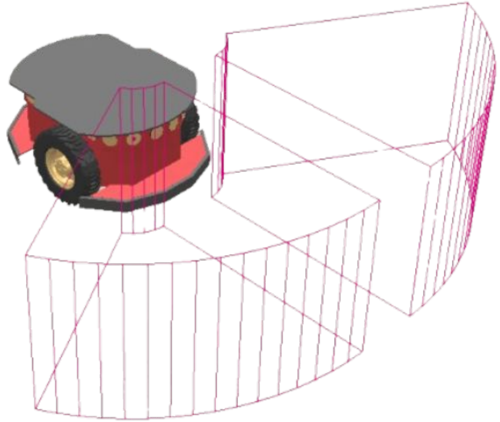
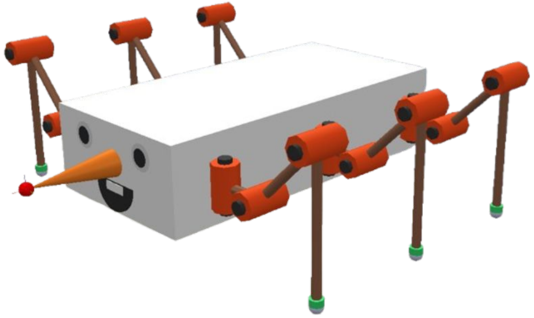
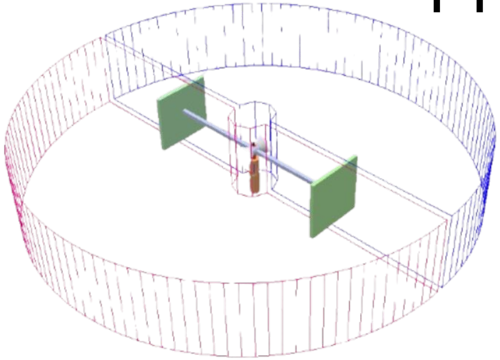
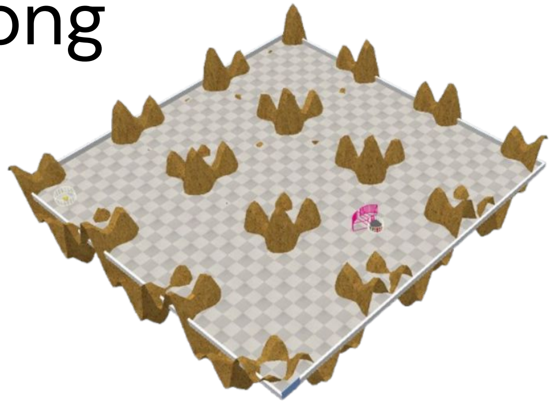




Embodied AI Class Experiments

Prof. Poramate Manoonpong





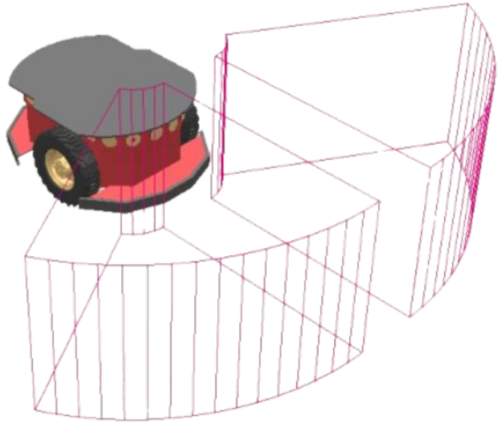
Introduction

These assignments constitute an integral component of the **Embodied AI** course taught by Prof. Poramate Manoonpong. They are specifically designed to ensure that students develop a thorough understanding of the theoretical principles and methodological foundations of Embodied Artificial Intelligence. Furthermore, the assignments aim to enable students to effectively apply the acquired knowledge and skills to the implementation of computational tasks as well as to practical real-world applications, thereby reinforcing the connection between theory and practice.

To support this objective, the assignments will be conducted using the **CoppeliaSim simulation** environment, with control algorithms implemented through **Lua scripting**, allowing students to design, test, and evaluate embodied controllers in a realistic and interactive simulated setting.

Table of Content

Contents



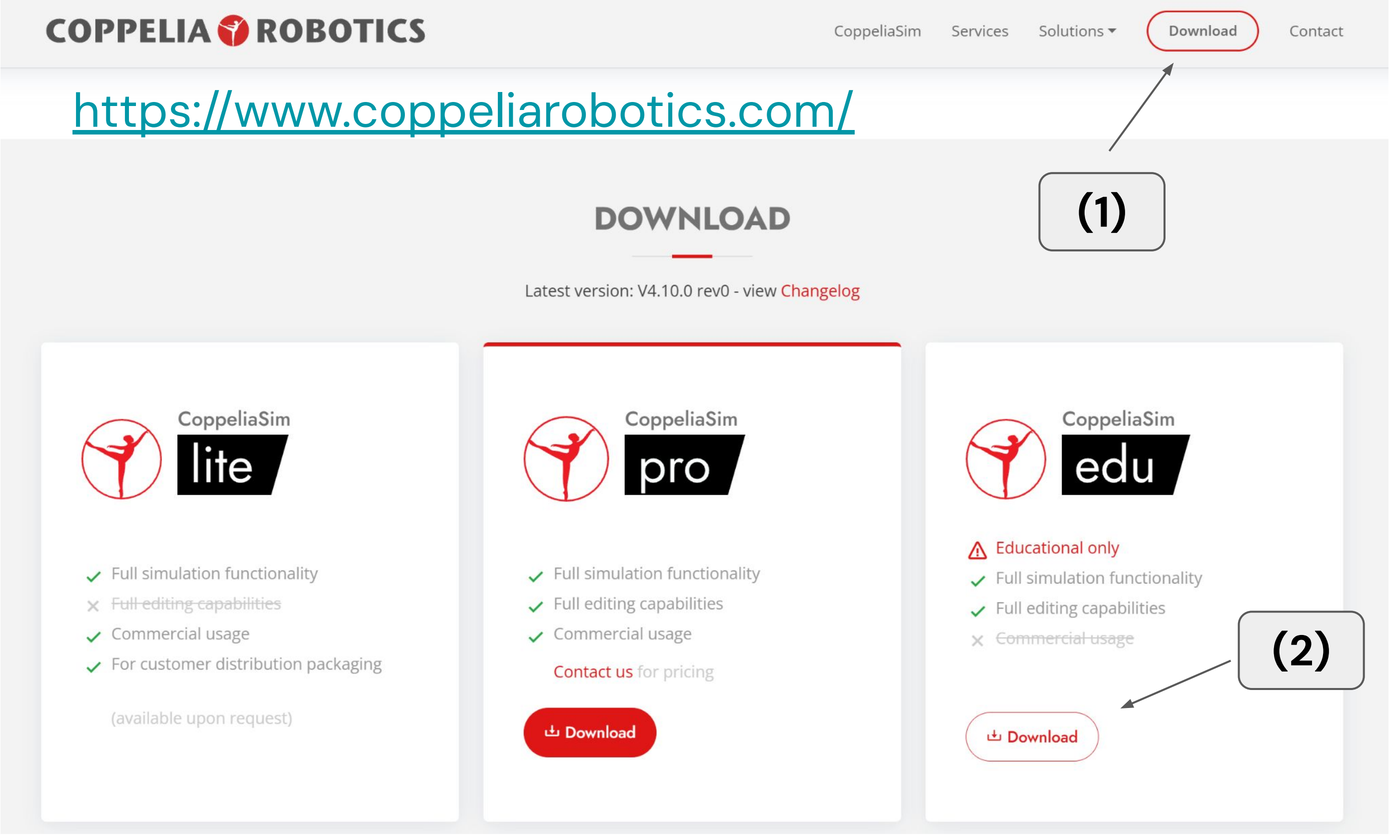
Download CoppeliaSim simulation

Download click here

- On the web page Goto (1) and (2)

