pyrosim Documentation

Release 0.1.1

Josh Bongard, Collin Cappelle, UVM MECL

Jun 08, 2017

CONTENTS

(Py)thon (Ro)bot (Sim)ulator is a python interface for ODE, primarily focused on neurally controlled robots. With it you can send bodies (cylinders, boxes, etc.), joints, and neural networks. Pyrosim is developed and maintained by the members of the MEC-Lab at The University of Vermont.

Requirements:

Pyrosim currently only requires numpy.

Contents:

CHAPTER

ONE

AUTO GENERATED DOCUMENTATION

class pyrosim.**Simulator**(*play_blind=False, play_paused=False, eval_time=100, dt=0.05, gravity=-1.0, xyz=[0.8317, -0.9817, 0.8], hpr=[121, -27.5, 0.0], use_textures=False, debug=False*)

Python interface for ODE simulator

Attributes

play_blind	(bool, optional) If True the simulation runs without graphics (headless) else if False the simulation runs with graphics (the default is False)
play_paused	(bool, optional) If True the simulation starts paused else if False the simulation starts running. With simulation window in focus use Ctrl-p to toggle pausing the simulation. (the default is False)
eval_time	(int, optional) The number of discrete steps in the simulation (the default is 100)
dt	(float, optional) The time in seconds between physics world steps. Larger dt values create more unstable physics. (the default is 0.05)
gravity	(float, optional) The gravity in the system. Negative values implie normal downward force of gravity. (default is -1.0)
xyz	(list of 3 floats) The xyz position of the camera (default is [0.8317,-0.9817,0.8000])
hpr	The heading, pitch, and roll of the camera (default is [121,-27.5,0.0])
use_textures	(bool, optional) Draw default ODE textures or not during simulation. (default is False)
debug	(bool, optional) If True print out every string command sent through the pipe to the simulator (the default is False)

Methods

film_body(*body_id*, *method='follow'*)

Sets the camera to film a body

Camera has two modes: 'follow' moves the camera's position based on where the body is moving and 'track' rotates the camera to look at the body

Parameters **body_id** : int

The id tag of the body to be filmed

method : str, optional

The way the camera should move to film the body. Either 'follow' or 'track' (default is 'follow')

Returns bool

True if successful, False otherwise

get_data()
Get all sensor data back as numpy matrix

get_num_bodies()
Returns the number of bodies

get_num_joints()
Returns the number of joints

get_num_neurons()
Returns the number of neurons

get_num_sensors()
Returns the number of sensors

get_sensor_data(*sensor_id*, *svi=0*)
Get the post simulation data from a specified sensor

Parameters **sensor_id** : int

the sensors id tag

svi : int , optional

The sensor value index. Certain sensors have multiple values (e.g. the position sensor) and the svi specifies which to access (e.g. for a position sensor, svi=0 corresponds to the x value of that sensor)

Returns list of float

Returns the list of sensor values over the simulation.

send_bias_neuron()
Send bias neuron to simulator.

Bias neurons emit a constant value of 1.0

Returns int

id tag of the neuron

send_box(*x=0*, *y=0*, *z=0*, *length=0.1*, *width=0.1*, *height=0.1*, *r=1*, *g=1*, *b=1*)
Send box body to the simulator

Parameters **x** : float, optional

The x position coordinate of the center

y : float, optional

The y position coordinate of the center

z : float, optional

The z position coordinate of the center

length : float, optional

The length of the box

width : float, optional

The width of the box

height : float, optional

The height of the box

r : float, optional

The amount of the color red in the box (r in [0,1])

g : float, optional

The amount of the color green in the box (g in [0,1])

b : float, optional

The amount of the color blue in the box (b in [0,1])

Returns int

id tag of the box

send_camera(*xyz*, *hpr*)

Sends camera position to simulator in eulerian coordinates

Parameters **xyz** : list of floats

A length 3 list specifying the x,y,z position of the camera in simulation

hpr : list of floats

A length 3 list specifying the heading, pitch, and roll of the camera

Returns bool

True if successful, False otherwise

send_cylinder(*x=0*, *y=0*, *z=0*, *r1=0*, *r2=0*, *r3=1*, *length=1.0*, *radius=0.1*, *r=1*, *g=1*, *b=1*)

