrapper.py` (`_controller()`):

```
torque = p_gain * error + d_gain * (clipped_vel - current_vel) + i_error
```

Økning av `p_gain` gir sterkere slag.

Reduksjon av `d_gain` gir jevnere bevegelser.

`p_gains` er satt til: `p_gain, d_gain, i_gain = 10.0, 1.0, 0.1`  
i `main` for øyeblikket

### 3. Justere AI-responstid

Endre `challenge_core.py` (`_step()`-funksjon):

```
start_time = time.time()
action = self.agent.draw_action(self._state)
end_time = time.time()
duration = (end_time - start_time)
```

Legg til en forsinkelse for å simulere reaksjonstid.

Endre hvordan AI velger handlinger for padler.

## Viktig:

kode i `challenge_core` må omskrives for vår prosjekt da den inneholder tournament kode for no vis vi trenger å styre steps fra der.

`air_hockey_challenge_wrapper.py` er for øyeblikket ikke i bruk i `main.py` eller noen andre aktivt refererte filer. Den er definert i prosjektet, men verken importert eller kalt noe sted. Dette betyr at verken AI eller en manuell agent er i bruk med simulasjonsmiljøet.

### Hvordan kan vi bruke filen

For å integrere `air_hockey_challenge_wrapper.py`, kan den erstatte direkte fysikkoppdateringer via MuJoCo (`mj_step()`) med `env.step(action)`. Dette vil da Standardisere AI-interaksjoner, noe som forenkler implementering av RL algoritmer. Gi en mer strukturert tilnærming til hvordan handlinger blir utført i miljøet. Tilrettelegge for forskjellige spillmoduser (som 3dof-hit, 3dof-defend).

Eksempel fra chatgpt:

```
from environment.env_settings.framework.air_hockey_challenge_wrapper import
AirHockeyChallengeWrapper

# Initialize the environment with a specific mode (e.g., '3dof-hit')
env = AirHockeyChallengeWrapper(env="3dof-hit")

# Manual AI Loop
while True:
    state = env.reset() # Reset environment at the beginning
    action = compute_manual_action(state) # Replace with AI or manual strategy

    # Step through the environment
    next_state, reward, done, info = env.step(action)

    if done:
        break # Restart if episode ends
```

## to long to read

- Fysikk er definert i `table.xml` (friksjon, demping).
- `position_control_wrapper.py` håndterer padlebevegelser og krefter.
- Simuleringsoppdateringer skjer i `main.py` (via `mj_step()`).
- AI kan modifisere kraft, friksjon og responstid.
- vi kan bruke `challenge_core.py` or `air_hockey_challenge_wrapper.py` for oppdatering av fysikk

for AI-integrasjon så langt jeg forstår

- Bruk `position_control_wrapper.py` for å programmere padlebevegelser.
- Endre `table.xml` for å justere puckens interaksjon med bordet dersom det trengs, men skal ikke kunne trenges.
- Eksperimenter med `challenge_core.py` for å justere AI-beslutninger.
- Implementer læringsbasert baneforutsigelse for smartere responser