- Then following the installation

https://www.coppeliarobotics.com/



CoppeliaSim shortcuts

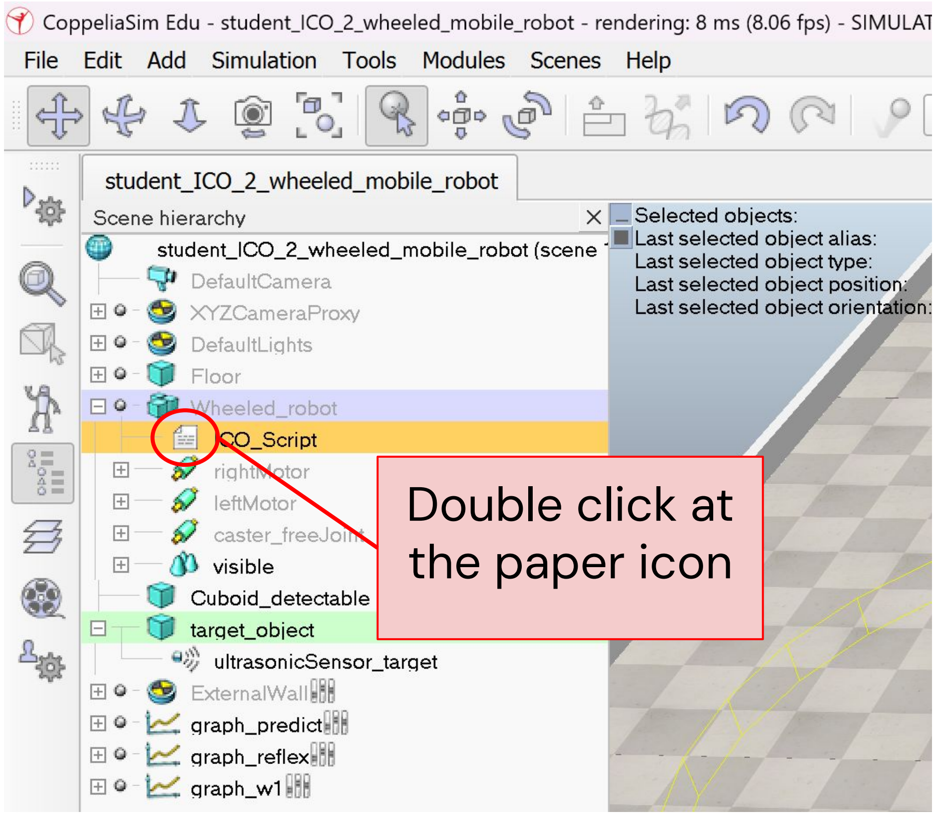
Shortcuts

When the focus is on the scene hierarchy or a page, following shortcut keys are supported:

- **CTRL+A**: select all
- **<esc>**: clear the selection
- **CTRL+C**: copy the selection
- **CTRL+V**: paste the copy buffer
- **CTRL+X**: cut the selection
- **<delete>**: delete the selection
- **<backspace>**: delete the selection
- **CTRL+O**: open a scene
- **CTRL+N**: open a new scene
- **CTRL+S**: save the scene
- **CTRL+W**: close a scene
- **CTRL+Q**: quit the application
- **CTRL+<space>**: start/stop the simulation
- **CTRL+E**: toggle between 1) normal, 2) object translation and 3) object rotation mouse mode
- **CTRL+D**: open the object property dialog
- **CTRL+G**: open the calculation module dialog
- **CTRL+B**: adjust the view to fit selected objects, or the whole scene if no object is selected. The focus needs to be on a view.
- **CTRL+ALT+C**: set focus on the commander
- **CTRL+L**: clear status bar (when focus is on the commander)

Table of content

To open lua script



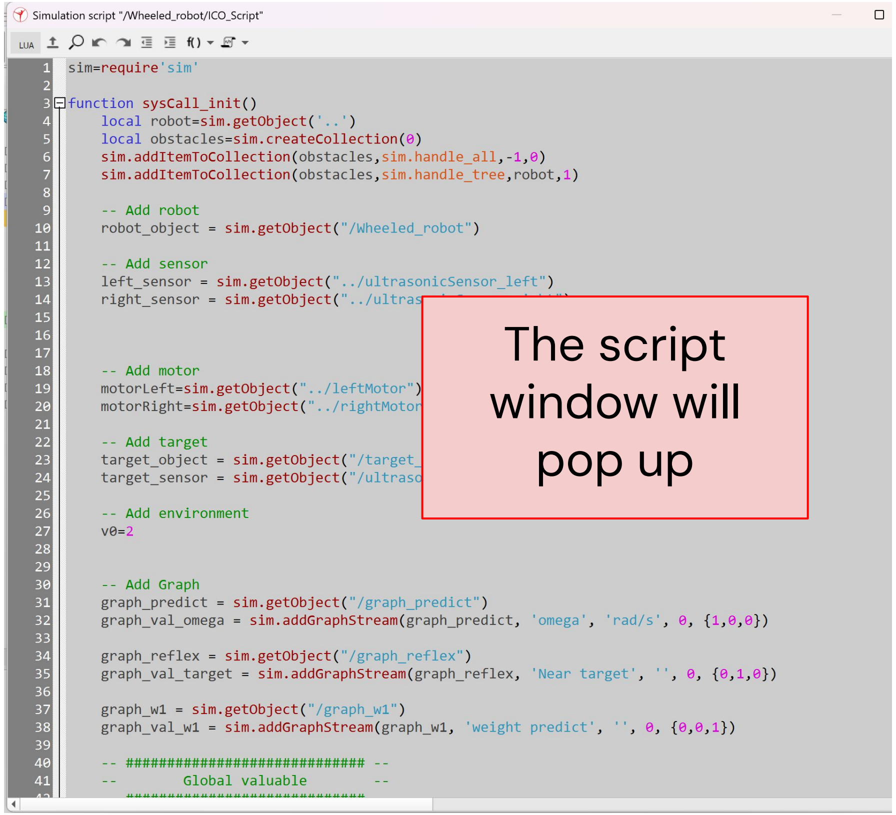
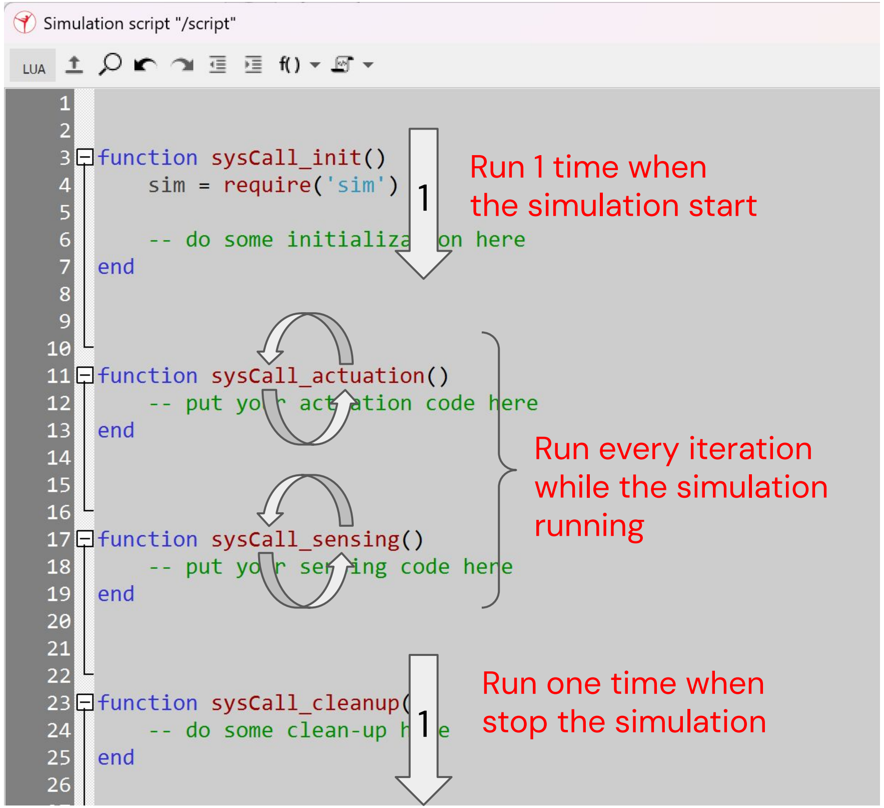


