My Project

Generated by Doxygen 1.8.13

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

Class Documentation

2.1 TurtlesimSIU.TurtlesimSIU.ColorSensor Class Reference

Public Member Functions

- def __init__ (self, owner)

  The ColorSensor class initializer.
- def topic_callback (self, data)

  Updates current color below the turtle.
- def check (self)

  Returns last color received by the sensor.

Public Attributes

- **owner**
- **colour**

2.1.1 Constructor & Destructor Documentation

2.1.1.1 __init__()

```
def TurtlesimSIU.TurtlesimSIU.ColorSensor.__init__ (
              self,
              owner )
```

The ColorSensor class initializer.

Parameters

owner	The name of the turtle that owns the sensor.

Returns

An instance of the ColorSensor class initialized with the specified turtle name.

2.1.2 Member Function Documentation

2.1.2.1 check()

```
def TurtlesimSIU.TurtlesimSIU.ColorSensor.check (
              self )
```

Returns last color received by the sensor.

Returns

last color received by the sensor. The returned object has 'r', 'g' and 'b' fields and their values are between 0-255.

2.1.2.2 topic_callback()

```
def TurtlesimSIU.TurtlesimSIU.ColorSensor.topic_callback (
              self,
              data )
```

Updates current color below the turtle.

It is called each 16 milisecond

The documentation for this class was generated from the following file:

- TurtlesimSIU.py

2.2 TurtlesimSIU.TurtlesimSIU.TurtlesimSIU Class Reference

Public Member Functions

- def __init__ (self)

  The TurtlesimSIU class initializer.
- def getFrameSize (self)

  Returns size of the environment.
- def getPose (self, turtle_name)

  Returns current pose of the given turtle.
- def setVel (self, turtle_name, vel)

  Sets velocity to the given turtle.
- def setPen (self, turtle_name, req)

Sets the given turtle's pen.
- def hasTurtle (self, turtle_name)

  Checks if the given turtle exists.
- def killTurtle (self, turtle_name)

  Kill the given turtle and remove its velocity publisher and teleport service client.
- def spawnTurtle (self, turtle_name, pose)

  Spawns the given turtle in the given localisation.
- def readSonar (self, fov_center, fov_range, range_min, range_max, owner)

  Checks the closes turtle in the area given by the parameters.
- def readCamera (self, name='turtle1', frame_pixel_size=200, cell_count=16, x_offset=0, goal=Pose(), show←_matrix_cells_and_goal=False)

  Reads image from the given turtles camera.
- def getColisions (self, names, collision_range)

  Check collistions between the given turtles.
- def setPose (self, turtle_name, pose, mode='absolute')

  Teleport the given turtle.
- def pixelsToScale (self)

  Returns the pixels/meter scaling factor.

Public Attributes

- **get_turtles**
- **get_pose**
- **spawn**
- **get_sonar**
- **get_camera_image**
- **has_turtle**
- **kill_turtle**
- **get_frame_size**
- **vel_publishers**
- **teleport_srvs**

2.2.1 Detailed Description

```
docstr for TurtlesimSIU
```

2.2.2 Constructor & Destructor Documentation

2.2.2.1 __init__()

```
def TurtlesimSIU.TurtlesimSIU.TurtlesimSIU.__init__ (
              self )
```

The TurtlesimSIU class initializer.

It should be called AFTER the turtle environment startup.

Returns

An instance of the TurtlesimSIU class.

2.2.3 Member Function Documentation

2.2.3.1 getColisions()

```
def TurtlesimSIU.TurtlesimSIU.TurtlesimSIU.getColisions (
              self,
              names,
              collision_range )
```

Check collistions between the given turtles.

Parameters

names	The names of the turtles.
collision_range	The minimal distance between the turtles.

Returns

list of the turtle pairs that colide

2.2.3.2 getFrameSize()

```
def TurtlesimSIU.TurtlesimSIU.TurtlesimSIU.getFrameSize (
              self )
```

Returns size of the environment.

Returns

is an object :
float32 width
(unit: meter) float32 height
(unit: meter)

2.2.3.3 getPose()

```
def TurtlesimSIU.TurtlesimSIU.TurtlesimSIU.getPose (
              self,
              turtle_name )
```

Returns current pose of the given turtle.

Parameters

turtle_name	The name of the turtle.

