pyrosim Documentation

Release 0.0.1

Collin Cappelle

Jun 05, 2017

CONTENTS

Contents:

PYROSIM CLASS DOCUMENTATION

class pyrosim.**PYROSIM**(*playBlind=False, playPaused=False, evalTime=100, dt=0.05, xyz=[0.8317, -0.9817, 0.8], hpr=[121, -27.5, 0.0], 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)
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])
debug	(bool, optional) If True print out every string command sent through the pipe to the simulator (the default is False)

Methods

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

> **Parameters sensorID** : int , optional
>
>> 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(*neuronID=0*)
Send bias neuron to simulator

> **Parameters neuronID** : int, optional

User specified ID tag for the neuron

> **Returns** bool
>
> True if successful, False otherwise

Send_Box (*objectID=0*, *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 objectID** : int, optional
>
> > User specified body ID tag for the box
>
> **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** bool
>
> True if successful, False otherwise

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** None

Send_Cylinder (*objectID=0*, *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 objectID** : int, optional

User specified body ID tag for the cylinder (default is 0)

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 bool

True if successful, False otherwise

Send_Developing_Synapse (*sourceNeuronID=0*, *targetNeuronID=0*, *startWeight=0.0*, *endWeight=0.0*, *startTime=0.0*, *endTime=1.0*)
Sends a synapse to the simulator

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

Parameters **sourceNeuronID** : int, optional

The ID of the source neuron of the synapse

targetNeuronID : int, optional

The ID of the target neuron of the synapse

startWeight : float, optional

The starting edge weight of the synapse

endWeight : float, optional

The ending edge weight of the synapse

startTime : float, optional

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

endTime : float, optional

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

Returns:

——

bool

True if successful, False otherwise

Send_Function_Neuron (*neuronID=0*, *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 **neuronID** : int, optional

The user specified ID tag of the neuron

function : function, optional

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

Returns bool

True if successful, False otherwise

Send_Hidden_Neuron (*neuronID=0*, *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 **neuronID** : int, optional

The user specified ID tag of the neuron

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 bool

True if successful, False otherwise

Send_Joint (*jointID=0*, *firstObjectID=0*, *secondObjectID=1*, *x=0*, *y=0*, *z=0*, *n1=0*, *n2=0*, *n3=1*, *lo=-0.7853981633974483*, *hi=0.7853981633974483*, *speed=1.0*, *torque=10.0*, *positionControl=True*)

Send a hinge joint to the simulator

Parameters **jointID** : int, optional

User specified ID tag for the joint (default is 0)

firstObjectID : int, optional

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 (default is 0)

secondOjbectID : int, optional

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 (default is 1)

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)

positionControl : 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 bool

True if successful, False otherwise

Send_Light_Sensor (*sensorID=0*, *objectID=0*)
    Attaches a light sensor to a body in simulation

> **Parameters sensorID** : int, optional
>
> > The user defined ID of the sensor
>
> **objectID** : int, optional
>
> > The body ID of the body to connect the sensor to
>
> **Returns** bool
>
> > True if successful

Send_Light_Source (*objectIndex=0*)
    Attaches light source to a body in simulation

> **Parameters objectIndex** : int, optional
>
> > The body ID of the body to attach the light to
>
> **Returns** bool
>
> > True if successful, False otherwise

Send_Motor_Neuron (*neuronID=0*, *jointID=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 neuronID** : int, optional
>
> > The user specified ID tag of the neuron
>
> **jointID** : 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** bool
>
> > True if successful, False otherwise

Send_Position_Sensor (*sensorID=0*, *objectID=0*)
    Attaches a position sensor to a body in simulation

> **Parameters sensorID** : int, optional
>
> > The user defined ID of the sensor
>
> **objectID** : int, optional
>
> > The body ID of the body to connect the sensor to
>
> **Returns** bool
>
> > True if successful

Send_Proprioceptive_Sensor (*sensorID=0*, *jointID=0*)
    Attaches a proprioceptive sensor to a joint in simulation

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

Parameters sensorID : int, optional

The user defined ID of the sensor

jointID : int, optional

The joint ID of the joint to connect the sensor to

Returns bool

True if successful

Send_Ray_Sensor (*sensorID=0*, *objectID=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 sensorID : int, optional

The user defined ID tag for the ray sensor

objectID : 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.

Send_Sensor_Neuron (*neuronID=0*, *sensorID=0*, *sensorValueIndex=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 neuronID : int, optional

The user defined ID of the neuron

sensorID : int, optional

The associated sensor ID for the neuron

sensorValueIndex : 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 bool

True if successful, False otherwise

Send_Synapse (*sourceNeuronID=0*, *targetNeuronID=0*, *weight=0.0*)
Sends a synapse to the simulator

Synapses are the edge connections between neurons

Parameters sourceNeuronID : int, optional

The ID of the source neuron of the synapse

targetNeuronID : int, optional

The ID of the target neuron of the synapse

weight : float, optional

The edge weight of the synapse

Send_Touch_Sensor (*sensorID=0*, *objectID=0*)
Send touch sensor to a body in the simulator

Parameters sensorID : int, optional

The user defined ID of the sensor

objectID : int, optional

The body ID of the associated body

Returns bool

True if successful, False otherwise

Send_User_Input_Neuron (*neuronID=0*, *values=1*)
Send neuron to the simulator which takes user defined values at each time step

Parameters neuronID : int, optional

The user specified ID tag of the neuron

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 bool

True if successful, False otherwise

Send_Vestibular_Sensor (*sensorID=0*, *objectID=0*)
Connects a vestibular sensor to a body

Vestibular sensors return a bodies orrientation in space

Parameters sensorID : int, optional

The user defined ID of the sensor

> > > **objectID** : int, optional
> > >
> > > > The body ID of the associated body
> >
> > **Returns** bool
> >
> > > True if successful, False otherwise

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

INDICES AND TABLES

- genindex
- modindex
- search

p

pyrosim, 3

G

P

S

W