Table of content

Lua script structure



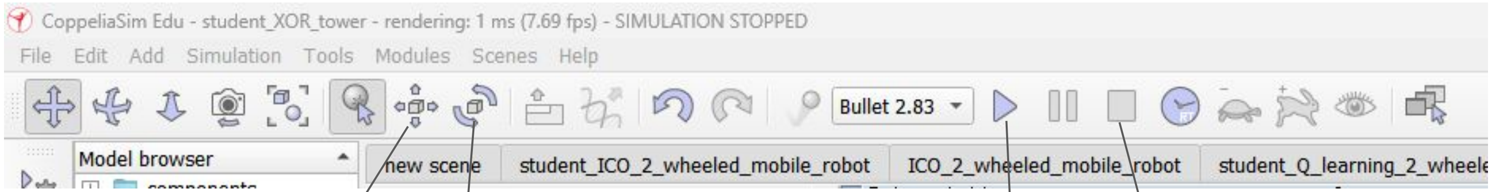
SysCall_init():
define global parameter, set initial value

SysCall_actuation():
command motor, set object position/orientation

SysCall_sensing():
Get sensor value, read object position/orientation

SysCall_cleanup():
clean the scene, save record value to file

Run and stop simulation



Move object

Rotate object

Run sim

End sim

press and hold to **move** scene

press and hold to **rotate** scene

Table of content

Download Task scenes

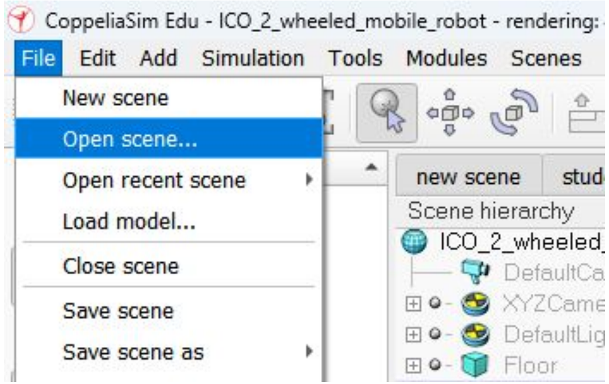
2-wheeled robot

- ICO learning
- Q-learning and SARSA

Simple tower defence

- XNOR neural network

Once you've finished loading the simulation, you can open the task scene by
1) double-clicking the scene file (file.ttt) directly
—or—
2) opening the simulation > Files > open scene... > *[choose your scene]*



Download

https://github.com/VISTEC-IST-ROBOTICS-PUBLIC/Embodied-AI-Course/

Table of content

Supporting function in Lua

Normal functions in Lua → they are in math library

- math.abs()
- math.cos()
- math.sin()
- ....

Function you might need but there is not in Lua → we have created and you can use

- max_value_index = argmax(array)
- val = randomFloatNumber(minVal, maxVal)
- val = sigmoid(val)
- val = dsigmoid(val) → differential of sigmoid function
- val = tanh(val)
- val = dtanh(val) → differential of tanh function

Table of content

Record data to CSV file

```
-- Add record File
logFile = false
fileHeader = 'Time, data1, data2, data3, data4, data5' --<<<<<

date_time = os.date("%Y-%m-%d_%H-%M-%S")
file = ''
if logFile then
    print('Data recording ...')
    scenePath = sim.getStringParam(sim.stringparam_scene_path)

    file = io.open(scenePath .. '/' .. string.format(date_time) .. '_log.csv',
    -- write header
    file:write(fileHeader)
end

```

```
function sysCall_sensing()
    -- Record data to CSV file
    simTime = sim.getSimulationTime()
    if logFile then
        dataArray = {simTime,1,2,3,4,5}
        recordData(dataArray)
    end

```

This function will record the data

1) Enable logging file by changing
logFile = true

2) You can change or add header name of each data (please follow the format)
fileHeader = " header1, header1, … headerN"

3) Change or add the data you need to record following the defined header in the **dataArray**
For example
- dataArray = {myVal1, myVal2, … myValN}

Note: The number of members in **dataArray** must be equal to the menbers in the defined headers.

