

Handshaking

Generated by Doxygen 1.8.12

Contents

1	README	1
2	Class Index	3
2.1	Class List	3
3	Class Documentation	5
3.1	Robot_VREP Class Reference	5
3.1.1	Constructor & Destructor Documentation	5
3.1.1.1	Robot_VREP()	5
3.1.1.2	Robot_VREP(const int clientId, std::vector< std::string > jointNames)	5
3.1.1.3	~Robot_VREP()	6
3.1.2	Member Function Documentation	6
3.1.2.1	getAngularForce(const unsigned int jointID)	6
3.1.2.2	getAngularPosition(const unsigned int jointID)	6
3.1.2.3	getAngularVelocity(const unsigned int jointID)	7
3.1.2.4	getMicroseconds()	7
3.1.2.5	getSeconds()	7
3.1.2.6	lockJoint(const int jointHandle)	8
3.1.2.7	setForce(const unsigned int jointID, const double force)	9
3.1.2.8	setPosition(const unsigned int jointID, const double pos)	9
3.1.2.9	setVelocity(const unsigned int jointID, const double speed)	10
3.1.2.10	startClock()	10
3.1.2.11	updateAngularForce(const unsigned int index=-1)	10
3.1.2.12	updateAngularPosition(const unsigned int index=-1)	11

3.1.2.13	<code>updateAngularVelocity(const unsigned int index=-1)</code>	11
3.1.2.14	<code>updateClock()</code>	11
3.2	Simulation Class Reference	12
3.2.1	Constructor & Destructor Documentation	12
3.2.1.1	<code>Simulation()</code>	12
3.2.1.2	<code>Simulation(std::vector< double > variables, std::vector< std::string > jointNames)</code>	12
3.2.1.3	<code>~Simulation()</code>	12
3.2.2	Member Function Documentation	12
3.2.2.1	<code>init()</code>	12
3.2.2.2	<code>run()</code>	13
3.2.2.3	<code>saveToFile(const std::string &path)</code>	13
3.2.2.4	<code>start()</code>	13
3.2.2.5	<code>step()</code>	13
Index		15

Chapter 1

README

IMPORTANT: The project is correctly set up in the XCode project but if for some reason you can't use XCode (like not having a Mac...), you will need to follow these instructions (until I write the appropriate makefile...): To compile the C++ files, you need to add the preprocessor macros: `NON_MATLAB_PARSING=1` and `MAX_EXT_API_CONNECTIONS=255`; You also need to compile `extApi.c` and `extApiPlatform`.

Chapter 2

Class Index

2.1 Class List

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

Robot_VREP	5
Simulation	12

Chapter 3

Class Documentation

3.1 Robot_VREP Class Reference

Public Member Functions

- [Robot_VREP](#) ()
- [Robot_VREP](#) (const int clientId, std::vector< std::string > jointNames)
- [~Robot_VREP](#) ()
- double [getAngularPosition](#) (const unsigned int jointID)
- double [getAngularVelocity](#) (const unsigned int jointID)
- double [getAngularForce](#) (const unsigned int jointID)
- bool [setPosition](#) (const unsigned int jointID, const double pos)
- bool [setVelocity](#) (const unsigned int jointID, const double speed)
- bool [setForce](#) (const unsigned int jointID, const double force)
- bool [updateAngularPosition](#) (const unsigned int index=-1)
- bool [updateAngularVelocity](#) (const unsigned int index=-1)
- bool [updateAngularForce](#) (const unsigned int index=-1)
- bool [lockJoint](#) (const int jointHandle)
- void [startClock](#) ()
- void [updateClock](#) ()
- double [getMicroseconds](#) ()
- double [getSeconds](#) ()

3.1.1 Constructor & Destructor Documentation

3.1.1.1 Robot_VREP::Robot_VREP ()

[Robot_VREP](#) Constructor

See also

[Robot_VREP\(const int clientId, std::vector<std::string> jointNames\)](#)

3.1.1.2 Robot_VREP::Robot_VREP (const int *clientId*, std::vector< std::string > *jointNames*)

[Robot_VREP](#) Constructor

Parameters

<i>clientId</i>	Id of the VREP client
<i>jointNames</i>	list of the joint names of the robot

See also

[Robot_VREP\(\)](#)

3.1.1.3 `Robot_VREP::~~Robot_VREP ()`

[Robot_VREP](#) Destructor

3.1.2 Member Function Documentation

3.1.2.1 `double Robot_VREP::getAngularForce (const unsigned int jointID)`

Gets the angular force of a given joint

Parameters

<i>jointID</i>	id of the joint we want to get the force of
----------------	---

Returns

the angular force of the joint in radians (double)

