

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	CPG Class Reference	5
3.1.1	Constructor & Destructor Documentation	5
3.1.1.1	CPG()	5
3.1.1.2	CPG(const double *params, std::vector< double > var)	5
3.1.1.3	~CPG()	6
3.1.2	Member Function Documentation	6
3.1.2.1	finalise()	6
3.1.2.2	getData()	6
3.1.2.3	init()	6
3.1.2.4	operator=(const CPG &cpg)	6
3.1.2.5	setInput(const double f)	6
3.1.2.6	step(const double vmes, const double t, const double dt)	7
3.1.3	Friends And Related Function Documentation	7
3.1.3.1	coupleCPG	7
3.2	PIDController Class Reference	7
3.2.1	Constructor & Destructor Documentation	7
3.2.1.1	PIDController()	7
3.2.1.2	PIDController(const double kp, const double ki, const double kd, const double dt=0.01)	8
3.2.1.3	~PIDController()	8
3.2.2	Member Function Documentation	8
3.2.2.1	update(const double currentPos, const double targetVel)	8
	Index	9

Chapter 1

README

IMPORTANT:

This code has been tested on Mac OSX and Ubuntu. We do NOT support Windows. If you endeavour to run it on Windows, you're on your own.

REQUIRED DEPENDENCIES:

OpenGL <https://www.opengl.org>

GLFW <http://www.glfw.org>

GLEW (Linux ONLY) <http://glew.sourceforge.net>

V-REP <http://www.coppeliarobotics.com>

COMPILING

First, make sure that all the required dependencies (see above) are installed.

For Mac, you can use the .xcodfile and simply run it or follow these Ubuntu instructions:

```
cd wherever_the_code_is/build
```

```
cmake ..
```

```
make
```

RUNNING

```
./Graphical
```

Before running ./Graphical, open the VREP file mico.ttt in the V-REP file.

Chapter 2

Class Index

2.1 Class List

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

CPG	5
PIDController	7

Chapter 3

Class Documentation

3.1 CPG Class Reference

Public Member Functions

- [CPG](#) ()
- [CPG](#) (const double *params, std::vector< double > var)
- [~CPG](#) ()
- void [setInput](#) (const double f)
- double [step](#) (const double vmes, const double t, const double dt)
- void [init](#) ()
- void [finalise](#) ()
- [CPG](#) & [operator=](#) (const [CPG](#) &cpg)
- std::vector< double > [getData](#) ()

Friends

- void [coupleCPG](#) ([CPG](#) *cpg1, [CPG](#) *cpg2)

3.1.1 Constructor & Destructor Documentation

3.1.1.1 [CPG::CPG](#) ()

[CPG](#) constructor

See also

[CPG](#)(const double* params, std::vector<double> var)

3.1.1.2 [CPG::CPG](#) (const double * *params*, std::vector< double > *var*)

[CPG](#) constructor

See also

[CPG](#)()

3.1.1.3 CPG::~~CPG ()

CPG destructor

3.1.2 Member Function Documentation

3.1.2.1 void CPG::finalise ()

finalisation function, to be called after init

See also

[CPG\(const double* params, std::vector<double> var\)](#)

3.1.2.2 std::vector< double > CPG::getData ()

data getter function

Returns

a vector

See also

[step\(const double vmes, const double t, const double dt\)](#)

3.1.2.3 void CPG::init ()

CPG initialisation function

See also

[finalize\(\)](#)

3.1.2.4 CPG& CPG::operator= (const CPG & *cpg*) `[inline]`

CPG equal operator

3.1.2.5 void CPG::setInput (const double *f*)

sets the GPS input

Parameters

<i>f</i>	input force
----------	-------------

3.1.2.6 `double CPG::step (const double vmes, const double t, const double dt)`

step function

Parameters

<i>vmes</i>	current joint velocity
<i>t</i>	current time
<i>dt</i>	timestep

3.1.3 Friends And Related Function Documentation

3.1.3.1 `void coupleCPG (CPG * cpg1, CPG * cpg2)` [[friend](#)]

[CPG](#) coupling function

Parameters

<i>cpg1</i>	first CPG
<i>cpg2</i>	secong CPG

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

- [CPG.h](#)
- [CPG.cpp](#)

3.2 PIDController Class Reference

Public Member Functions

- [PIDController](#) ()
- [PIDController](#) (const double *kp*, const double *ki*, const double *kd*, const double *dt*=0.01)
- [~PIDController](#) ()
- double [update](#) (const double *currentPos*, const double *targetVel*)

3.2.1 Constructor & Destructor Documentation

3.2.1.1 `PIDController::PIDController ()`

PID Controller Constructor

See also

[PIDController\(const double *kp*, const double *ki*, const double *kd*, const double *dt*\)](#)

3.2.1.2 `PIDController::PIDController (const double kp, const double ki, const double kd, const double dt = 0.01)`

PID Controller Constructor

Parameters

<i>kp</i>	P gain (double)
<i>ki</i>	I gain (double)
<i>kd</i>	D gain (double)
<i>dt</i>	timestep (double)

See also

[PIDController\(\)](#)

3.2.1.3 `PIDController::~~PIDController ()`

PID Controller Destructor

3.2.2 Member Function Documentation

3.2.2.1 `double PIDController::update (const double currentPos, const double targetVel)`

updates the controller. Here we transform velocity control into position control to use the traditional PID formulaes.

Parameters

<i>currentPos</i>	current joint angular position (double)
<i>targetVel</i>	target velocity (double)

Returns

velocity to be applied to the joint (double)

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

- PIDController.h
- PIDController.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