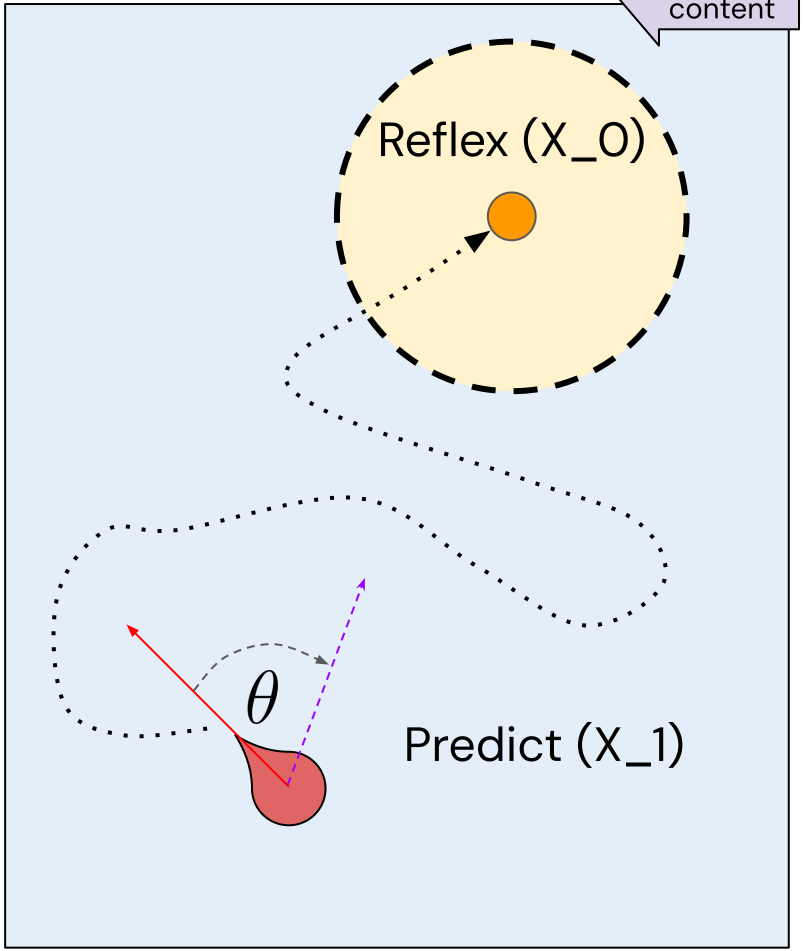
ICO learning

Table of content

ICO learning task

The robot know the theta (angle direction heading to the target). If the robot turns towards the goal, theta is zero (it means the robot doesn't have to turn left (-theta) or turn right (+thera)).

Predict signal (X_1) = theta all the time
Reflex signal (X_0) = theta (if the robot is in yellow area, if not X_0 = 0)
Reflex weight (W_0) = 1
Predict weight (W_1) = find it from ICO !!!!



ICO learning task

The robot is initiated move forward and random turn. Onec, the robot move to yellow area, the robot will move directly to the target because the reflex signal and its weight (=1)

Your task is code the ICO equation to learn the predict weight (W_1) to drive the robot to the target

Ps. If the robot hits an obstacle (target or wall), the robot and target will respawn at a new random position and direction.

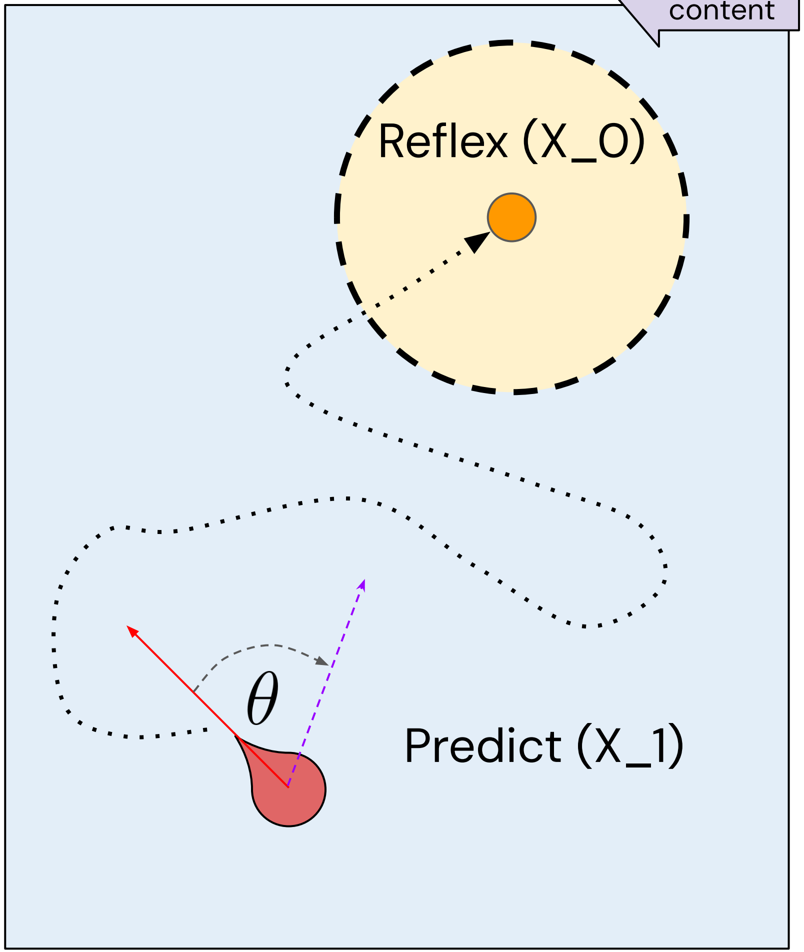
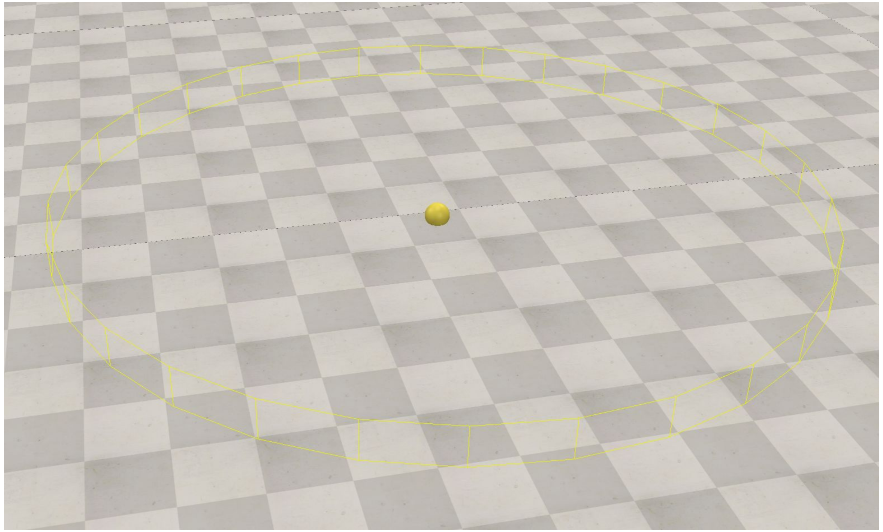