See also

[updateAngularForce\(const unsigned int index\)](#)
[setForce\(const unsigned int jointID, const double force\)](#)
[getAngularPosition\(const unsigned int jointID\)](#)
[getAngularVelocity\(const unsigned int jointID\)](#)

3.1.2.2 `double Robot_VREP::getAngularPosition (const unsigned int jointID)`

Gets the angular position of a given joint

Parameters

<i>jointID</i>	id of the joint we want to get the position of
----------------	--

Returns

the angular position of the joint in radians (double)

See also

[updateAngularPosition\(const unsigned int index\)](#)
[setPosition\(const unsigned int jointID, const double pos\)](#)
[getAngularVelocity\(const unsigned int jointID\)](#)
[getAngularForce\(const unsigned int jointID\)](#)

3.1.2.3 double Robot_VREP::getAngularVelocity (const unsigned int *jointID*)

Gets the angular velocity of a given joint

Parameters

<i>jointID</i>	id of the joint we want to get the velocity of
----------------	--

Returns

the angular velocity of the joint in radians (double)

See also

[updateAngularVelocity\(const unsigned int index\)](#)
[setVelocity\(const unsigned int jointID, const double vel\)](#)
[getAngularPosition\(const unsigned int jointID\)](#)
[getAngularForce\(const unsigned int jointID\)](#)

3.1.2.4 double Robot_VREP::getMicroseconds ()

Updates the clock

Returns

elapsed time in microseconds (double)

See also

[startClock\(\)](#)
[updateClock\(\)](#)
[getSeconds\(\)](#)

3.1.2.5 double Robot_VREP::getSeconds ()

Updates the clock

Returns

elapsed time in seconds (double)

See also

[startClock\(\)](#)
[updateClock\(\)](#)
[getMicroseconds\(\)](#)

3.1.2.6 `bool Robot_VREP::lockJoint (const int jointHandle)`

Is supposed to lock the joint so that it can't move (doesn't seem to be working...)

Parameters

<i>jointHandle</i>	handle of the joint to lock
--------------------	-----------------------------

Returns

true if the operation succeeded, false otherwise (bool)

3.1.2.7 bool Robot_VREP::setForce (const unsigned int *jointID*, const double *force*)

Sets the angular force of a given joint

Parameters

<i>jointID</i>	id of the joint we want to set the force of
----------------	---

Returns

true if the operation succeeded, false otherwise (bool)

See also

[updateAngularForce\(const unsigned int index\)](#)
[getAngularForce\(const unsigned int jointID\)](#)
[setPosition\(const unsigned int jointID, const double pos\)](#)
[setVelocity\(const unsigned int jointID, const double speed\)](#)

3.1.2.8 bool Robot_VREP::setPosition (const unsigned int *jointID*, const double *pos*)

Sets the angular position of a given joint

Parameters

<i>jointID</i>	id of the joint we want to set the position of
----------------	--

Returns

true if the operation succeeded, false otherwise (bool)

See also

[updateAngularPosition\(const unsigned int index\)](#)
[getAngularPosition\(const unsigned int jointID\)](#)
[setVelocity\(const unsigned int jointID, const double speed\)](#)
[setForce\(const unsigned int jointID, const double force\)](#)

3.1.2.9 `bool Robot_VREP::setVelocity (const unsigned int jointID, const double speed)`

Sets the angular velocity of a given joint

Parameters

<i>jointID</i>	id of the joint we want to set the velocity of
----------------	--

Returns

true if the operation succeeded, false otherwise (bool)

See also

[updateAngularVelocity\(const unsigned int index\)](#)
[getAngularVelocity\(const unsigned int jointID\)](#)
[setPosition\(const unsigned int jointID, const double pos\)](#)
[setForce\(const unsigned int jointID, const double force\)](#)

3.1.2.10 `void Robot_VREP::startClock ()`

Starts the clock

See also

[updateClock\(\)](#)
[getMicroseconds\(\)](#)
[getSeconds\(\)](#)

3.1.2.11 `bool Robot_VREP::updateAngularForce (const unsigned int index = -1)`

Updates the angular force of a given joint (Be careful not to call this function more than necessary since it's terribly slow)

Parameters

<i>index</i>	(optional) id of the joint we want to update. If not set, will update every joint
--------------	---

Returns

true if the operation succeeded, false otherwise (bool)

See also

[updateAngularPosition\(const unsigned int index\)](#)
[updateAngularVelocity\(const unsigned int index\)](#)
[getAngularForce\(const unsigned int jointID\)](#)
[setForce\(const unsigned int jointID, const double force\)](#)

3.1.2.12 `bool Robot_VREP::updateAngularPosition (const unsigned int index = -1)`

Updates the angular position of a given joint

Parameters

<i>index</i>	(optional) id of the joint we want to update. If not set, will update every joint
--------------	---