Send cylinder body to the simulator

Parameters **x** : float, optional

The x position coordinate of the center (default is 0)

y : float, optional

The y position coordinate of the center (default is 0)

z : float, optional

The z position coordinate of the center (default is 0)

r1 : float, optional

The orientation along the x axis. The vector [r1,r2,r3] specify the direction of the long axis of the cylinder. (default is 0)

r2 : float, optional

The orientation along the y axis. The vector [r1,r2,r3] specify the direction of the long axis of the cylinder. (default is 0)

r3 : float, optional

The orientation along the z axis. The vector [r1,r2,r3] specify the direction of the long axis of the cylinder. (default is 1)

length : float, optional

The length of long axis of the cylinder (default is 1.0)

radius : float, optional

The radius of the short axis of the cylinder (default is 0.1)

r : float, optional

The amount of the color red in the box (r in [0,1])

g : float, optional

The amount of the color green in the box (g in [0,1])

b : float, optional

The amount of the color blue in the box (b in [0,1])

Returns int

The id tag of the cylinder

send_developing_synapse(*source_neuron_id=0*, *target_neuron_id=0*, *start_weight=0.0*, *end_weight=0.0*, *start_time=0.0*, *end_time=1.0*)

Sends a synapse to the simulator

Developing synapses are synapses which change over time. The synapse will interpolate between the start_weight and end_weight over the desired time range dictated by start_time and end_time. start_time and end_time are in [0,1] where 0 maps to time step 0 and 1 maps to the eval_time of the simulation. Setting start_time equal to end_time results in a discrete change from start_weight to end_weight in the synapse at the specified time step. If start_time >= end_time times are changed such that end_time = start_time.

Parameters **source_neuron_id** : int, optional

The id of the source neuron of the synapse

target_neuron_id : int, optional

The id of the target neuron of the synapse

start_weight : float, optional

The starting edge weight of the synapse

end_weight : float, optional

The ending edge weight of the synapse

start_time : float, optional

The starting time of development. start_time in [0,1]

end_time : float, optional

The ending time of development. end_time in [0,1]

Returns

——-

bool

True if successful, False otherwise

send_function_neuron(*function=<built-in function sin>*)

Send neuron to simulator which takes its value from the user defined function

The function is mapped to the specific time in the simulation based on both the discrete evaluation time and the dt space between time steps. For example if evalTime=100 and dt=0.05 the function will be evaluated at [0,0.05,...,5]

Parameters **function** : function, optional

The function which defines the neuron value. Valid functions return a single float value over the time domain.

Returns int

The id tag of the neuron

send_hidden_neuron(*tau=1.0*)

Send a hidden neuron to the simulator

Hidden neurons are basic neurons which can have inputs and outputs. They 'hidden' between input neurons (sensors, bias, function) and output neurons (motors)

Parameters **tau** : float, optional

The 'learning rate' of the neuron. Increasing tau increases how much of value of the neuron at the current time step comes from external inputs vs. the value of the neuron at the previous time step

Returns int

The id tag of the neuron

send_hinge_joint(*first_body_id*, *second_body_id*, *x=0*, *y=0*, *z=0*, *n1=0*, *n2=0*, *n3=1*, *lo=-0.7853981633974483*, *hi=0.7853981633974483*, *speed=1.0*, *torque=10.0*, *position_control=True*)

Send a hinge joint to the simulator

Parameters **first_body_id** : int

The body id of the first body the joint is connected to. If set equal to -1, the joint is connected to a point in space

secondbodyid : int

The body id of the second body the joint is connected to. If set equal to -1, the joint is connected to a point in space

x : float, optional

The x position coordinate of the joint (default is 0)

y : float, optional

The y position coordinate of the joint (default is 0)

z : float, optional

The z position coordinate of the joint (default is 0)

n1 : float, optional

The orientation along the x axis. The vector [n1,n2,n3] specifies the axis about which the joint rotates (default is 0)

n2 : float, optional

The orientation along the y axis. The vector [n1,n2,n3] specifies the axis about which the joint rotates (default is 0)

n3 : float, optional

The orientation along the z axis. The vector [n1,n2,n3] specifies the axis about which the joint rotates (default is 1)

lo : float, optional

The lower limit in radians of the joint (default is -pi/4)

hi : float, optional

The upper limit in radians of the joint (default is pi/4)

speed : float, optional

The speed of the motor of the joint (default is 1.0)

torque : float, optional

The amount of torque the motor in the joint has (default is 10.0)

position_control : bool, optional

True means use position control. This means the motor neuron output is treated as a target angle for the joint to actuate to. False means the motor neuron output is treated as a target actuation rate.