Table of content

Table of content

ICO learning Goal



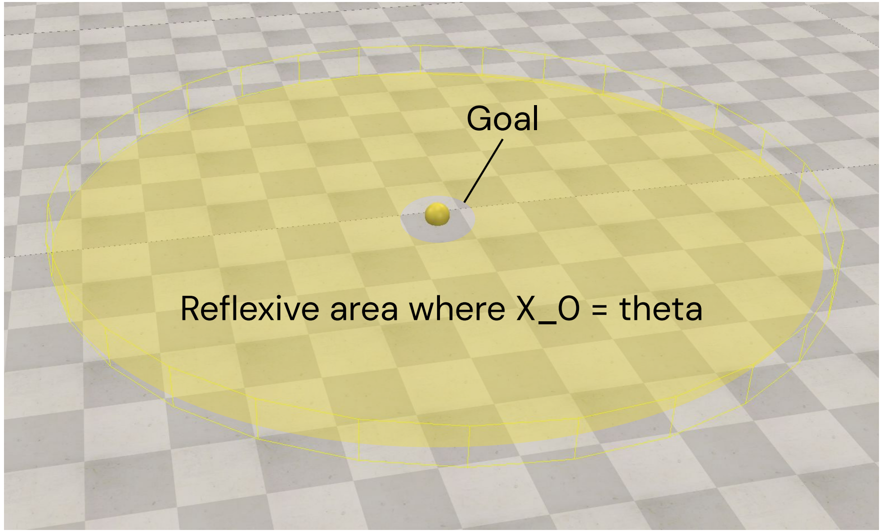


Table of content

ICO learning task

```lua
-- ############################## --
--               ICO              --
-- ############################## --
--
--
-- Predict(X1)---------\-------[X>--\
--                      \       /     \
--                      (X)---/        \
--                      /               O ---->
--           [dx0/dt]--/               /
--              |                      /
-- Reflex(x0)---|-------------------/

output_angle = 0
neural_output = 0
noise = 0

-- ===================================================== --
--             Student Initial parameter Here             --
-- ===================================================== --
-- Must use parameter

x0 = 0          -- reflec signal
x1 = 0          -- predict signal
alpha = 0.01    -- learning rate

-- You can edit and add parameters
example_val_1 = 0
example_val_2 = 2.1
example_val_3 = 44
example_val_4 = 100


-- ===================================================== --
-- ===================================================== --
```

Initial parameters in this zone

```lua

function ICO_controller()

    -- make noise every 4 seconds
    if math.floor(simTime)%4 then
        noise = randomFloatNumber(-2,2)
    end

    -- add theta to signals
    if sensor_target_result == 1 then
        reflex_sig = robot_target_omega
    else
        reflex_sig = 0
    end

    x0 = reflex_sig            -- Reflex signal
    x1 = robot_target_omega    -- predictive signal

    -- ========================================= --
    --             Student Edit code Here         --
    -- ========================================= --




    -- ========================================= --
    -- ========================================= --

    -- add noise to output
    output_angle = neural_output + noise*math.max(0, (1-w1))

    -- ============ drive to the robot wheel ============= --
    -- forward velocity
    v0 = 2
    wheel_left = -output_angle + v0
    wheel_right = output_angle + v0
    -- ================================================= --

end -- ICO controller

```

Edit code only in **ICO_controller function**

Get predict and reflex signals

Add ICO equation here

add noise to neural output

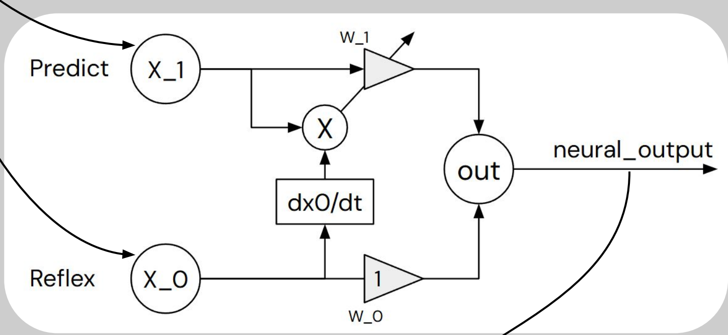
drive the output to the robot wheels with forward movement

Table of content

Hint 1

```
x0 = reflex_sig             -- Reflex signal
x1 = robot_target_omega  -- predictive signal

-- ======================================== --
--               Student Edit code Here         --
-- ======================================== --
```




```
-- ======================================== --
-- ======================================== --

-- add noise to output
output_angle = neural_output + noise*math.max(0, (1-w1))
```

Table of content

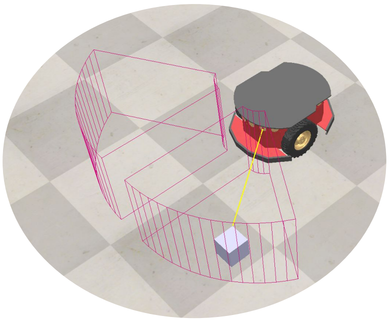
Q-learning

Table of content

Q-learning task

The robot has ultrasonic sensors on left and right side (see next slide). If a sensor detects an obstacle (obstacle in sensor area), its value is one, if not, the value is zero.

Your task is Use Q-learning for controlling the robot avoid hitting the obstacle.



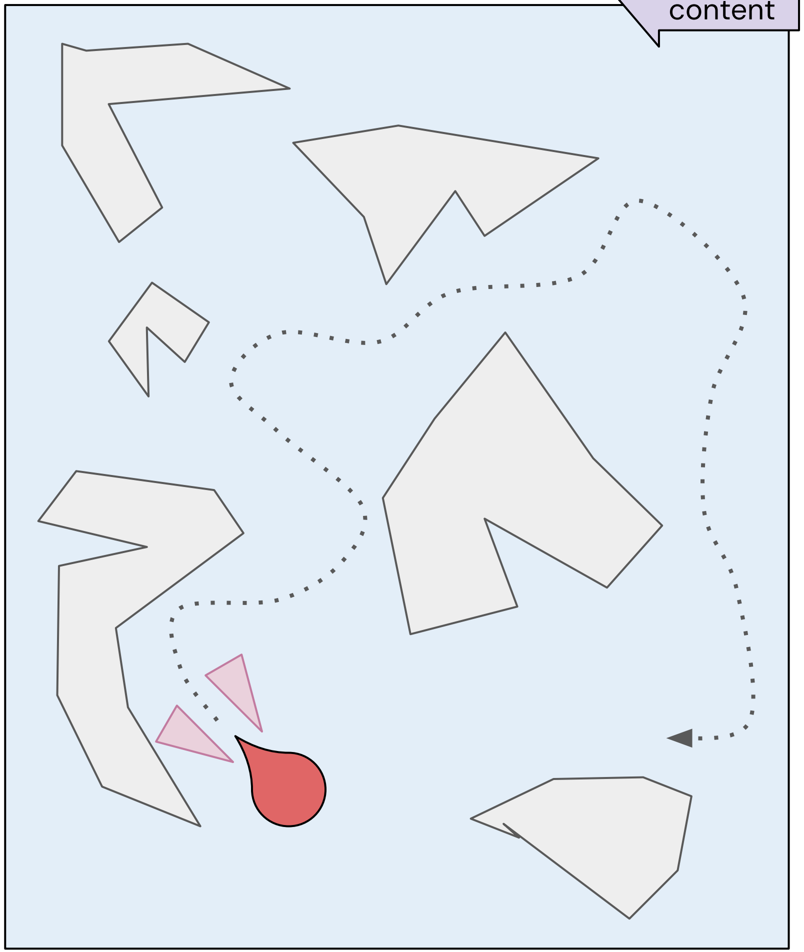
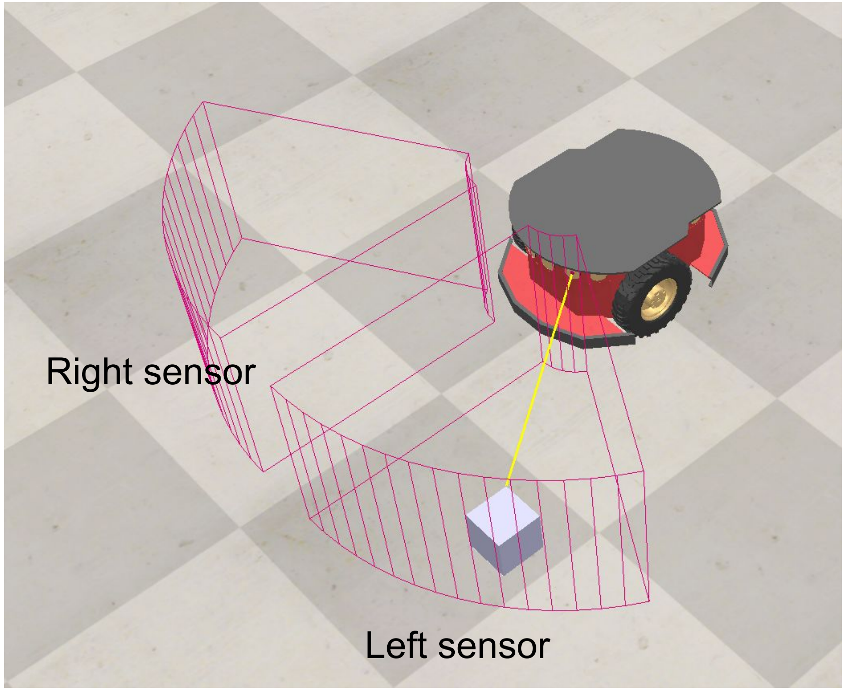


