-> Single class manages environment

-> task class sets up scene, process actions, compute rewards, observations,

Prims: basic building blocks of a scene
- light -> transform
- -> mesh
  - -> can have other prims/objects under it.

Attributes: key - value pair
e.g color: red

relationships: pointers
- -> relationship
- -> mesh can be related to light for shading.

Sim. spawners is a wrapper for USD API

When simulation starts only alter properties of prim.

The first path in prims is the objects heirchy in USD.

world
|— Cone1
|—— xform

-> this is how prims are organize

— gibson
down

in the
world.

```
# spawn a green cone with colliders and rigid body
cfg_cone_rigid = sim_utils.ConeCfg(
    radius=0.15,
    height=0.5,
    rigid_props=sim_utils.RigidBodyPropertiesCfg(),
    mass_props=sim_utils.MassPropertiesCfg(mass=1.0),
    collision_props=sim_utils.CollisionPropertiesCfg(),
    visual_material=sim_utils.PreviewSurfaceCfg(diffuse_color=(0.0, 1.0, 0.0)),
)
cfg_cone_rigid.func(
    "/World/ConeRigid", cfg_cone_rigid, translation=(-0.2, 0.0, 2.0), orientation=(0.5, 0.0, 0.5, 0.0)
)
```

1

2

path in USD heirchy

config

-> then

1 create config for shape / mesh

reference the API in isaac lab in this case it was the ConeCfg method

2

translations ....

RL

Nov 7th, 2024

policy: takes State of the world and produces an action.

-> its a function
-> because its a function it can be a neural network

-> The policy can take in images instead of velocities, position etc....
-> each neuron will adapt to one pattern of the same (ball / paddle position)
-> This is called policy gradient.

Bellman equation: relationship between The value of a state and the value of the next state.

state : gas runs out, tire wears out

gas is proportional to speed (1)

(2) tire is determined by type of tire
-> different tires allow for different top speed

(1) options: make a pit stop
-> 10 sec penalty
run out of gas:
-> DNF

(2) options: choose between tyre 1, 2, 3

1) higher speed, lower life

2) med speed, med life

3) lower speed, highest life

-> pit stop
-> 10 sec penalty
don't stop
-> DNF

high level model:

at each time step: input [gas life, Tyre life, state] -> linked to reward

output at each time step = [Action space]

Action space = [steering, gas, break, pit stop]

it will be pelized if it DNFs, or sets a slower lap time against adverary

reward = $100 \; \cancel{-} \; \frac{10000}{100} + \frac{t}{100}$ if it take 10.000 steps its 0

$$\text{reward function} = \underbrace{100 - \frac{t}{100}}_{\text{driving score}} + \frac{\theta}{100} + \Omega$$

100: Total score for being a good driver	$\frac{t}{100}$: reward lost for how long it falie to drive	$\frac{\theta}{100}$: for every second its lap time is faster than opponent.

$\Omega = -10000000$ if DNF

for calculating lap time:

(current time in lap)

365 sec to complete

30%

300 sec

100%

thread ①

model 1

lap time report

thread 2

running model 2

lap time report

Reward function = Total reward $- \frac{t}{100} + \frac{\theta}{100} - \frac{P}{100} + w$

total reward = how perfect can you drive

t = penalizing for how perfect you can drive (time in lap)

θ = reward for every step which your time is less than your competitors

P = penalty for fuel/tire being less than 40% and increased greatly when bellow 0.

W = reward/penalty for winning or losing.

1. trained the high Level ✓
   -> generate graphs

2. get single agent racing ✓

3.

End goals: assuming there is a pit stop method, agent to be competetive against human.

↳ We need a robust pit stop method.

pit stop method:

currently: Total fuel,

Total tire level,

fixed rate of consumption

→ you can race a whole lap without pit stopping

↓

increase rate

↳ doesn't matter because both agents will just have to pit stop at least once.

↳ randomize the rate of consumption [clip it]

↳ Just adding a randomized rate of consumption will not add any value or complexity to the system because the model is learning to pit stop based on fuel left, other in words rate of consumption is related to total and we are only observing the related variable.

Essentially the problem we are trying to solve is least number of pit stops in order to complete a lap.

in real life or in sim, multi agent or not the objective is to make it to a pit stop with least amount of fuel possible.

and the skill comes in

Where how well can you drive to make your car last as long as possible and minimize pit stops while minimizing time taken

(pit stop) minimizing use of supplies

which is related to

howell you drive / racing line, speeds / breaking, acceleration

While you minimize time around a track which means you want to maximize speed

minimize breaking ←

and minimize racing line

with in rules,

very recursive,

reward function of higher level model should minimize pit stops.

- The reason why we removed the randomization in consoption rate, is because it will add variability to the threshold at which the model makes a decision to take a pitstop.
  - Eg: Let's say it's at 0.3 rate and we wan't the model to take a pitstop at fuel level 0.2, in time step 1 it's at 0.7, then next it's at 0.4 then 0.1, → This forces the model to take a pit stop at 0.4 or 0.1, both at which are not a ideal condition. Because the way the reward function is setup it will get punished either option. We saw af

We saw after training, that the reward wasn't converging and was continiusly going into the negatives, and the model was continously getting punished and wasn't learning. Regardless of randomizingf the consumption rate, the variable that is being observed by the model is the total fuel and tire trend level left, therefore changing the consumption rate, only changes the number of timestamps required for the fuel level and tire trend level to reach the threshold for the model to make a decision, therefore it added only unessessary complexity and variance on how close the level of the fuel and tire trend will reach the threshold. Realistically it only makes sense to have the consumption rate to be related to the driving behaviour to the lower models, but it was not possible to do this because we did not have the ability to train the lower model because of the computational limitation, thus we used a pretrteianed lower level model.`