Returns

is a Pose object :
float32 x
(unit: meter) float32 y
(unit: meter) float32 theta
(unit: radian) float32 linear_velocity
(unit: meter/sec) float32 angular_velocity
(unit: radian/sec)

2.2.3.4 hasTurtle()

```
def TurtlesimSIU.TurtlesimSIU.TurtlesimSIU.hasTurtle (
              self,
              turtle_name )
```

Checks if the given turtle exists.

Parameters

turtle_name	The name of the turtle.

Returns

True if the turtle exists.

2.2.3.5 killTurtle()

```
def TurtlesimSIU.TurtlesimSIU.TurtlesimSIU.killTurtle (
              self,
              turtle_name )
```

Kill the given turtle and remove its velocity publisher and teleport service client.

Parameters

turtle_name	The name of the turtle.

2.2.3.6 pixelsToScale()

```
def TurtlesimSIU.TurtlesimSIU.TurtlesimSIU.pixelsToScale (
              self )
```

Returns the pixels/meter scaling factor.

Returns

The pixels/meter scaling factor.

2.2.3.7 readCamera()

```
def TurtlesimSIU.TurtlesimSIU.TurtlesimSIU.readCamera (
              self,
              name = 'turtle1',
              frame_pixel_size = 200,
              cell_count = 16,
              x_offset = 0,
              goal = Pose(),
              show_matrix_cells_and_goal = False )
```

Reads image from the given turtles camera.

The camera localisation and sensor size is configurable in the arguments.

Parameters

name	The name of the turtle owning the camera.
frame_pixel_size	The size of the camera sensor in pixels. The sensor is a square which a=frame_pixel_size.
cell_count	The count of the returned matrix cells. The matrix is square and is divided into the given number of cells.
x_offset	The offset in x direction (turtle front) of the camera localisation. If equals 0, the camera is in front of the turtle, and if equals -frame_pixel_size/2, the turtle is in the image center.
goal	The goal of the turtle to calculate distance from each cell to the goal. It is given by turtlesim.msg.Pose(x,y,theta). 'x' and 'y' in meters, theta in radians.
show_matrix_cells_and_goal	Triggers visualisation of the cells, the goal and the turtle pose

Returns

The NxN matrix, and each cell of the matrix has 4 fields: cell.red, cell.green, cell.blue, cell.distance. The latter is the distance to the specified goal. The size 'N' of the matrix is a square root of cell_count argument.

2.2.3.8 readSonar()

```
def TurtlesimSIU.TurtlesimSIU.TurtlesimSIU.readSonar (
              self,
              fov_center,
              fov_range,
              range_min,
              range_max,
              owner )
```

Checks the closes turtle in the area given by the parameters.

Parameters

owner	The name of the turtle owning the sonar.
fov_center	The direction of the sonar center.
fov_range	The angle of the sonar's field of view.
range_min	The min range of the sonar.
range_max	The max range of the sonar.

Returns

distance to the closes turtle in the given area

2.2.3.9 setPen()

```
def TurtlesimSIU.TurtlesimSIU.TurtlesimSIU.setPen (
              self,
              turtle_name,
              req )
```

Sets the given turtle's pen.

Parameters

turtle_name	The name of the turtle.
req	turtlesim.srv.SetPenRequest(r, g, b, width, off) object specifying the pen configuration.

2.2.3.10 setPose()

```
def TurtlesimSIU.TurtlesimSIU.TurtlesimSIU.setPose (
              self,
              turtle_name,
              pose,
              mode = 'absolute' )
```

Teleport the given turtle.

Parameters

turtle_name	The turtle_name of the turtle.
pose	The destination pose of the turtle.
mode	the mode of the teleportaiton ('absolute', 'relative'). **For 'absolute' translate and rotate afterwards, and for 'relative' rotate and translate afterwards**

Returns

True if succeeded

2.2.3.11 setVel()

```
def TurtlesimSIU.TurtlesimSIU.TurtlesimSIU.setVel (
              self,
              turtle_name,
              vel )
```

Sets velocity to the given turtle.

Parameters

turtle_name	The name of the turtle.
vel	geometry_msgs.msg.Twist object specifying the velocity.

Returns

True if the velocity was set

2.2.3.12 spawnTurtle()

```
def TurtlesimSIU.TurtlesimSIU.TurtlesimSIU.spawnTurtle (
              self,
              turtle_name,
              pose )
```

Spawns the given turtle in the given localisation.

Parameters

turtle_name	The name of the turtle.
pose	The pose of the turtle given by turtlesim.msg.Pose(x,y,theta). 'x' and 'y' in meters, theta in radians.

Returns

True if succeeded

The documentation for this class was generated from the following file:

- TurtlesimSIU.py