Table of content

2-wheeled robot



The wheeled robot consists of two wheels and two ultrasonic sensors (left and right sensors). **The red lines** indicate detecting area, if there is an object being in the area, you can see **a yellow line** pointing to the object. In this case, the left sensor value is equal to one, while the right sensor value is zero.

Table of content

Scene in Simulation

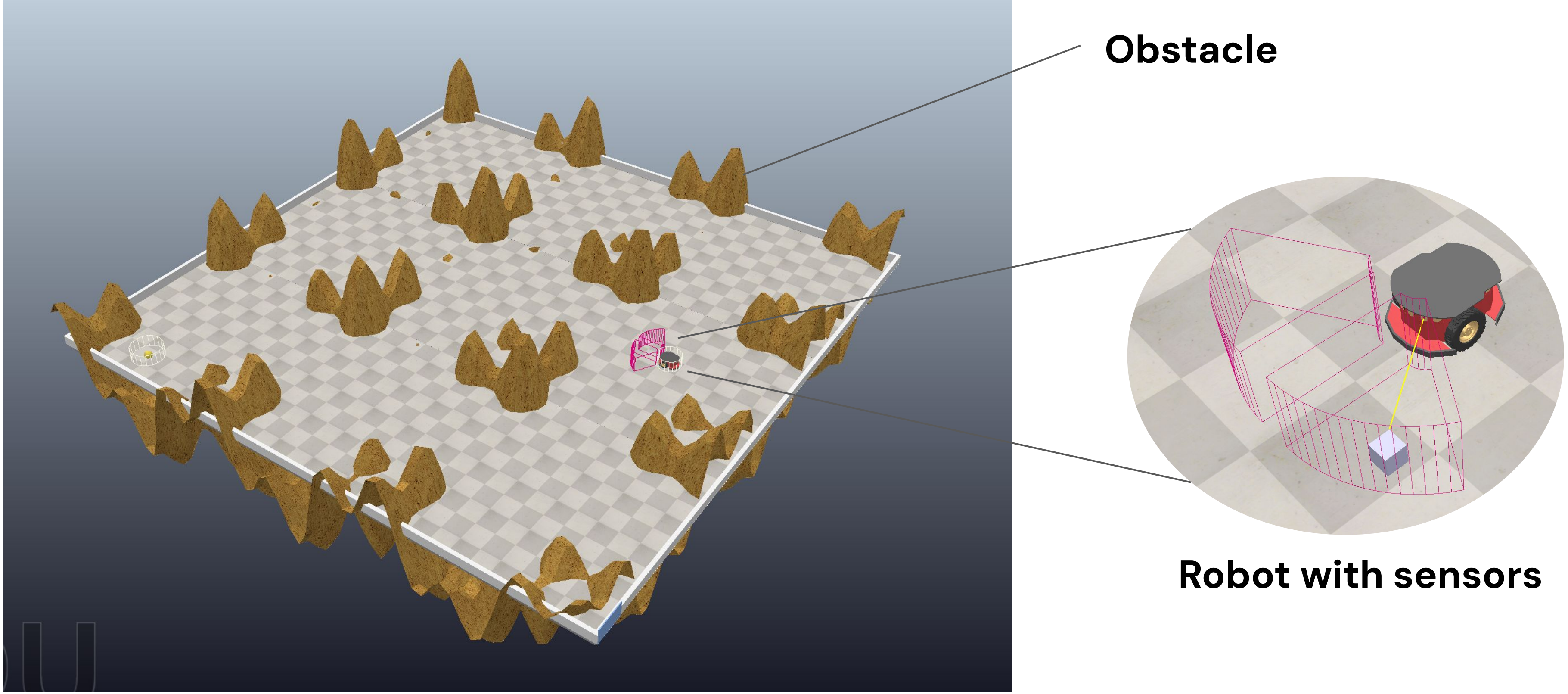


Table of content

Global parameter initiation

```lua
-- ############################## --
--       Q-Learning or SARSA      --
-- ############################## --

-- ========================================================== --
--              Student Initial parameter Here                --
-- ========================================================== --

-- state|          action (move)
--      | forward | left | right | back|
-- {0,0}|    0    |  0   |   0   |  0  |
-- {0,1}|    0    |  0   |   0   |  0  |
-- {1,0}|    0    |  0   |   0   |  0  |
-- {1,1}|    0    |  0   |   0   |  0  |

Q =  {{0,0,0,0},
      {0,0,0,0},
      {0,0,0,0},
      {0,0,0,0}}


alpha = 0.1 -- learning rate
gamma = 0.9 -- discount factor
epsilon = 0.1 --exploration rate
reward = 0
max_episodes = 100
current_episodes = 0

state = 0
next_state = 0
action = 0

-- ========================================================== --
-- ========================================================== --

end -- sysCall_init
```

For example

Initial parameters in this zone

Hint: Create Array in Lua code

Array_2D = { {1, 2, 3},
{4,5, 6,} }

Table of content

Editable code in sysCall_thread()

```lua
function sysCall_thread()

    local dt = 0.2   -- 200 ms
    while sim.getSimulationState() ~= sim.simulation_advancing_abouttostop do
    -- ========================================================= --
    --                  Student Edit code Here                   --
    -- ========================================================= --

        -- ===== 1. READ SENSORS =====
        sensor_left, sensor_right = get_sensor()
        print(sensor_left, sensor_right)

        -- ===== 2. SELECT ACTION =====


        -- ===== 3. EXECUTE ACTION =====


        -- ===== 4. WAIT CONTROL PERIOD =====
        sim.wait(dt) -- Don't delete this
        -- ==================================

        -- ===== 5. OBSERVE NEXT STATE & REWARD =====


        -- ===== 6. Q-LEARNING UPDATE =====


    -- ========================================================= --
    -- ========================================================= --

    end
end
```

You have to code only in this while-loop in **the sysCall_thread()** function. This loop will be run every 200 ms. You don't need to create any for-loop.