Returns

true if the operation succeeded, false otherwise (bool)

See also

[updateAngularVelocity\(const unsigned int index\)](#)
[updateAngularForce\(const unsigned int index\)](#)
[getAngularPosition\(const unsigned int jointID\)](#)
[setPosition\(const unsigned int jointID, const double pos\)](#)

3.1.2.13 `bool Robot_VREP::updateAngularVelocity (const unsigned int index = -1)`

Updates the angular velocity of a given joint

Parameters

<i>index</i>	(optional) id of the joint we want to update. If not set, will update every joint
--------------	---

Returns

true if the operation succeeded, false otherwise (bool)

See also

[updateAngularPosition\(const unsigned int index\)](#)
[updateAngularForce\(const unsigned int index\)](#)
[getAngularVelocity\(const unsigned int jointID\)](#)
[setVelocity\(const unsigned int jointID, const double speed\)](#)

3.1.2.14 `void Robot_VREP::updateClock ()`

Updates the clock

See also

[startClock\(\)](#)
[getMicroseconds\(\)](#)
[getSeconds\(\)](#)

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

- Robot_VREP.h
- Robot_VREP.cpp

3.2 Simulation Class Reference

Public Member Functions

- [Simulation](#) ()
- [Simulation](#) (std::vector< double > variables, std::vector< std::string > jointNames)
- [~Simulation](#) ()
- void [start](#) ()
- void [init](#) ()
- void [run](#) ()
- std::vector< double > [step](#) ()
- void [saveToFile](#) (const std::string &path)

3.2.1 Constructor & Destructor Documentation

3.2.1.1 [Simulation::Simulation](#) ()

[Simulation](#) Constructor

See also

[Simulation](#)(std::vector<double> variables, std::vector<std::string> jointNames)

3.2.1.2 [Simulation::Simulation](#) (std::vector< double > *variables*, std::vector< std::string > *jointNames*)

[Simulation](#) Constructor

Parameters

<i>variables</i>	initial values of the variables V1, y1, ss1, V2, y2, ss2
<i>jointNames</i>	joint names of the robot

See also

[Simulation](#)()

3.2.1.3 [Simulation::~~Simulation](#) ()

[Simulation](#) Destructor

3.2.2 Member Function Documentation

3.2.2.1 void [Simulation::init](#) ()

initializes [Simulation](#): initializes functions, Integrator and updates forces.

3.2.2.2 void Simulation::run ()

runs whole [Simulation](#) (Do NOT use this function with the Grapher! Use step only)

3.2.2.3 void Simulation::saveToFile (const std::string & *path*)

saves the recorded values into a csv file

Parameters

<i>path</i>	path of the file where the values are saved
-------------	---

3.2.2.4 void Simulation::start ()

[Simulation](#) start: Connects to remote API server and retrieves sphere informations

3.2.2.5 std::vector< double > Simulation::step ()

[Simulation](#) step

Returns

the new variables values for this step (std::vector<double>)

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

- Simulation.h
- Simulation.cpp

Index

- ~Robot_VREP
 - Robot_VREP, 6
- ~Simulation
 - Simulation, 12
- getAngularForce
 - Robot_VREP, 6
- getAngularPosition
 - Robot_VREP, 6
- getAngularVelocity
 - Robot_VREP, 7
- getMicroseconds
 - Robot_VREP, 7
- getSeconds
 - Robot_VREP, 7
- init
 - Simulation, 12
- lockJoint
 - Robot_VREP, 7
- Robot_VREP, 5
 - ~Robot_VREP, 6
 - getAngularForce, 6
 - getAngularPosition, 6
 - getAngularVelocity, 7
 - getMicroseconds, 7
 - getSeconds, 7
 - lockJoint, 7
 - Robot_VREP, 5
 - setForce, 9
 - setPosition, 9
 - setVelocity, 9
 - startClock, 10
 - updateAngularForce, 10
 - updateAngularPosition, 10
 - updateAngularVelocity, 11
 - updateClock, 11
- run
 - Simulation, 12
- saveToFile
 - Simulation, 13
- setForce
 - Robot_VREP, 9
- setPosition
 - Robot_VREP, 9
- setVelocity
 - Robot_VREP, 9
- Simulation, 12
 - ~Simulation, 12
 - init, 12
 - run, 12
 - saveToFile, 13
 - Simulation, 12
 - start, 13
 - step, 13
- start
 - Simulation, 13
- startClock
 - Robot_VREP, 10
- step
 - Simulation, 13
- updateAngularForce
 - Robot_VREP, 10
- updateAngularPosition
 - Robot_VREP, 10
- updateAngularVelocity
 - Robot_VREP, 11
- updateClock
 - Robot_VREP, 11