Returns int

The id tag for the hinge joint

send_light_sensor(*body_id=0*)

Attaches a light sensor to a body in simulation

Parameters **body_id** : int, optional

The body id of the body to connect the sensor to

Returns int

The id tag of the sensor

send_light_source(*body_id=0*)

Attaches light source to a body in simulation

Parameters **body_id** : int, optional

The body id of the body to attach the light to

Returns int

The id tag of the body the light source is attached to.

send_motor_neuron(*joint_id=0*, *tau=1.0*)

Send motor neurons to simulator

Motor neurons are neurons which connecto to a specified joint and determine how the joint moves every time step of simulation

Parameters **joint_id** : int, optional

The joint id tag of the joint we want the neuron to connect to

tau :

The 'learning rate' of the neuron. Increasing tau increases how much of value of the neuron at the current time step comes from external inputs vs. the value of the neuron at the previous time step

Returns int

The id tag of the neuron

send_position_sensor(*body_id=0*)

Attaches a position sensor to a body in simulation

Parameters **body_id** : int, optional

The body id of the body to connect the sensor to

Returns int

The id tag of the sensor

send_proprioceptive_sensor(*joint_id=0*)

Attaches a proprioceptive sensor to a joint in simulation

Proprioceptive sensors returns the angle of the joint at each time step

Parameters **joint_id** : int, optional

The joint id of the joint to connect the sensor to

Returns int

The id tag of the sensor

send_ray_sensor(*body_id=0*, *x=0*, *y=0*, *z=0*, *r1=0*, *r2=0*, *r3=1*)

Sends a ray sensor to the simulator connected to a body

Ray sensors return four values each time step, the distance and color (r,g,b).

Parameters **body_id** : int, optional

The body id of the associated body the ray sensor is connected to. When this body moves the ray sensor moves accordingly

x : float, optional

The x position of the sensor

y : float, optional

The y position of the sensor

z : float, optional

The z position of the sensor

r1 : float, optional

The x direction of the sensor. The array [r1,r2,r3] is the direction the ray sensor is pointing in the time step.

r2 : float, optional

The y direction of the sensor. The array [r1,r2,r3] is the direction the ray sensor is pointing in the time step.

r3 : float, optional

The z direction of the sensor. The array [r1,r2,r3] is the direction the ray sensor is pointing in the time step.

Returns int

The id tag of the sensor

send_sensor_neuron(*sensor_id=0*, *svi=0*, *tau=1.0*)

Sends a sensor neuron to the simulator

Sensor neurons are input neurons which take the value of their associated sensor

Parameters **sensor_id** : int, optional

The associated sensor id for the neuron to draw values from.

svi : int, optional

The sensor value index is the offset index of the sensor. SVI is used for sensors which return a vector of values (position, ray sensors, etc.)

tau : int, optional

not used for sensor neurons

Returns int

The id tag of the neuron

send_sphere(*x=0*, *y=0*, *z=0*, *radius=0.5*, *r=1*, *g=1*, *b=1*)

Sends a sphere to the simulator

Parameters **x** : float, optional

The x position of the center

y : float, optional

The y position of the center

z : float, optional

The z position of the center

radius : float, optional

The radius of the sphere (default is 0.5)

r : float, optional

The amount of the color red in the box (r in [0,1])

g : float, optional

The amount of the color green in the box (g in [0,1])

b : float, optional

The amount of the color blue in the box (b in [0,1])

Returns int

The id tag of the sphere

send_synapse(*source_neuron_id=0*, *target_neuron_id=0*, *weight=0.0*)

Sends a synapse to the simulator

Synapses are the edge connections between neurons

Parameters **source_neuron_id** : int, optional

The id of the source neuron of the synapse

target_neuron_id : int, optional

The id of the target neuron of the synapse

weight : float, optional

The edge weight of the synapse

Returns bool

True if successful, False otherwise

send_touch_sensor(*body_id=0*)

Send touch sensor to a body in the simulator

Parameters **body_id** : int, optional

The body id of the associated body

Returns int

The id tag of the sensor

send_user_input_neuron(*in_values*)
Send neuron to the simulator which takes user defined values at each time step

Parameters **in_values** : list of floats or float, optional

The user specified values for the neuron. If length of values < the number of time steps, the values are continually looped through until every time step has a corresponding value

Returns int

The id tag of the neuron.

send_vestibular_sensor(*body_id=0*)
Connects a vestibular sensor to a body

Vestibular sensors return a bodies orrientation in space

Parameters **body_id** : int, optional

The body id of the associated body

Returns int

The id tag of the sensor

start()
Starts the simulation

wait_to_finish()
Waits to for the simulation to finish and collects data

Returns numpy matrix

A matrix of the sensor values for each time step of the simulation

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

p

F

G

P

S

W