We have provided you a procedural step 1 - 6, which you can follow or not.

You can create a new function for operating in this while-loop such as get_action(), and get_state().

Table of content

Create custom function

```lua
-- ============================================================================ --
--                       Student Create Function Here
-- ============================================================================ --















-- ===================== --
function example_create_function(a,b,c)
    print(a,b,c)
    return a+b+c, a-b-c
end

```

Example Function with two returning values

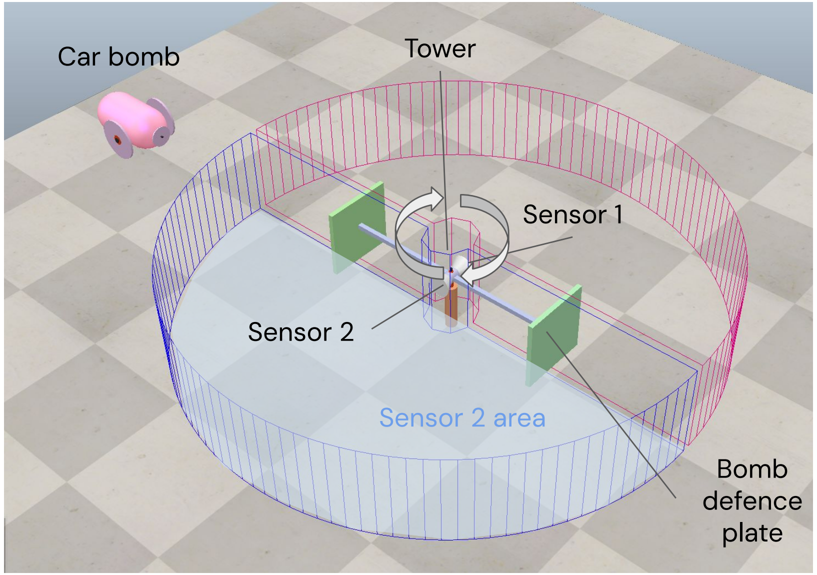
You can create a new function and use it in this while-loop such as get_action(), and get_state().

Table of content

XOR tower defence

Table of content

Short story for tower defence



During World War, enemy forces invaded your territory and attempted to destroy a critical communication tower used to contact allied intelligence units. To stop this, the enemy sent explosive vehicles toward the tower.

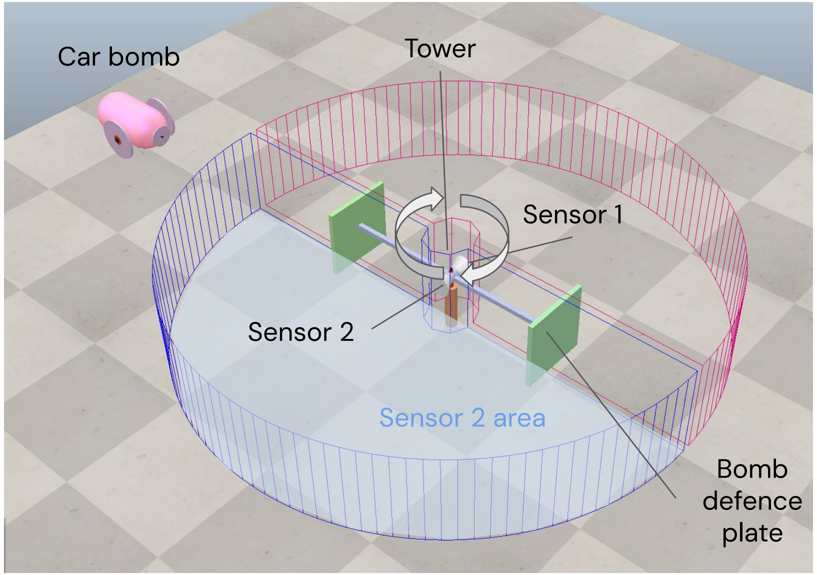
Due to limited time and resources, you built two small movable titanium walls that can withstand explosions. Using a one-directional motor salvaged from a damaged fighter aircraft, the walls rotate around the tower to block incoming explosives.

Two damaged radars are installed on the tower. Although each radar can scan up to 180 degrees, they can only detect the presence of an object, not its exact position.

Your task is to develop a neural network that controls the rotating walls using only the limited input from the two radars to protect the communication tower.

Table of content

Short story for tower defence



The tower consists of one directional motor, two ultrasonic sensors with 180 degrees range detection, and two bomb defence plates for stopping and bombing the car bomb.

Your task is rotate the bomb defence plate to the car bomb direction to protect the tower by using XOR logic.

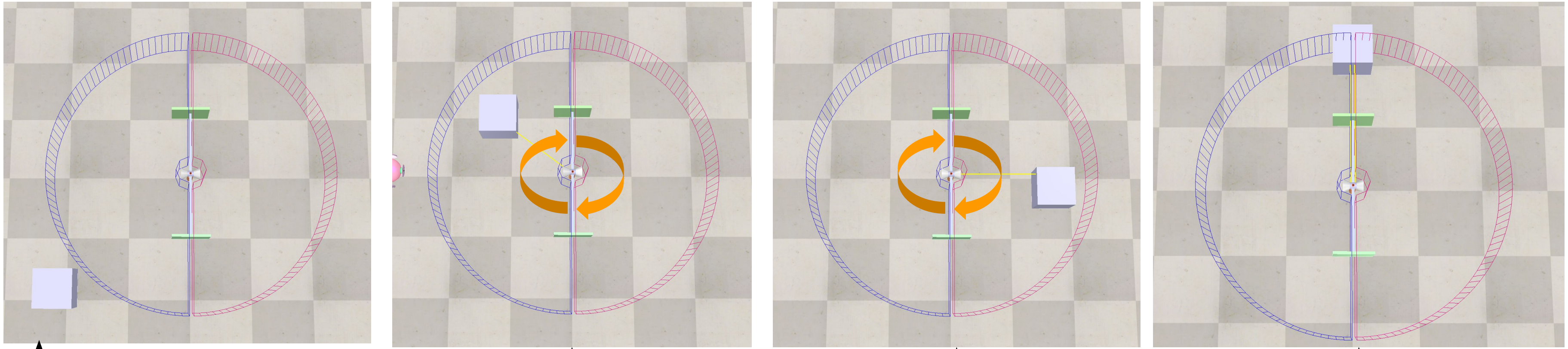
Hint:
The motor should stop when both sensor can detect object simultaneously
||
The plate will fit perfectly in front of the car bomb.

