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 AIer)
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 `mujoco.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()`.

Viktige metoder i hver fil

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

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.
- Manuell AI kan forbedres med adaptiv beslutningstaking og prediktivt forsvar.

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