XOR logic

Sens 1	Sens 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

Table of content

XOR logic with tower action



XOR logic

Sens 1	Sens 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

Table of content

CoppeliaSim code (Parameter Initiation)

Parameter initiation

```
    -- ========================================================= --
    --                Student Initial parameter Here              --
    -- ========================================================= --

    -- ================ XOR ================== --
    -- A, B, Output of XOR logic
    training_data = {{0,0,0},
                     {0,1,1},
                     {1,0,1},
                     {1,1,0}}

    training = true

    -- ========================================================= --
    -- ========================================================= --
end -- End of SysCall Initial
```

Initial parameters in this zone

CoppeliaSim code (Training and testing functions)

```
function XOR_training()
    -- Neural
    -- [In1]------W_1-------\           Target
    --       \               \             |
    --     W_2\               \            |
    --         \               \           V
    --          [h1]---W_5---> [Out]---->(+)----> error
    --         /   /           /   /
    --     W_3/   /W_6        /   /W_7
    --       /   /           /   /
    -- [In2]----/---W_4-----/   /
    --         /               /
    --      [B1]       [B2]---/
    --

    -- ======================================================== --
    --                    Student Edit code Here                  --
    -- ======================================================== --

    for epoch = 1, 10000 do
        -- print(sen_val_1, sen_val_2)
        for data_i=1 , 4 do

            -- Neural output equation --

            -- Find Error & delta error --

            -- Update weights --

        end -- each data
        if epoch % 1000 == 0 then
            print('epoch = ' .. epoch)
        end
    end -- each epoch

    -- ======================================================== --
    -- ======================================================== --

end -- end of XOR_training function
```

Code here for training neural network

```
function XOR_testing(input_1, input_2)
    neural_output = 0
    -- ======================================================== --
    --                    Student Edit code Here                  --
    -- ======================================================== --

    -- Neural equation with updated weights

    -- Example output, student should delete thses when do the task
    if input_1 == input_2 then
        neural_output = 0
    else
        neural_output = 1
    end


    -- ======================================================== --
    -- ======================================================== --

    return neural_output

end -- end of XOR_testing function
```

Code here for Testing neural network

Table of content

CoppeliaSim code (driving neural output)

You don’t need to edit these function, just understand what they do in the program

```lua

-- This funciton will enable the XOR_testing fucntion
-- when XOR_training funciton have finished training process
function sysCall_thread()
    if training == true then
        print("training")
        XOR_training()
        training = false
    end
end



```

This function will enable XOR_testing function after training process finishes

```lua
--==========================================================================--
--==========================================================================--

function sysCall_actuation()
    -- When training process end, XOR_testing function will be executed
    drive = 0
    if training == false then
        drive = XOR_testing(sen_result_1, sen_result_2)
    end

    -- Drive to the motor
    sim.setJointTargetVelocity(joint, drive*0.5)

end

```

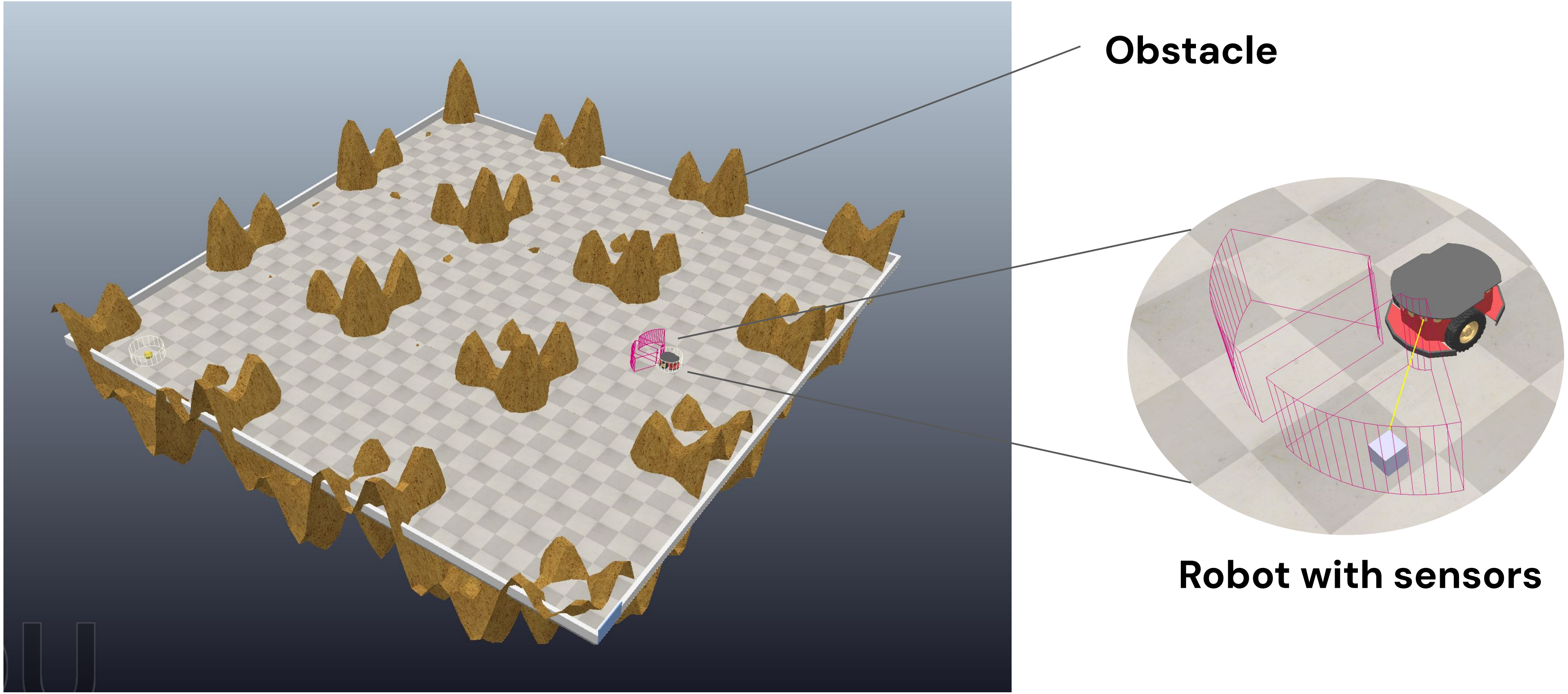
This function execute XOR_testing and drive the neural output to the motor

End

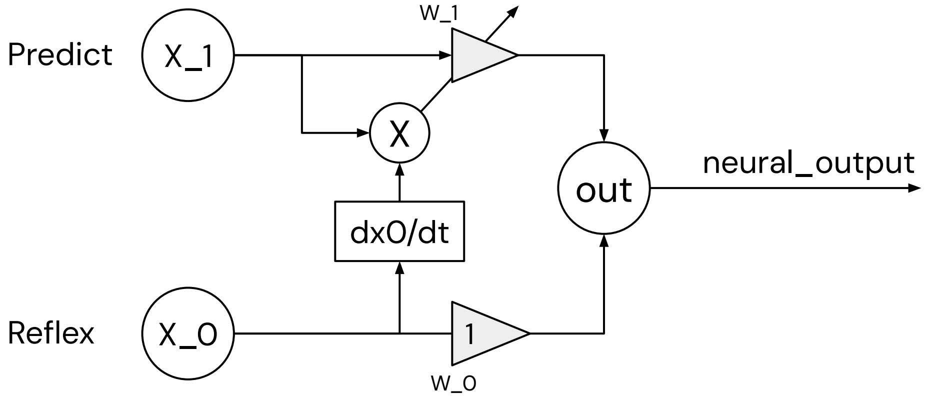
Table of content

SARSA

Scene in Simulation



ICO Answer


Predict
X_1
W_1
X
out
neural_output
dx0/dt
Reflex
X_0
1
W_0